Informatica

Informatica® PowerCenter
10.1.1

Designer Guide

Informatica PowerCenter Designer Guide
10.1.1
December 2016

© Copyright Informatica LLC 1999, 2018

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.

Informatica, the Informatica logo, and PowerCenter 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, including without limitation: Copyright DataDirect Technologies. All rights reserved. Copyright © Sun Microsystems. All rights reserved. Copyright © RSA Security Inc. All Rights Reserved. Copyright © Ordinal Technology Corp. All rights reserved. Copyright © Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright © Meta Integration Technology, Inc. All rights reserved. Copyright © Intalio. All rights reserved. Copyright © Oracle. All rights reserved. Copyright © Adobe Systems Incorporated. All rights reserved. Copyright © DataArt, Inc. All rights reserved. Copyright © ComponentSource. All rights reserved. Copyright © Microsoft Corporation. All rights reserved. Copyright © Rogue Wave Software, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © Yahoo! Inc. All rights reserved. Copyright © Glyph & Cog, LLC. All rights reserved. Copyright © Thinkmap, Inc. All rights reserved. Copyright © Clearpace Software Limited. All rights reserved. Copyright © Information Builders, Inc. All rights reserved. Copyright © OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved. Copyright Cleo Communications, Inc. All rights reserved. Copyright © International Organization for Standardization 1986. All rights reserved. Copyright © ej-technologies GmbH. All rights reserved. Copyright © Jaspersoft Corporation. All rights reserved. Copyright © International Business Machines Corporation. All rights reserved. Copyright © yWorks GmbH. All rights reserved. Copyright © Lucent Technologies. All rights reserved. Copyright © University of Toronto. All rights reserved. Copyright © Daniel Veillard. All rights reserved. Copyright © Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright © MicroQuill Software Publishing, Inc. All rights reserved. Copyright © PassMark Software Pty Ltd. All rights reserved. Copyright © LogiXML, Inc. All rights reserved. Copyright © 2003-2010 Lorenzi Davide, All rights reserved. Copyright © Red Hat, Inc. All rights reserved. Copyright © The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright © EMC Corporation. All rights reserved. Copyright © Flexera Software. All rights reserved. Copyright © Jinfonet Software. All rights reserved. Copyright © Apple Inc. All rights reserved. Copyright © Telerik Inc. All rights reserved. Copyright © BEA Systems. All rights reserved. Copyright © PDFlib GmbH. All rights reserved. Copyright © Orientation in Objects GmbH. All rights reserved. Copyright © Tanuki Software, Ltd. All rights reserved. Copyright © Ricebridge. All rights reserved. Copyright © Sencha, Inc. All rights reserved. Copyright © Scalable Systems, Inc. All rights reserved. Copyright © jQWidgets. All rights reserved. Copyright © Tableau Software, Inc. All rights reserved. Copyright© MaxMind, Inc. All Rights Reserved. Copyright © TMate Software s.r.o. All rights reserved. Copyright © MapR Technologies Inc. All rights reserved. Copyright © Amazon Corporate LLC. All rights reserved. Copyright © Highsoft. All rights reserved. Copyright © Python Software Foundation. All rights reserved. Copyright © BeOpen.com. All rights reserved. Copyright © CNRI. All rights reserved.

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

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

The product includes ACE(TM) and TAO(TM) software copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (©) 1993-2006, all rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (copyright The OpenSSL Project. All Rights Reserved) and redistribution of this software is subject to terms available at http://www.openssl.org and http://www.openssl.org/source/license.html.

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

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

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

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

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

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

This product includes software copyright © 2007 The Eclipse Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http:// www.eclipse.org/org/documents/epl-v10.php and at http://www.eclipse.org/org/documents/edl-v10.php.

This product includes software licensed under the terms at http://www.tcl.tk/software/tcltk/license.html, http://www.bosrup.com/web/overlib/?License, http:// www.stlport.org/doc/ license.html, http://asm.ow2.org/license.html, http://www.cryptix.org/LICENSE.TXT, http://hsqldb.org/web/hsqlLicense.html, http:// httpunit.sourceforge.net/doc/ license.html, http://jung.sourceforge.net/license.txt , http://www.gzip.org/zlib/zlib_license.html, http://www.openldap.org/software/ release/license.html, http://www.libssh2.org, http://slf4j.org/license.html, http://www.sente.ch/software/OpenSourceLicense.html, http://fusesource.com/downloads/ license-agreements/fuse-message-broker-v-5-3- license-agreement; http://antlr.org/license.html; http://aopalliance.sourceforge.net/; http://www.bouncycastle.org/ licence.html; http://www.jgraph.com/jgraphdownload.html; http://www.jcraft.com/jsch/LICENSE.txt; http://jotm.objectweb.org/bsd_license.html; . http://www.w3.org/ Consortium/Legal/2002/copyright-software-20021231; http://www.slf4j.org/license.html; http://nanoxml.sourceforge.net/orig/copyright.html; http://www.json.org/ license.html; http://forge.ow2.org/projects/javaservice/, http://www.postgresql.org/about/licence.html, http://www.sqlite.org/copyright.html, http://www.tcl.tk/ software/tcltk/license.html, http://www.jaxen.org/faq.html, http://www.jdom.org/docs/faq.html, http://www.slf4j.org/license.html; http://www.iodbc.org/dataspace/ iodbc/wiki/iODBC/License; http://www.keplerproject.org/md5/license.html; http://www.toedter.com/en/jcalendar/license.html; http://www.edankert.com/bounce/ index.html; http://www.net-snmp.org/about/license.html; http://www.openmdx.org/#FAQ; http://www.php.net/license/3_01.txt; http://srp.stanford.edu/license.txt; http://www.schneier.com/blowfish.html; http://www.jmock.org/license.html; http://xsom.java.net; http://benalman.com/about/license/; https://github.com/CreateJS/ EaselJS/blob/master/src/easeljs/display/Bitmap.js; http://www.h2database.com/html/license.html#summary; http://jsoncpp.sourceforge.net/LICENSE; http:// jdbc.postgresql.org/license.html; http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto; https://github.com/rantav/hector/blob/master/ LICENSE; http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html; http://jibx.sourceforge.net/jibx-license.html; https://github.com/lyokato/libgeohash/blob/ master/LICENSE; https://github.com/hjiang/jsonxx/blob/master/LICENSE; https://code.google.com/p/lz4/; https://github.com/jedisct1/libsodium/blob/master/

LICENSE; http://one-jar.sourceforge.net/index.php?page=documents&file=license; https://github.com/EsotericSoftware/kryo/blob/master/license.txt; http://www.scala-lang.org/license.html; https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt; http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html; https://aws.amazon.com/asl/; https://github.com/twbs/bootstrap/blob/master/LICENSE; https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt; https://github.com/documentcloud/underscore-contrib/blob/master/LICENSE, and https://github.com/apache/hbase/blob/master/LICENSE.txt.

This product includes software licensed under the Academic Free License (http://www.opensource.org/licenses/afl-3.0.php), the Common Development and Distribution License (http://www.opensource.org/licenses/cddl1.php) the Common Public License (http://www.opensource.org/licenses/cpl1.0.php), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (http:// www.opensource.org/licenses/bsd-license.php), the new BSD License (http://opensource.org/licenses/BSD-3-Clause), the MIT License (http://www.opensource.org/licenses/mit-license.php), the Artistic License (http://www.opensource.org/licenses/artistic-license-1.0) and the Initial Developer's Public License Version 1.0 (http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/).

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

This product includes software Copyright (c) 2013 Frank Balluffi and Markus Moeller. All rights reserved. Permissions and limitations regarding this software are subject to terms of the MIT license.

See patents at https://www.informatica.com/legal/patents.html.

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

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

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

The information in this documentation is subject to change without notice. If you find any problems in this documentation, please report them to us in writing at Informatica LLC 2100 Seaport Blvd. Redwood City, CA 94063.

INFORMATICA LLC 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: 2018-06-25

Table of Contents

Preface 15

Informatica Resources. 15

Informatica Network. 15

Informatica Knowledge Base. 15

Informatica Documentation. 15

Informatica Product Availability Matrixes. 16

Informatica Velocity. 16

Informatica Marketplace. 16

Informatica Global Customer Support. 16

Chapter 1: Using the Designer. 17

Using the Designer Overview. 17

Designer Tools 17

Designer Windows. 18

Configuring Designer Options. 19

Configuring General Options. 19

Configuring Tables Options. 21

Configuring Format Options. 22

Configuring Debug Options. 24

Configuring Web Service Options 24

Configuring Miscellaneous Options. 25

Using Toolbars. 26

Designer Toolbars. 26

Workflow Manager Toolbars. 26

Workflow Monitor Toolbars. 27

Repository Manager Toolbars. 27

Displaying a Toolbar. 27

Creating a Toolbar. 28

Configuring a Toolbar. 28

Adding Custom Tools. 28

Navigating the Workspace. 29

Search Tools 29

Iconizing and Restoring Workspace Objects 31

Arranging Workspace Objects. 32

Zooming the Workspace. 32

Viewing the Workspace. 33

Designer Tasks 33

Adding a Repository. 33

Printing the Workspace. 34

Viewing the Last Saved Date/Time 34

Opening and Closing a Folder. . . . 34
Creating Shortcuts. . . . 34
Checking Out and In Versioned Objects. . . . 35
Searching for Objects. . . . 35
Entering Descriptions for Repository Objects. . . . 35
Viewing and Comparing Versioned Repository Objects 35
Reverting to a Previous Object Version. . . . 36
Copying Designer Objects. . . . 36
Exporting and Importing Objects. . . . 37
Working with Multiple Ports or Columns. . . . 37
Renaming Ports. . . . 38
Using Shortcut Keys. . . . 38
Previewing Data. . . . 39
Previewing Relational Data. . . . 39
Previewing Flat File Data. . . . 40
Previewing XML Data. . . . 40
Working with Metadata Extensions. . . . 41
Creating Metadata Extensions. . . . 41
Editing Metadata Extensions. . . . 42
Deleting Metadata Extensions. . . . 43
Using Business Names. . . . 43
Adding Business Names to Sources or Targets. . . . 43
Displaying Business Names in the Navigator. . . . 43
Displaying Business Names as Column Names. . . . 44
Using Business Names as Port Names in Source Qualifiers. . . . 44
Using Business Documentation. . . . 44
Specifying the Documentation Path. . . . 44
Creating Links to Documentation Files. . . . 45
Viewing Business Documentation. . . . 45
Viewing Mapplet and Mapping Reports (Deprecated). . . . 46
Viewing a Mapplet Composite Report. . . . 46
Viewing a Mapping Composite Report. . . . 46

Chapter 2: Working with Sources. . . . 47

Working with Sources Overview 47
Oracle Sources. . . . 48
Special Character Handling in Source Definitions. . . . 48
Updating Source Definitions. . . . 49
Creating Sessions. . . . 50
Working with Relational Sources. . . . 50
Relational Source Definitions. . . . 50
Connectivity for Relational Sources. . . . 51
Configuring a Third-Party ODBC Data Source. . . . 52

Importing Relational Source Definitions. . . . 52
Updating a Relational Source Definition. . . . 53
Creating a Source Definition from a Target Definition. . . . 55
Working with COBOL Sources. . . . 56
Importing COBOL Sources. . . . 56
Working with COBOL Copybooks. . . . 57
Steps to Import a COBOL Source Structure. . . . 57
Components in a COBOL Source File. . . . 58
FD Section. . . . 58
Fields. . . . 58
OCCURS. . . . 58
REDEFINES. . . . 58
Configuring COBOL Source Definitions. . . . 59
Configuring the Table Tab. . . . 59
Configuring Advanced Properties. . . . 60
Configuring the Columns Tab. . . . 60
Importing a Microsoft Excel Source Definition. . . . 62
Defining Ranges. . . . 62
Formatting Columns of Numeric Data. . . . 62
Steps to Import a Microsoft Excel Source Definition. . . . 63
Manually Creating a Source Definition. . . . 63
Troubleshooting Sources. . . . 64

Chapter 3: Working with Flat Files. . . . 65

Working with Flat Files Overview. . . . 65
Creating Sessions with Flat File Sources and Targets. . . . 65
Importing Flat Files. . . . 65
Special Character Handling. . . . 66
Selecting Code Pages. . . . 66
Changing Display Fonts. . . . 66
Importing Fixed-Width Flat Files. . . . 67
Importing Delimited Flat Files. . . . 69
Editing Flat File Definitions. . . . 71
Editing Table Options. . . . 72
Editing Columns. . . . 73
Updating Fixed-Width File Properties. . . . 73
Updating Delimited File Properties. . . . 76
Formatting Flat File Columns. . . . 79
Formatting Numeric Columns. . . . 80
Formatting Datetime Columns. . . . 82
Defining Default Datetime and Numeric Formats. . . . 84
Working with File Lists. . . . 85
Working with Shift-Sensitive Flat Files. . . . 85

Importing Flat Files with Shift Keys. 86
Importing Flat Files without Shift Keys. 86
Working with Multibyte Data in Fixed-Width Targets. 87
Troubleshooting Flat Files. 87

Chapter 4: Working with Targets. 88

Working with Targets Overview. 88
Creating Target Definitions. 88
Maintaining Targets and Target Definitions. 89
Oracle Targets. 89
Target Code Pages. 89
Special Character Handling in Target Definitions. 89
Importing a Target Definition. 91
Relational Target Definitions. 91
Connectivity for Relational Targets. 91
Configuring a Third-Party ODBC Data Source. 93
Importing Relational Target Definitions. 93
Creating a Target Definition from a Source Definition. 93
Creating a Target Definition from a Relational Source. 94
Creating a Target Definition from a Flat File Source. 94
Creating a Normalized Target from a COBOL Source. 94
Steps to Create a Target Definition from a Source Definition. 94
Creating a Target Definition from a Transformation. 95
Creating a Target from a Transformation with One Output Group. 95
Creating a Target from a Transformation with Multiple Output Groups. 95
Creating a Target from a Normalizer Transformation. 96
Creating a Target from a Mapplet. 97
Transformation and Target Datatypes. 97
Steps to Create a Target. 98
Manually Creating a Target Definition. 99
Maintaining Relational Target Definitions. 99
Reimporting a Relational Target Definition. 101
Creating a Primary Key-Foreign Key Relationship. 101
Editing Table Options. 101
Editing Columns. 102
Defining Indexes. 103
Creating a Target Table. 104
SQL DDL Commands in the Designer. 104
Dropping and Re-creating Indexes. 105
Re-creating Targets. 105
Troubleshooting Targets. 105

Chapter 5: Mappings. 107
Mappings Overview. 107
Object Dependency. 108
Developing a Mapping. 108
Working with Mappings. 109
Creating a Mapping. 109
Opening a Mapping. 109
Copying a Mapping. 110
Copying Mapping Segments. 110
Copying Mapping Objects. 111
Exporting and Importing a Mapping. 111
Editing a Mapping. 111
Debugging a Mapping. 112
Deleting a Mapping. 112
Viewing Link Paths to a Port. 112
Viewing Source Column Dependencies. 113
Connecting Mapping Objects 114
Options for Linking Ports. 114
Rules and Guidelines for Connecting Mapping Objects. 115
Linking Ports. 115
Manually Linking Ports. 116
Linking Ports by Position. 116
Linking Ports by Name. 117
Propagating Port Attributes. 118
Understanding Dependency Types. 118
Propagating Dependencies in a Link Path. 118
Propagating Implicit Dependencies. 119
Propagated Attributes by Transformation. 120
Rules and Guidelines for Propagating Ports and Attributes. 122
Steps to Propagate Port Attributes. 123
Working with Sources in a Mapping. 123
Working with Relational Sources in a Mapping. 124
Working with Transformations in a Mapping. 125
Working with Mapplets in a Mapping. 125
Working with Targets in a Mapping. 125
Configuring Relational Targets in a Mapping. 126
Configuring Flat File Targets in a Mapping. 126
Configuring XML Targets in a Mapping. 126
Setting the Target Load Order. 126
Creating Target Files by Transaction. 127
Configuring the Target. 127

Configuring the Mapping. 128
Running the Session. 128
Rules and Guidelines for Creating Target Files by Transaction. 128
Example. 128
Working with Relational Targets in a Mapping. 129
Rejecting Truncated and Overflow Data 130
Configuring the Target Update Override. 130
Configuring the Table Name Prefix. 132
Adding Pre- and Post-Session SQL Commands. 132
Overriding the Target Table Name. 133
Validating a Mapping. 133
Connection Validation. 133
Expression Validation. 134
Object Validation. 134
Data Flow Validation. 134
Steps to Validate a Mapping. 135
Using the Workflow Generation Wizard. 136
Workflow Generation Wizard Steps. 136
Troubleshooting Mappings. 137

Chapter 6: Mapplets. 139

Mapplets Overview. 139
Understanding Mapplet Input and Output 140
Mapplet Input. 140
Mapplet Output. 140
Viewing Mapplet Input and Output. 140
Using the Mapplet Designer. 142
Creating a Mapplet. 142
Validating Mapplets. 142
Editing Mapplets. 143
Mapplets and Mappings. 143
Using Mapplets in Mappings. 144
Creating and Configuring Mapplet Ports 145
Connecting to Mapplet Input Ports. 145
Connecting to Mapplet Output Groups. 145
Viewing the Mapplet. 146
Setting the Target Load Plan. 146
Pipeline Partitioning. 146
Rules and Guidelines for Mapplets. 146
Tips for Mapplets. 147

Chapter 7: Mapping Parameters and Variables. 149

Mapping Parameters and Variables Overview. 149

Mapping Parameters. 150
Mapping Variables. 150
Using Mapping Parameters and Variables. 151
Initial and Default Values. 151
Using String Parameters and Variables. 152
Using Datetime Parameters and Variables. 152
Code Page Relaxation. 152
Mapping Parameters. 153
Step 1. Create a Mapping Parameter. 153
Step 2. Use a Mapping Parameter. 154
Step 3. Define a Parameter Value. 155
Mapping Variables. 155
Variable Values. 155
Variable Datatype and Aggregation Type. 156
Variable Functions. 158
Mapping Variables in Mapplets. 158
Using Mapping Variables. 158
Defining Expression Strings in Parameter Files. 161
Tips for Mapping Parameters and Variables. 161
Troubleshooting Mapping Parameters and Variables. 162

Chapter 8: Working with User-Defined Functions. 163

Working with User-Defined Functions Overview. 163
Example. 163
Creating User-Defined Functions. 163
Configuring the Function Type. 164
Configuring Public Functions that Contain Private Functions. 164
Steps to Create a User-Defined Function. 165
Managing User-Defined Functions. 165
Editing User-Defined Functions. 166
Deleting User-Defined Functions. 166
Exporting User-Defined Functions. 166
Validating User-Defined Functions. 167
Copying and Deploying User-Defined Functions. 167
Creating Expressions with User-Defined Functions. 167

Chapter 9: Using the Debugger. 168

Using the Debugger Overview. 168
Debugger Session Types. 168
Debug Process. 169
Creating Breakpoints. 170
Selecting the Instance Name. 171
Creating Error Breakpoints. 172

Creating Data Breakpoints . . . 172
Entering the Data Breakpoint Condition. . . . 173
Steps to Enter Breakpoints. . . . 175
Editing a Breakpoint. . . . 175
Configuring the Debugger. . . . 176
Step 1. Debugger Introduction. . . . 176
Step 2. Select Integration Service and Session Type. . . . 177
Step 3. Select Session Information. . . . 177
Step 4. Set Session Configuration. . . . 178
Step 5. Set Target Options. . . . 178
Running the Debugger. . . . 179
Initializing State. . . . 179
Running State. . . . 179
Paused State. . . . 179
Debugger Tasks. . . . 180
Working with Persisted Values. . . . 181
Designer Behavior. . . . 181
Monitoring the Debugger. . . . 182
Monitoring Debug Indicators. . . . 182
Monitoring Transformation Data. . . . 183
Continuing the Debugger. . . . 184
Monitoring Target Data. . . . 184
Monitoring the Debug Log. . . . 185
Using the Workflow Monitor. . . . 186
Modifying Data. . . . 187
Restrictions. . . . 187
Evaluating Expressions. . . . 188
Evaluating Expressions Using Mapping Variables. . . . 188
Steps to Evaluate Expressions. . . . 189
Copying Breakpoint Information and Configuration. . . . 189
Transferring Breakpoints and Configuration. . . . 189
Troubleshooting the Debugger. . . . 190

Chapter 10: Viewing Data Lineage. . . . 191

Viewing Data Lineage Overview. . . . 191
Configuring Data Lineage Access. . . . 191
Running Data Lineage from the Designer. . . . 192

Chapter 11: Comparing Objects. . . . 193

Comparing Objects Overview. . . . 193
Comparing Sources, Targets, and Transformations . . . 194
Comparing Mappings and Mapplets . . . 195

Chapter 12: Managing Business Components. 197

Managing Business Components Overview. 197

Business Component Locking. 198

Creating Links to Business Component Documentation. 198

Managing Business Components and Directories. 198

Creating and Editing a Directory. 198

Creating a Business Component 199

Deleting a Directory or Business Component. 199

Copying a Directory or Business Component. 200

Chapter 13: Creating Cubes and Dimensions. 201

Creating Cubes and Dimensions Overview. 201

Understanding Multi-Dimensional Metadata. 201

Key Elements of Multi-Dimensional Metadata 202

Creating a Dimension. 203

Step 1. Create a Dimension. 203

Step 2. Add Levels to the Dimension. 203

Step 3. Add Hierarchies to the Dimension. 204

Step 4. Add Levels to the Hierarchy. 204

Creating a Cube. 204

Editing a Cube. 205

Editing a Dimension. 205

Deleting a Cube or Dimension. 205

Opening and Closing a Cube. 206

Viewing Metadata for Cubes and Dimensions. 206

Tips for Cubes and Dimensions. 206

Chapter 14: Using the Mapping Wizards. 208

Maintaining Star Schemas. 208

Understanding the Mapping Wizards. 210

Using the Getting Started Wizard 210

Using the Slowly Changing Dimensions Wizard. 210

Choosing Sources for the Mappings. 211

Creating a Pass-Through Mapping. 212

Understanding the Mapping. 212

Steps to Create a Pass-Through Mapping. 213

Customizing the Mapping. 213

Configuring a Pass-Through Session. 213

Creating a Slowly Growing Target Mapping. 213

Handling Keys. 214

Understanding the Mapping. 214

Steps to Create a Slowly Growing Target Mapping. 215

Configuring a Slowly Growing Target Session. . . . 216
Creating a Type 1 Dimension Mapping. . . . 216
Handling Keys. . . . 216
Understanding the Mapping. . . . 216
Steps to Create a Type 1 Dimension Mapping. . . . 219
Configuring a Type 1 Dimension Session. . . . 220
Creating a Type 2 Dimension/Version Data Mapping. . . . 220
Handling Keys. . . . 221
Numbering Versions. . . . 221
Understanding the Mapping. . . . 222
Steps to Create a Type 2 Dimension/Version Data Mapping. . . . 224
Customizing the Mapping. . . . 225
Configuring a Type 2 Dimension/Version Data Session. . . . 225
Creating a Type 2 Dimension/Flag Current Mapping. . . . 226
Flagging the Current Value. . . . 226
Handling Keys. . . . 226
Understanding the Mapping. . . . 227
Steps to Create a Type 2 Dimension/Flag Current Mapping. . . . 231
Configuring a Type 2 Dimension/Flag Current Session. . . . 232
Creating a Type 2 Dimension/Effective Date Range Mapping. . . . 232
Maintaining the Effective Date Range. . . . 232
Handling Keys. . . . 233
Understanding the Mapping. . . . 233
Steps to Create a Type 2 Dimension/Effective Date Range Mapping. . . . 237
Configuring a Type 2 Dimension/Effective Date Range Session. . . . 238
Creating a Type 3 Dimension Mapping. . . . 238
Saving Previous Values. . . . 238
Handling Keys. . . . 239
Marking the Effective Date. . . . 239
Understanding the Mapping. . . . 239
Steps to Create a Type 3 Dimension Mapping. . . . 242
Configuring a Type 3 Dimension Session. . . . 243
Creating Targets in the Target Database. . . . 244
Scheduling Sessions and Workflows. . . . 244
Creating a Mapping from the Informatica Mapping Templates. . . . 245
Step 1. Select the Mapping Template. . . . 245
Step 2. Specify Mapping Details and Parameter Values. . . . 246
Step 3. Create Mappings and Save Parameter Values. . . . 247
Step 4. Import Mappings into the Repository. . . . 247

Appendix A: Datatype Reference. . . . 248

Datatype Reference Overview. . . . 248
Transformation Data Types. . . . 249

Integer Datatypes. . . . 250
Binary Datatype. . . . 252
Date/Time Datatype. . . . 252
Decimal and Double Data Types. . . . 252
String Datatypes. . . . 254
IBM DB2 and Transformation Datatypes. . . . 254
Informix and Transformation Datatypes. . . . 255
Datatype Synonyms. . . . 256
Microsoft SQL Server and Transformation Datatypes. . . . 257
Datatype Synonyms. . . . 258
Uniqueidentifier Datatype. . . . 258
Oracle and Transformation Datatypes. . . . 259
Number(P,S) Datatype. . . . 259
Char, Varchar, Clob Datatypes. . . . 260
SAP HANA and Transformation Datatypes. . . . 260
Sybase and Transformation Datatypes. . . . 261
Datatype Synonyms. . . . 263
Binary and Varbinary Datatypes for Sybase IQ. . . . 263
Teradata and Transformation Datatypes. . . . 264
Datatype Synonyms. . . . 264
ODBC and Transformation Datatypes. . . . 265
COBOL and Transformation Datatypes. . . . 266
Flat File and Transformation Datatypes. . . . 266
Number Datatype. . . . 267
XML and Transformation Datatypes. . . . 267
Converting Data. . . . 267
Port-to-Port Data Conversion. . . . 267
Converting Strings to Dates. . . . 268
Converting Strings to Numbers. . . . 268

Appendix B: Configure the Web Browser. . . . 269

Configure the Web Browser. . . . 269

Index. . . . 270

Preface

The *PowerCenter® Designer Guide* is written for developers and software engineers responsible for implementing a data warehouse. The *PowerCenter Designer Guide* assumes that you have a solid understanding of your operating systems, relational database concepts, and database engines, flat files, or mainframe system in your environment. This guide also assumes that you are familiar with the interface requirements for your supporting applications.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- 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 search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.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

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

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

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access 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. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access 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 augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

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

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Using the Designer

This chapter includes the following topics:

- Using the Designer Overview, 17
- Configuring Designer Options, 19
- Using Toolbars, 26
- Adding Custom Tools, 28
- Navigating the Workspace, 29
- Designer Tasks , 33
- Previewing Data, 39
- Working with Metadata Extensions, 41
- Using Business Names, 43
- Using Business Documentation, 44
- Viewing Mapplet and Mapping Reports (Deprecated), 46

Using the Designer Overview

The Designer has tools to help you build mappings and mapplets so you can specify how to move and transform data between sources and targets. The Designer helps you create source definitions, target definitions, and transformations to build the mappings.

The Designer includes windows so you can view folders, repository objects, and tasks. You can work in multiple folders and repositories at one time.

You can configure general Designer settings, such as font and background color. You can also configure specific tool settings for each Designer tool.

Designer Tools

The Designer provides the following tools:

- **Source Analyzer.** Import or create source definitions for flat file, XML, COBOL, Application, and relational sources.
- **Target Designer.** Import or create target definitions.
- **Transformation Developer.** Create reusable transformations.
- **Mapplet Designer.** Create mapplets.

- **Mapping Designer.** Create mappings.

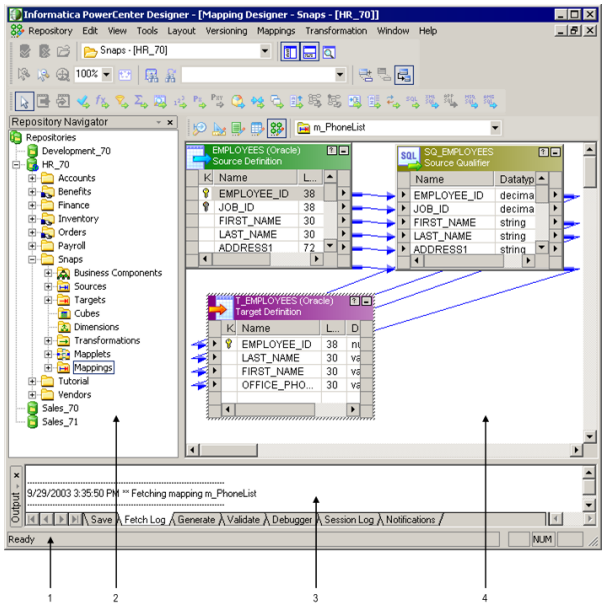
Designer Windows

The Designer consists of the following windows:

- **Navigator.** Connect to multiple repositories and folders. You can also copy and delete objects and create shortcuts using the Navigator.
- **Workspace.** View or edit sources, targets, mapplets, transformations, and mappings. You work with a single tool at a time in the workspace, which has two formats: default and workbook. You can view multiple versions of an object in the workspace.
- **Status bar.** Displays the status of the operation you perform.
- **Output.** Provides details when you perform certain tasks, such as saving work or validating a mapping. Right-click the Output window to access window options, such as printing output text, saving text to file, and changing the font size.
- **Overview.** View workbooks that contain large mappings or a lot of objects. The Overview window outlines the visible area in the workspace and highlights selected objects in color. To open the Overview window, click View > Overview Window.
- **Instance Data.** View transformation data while you run the Debugger to debug a mapping.
- **Target Data.** View target data while you run the Debugger to debug a mapping.

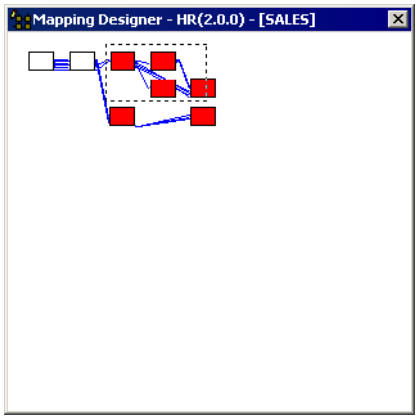
You can view a list of open windows, and you can switch from one window to another in the Designer. To view the list of open windows, click Window > Windows.

The following figure shows the Designer windows:



1. Status Bar
2. Navigator
3. Output
4. Workspace

The following figure shows the Overview window:



The objects within the outline display in the workspace. Objects filled with color are selected in the workspace.

Configuring Designer Options

You can configure how the Designer displays general information, tables, and tools. The Designer lets you specify display options including the background color of the tools, the organization of the navigator window, and the font used in different tools. You can also specify other options such as whether the Source Analyzer imports primary keys. Some changes require you to restart the Designer to apply the changes. The Designer warns you if you need to restart the program.

You can configure the following options in the Designer:

- **General.** You can configure general display options, XML import, file directories, and the organization of the Navigator window.
- **Tables.** You can configure the columns, size, and behavior of table definitions.
- **Format.** You can configure the color, font, and other format options of Designer tools.
- **Debug.** You can configure display and TCP/IP options of the Debugger.
- **Web services.** You can configure backward compatibility and WSDL creation options for the Web Services Hub.
- **Miscellaneous.** You can configure the available functions for the Copy Wizard and checkouts.

Configuring General Options

You can configure general options for the Navigator window, the Display window, and directories.

To configure general options:

1. Click Tools > Options.
2. Click the General tab.

3. You can configure the following general options:

General Option	Description
Reload Tables/Mappings when Opening a Folder	Reloads the last view of a tool when you open it. For example, if you have a mapping open when you disconnect from a repository, the mapping displays the next time you open the folder and the Mapping Designer.
Ask Whether to Reload the Tables/Mappings	Appears only when you select Reload Tables/Mappings when Opening a Folder. Select to have the Designer prompt you to reload objects each time you open a folder.
Display Tool Names on Views	Displays the name of the tool in the upper left corner of the workspace or workbook.
Delay Overview Window Pans	By default, when you drag the focus of the Overview window, the focus of the workbook moves concurrently. When you select this option, the focus of the workspace does not change until you release the mouse button.
Receive Notifications from Repository Service	You can receive notification messages in the Designer and view them in the Output window. Notification messages include information about objects that another user creates, modifies, or deletes. You receive notifications about mappings, mapplets, shortcuts, source definitions, target definitions, and transformations. The Repository Service notifies you of the changes so you know that objects you are working with may be out of date. For the Designer to receive a notification, the folder containing the object must be open in the Navigator. For the Designer to receive notification of a modified or deleted object, the object must be open in the workspace. You also receive user-created notifications that are posted by the user who manages the Repository Service. Default is enabled.
Save All MX Data	Saves all MX data when you save mappings in the Designer. Select this option to use MX views for third-party repository tools. When you save MX data for mappings, PowerCenter creates a field expression for each target field in the mappings. These expressions are different from the transformation expressions that you create in the Expression Editor. You must save and check in the mapping before you can access the MX data. Default is disabled. **Note:** Saving MX data can impact repository performance. Use this option only when you want to use MX views.
Save Only Source/Target Dependencies	Saves only dependency-related MX data when you save mappings in the Designer. Select this option if you do not need to view expressions of fields in MX views, but you use the Repository Manager to view source/target dependencies. Default is disabled.
Group Source by Database	Displays sources grouped by database in the Navigator. Otherwise, sources appear in alphabetic order by source name, with the database name in parentheses.
Display Sources Using Business Names	Displays sources by business names in the Navigator. If you enable the option to display sources by business names, the Navigator displays the business names first, and then the source type name and table name in parentheses.
Display Targets Using Business Names	Displays targets by business names in the Navigator. If you enable the option to display targets by business names, the Navigator displays the business names first, then the target type name and table name in parentheses.

General Option	Description
Workspace File	The directory for workspace files that are created by the Designer. Workspace files maintain the last source or target that you opened, or the last mapping that you saved. This directory should be a local directory to prevent file corruption or overwrites by multiple users. By default, the Designer creates files in the installation directory.
HTML Documentation Path or URL	HTML or URL path to enter links to business documentation.

Configuring Tables Options

You can configure the columns, size, and behavior of table definitions.

To configure table definition options:

1. Click Tools > Options.
2. Click the Tables tab.
3. From the Tables list, select the repository object you want to configure.

   The Columns section lists available display parameters for each repository object. Use the up and down arrows to change the display order of the columns.
4. You can configure the following options for the selected object type:

Tables Option	Tool Availability	Description
Columns Default Width	All tools	Default width of table columns in bytes.
Columns Available	All tools	Columns that are available to display when the repository object is full-size.
Columns Selected	All tools	Columns that appear in the Designer when the repository object is full-size. To display a column, select it in the Columns Available list and click the double-arrow button to move the column into the Columns Selected list. You can also delete columns and change the order in which they appear.
Tools	All tools	Select the Designer tool you want to configure.
Import Primary Keys	Source Analyzer, Target Designer	Designer imports primary keys for relational source or target definitions.
Import Foreign Keys	Source Analyzer, Target Designer	Designer imports foreign keys for relational source or target definitions.
Create Source Qualifier When Opening Sources	Mapping Designer, Mapplet Designer	Designer creates a Source Qualifier or a Normalizer transformation for each source you add to the mapping. Clear the option to manually create all Source Qualifiers and Normalizers. You might want to create a Source Qualifier manually when you use the Source Qualifier to join relational tables.

Tables Option	Tool Availability	Description
Use Business Names as Column Names for Source Qualifier	Mapping Designer, Mapplet Designer	Source Qualifier uses business names as column names.
General Default Width	All tools	Default width of all repository objects.
General Default Height	All tools	Default height of all repository objects.
Show Tooltips	All tools	Shows the column or repository object name when you move the pointer over a column or the title bar of an object. When selected, this option also displays business names when you move the pointer over the icon on the title bar of an object. This option applies to all repository objects.

Some options on the Tables tab are disabled, depending on the tool that you select to edit. Click Reset All to revert to default setting for this tab.

5. Repeat steps 3 to 4 for each repository object type you want to configure.

Configuring Format Options

You can configure the color, font, and other format options for each tool in the Designer.

To configure format options:

1. Click Tools > Options.
2. Click the Format tab.
3. To apply a color theme, click Select Theme.
4. In the Workspace Colors section, select a Designer tool from the Tools menu to configure the workspace colors.
5. Select a workspace element, and click Color to change the color of the element.

   Elements in the Workspace Colors section depend on the tool that you select to configure. You can configure the color of the following elements:

Element	Tool Availability	Description
Background	All tools	Background of the workspace area.
Foreground Text	All tools	Text that appears in the workspace area.
Link Selection	Source Analyzer, Target Designer, Mapplet Designer, Mapping Designer	Selected link between repository objects in the workspace.
Relationship Link	Source Analyzer, Target Designer, Mapplet Designer, Mapping Designer	Link showing primary-key/foreign-key relationship between two tables.

Element	Tool Availability	Description
Propagate Link	Mapplet Designer, Mapping Designer	Link affected by port attribute propagation.
Data Flow Link	Mapplet Designer, Mapping Designer	Link between ports of repository objects in a mapplet or mapping in the workspace.
Metadata Flow Link	Mapplet Designer, Mapping Designer	Link between an MQ Source Qualifier transformation and the associated source.

6. In the Caption Colors section, select an object type from the Tables menu to configure the title bar text and background colors.

   You can configure the following colors:

Option	Description
Foreground	Color of the table caption text.
Background	Background color of the table caption.
Background(2)	Second background color of the table caption. You can combine two background colors to create a color gradient in the title bar of the repository object. Choose the same color for Background and Background(2) if you do not want a color gradient.

7. In the Fonts section, select a component of the Designer from the Categories menu to view or change its font.

   The Current Font field shows the font of the currently selected Designer component. Click Change to change the display font and language script for the Designer component selected in the Categories menu. You might want to do this if the PowerCenter Client code page is different from the source file code page.

Using Color Themes

Use color themes to quickly select the colors of the workspace elements in the Designer tools. When you apply a color theme, you update the colors of the workspace elements in all the Designer tools collectively. You can choose from the following standard color themes:

- **Informatica Classic.** This is the standard color scheme for workspace elements. The workspace background is gray, the workspace text is white, and the link colors are blue, red, blue-gray, dark green, and black.
- **High Contrast Black.** Bright link colors stand out against the black background. The workspace background is black, the workspace text is white, and the link colors are purple, red, light blue, bright green, and white.
- **Colored Backgrounds.** Each Designer tool has a different pastel-colored workspace background. The workspace text is black, and the link colors are the same as in the Informatica Classic color theme.

After you select a color theme for the Designer tools, you can modify the color of individual workspace elements. Changes that you make to individual elements do not appear in the Preview section of the Theme Selector dialog box.

To select a color theme for the Designer tools:

1. In the Designer, click Tools > Options.

2. Click the Format tab.
3. In the Color Themes section of the Format tab, click Select Theme.
4. Select a theme from the Theme menu.
5. Click the tabs in the Preview section to see how the workspace elements appear in each of the Designer tools.
6. Click OK to apply the color theme.

Configuring Debug Options

You can configure Debugger display and TCP/IP options on the Debug tab.

To configure Debug options:

1. Click Tools > Options.
2. Click the Debug tab.
3. You can configure the following options for the Debugger:

Option	Description
Data Display	Select information to display in the Target Data and Instance Data windows.
TCP/IP	Select a specific TCP/IP port or a range of ports for the Designer connection to the Integration Service.

Configuring Web Service Options

You can configure web service options on the WebService tab.

To configure web service options:

1. Click Tools > Options.
2. Click the WebService tab.

3. You can configure the following options for web services:

Option	Description
Create separate groups for message and header ports	Indicates whether to create separate XML views for message and header ports when you add message ports to web service source and target definitions. If you create separate groups for message and header ports, the Designer creates one view for the message ID and another view for the header ports. Web service source and target definitions that include message and header ports in separate views match the format of previous versions of web service source or target definitions. This option is provided for backward compatibility to PowerCenter version 8.1.x.
Create separate targets for faults	Indicates whether to create separate target definitions for fault responses. If you create separate target definitions for fault responses, the Designer creates a target definition for each fault in the output message. Web service targets that include a separate definition for each fault in the output message match the format of previous versions of web service target definitions. This option is provided for backward compatibility to PowerCenter version 8.1.x.
Create WSDL	Indicates whether to generate the WSDL for mappings generated from relational or flat file sources or targets, transformations, or mapplets. If you generate a web service mapping without a WSDL, the Designer can generate the WSDL after you create the mapping. Click the browse button to select the directory in which the Designer creates the WSDL files. The name of the WSDL file is the name of the mapping with a .wsdl extension.

Configuring Miscellaneous Options

You can configure Copy Wizard and versioning options on the Miscellaneous tab.

To configure Miscellaneous options:

1. Click Tools > Options.
2. Click the Miscellaneous tab.
3. You can configure the following options:

Option	Description
Generate Unique Name When Resolved to “Rename”	Generates unique names for copied objects if you select the Rename option in the Copy Wizard window. For example, if the source object s_customers has the same name as a source in the destination folder, the Rename option generates the unique name s_customers1.
Show Checkout Image in Navigator	Displays the Check Out icon when you check out an object.
Allow Delete Without Checkout	You can delete versioned repository objects without first checking them out. You cannot, however, delete an object that another user has checked out. If you select this option, the Repository Service checks out an object to you when you delete it.

Option	Description
Check In Deleted Objects Automatically After They Are Saved	Checks in deleted objects after you save the changes to the repository. When you clear this option, the deleted object remains checked out and you must check it in from the results view.
Reset All	Resets all Miscellaneous options to the default values.

Using Toolbars

Toolbars allow you to select tools and tasks quickly. You can configure the toolbars you want to display or hide. You can also create a new toolbar and add or remove toolbar buttons.

Designer Toolbars

You can configure the Designer to display the following toolbars:

- **Standard.** Contains buttons to connect to and disconnect from repository and folders, and to toggle views and columns.
- **Repository.** Contains buttons to connect to and disconnect from repositories and folders, export and import objects, save changes, and print the workspace.
- **View.** Contains buttons to configure toolbars, toggle windows, toggle full-screen view, change workspace mode, and view properties.
- **Layout.** Contains buttons to iconize, and arrange repository objects, copy, link, and move columns, and zoom in/out of the workspace.
- **Mappings/Mapplets.** Contains buttons to create, edit, and parse mappings and mapplets.
- **Transformations.** Contains buttons to create transformations.
- **Advanced Transformations.** Contains buttons to create advanced transformations.
- **Versioning.** Contains buttons to check in objects, undo checkouts, find checkouts, view history, undo labels, and manage queries.
- **Debugger.** Contains buttons to start, stop, and continue the Debugger.
- **Tools.** Contains buttons to connect to the other PowerCenter Client applications. When you use a Tools button to open another PowerCenter Client application, PowerCenter uses the same repository connection to connect to the repository and opens the same folders.

If you do not see all buttons in the toolbar, you can configure the toolbar.

Workflow Manager Toolbars

The Workflow Manager can display the following toolbars to help you select tools and perform operations quickly:

- **Standard.** Contains buttons to connect to and disconnect from repositories and folders, toggle windows, zoom in and out, pan the workspace, and find objects.
- **Connections.** Contains buttons to create and edit connections, services, and server grids.
- **Repository.** Contains buttons to connect to, disconnect from, and add repositories, open folders, close tools, save changes to repositories, and print the workspace.

- **View.** You can customize toolbars, toggle the status bar and windows, toggle full-screen view, create a new workbook, and view the properties of objects.
- **Layout.** Contains buttons to arrange objects in the workspace, find objects, zoom in and out, and pan the workspace.
- **Tasks.** Contains buttons to create tasks.
- **Workflow.** Contains buttons to edit workflow properties.
- **Run.** Contains buttons to schedule the workflow, start the workflow, or start a task.
- **Versioning.** Contains buttons to check in objects, undo checkouts, compare versions, compare versions, list checked-out objects, and list repository queries.
- **Tools.** Contains buttons to connect to the other PowerCenter Client applications. When you use a Tools button to open another PowerCenter Client application, PowerCenter uses the same repository connection to connect to the repository and opens the same folders. By default, PowerCenter displays the Tools toolbar.

Workflow Monitor Toolbars

The Workflow Monitor can display the following toolbars to help you perform operations quickly:

- **Standard.** Contains buttons to connect to and disconnect from repositories, print, view print previews, search the workspace, show or hide the navigator in task view, and show or hide the output window.
- **Integration Service.** Contains buttons to connect to and disconnect from Integration Services, ping Integration Service, and perform workflow operations.
- **View.** Contains buttons to configure time increments and show properties, workflow logs, or session logs.
- **Filters.** Contains buttons to display most recent runs, and to filter tasks, Integration Services, and folders.

Repository Manager Toolbars

The Repository Manager displays the following toolbars by default:

- **Standard.** Contains buttons to connect to and disconnect from repositories, view dependencies for selected objects, search by keyword, view object properties, close the Dependency window, and toggle the Navigator and Output windows.
- **Tools.** Contains buttons to connect to the other PowerCenter Client applications. When you use a Tools button to open another PowerCenter Client application, PowerCenter uses the same repository connection to connect to the repository and opens the same folders.

Displaying a Toolbar

You can configure a toolbar to display at all times. You can configure the toolbars to display in the Designer, Workflow Manager, and the Workflow Monitor.

To display a toolbar:

1. Click Tools > Customize.
2. On the Toolbars tab, select the toolbars you want to display.
3. To show tooltips as you hold the pointer over a toolbar button, select Show Tooltips.
4. Click OK.

Creating a Toolbar

You can create a new toolbar and choose buttons for the new toolbar. You can create toolbars in the Designer, Workflow Manager, and the Workflow Monitor.

To create a new toolbar:

1. Click Tools > Customize.
2. On the Toolbars tab, click New.
3. Enter a name for the new toolbar and click OK.

   A new floating toolbar appears, and the toolbar name appears in the toolbar list.
4. Click the Commands tab.
5. Select a toolbar from the Categories list to view the buttons available.
6. Drag any button from the Buttons area to the new floating toolbar.
7. Click OK when you finish creating the toolbar.

Configuring a Toolbar

To configure a toolbar, add or remove buttons on the toolbar. You can configure toolbars to display in the Designer, Workflow Manager, and the Workflow Monitor.

To configure a toolbar:

1. Click Tools > Customize.
2. Verify you select the toolbar you want to configure.

   The toolbar appears.
3. To remove a button from the toolbar, drag the button from the toolbar to the Customize dialog box.
4. To add a button, click the Commands tab.
5. Select a toolbar from the Categories list to view the buttons available.
6. Drag any button from the Buttons area to the customized toolbar.
7. Click OK when you finish customizing the toolbar.

Adding Custom Tools

The Designer lets you add custom tools to the Tools menu. You can start programs that you frequently use from within the Designer. For example, you can start the Business Objects Designer or a web browser from within the Designer.

When you add custom tools, the Designer appends the tool names to the bottom of the Tools menu. To start a tool, click the Tools menu and select the custom tool.

You can add, delete, and rearrange the order of custom tools. Click Tools > Customize and click the Tools tab.

The Menu Contents list displays the custom tools. The Designer places the tool names at the bottom of the Tools menu in the order in which they appear in this list.

To add a custom tool:

1. Click the Add Custom Tool button.

   You can add up to nine tools to the Tools menu.

2. In the Menu Contents field, enter a unique name for the custom tool. The Designer displays this name on the Tools menu.

   Tip: Enter an ampersand (&) before a character in the tool name to use that character as a quick access key.

3. Use the arrow buttons to place the new tool in the appropriate position on the Tools menu.
4. Enter the following information for the tool:

Option	Description
Command	Name and file path of the tool executable file. Click the Browse button to select the executable file.
Arguments	Arguments that the Designer passes to the custom tool. The arguments are optional or required, depending on the tool. If you want the Designer to prompt you for arguments when you start the custom tool, select Prompt for Arguments.
Initial Directory	Directory in which the custom tool should start. If you do not enter an initial directory, the Designer uses the custom tool program executable directory.

5. Click OK.

Navigating the Workspace

When you view or edit repository objects in the workspace, complete the following tasks to navigate the workspace easily:

- Search for columns or ports.
- Resize repository objects.
- Zoom in or zoom out of the workspace.

Search Tools

The Designer includes the Find Next and Find in Workspace tools to help you find columns or ports in repository objects or strings in the output window.

Find Next

Use the Find Next tool to search for a column or port name in:

- Transformations
- Mapplets
- Source definitions
- Target definitions

With the Find Next tool, you can search one object at a time. You cannot search multiple objects at the same time. Use Find Next in each Designer tool. Select a single transformation or click in the Output window before performing the search.

The Designer saves the last 10 strings searched in the Find Next box on the Standard toolbar.

To find a column or port name:

1. Select the transformation, mapplet, source or target definition, or click the Output window.
2. Enter the text you want to search in the Find box on the Standard toolbar. For example, you might enter **add** to locate an address column.

   Tip: The search is not case sensitive.
3. Click Edit > Find Next, click the Find Next button, or press Enter to search for the string.

   The Designer finds the first occurrence of the search string.

   Tip: You can also press F3 to search for the string.
4. Press Enter again to search for the next occurrence of the search string. The Designer searches each port or column name in the transformation and wraps to the top of the transformation until it finds a match.

Find in Workspace

You can search for a string in the Save, Generate, or Validate tabs in the Output window. The Find in Workspace tool searches for a field name or transformation name in all transformations in the workspace.

The Find in Workspace tool lets you to search all of the transformations in the workspace for port or transformation names. You can search for column or port names or table names matching the search string. You can specify whether to search across all names in the workspace, or across the business name of a table, column, or port. You can also choose to search for whole word matches for the search string or matches which match the case of the search string.

To find a column, port, or transformation name in the workspace:

1. In a Designer tool, click the Find in Workspace button, or click Edit > Find in Workspace.

   The Find in Workspace dialog box opens.
2. Choose to search for field names or table names matching the search text.

Look for Option	Description
Fields	Designer searches for column or port names matching the search text.
Tables	Designer searches for table names matching the search text.

3. Specify the search string in the Find What entry field, or select a search string from the list.

   The Designer saves the last 10 search strings in the list.

4. Specify whether you want to look for matches to the search text among all names or business names.

Find In Option	Description
Names	Designer searches across all names of tables, columns, or ports in the workspace.
Business Names	Designer searches across the business names of tables, columns, or ports in the workspace.

5. Specify search criteria.

Search Option	Description
Look in all Tables	Select this option to search across all tables in the workspace. If you do not select this option, the Designer searches across all currently selected tables in the workspace. This option is selected when you search for table names matching the search string.
Match Whole Word Only	Select this option to find names or business names that match the specified string.
Match Case	Select this option if you want to search for strings that match the case of the specified search string.

6. Click Find Now.

   All matches appear in the table at the bottom of the Find in Workspace dialog box.

7. Click Close.

Iconizing and Restoring Workspace Objects

Each Designer tool displays different objects in the workspace. You might view source definitions, target definitions, transformations, mapplets, or entire mappings, depending on which tool you have open. The Designer can display repository objects in the following forms:

- **Normal.** The Designer displays the information in each object in columns. The Designer displays objects normally by default.
- **Iconized.** The Designer reduces objects to a named icon. You might iconize objects when working with large mappings. You can view descriptions associated with iconized objects by holding the pointer above the icon.
- **Zoom.** The Designer changes the magnification level of the normal and iconized objects in the workspace when you use the zoom feature. You can apply magnification levels from 30 to 100 percent in 10 percent increments.

Iconizing Workspace Objects

You can iconize objects in the workspace.

To iconize workspace objects:

1. Select the object in the workspace.

   To select more than one object, Shift-click or Ctrl-click multiple objects. You can also select multiple objects by clicking Edit > Select All or dragging a rectangle around those objects.

2. Click Layout > Arrange All Iconic.

   Tip: You can also right-click and select Arrange All Iconic.

Restoring Workspace Objects

You can restore iconized objects to normal size.

To restore iconized workspace objects to normal size:

1. Select the object.

   To select more than one object, Shift-click or Ctrl-click multiple objects.

   You can also select multiple objects by clicking Edit > Select All or dragging a rectangle around those objects.
2. Click Layout > Arrange.

   Tip: You can also restore an object by double-clicking the iconized object or right-clicking the object and selecting Arrange.

Arranging Workspace Objects

You can display specific pipelines in the Mapping Designer.

To display specific pipelines in the Mapping Designer:

1. Click Layout > Arrange.
2. Select pipelines by target.

   You can select the Iconic option to display arranged pipelines in iconized form.
3. Click OK.

Zooming the Workspace

You can zoom in and out of the workspace. Use the toolbar or the Layout menu options to set zoom levels. The toolbar has the following zoom options:

- **Zoom in 10% on button.** Uses a point you select as the center point from which to increase the current magnification in 10 percent increments.
- **Zoom out 10% on button.** Uses a point you select as the center point from which to decrease the current magnification in 10 percent increments.
- **Zoom in based on rectangle.** Increases the current magnification of a rectangular area you select. Degree of magnification depends on the size of the area you select, workspace size, and current magnification.
- **Drop-down list.** Maintains the center point of the workspace and sets the zoom level to the percent you select from the list.
- **Scale to fit.** Scales all workspace objects to fit the workspace.

The Layout menu has the following zoom options:

- **Zoom Center.** Maintains the center point of the workspace and zooms in or out by 10 percent increments.
- **Zoom Point.** Uses a point you select as the center point from which to zoom in or out by 10 percent increments.
- **Zoom Rectangle.** Increases the current magnification of a rectangular area you select. Degree of magnification depends on the size of the area you select, workspace size, and current magnification.
- **Zoom Normal.** Sets the zoom level to 100 percent.

- **Scale to Fit.** Scales all workspace objects to fit the workspace.
- **Zoom Percent.** Maintains the center point of the workspace and sets the zoom level to the percent you choose.

When you add objects to a workspace, the Designer uses the magnification level of the other objects. The Designer saves the zoom level for each Designer tool in a folder when you exit.

Viewing the Workspace

You can maximize the workspace window by clicking View > Full Screen. You can see the Standard toolbar and the Transformation toolbar when you are in full screen view. The full screen view hides the menu, the Navigator and Output windows, and the title bar. To use the menu when you are in full screen view, point anywhere on the top of the screen to show the menu.

To go back to normal view, click the Close Full Screen button or press Esc.

Designer Tasks

You can complete the following tasks in each Designer tool:

- Add a repository.
- Print the workspace.
- View date and time an object was last saved.
- Open and close a folder.
- Create shortcuts.
- Check out and in repository objects.
- Search for repository objects.
- Enter descriptions for repository objects.
- View older versions of objects inthe workspace.
- Revert to a previously saved object version.
- Copy objects.
- Export and import repository objects.
- Work with multiple objects, ports, or columns.
- Rename ports.
- Use shortcut keys.

You can also view object dependencies in the Designer.

Adding a Repository

To edit a repository object, first add a repository in the Navigator so you can access the repository object. To add a repository in the Navigator, click Repository > Add. Use the Add Repositories dialog box to add the repository.

Printing the Workspace

To print the workspace:

- Click Repository > Print.

You can also right-click the workspace and choose Print.

Setting up the Workspace Printout

To specify a header and a footer for the workspace print out:

- Click Repository > Page Setup.

Previewing the Workspace Print Out

To preview the workspace before you print:

- Click Repository > Print Preview.

Viewing the Last Saved Date/Time

You can view the date and time an object was last saved in the repository. To view the “Last Saved” date and time, select the object in the Navigator and click View > Properties.

Note: Sources do not have a “Last Saved” date and time.

For Windows, use the Regional Settings in the Control Panel to configure the “Last Saved” date and time format.

Opening and Closing a Folder

Double-click a folder to open a folder and tool at the same time. When you double-click the folder, the Designer opens the folder in the Navigator and displays the last tool that was active within the folder.

You can also select a folder and then choose a tool to open a workspace. If you select the Reload Tables/Mappings When Opening a Folder option, the Designer also displays the last objects open when using that tool.

For example, if you close a folder with the Source Analyzer active, the next time you open the folder using the Open Folder button, the Designer displays the Source Analyzer.

To close a folder:

- Select the folder in the Navigator and click the Disconnect button. To close all open folders and tools, click Repository > Close All Tools.

Creating Shortcuts

To create a shortcut to an object in a shared folder, drag the object into the destination folder or into the mapping. For example, to create a shortcut to a source, drag the source from the shared folder into the mapping you have open in the workspace. You can also create a shortcut by dropping the source into the destination folder. To use the new shortcut, drag it into the workspace.

You can create a shortcut to a shared folder in the same repository. You can also create a shortcut in a local repository that points to a shared folder in the global repository, as long as both repositories are in the same domain.

As with copying objects, the destination folder must be open.

Note: You cannot create shortcuts to objects in non-shared folders.

Checking Out and In Versioned Objects

When you work with versioned objects, you must check out an object to modify it, and save it to commit changes to the repository. Checking in an object adds a new version to the object history. You must check in the object to allow other users to make changes to it.

Searching for Objects

Use an object query to search for objects in the repository that meet specified conditions. When you run a query, the repository returns results based on those conditions. You may want to create an object query to complete the following tasks:

- **Track repository objects during development**. You can add Label, User, Last saved, or Comments parameters to queries to track objects during development.
- **Associate a query with a deployment group**. When you create a dynamic deployment group, you can associate an object query with it.

To create an object query, click Tools > Queries to open the Query Browser.

From the Query Browser, you can create, edit, and delete queries. You can also configure permissions for each query from the Query Browser. You can run any query for which you have read permissions from the Query Browser.

Entering Descriptions for Repository Objects

You can enter descriptions and comments for each repository object. You can enter a maximum number of characters equal to 2,000 bytes/K. K is the maximum number of bytes a character contains in the selected repository code page. For example, if the repository code page is a Japanese code page where K=2, each description and comment field lets you enter up to 1,000 characters.

Viewing and Comparing Versioned Repository Objects

You can view and compare versions of objects. If an object has multiple versions, you can find the versions of the object in the View History window. In addition to comparing versions of an object in a window, you can view the various versions of an object in the workspace to graphically compare them.

Rules and Guidelines for Viewing and Comparing Versioned Repository Objects

Use the following rules and guidelines when you view older versions of objects in the workspace:

- You cannot simultaneously view multiple versions of composite objects, such as mappings and mapplets.
- Older versions of composite objects might not include the child objects that were used when the composite object was checked in. If you open a composite object that includes a child object version that is purged from the repository, the preceding version of the child object appears in the workspace as part of the composite object. For example, you want to view version 5 of a mapping that originally included version 3 of a source definition, but version 3 of the source definition is purged from the repository. When you view version 5 of the mapping, version 2 of the source definition appears as part of the mapping.
- Shortcut objects are not updated when you modify the objects they reference. When you open a shortcut object, you view the same version of the object that the shortcut originally referenced, even if subsequent versions exist.

Viewing an Older Version of a Repository Object

To open an older version of an object in the workspace:

1. In the workspace or Navigator, select the object and click Versioning > View History.
2. Select the version you want to view in the workspace and click Tools > Open in Workspace.

   Note: An older version of an object is read-only, and the version number appears as a prefix before the object name. You can simultaneously view multiple versions of a non-composite object in the workspace.

Comparing Versions of a Repository Object

To compare two versions of an object:

1. In the workspace or Navigator, select an object and click Versioning > View History.
2. Select the versions you want to compare, and then click Compare > Selected Versions.

   To compare a version of the object with the previous version, select a version and click Compare > Previous Version.

   A window appears where you can view detailed information about both versions of the object.

   Note: You can also access the View History window from the Query Results window when you execute a query.

Reverting to a Previous Object Version

When you edit an object in the Designer, you can revert to a previously saved version, undoing changes you entered since the last save. You can revert to the previously saved versions of multiple objects at the same time.

To revert to a previously saved version of an object:

1. Open the object in the workspace.
2. Select the object and click Edit > Revert to Saved.
3. Click Yes. If you selected more than one object, click Yes to All.

   The Designer removes all changes entered since the last time you saved the object.

Copying Designer Objects

You can copy Designer objects within the same folder, to a different folder, or to a different repository. You can copy any of the Designer objects such as sources, targets, mappings, mapplets, transformations, and dimensions. You must open the target folder before you can copy objects to it.

Use the Copy Wizard in the Designer to copy objects. The Copy Wizard checks for conflicts in the target folder and provides choices to resolve the conflicts. For example, if an item already exists in the target folder, a description of the problem displays in the screen. The Copy Wizard displays possible resolutions. For a duplicate object, you can rename, reuse, replace, or skip copying the object.

To configure display settings and functions of the Copy Wizard, click Tools > Options in the Designer.

You can import objects from an XML file through the Import Wizard in the Designer.

The Import Wizard provides the same options to resolve conflicts as the Copy Wizard.

Copying Mapping Segments

You can copy segments of mappings and mapplets when you want to reuse a portion of the mapping logic. A segment consists of one of more objects in a mapping or mapplet. The objects can be sources, targets, shortcuts, transformations, and mapplets. To copy mapping segments, select and copy the segments from one mapping and paste them into a target mapping. You can paste segments of mappings or mapplets into an empty mapping or mapplet workspace. You can also copy segments across folders or repositories.

To copy a segment of a mapping or mapplet:

1. Open a mapping or mapplet.
2. Select a segment by highlighting each object you want to copy.

   You can select multiple objects. You can also select segments by dragging the pointer in a rectangle around objects in the workspace.
3. Copy the segment to the clipboard.
4. Open a target mapping or mapplet.

   Optionally, you can open an empty mapping or mapplet workspace.
5. Click Edit > Paste or press Ctrl+V.

   The Copy Wizard opens if you have copy conflicts.

Copying Objects as an Image

In the Designer workspace, you can copy an image of objects such as mappings or transformations to the Clipboard. You can then paste the image file into the workspace of any application that uses graphics.

To copy objects in the Designer workspace as an image:

1. In the Designer, open the tool associated with the object you want to copy.

   For example, open the Mapping Designer if you want to copy a mapping as an image.
2. Open the object you want to copy.

   If you choose to show iconized mappings, the copied image does not show mapping links. You can copy the mapping links by selecting them manually.
3. Drag the pointer to create a rectangle around the objects you want to select.

   You can also Ctrl-click to select individual objects. However, if you use this method to select objects, the copied image does not show mapping links.
4. Press Shift+Alt+C or click Edit > Copy As Image.

   You can copy one image to the Clipboard at a time.
5. Paste the image into the workspace of any application that uses graphic files.

   For example, you can paste the image into a Microsoft Word document.

Exporting and Importing Objects

To export an object to an XML file, right-click the object and choose Export Objects in the Designer. The Designer saves all necessary information to import that object back into a repository.

To import an object from an XML file in the Designer, click Repository > Import Objects.

Working with Multiple Ports or Columns

In all Designer tools, you can move or delete multiple ports or columns at the same time.

Note: You cannot select multiple ports or columns when editing COBOL sources in the Source Analyzer.

To select consecutive ports or columns:

1. Double-click the source, target, or transformation.
2. In the Edit Tables dialog box, choose the Ports or Columns tab.
3. Select the first port or column using the row header number in the transformation.
4. To select a range to move or delete, hold down the Shift key as you click the last port or column in the range. To select individual ports or columns, hold down the Ctrl key and click each port or column using the row header number you want to move or delete.

   Note: When you select multiple ports or columns, the Designer disables add, copy, and paste.
5. Click Delete.

Renaming Ports

You can rename ports in sources, targets, and transformations. To rename a port in a source, target, or transformation, double-click the port, and type in the new name. The Designer propagates the new name to the mappings and mapplets that use this source, target, or transformation. You can rename ports in the following Designer tools:

- **Source Analyzer.** To rename a port in a source.
- **Target Designer.** To rename a port in a target.
- **Transformation Developer.** To rename a port in a reusable transformation.
- **Mapplet Designer and Mapping Designer.** To rename a port in a non-reusable transformation. You can propagate the new name to other non-reusable transformations that access this port through the Designer.

Using Shortcut Keys

When editing a repository object, use shortcuts on the Ports or Columns tab.

The following table lists the Designer shortcuts:

Task	Shortcut
Add a new field or port.	Alt+F
Cancel editing in a cell.	Esc
Select or clear a port type check box.	Space bar
Copy a row.	Alt+O
Copy text in a cell.	Ctrl+C
Cut a row.	Alt+C
Edit the text of a cell.	F2, then move the cursor to the location inside the cell.
Find all combination and list boxes.	Type the first letter on the list.

Task	Shortcut
Find tables or fields in the workspace.	Ctrl+F
Move current row down.	Alt+W
Move current row up.	Alt+U
Open the Expression Editor from the expression field.	F2, then press F3
Paste a row.	Alt+P
Paste copied text into a cell.	Ctrl+V
Select the text of a cell.	F2
Validate the default value in a transformation.	Alt+V

Previewing Data

You can preview source and target data in the Designer. Preview source or target data before you create a mapping or while you work with a mapping.

You can preview the following types of data:

- **Relational tables and views.** Preview relational sources and targets. You can preview data for a valid relational source definition or target definition in the Designer. A source definition or shortcut to a source definition is valid if it matches the source table. A target definition is valid if it matches the target table. You must be able to connect to the source or target database to preview relational data.
- **Fixed-width and delimited flat files.** Preview flat file sources and targets that do not contain binary data. You can preview data for a valid flat file source or target definition in the Designer. A source or target definition is valid if it matches the source file.
- **XML files**. Preview XML file data in the XML Editor. You can preview XML data using an XML definition and an external XML file. To preview data, you must have a valid XML definition in the repository and data in an external XML file that is valid for the definition. You can view data for one XML view at a time.

  Tip: Preview source data to verify that you extract data from the correct source. You can also preview source data to determine which columns to use in a mapping. You can preview target data before you truncate target tables or when you implement an update strategy.

Previewing Relational Data

To preview relational source or target data:

1. Select a relational source or target definition in the workspace.
2. Right-click the source or target definition in the workspace and choose Preview Data.
3. Select a data source name.
4. Enter a database user name and password.

   The user name must have database permissions to view the object.

5. Enter the database table owner name.
6. Enter the number of rows you want to preview.

   The Preview Data dialog box can display up to 500 rows and up to 65,000 columns.
7. Click Connect.
8. To change the number of rows you want to preview, enter a new number and click Refresh.
9. Click Close.

Previewing Flat File Data

To preview flat file source and target data:

1. Select a flat file source or target definition in the workspace.
2. Right-click the source or target definition in the workspace and choose Preview Data.
3. Click the Browse button to select a flat file.
4. Locate and select the file you want to preview and click Open.
5. Select a code page from the Preview Data dialog box.

   The code page must match the code page of the source. If the code page does not match the code page of the source, the data might display incorrectly.

   The Designer lists the five code pages you have most recently selected. Then it lists all remaining code pages in alphabetical order.
6. Enter the starting row number and the number of rows you want to preview.

   The Preview Data dialog box can display up to 500 rows and up to 65,000 columns.
7. Click Open.

   Note: The Preview Data dialog box displays shift characters display as periods (.) in shift-sensitive flat files.
8. To change the starting row number or the number of rows you want to preview, enter a new number and click Refresh.
9. Click Close.

Previewing XML Data

To preview the XML data:

1. Double-click the XML definition in the workspace.

   The XML editor appears.
2. Select a view in the XML editor workspace.
3. Click XML Views > Preview Data.
4. Browse for the XML file containing data to preview.

   The Preview XML Data dialog box displays data from the XML file using the view that you chose.
5. If you want to use a different XML file, click the Select File icon on the dialog box.
6. Click OK.

Working with Metadata Extensions

You can extend the metadata stored in the repository by associating information with individual repository objects. For example, you may want to store contact information with the sources you create. If you create an Aggregator transformation, you may want to store an email address with that transformation. You associate information with repository objects using metadata extensions.

Repository objects can contain both vendor-defined and user-defined metadata extensions. You can view and change the values of vendor-defined metadata extensions, but you cannot create, delete, or redefine them. You can create, edit, delete, and view user-defined metadata extensions and change their values.

You can create metadata extensions for the following objects in the Designer:

- Source definitions
- Target definitions
- Transformations
- Mappings
- Mapplets

You can create either reusable or non-reusable metadata extensions. You associate reusable metadata extensions with *all* repository objects of a certain type, such as all source definitions or all Expression transformations. You associate non-reusable metadata extensions with a single repository object, such as one target definition or one mapping.

Creating Metadata Extensions

You can create user-defined, reusable, and non-reusable metadata extensions for repository objects using the Designer. To create a metadata extension, you edit the object for which you want to create the metadata extension, and then add the metadata extension to the Metadata Extension tab.

If you have multiple reusable metadata extensions to create, it is easier to create them using the Repository Manager.

To create a metadata extension:

1. Open the appropriate Designer tool.
2. Drag the appropriate object into the workspace.
3. Double-click the title bar of the object to edit it. If the object is a mapping or mapplet, click Mappings > Metadata Extensions or Mapplets > Metadata Extensions.
4. Click the Metadata Extensions tab.

   This tab lists the existing user-defined and vendor-defined metadata extensions. User-defined metadata extensions appear in the User Defined Metadata Domain. If they exist, vendor-defined metadata extensions appear in their own domains.
5. Click the Add button.

   A new row appears in the User Defined Metadata Extension Domain.

6. Enter the following information:

Field	Description
Extension Name	Name of the metadata extension. Metadata extension names must be unique for each type of object in a domain. Metadata extension names cannot contain special characters except underscores and cannot begin with numbers.
Datatype	Select numeric (integer), string, or boolean.
Precision	Maximum length for string metadata extensions.
Value	For a numeric metadata extension, the value must be an integer between -2,147,483,647 and 2,147,483,647. For a boolean metadata extension, choose true or false. For a string metadata extension, click the Open button in the Value field to enter a value of more than one line, up to 2,147,483,647 bytes.
Reusable	Makes the metadata extension reusable or non-reusable. Select to apply the metadata extension to all objects of this type (reusable). Clear to make the metadata extension apply to this object only (non-reusable). If you create a reusable metadata extension for a transformation, the metadata extension applies to all transformations of that type (for example, all Aggregator transformations or all Router transformations), and not to all transformations. **Note:** If you make a metadata extension reusable, you cannot change it back to non-reusable. The Designer makes the extension reusable as soon as you confirm the action.
UnOverride	Restores the default value of the metadata extension when you click Revert. This column appears only if the value of one of the metadata extensions was changed.
Description	Description of the metadata extension.

7. Click OK.

Editing Metadata Extensions

You can edit user-defined, reusable, and non-reusable metadata extensions for repository objects using the Designer. To edit a metadata extension, you edit the repository object, and then make changes to the Metadata Extension tab.

What you can edit depends on whether the metadata extension is reusable or non-reusable. You can promote a non-reusable metadata extension to reusable, but you cannot change a reusable metadata extension to non-reusable.

Editing Reusable Metadata Extensions

If the metadata extension you want to edit is reusable and editable, you can change the value of the metadata extension, but not any of its properties. However, if the vendor or user who created the metadata extension did not make it editable, you cannot edit the metadata extension or its value.

To edit the value of a reusable metadata extension, click the Metadata Extensions tab and modify the Value field. To restore the default value for a metadata extension, click Revert in the UnOverride column.

Editing Non-Reusable Metadata Extensions

If the metadata extension you want to edit is non-reusable, you can change the value of the metadata extension and its properties. You can also promote the metadata extension to a reusable metadata extension.

To edit a non-reusable metadata extension, click the Metadata Extensions tab. You can update the Datatype, Value, Precision, and Description fields.

To make the metadata extension reusable, select Reusable. If you make a metadata extension reusable, you cannot change it back to non-reusable. The Designer makes the extension reusable as soon as you confirm the action.

To restore the default value for a metadata extension, click Revert in the UnOverride column.

Deleting Metadata Extensions

You delete reusable metadata extensions in the Repository Manager.

You can delete non-reusable metadata extensions in the Designer. To delete metadata extensions, edit the repository object, and then delete the metadata extension from the Metadata Extension tab.

Using Business Names

You can add business names to sources, targets, and columns. Business names are descriptive names that you give to a source, target, or column. They appear in the Navigator in the Business Components source node and in the source and target nodes. Business names can also appear as column names of the source and target definition in the workspace. You can also create source qualifiers to display business names as column names in the Mapping and Mapplet Designers.

Adding Business Names to Sources or Targets

You can add business names to source and target definitions. When you import source definitions for PeopleSoft and SAP, the Designer imports business names.

To add a business name to a source or target:

1. In the Source Analyzer or Target Designer, open the source or target definition.
2. In the Edit Tables dialog box, click Rename.
3. Enter the business name in the Business Name field.
4. Click OK.

To display business names in the Navigator, enable the Display Table As Business Name option. The business name appears in the Navigator with the table name in parentheses.

Displaying Business Names in the Navigator

You can configure the Designer to display business names if they exist for source and target definitions in the Navigator. When you enable business names to display in the Navigator, the table names appear in parentheses.

For example, if you create a source definition with the table name EMPLOYEES and business name Employee Data, the Navigator displays the business name with the table name in parentheses.

If you create a shortcut to a source or target when you enable business names to display in the Navigator, the Designer names the shortcut Shortcut_To_*BusinessName*.

Displaying Business Names as Column Names

You can display column business names in source and target definitions. You can configure the Designer to select business names as column names in the workspace. You can have duplicate column business names for source and target definitions.

Using Business Names as Port Names in Source Qualifiers

Use business names of source columns as port names in Source Qualifier transformations.

To add business names to existing Source Qualifiers in a mapping, you must enable the option to display business names. Then delete the existing source and reimport it with the Source Qualifier.

If the source has no business names, the Source Qualifier contains the port names. If the business name contains characters that are not allowed as a port name, the Designer replaces each of the characters with an underscore (_). Business names usually contain spaces between words, which is not allowed in a port name. For example, the business name Employee Data becomes Employee_Data.

Tip: Avoid using PowerCenter reserved words, such as DD_INSERT, for business and port names.

Using Business Documentation

Business documentation provides details about a repository object or transformation expression. You can create and edit links to business documentation that you have developed for repository objects through the Designer. The documentation must reside on a local machine, network server, or company intranet or internet web site in a Windows environment.

You can develop business documentation in HTML, PDF, or any text format, for the following repository objects:

- Source and target tables and table instances
- All transformations and transformation instances
- Mapplets
- Mappings
- Business component directories

To access business documentation, you need to complete the following tasks:

- Specify the documentation path in the Designer.
- Create a link in the repository object.
- Click the link to view the documentation.

Specifying the Documentation Path

You specify a documentation path or root where you store the documentation files. You reference this path in the links you add to each repository object.

If you store the files on a local machine or network server, use a file path. If you place the files on the organization's intranet or internet web site, use a URL.

The following formats are valid documentation paths or roots:

- File path to a local drive, such as c:\doc\informatica\
- File path to a network server, such as \\server5\doc\informatica\
- URL, such as http://www.internal.company.com/doc/

Related Topics:

- "Configuring General Options" on page 19

Creating Links to Documentation Files

You can create links in repository objects to documentation files. Click these links in the Properties or Edit dialog box of each object to view the business documentation.

The links must be a valid URL or file path to reference a business document. Use either of the following formats:

- **Root variable.** Use the string "file://$docroot" to refer to the documentation root you have specified in the documentation path field.

  For example, if a document root is http://internal.company.com/doc, the Designer points to http://internal.company.com/doc/finance/vendors_help.html, but displays file://$docroot as part of the link.
- **Complete file path or link.** Precede the file path with ://, such as file://c:\doc\help\ or http://internal.company.com/doc/help/. A file path must be preceded by file:// to be valid.

To create a link, edit the properties of the object. How you do this depends on the type of object you want to document.

The following table summarizes how to create a link for repository objects:

Repository Object	How to Create Documentation Links
Table/Transformation	- Double-click the table/transformation in the workspace to open it. - Enter the documentation link in the Description window.
Mapping/Mapplet	- Open the mapping/mapplet in the workspace. - Click Mappings > Edit or Mapplets > Edit. - Enter the documentation link in the comment field.
Business Component Directory	- Select the business component directory in the Navigator. - Click Repository > Business Component > Edit Properties. - Enter the documentation link in the comment field.

Tip: To add comments to individual business components, edit the properties of the original object it references.

Viewing Business Documentation

When you click a link to business documentation, Windows launches the corresponding application to view the file.

Viewing Mapplet and Mapping Reports (Deprecated)

You can view PowerCenter Repository Reports for mappings and mapplets in the Designer. View reports to get more information about the sources, targets, ports, and transformations in mappings and mapplets. When you view a report, the Designer launches JasperReports Server in a browser window and displays the report.

You can view the following reports:

- Mapplet Composite Report
- Mapping Composite Report

An administrator uses the Administrator tool to create a Reporting and Dashboards Service and adds a reporting source for the service. The reporting source must be the PowerCenter repository that contains the mappings and mapplets you want to report on.

Deprecated Behavior

Effective in version 10.1, Informatica deprecated the Reporting and Dashboards Service. Informatica plans to drop support for the Reporting and Dashboards Service in a future release.

You can create additional reports based on the PowerCenter repository views. You can use a third-party reporting tool to create reports. For information about the PowerCenter repository views, see the *Informatica PowerCenter Repository Guide*.

Viewing a Mapplet Composite Report

The Mapplet Composite Report includes information about a mapplet:

- **All objects.** Information about all objects in the mapplet.
- **Transformations.** Transformations used in the mapplet.

To view a Mapplet Composite Report:

1. In the Designer, open a mapplet.
2. Right-click in the workspace and choose View Mapplet Report.

The Designer launches JasperReports Server in the default browser for the client machine and runs the Mapplet Composite Report.

Viewing a Mapping Composite Report

View a mapping report to get more information about the objects in a PowerCenter mapping. The Mapping Composite Report includes information about the following components in the mapplet:

- **Source and target fields.** Fields used in mapping sources.
- **Port connections.** Port-level connections between objects.
- **Transformation ports.** Transformation ports for each transformation in the mapping.
- **Object-level connections.** Connections between all objects in the mapping.

To view a Mapping Composite Report:

1. In the Designer, open a mapplet.
2. Right-click in the workspace and choose View Mapping Report.

   The Designer launches JasperReports Server in the default browser for the client machine and runs the Mapping Composite Report.

CHAPTER 2

Working with Sources

This chapter includes the following topics:

- Working with Sources Overview , 47
- Working with Relational Sources, 50
- Creating a Source Definition from a Target Definition, 55
- Working with COBOL Sources, 56
- Components in a COBOL Source File, 58
- Configuring COBOL Source Definitions, 59
- Importing a Microsoft Excel Source Definition, 62
- Manually Creating a Source Definition, 63
- Troubleshooting Sources, 64

Working with Sources Overview

To extract data from a source, first define sources in the repository. You can import or create the following types of source definitions in the Source Analyzer:

- Relational tables, views, and synonyms
- Fixed-width and delimited flat files that do not contain binary data.
- COBOL files
- XML files
- Web Services Description Language (WSDL)
- Data models using certain data modeling tools through Metadata Exchange for Data Models (an add-on product)

You can import sources that use multibyte character sets. Source code pages must be a superset of the target code pages.

Source definitions can be single- or multi-group. A single-group source has a single group in the source definition. Relational sources use a single-group source definition. A multi-group source has multiple groups in the source definition. Non-relational sources such as XML sources use multi-group source definitions.

Note: Because source definitions must match the source, you should import definitions instead of creating them manually.

Oracle Sources

You can import Oracle sources that use basic compression and OLTP compression. You can also manually create source definitions for Oracle sources that use basic compression and OLTP compression.

Special Character Handling in Source Definitions

You can import, create, or edit source definitions with table and column names containing special characters, such as the slash (/) character through the Designer. When you use the Source Analyzer to import a source definition, the Designer retains special characters in table and field names.

However, when you add a source definition with special characters to a mapping, the Designer either retains or replaces the special character. Also, when you generate the default SQL statement in a Source Qualifier transformation for a relational source, the Designer uses quotation marks around some special characters. The Designer handles special characters differently for relational and non-relational sources.

The following table describes how the Designer handles special characters in relational sources:

Special Character	Source Analyzer Behavior	Mapping Designer Behavior
`@#$_`	- Retains the character in the source definition table name. - Retains the character in the source definition column names.	- Retains the character in the source instance table name. - Retains the character in the source instance column names. - Retains the character in the Source Qualifier transformation name. - Retains the character in the Source Qualifier transformation port names. - Does not use quotation marks around the table or column names in the SQL query.
`/+-=~`!%^&*()[]{}';?,<>| <space>`	- Retains the character in the source definition table name. - Retains the character in the source definition column names.	- Replaces the character in the source instance table name with the underscore character. - Retains the character in the source instance column names. - Replaces the character in the Source Qualifier transformation name with the underscore character. - Replaces the character in the Source Qualifier transformation port names with the underscore character. - Delimits table and column names including special characters with quotation marks in the SQL query.
`.":\t\r\n`	- Designer does not recognize these characters in relational source table and column names.	- Designer does not recognize these characters in relational source table and column names.
`.`	- Designer recognizes the period in ODBC sources.	- Designer recognizes the period in ODBC source instances.

Note: Although the Designer replaces slash characters with underscore characters for source table names, it retains slash characters in source definition port names.

The following table describes how the Designer handles special characters in non-relational sources:

Special Character	Source Analyzer Behavior	Mapping Designer Behavior
`@#$_`	- Retains the character in the source definition table name. - Retains the character in the source definition column names. **Note:** You cannot use the @ character as the first character in a table or column name.	- Retains the character in the source instance table name. - Retains the character in the source instance column names. - Retains the character in the Source Qualifier transformation name. - Retains the character in the Source Qualifier transformation port names. **Note:** You cannot use the @ character as the first character in a table or column name.
`/`	- Retains the character in the source definition table name. - Retains the character in the source definition column names.	- Replaces the character in the source instance table name with the underscore character. - Retains the character in the source instance column names. - Replaces the character in the Source Qualifier transformation name with the underscore character. - Replaces the character in the Source Qualifier transformation port names with the underscore character.
`.+-=~`!%^&*()[]{}'"` `;:?,<>\\\t\r\n` `<space>`	- Designer does not recognize these characters in non-relational source table and column names.	- Designer does not recognize these characters in non-relational source table and column names.

Some databases require special configuration or commands to allow table and field names containing the slash character. For more information, see the database documentation.

Updating Source Definitions

When you update a source definition, the Designer propagates the changes to all mappings using that source. Some changes to source definitions can invalidate mappings.

The following table describes how you can impact mappings when you edit source definitions:

Modification	Result
Add a column.	Mappings are not invalidated.
Change a column datatype.	Mappings may be invalidated. If the column is connected to an input port that uses a datatype incompatible with the new one, the mapping is invalidated.
Change a column name.	Mapping may be invalidated. If you change the column name for a column you just added, the mapping remains valid. If you change the column name for an existing column, the mapping is invalidated.
Delete a column.	Mappings can be invalidated if the mapping uses values from the deleted column.

When you add a new column to a source in the Source Analyzer, all mappings using the source definition remain valid. However, when you add a new column and change some of its properties, the Designer invalidates mappings using the source definition.

You can change the following properties for a newly added source column without invalidating a mapping:

- Name
- Datatype
- Format
- Usage
- Redefines
- Occurs
- Key type

If the changes invalidate the mapping, you must open and edit the mapping. Then click Repository > Save to save the changes to the repository. If the invalidated mapping is used in a session, you must validate the session.

Creating Sessions

When you create a session, you can specify a source location different from the location you use when you import the source definition. If the source is a file, you can override some of the file properties when you create a session.

Working with Relational Sources

You can add and maintain relational source definitions in the following ways:

- **Import source definitions.** Import source definitions into the Source Analyzer.
- **Update source definitions.** Update source definitions either manually or by reimporting the definition.

To import a relational source definition, you need to configure connectivity between the source database and the PowerCenter Client.

Relational Source Definitions

You can import relational source definitions from database tables, views, and synonyms. When you import a source definition, you import the following source metadata:

- Source name
- Database location
- Column names
- Datatypes
- Key constraints

Note: When you import a source definition from a synonym, you might need to manually define the constraints in the definition.

After importing a relational source definition, you can optionally enter business names for the table and columns. You can also manually define key relationships, which can be logical relationships created in the repository that do not exist in the database.

Connectivity for Relational Sources

To import a source definition, you must be able to connect to the source database from the client machine using a properly configured ODBC data source or gateway. You may also require read permission on the database object.

When you create an ODBC data source, you must also specify the driver that the ODBC driver manager sends database calls to. The following table shows the recommended ODBC drivers to use with each database:

Database	ODBC Driver	Requires Database Client Software
IBM DB2	DataDirect ODBC Wire Protocol driver	No
Informix	DataDirect ODBC Wire Protocol driver	No
Microsoft Access	Microsoft Access driver	Yes
Microsoft Excel	Microsoft Excel driver	Yes
Microsoft SQL Server	DataDirect ODBC Wire Protocol driver	No
Oracle	DataDirect ODBC Wire Protocol driver	No
Sybase ASE	DataDirect ODBC Wire Protocol driver	No
SAP HANA	SAP HANA ODBC driver	Yes
Teradata	Teradata ODBC driver	n/a

When you use a third-party ODBC data source to import a source definition, the Designer might display a message indicating that the third-party driver is not listed in powrmart.ini. The Designer attempts to import source definition metadata using a driver that is shipped with PowerCenter. If the third-party provides a driver to import metadata, configure powrmart.ini.

For example, if Vendor A provided a driver named `vendoraodbc.dll`, you can add an entry under the ODBCDLL heading based on the specified database:

```
Vendor A = pmodbc.dll
Vendor A = extodbc.dll
```

In the example, the Designer directly interacts with the system ODBC drivers to use the ODBC data source. The system ODBC drivers internally interacts with the third-party ODBC driver, `vendoraodbc.dll`.

The following table lists a sample database dependent entry to use with the PMODBC.ini system ODBC driver:

Database	Entry
MySQL	`MYSQL = PMODBC.DLL`
PowerExchange	`PWX = PMODBC.DLL`
dBase 4	`dBASE IV = PMODBC.DLL`
Visual FoxPro	`Visual FoxPro = PMODBC.DLL`

Database	Entry
Sybase IQ	`Adaptive Server IQ = PMODBC.DLL`
Lotus Notes	`Lotus Notes = PMODBC.DLL`

Configuring a Third-Party ODBC Data Source

PowerCenter uses the powrmart.ini file to recognize the third-party ODBC driver, import the ODBC database object with third-party ODBC drivers, and to preview data in Designer. To import a source definition with an ODBC driver that is not included in the powrmart.ini file, configure the file on the PowerCenter Client machine.

1. Open powrmart.ini in the following directory:

   `<Informatica installation directory>\clients\PowerCenterClient\client\bin`

2. Add an entry to the file under the ODBCDLL section that includes the name of the ODBC data source.

   Ensure that the entry directly points to `pmodbc.dll` or `extodbc.dll`.

3. Save and close powrmart.ini.
4. Restart the PowerCenter Client and import the source definition.

Importing Relational Source Definitions

To create a relational source definition, use the Source Analyzer to import source metadata.

To import a relational source definition:

1. In the Source Analyzer, click **Sources** > **Import from Database.**
2. Select the ODBC data source used to connect to the source database.

   If you need to create or modify an ODBC data source, click the **Browse** to open the ODBC Administrator. Create the data source, and click **OK**. Select the new ODBC data source.

3. Enter a database user name and password to connect to the database.

   Note: You must have the appropriate database permissions to view the object.

   You may need to specify the owner name for database objects you want to use as sources.

4. Optionally, use the search field to limit the number of tables that appear.

   Note: To enable case-sensitive search for a table, enclose the search string in double quotes.

5. Click **Connect.**

   If no table names appear, or if the table you want to import does not appear, click **All.**

6. Scroll down through the list of sources to find the source you want to import. Select the relational object or objects you want to import.

   You can hold down the Shift key to select a block of sources within one folder or hold down the Ctrl key to make non-consecutive selections within a folder. You can also select all tables within a folder by selecting the folder and clicking Select **All**. Use the **Select None** button to clear all highlighted selections.

7. Click **OK.**

   The source definition appears in the **Source Analyzer**. In the **Navigator**, the new source definition appears in the **Sources node** of the active repository folder, under the source database name.

Updating a Relational Source Definition

You can update a source definition to add business names or to reflect new column names, datatypes, or other changes. You can update a source definition in the following ways:

- **Edit the definition.** Manually edit the source definition if you need to configure properties that you cannot import or if you want to make minor changes to the source definition.
- **Reimport the definition.** If the source changes are significant, you may need to reimport the source definition. This overwrites or renames the existing source definition. You can retain existing primary key-foreign key relationships and descriptions in the source definition being replaced.

When you update a source definition, the Designer propagates the changes to all mappings using that source. Some changes to source definitions can invalidate mappings.

If the changes invalidate the mapping, you must open and edit the mapping. Then click Repository > Save to save the changes to the repository. If the invalidated mapping is used in a session, you must validate the session.

Editing Relational Source Definitions

You might want to manually edit a source definition to record properties that you cannot import from the source. You can edit a relational source definition to create key columns and key relationships. These relationships can be logical relationships. They do not have to exist in the database.

You can add descriptions or specify links to business documentation for source definitions at any time. Adding descriptions or business documentation links to source definitions is an easy way to document the purpose of a source definition. You can add or modify descriptions to any existing source definition.

To edit a relational source definition:

1. In the Source Analyzer, double-click the title bar of the source definition.
2. Edit the following settings:

Table Settings	Description
Select Table	Displays the source definition you are editing. To choose a different open source definition to edit, select it from the list.
Rename button	Opens a dialog box to edit the name of the source definition and enter a business name.
Owner Name	Table owner in the database.
Description	Optional description of source table. Character limit is 2,000 bytes/K, where K is the maximum number of bytes for each character in the repository code page. Enter links to business documentation.
Database Type	Indicates the source or database type. If necessary, select a new database type.

3. Click the Columns Tab.

4. Edit the following settings:

Column Settings	Description
Column Name	The column names in the source. When editing a relational source definition, edit the column name only if the actual source column name changed.
Datatype	The datatypes that display in the source definition depend on the source type of the source definition.
Precision and Scale	*Precision* is the maximum number of significant digits for numeric datatypes, or the maximum number of characters for string datatypes. Precision includes scale. *Scale* is the maximum number of digits after the decimal point for numeric values. Therefore, the value 11.47 has a precision of 4 and a scale of 2. The string *Informatica* has a precision (or length) of 11. All datatypes for relational sources have a maximum precision. For example, the Integer datatype has a maximum precision of 10 digits. Some numeric datatypes have a similar limit on the scale or do not allow you to set the scale beyond 0. Integers, for example, have a scale of 0, since by definition they never include decimal values. You can change the precision and scale for some datatypes to values that differ from the values defined in the database. However, changing the precision or scale can cause numeric overflow on numeric columns, truncation on character columns, or insertion of zeroes on datetime columns when the Integration Service reads from the source column.
Not Null	Choose whether you want to allow null data in the source.
Key Type	Select Primary, Foreign, Primary-Foreign, or Not a Key. Applies to relational sources only.
Business Name	Optionally, you can add business names to each source column.

5. Click OK.

Reimporting a Source Definition

Complete the following steps to reimport a source definition. You can retain the following information in the source definition being replaced:

- Primary key-foreign key relationships
- Source definition description
- Column or port description

To reimport a relational source definition:

1. In the Designer, connect to the repository containing the source definition you want to update.
2. Open the Source Analyzer and import the source definition again.
3. The Designer prompts you to rename or replace the existing source tables.
4. To view the differences between the table you are importing and the existing table, click Compare.

   A dialog box displays the attributes in each source side-by-side.
5. Specify whether you want to retain primary key-foreign key information or source descriptions in the existing source definition.

The following table describes the options in the Table Exists dialog box:

Option	Description
Apply to all Tables	Apply rename, replace, or skip all tables in the folder.
Retain User-Defined Pk-Fk Relationships	Keep the primary key-foreign key relationships in the source definition being replaced.
Retain User-Defined Descriptions	Retain the source description and column and port descriptions of the source definition being replaced.

6. Click Replace, Rename, or Skip:

Option	Description
Replace	Replace the source definition with a new one.
Rename	Enter a unique name for the new source definition.
Skip	Do not import the new source definition.

7. If you click Rename, enter the name of the source definition and click OK.

Creating a Source Definition from a Target Definition

You can create a source definition from a target definition or from a shortcut to a target definition.

Drag relational and flat file target definitions into the Source Analyzer to create source definitions. The Designer uses the target definition code page for the matching source definition.

You can create source definitions from the following types of target definitions:

- Flat file
- IBM DB2
- Informix
- Microsoft SQL Server
- MQ Series
- Netezza
- ODBC
- Oracle
- Sybase
- Teradata
- Vertica

You might want to create a source definition from a target definition if you are staging data, and you need to use the definition as a target in one mapping and a source in another mapping.

Working with COBOL Sources

To provide support for mainframe source data, you can import a COBOL file as a source definition in the Designer. COBOL files are fixed-width files that may contain text and binary data. PowerCenter supports the following code pages for COBOL files:

- 7-bit ASCII
- EBCDIC-US
- 8-bit ASCII
- 8-bit EBCDIC
- ASCII-based MBCS
- EBCDIC-based MBCS

You can import shift-sensitive COBOL files that do not contain shift keys. Define the shift states for each column in the COBOL source definition.

COBOL sources often denormalize data and compact the equivalent of separate table records into a single record. You use the Normalizer transformation to normalize these records in the mapping.

After you import a COBOL source definition, review and configure the COBOL file to create record groups. COBOL files often represent the functional equivalent of multiple source tables within the same set of records. When you review the structure of the COBOL file, you can adjust the description to identify which groups of fields constitute a single pseudo-table.

Importing COBOL Sources

The Designer uses the data structures stored in the Data Division of a COBOL program to create a source definition. When you import a COBOL file, the Designer looks for a specific COBOL file format, which is different than the standard ANSI format.

The Designer looks for a COBOL file format similar to the following example:

```
identification division.
program-id. mead.
environment division.
    select file-one assign to "fname".
data division.
file section.
fd  FILE-ONE.
 01  SCHOOL-REC.
    02  SCHOOL-ID                PIC 9(15).
    02  SCHOOL-NM                PIC X(25).
    02  CLASS-REC                OCCURS 2 TIMES.
        03  CLASS-ID             PIC 9(5).
        03  CLASS-NM             PIC X(25).
        03  STUDENT-REC       OCCURS 5 TIMES.
            04  STUDENT-ID       PIC 9(15).
            04  STUDENT-NM       PIC X(25).
            04  PARENT-REC    OCCURS 2 TIMES.
                05 PARENT-ID     PIC 9(15).
                05 PARENT-NM     PIC X(25).
         03  TEACHER-REC       OCCURS 3 TIMES.
            04  TEACHER-ID       PIC 9(15).
            04  TEACHER-NM       PIC X(25).
      02  SPORT-REC             OCCURS 2 TIMES.
         03  SPORT-TEAM          PIC X(30).
working-storage section.
procedure division.
    stop run.
```

Working with COBOL Copybooks

The Designer cannot recognize a COBOL copybook (.cpy file) as a COBOL file (.cbl file) because it lacks the proper format. To import a COBOL copybook in the Designer, you can insert it into a COBOL file template by using the COBOL statement “copy.” After you insert the copybook file into the COBOL file template, you can save the file as a .cbl file and import it in the Designer.

If the .cbl file and the .cpy file are not in the same local directory, the Designer prompts for the location of the .cpy file.

When the COBOL copybook file contains tabs, the Designer expands tabs into spaces. By default, the Designer expands a tab character into eight spaces. You can change this default setting in powrmart.ini. You can find powrmart.ini in the root directory of the PowerCenter Client installation.

To change the default setting, add the following text to powrmart.ini:

```
[AnalyzerOptions]
TabSize=n
```

where *n* is the number of spaces the Designer reads for every tab character. To apply changes, restart the Designer.

For example, the COBOL copybook file is called sample.cpy. The COBOL file below shows how to use the copy statement to insert the sample copybook into a COBOL file template:

```
identification division.
program-id. mead.
environment division.
    select file-one assign to "fname".
data division.
file section.
fd  FILE-ONE.
    copy “sample.cpy”.
working-storage section.
procedure division.
    stop run.
```

Steps to Import a COBOL Source Structure

To import a COBOL source structure, complete the following steps:

1. Open the Source Analyzer, and click Sources > Import from COBOL file.
2. Select the COBOL file you want to analyze.
3. Select the code page of the COBOL file.

   This is the code page of the COBOL file (.cbl), not the data file. The code page must be compatible with the PowerCenter Client code page.

   When you select this code page, the data file uses this code page by default. After you import the COBOL file, you can configure the code page of the source data when you adjust the source definition or when you run the workflow.

   The Designer lists the five code pages you have most recently selected. Then it lists all remaining code pages in alphabetical order.
4. Click OK.

The COBOL source definition appears in the Designer. More than one definition may appear if the COBOL file has more than one FD entry.

Components in a COBOL Source File

When you import a COBOL source, the Designer scans the file for the following components:

- FD Section
- Fields
- OCCURS
- REDEFINES

FD Section

The Designer assumes that each FD entry defines the equivalent of a source table in a relational source and creates a different COBOL source definition for each such entry. For example, if the COBOL file has two FD entries, CUSTOMERS and ORDERS, the Designer creates one COBOL source definition containing the fields attributed to CUSTOMERS, and another with the fields that belong to ORDERS.

Fields

The Designer identifies each field definition, reads the datatype, and assigns it to the appropriate source definition.

OCCURS

COBOL files often contain multiple instances of the same type of data within the same record. For example, a COBOL file may include data about four different financial quarters, each stored in the same record. When the Designer analyzes the file, it creates a different column for each OCCURS statement in the COBOL file. These OCCURS statements define repeated information in the same record. Use the Normalizer transformation to normalize this information.

For each OCCURS statement, the Designer creates the following items:

- One target table when you drag the COBOL source definition into the Target Designer.
- A primary-foreign key relationship
- A generated column ID (GCID)

REDEFINES

COBOL uses REDEFINES statements to build the description of one record based on the definition of another record. When you import the COBOL source, the Designer creates a single source that includes REDEFINES.

The REDEFINES statement lets you specify multiple PICTURE clauses for the sample physical data location. Therefore, you need to use Filter transformations to separate the data into the tables created by REDEFINES.

For each REDEFINES:

- The Designer creates one target table when you drag the COBOL source definition into the Target Designer.
- The Designer creates one primary-foreign key relationship.
- The Designer creates a generated key (GK).
- You need a separate Filter transformation in the mapping.

Configuring COBOL Source Definitions

After you import a COBOL source definition, you may need to configure some of the source properties. The COBOL source definition is similar to a fixed-width flat file definition. However, a COBOL file has some unique properties to consider when you configure the definition:

- OCCURS
- Field definition
- Word or byte storage
- Field attributes

Review the following tabs and dialog boxes when you configure a COBOL source definition:

- **Table tab.** Review storage.
- **Advanced properties.** Review properties for fixed-width data files.
- **Columns tab.** Review OCCURS, FD section, and field attributes.

Configuring the Table Tab

Configure the Table tab of a COBOL source definition similar to the way you configure a flat file definition. However, with a COBOL definition, you also need to consider storage type.

The Integration Service supports COMP-1 word storage in network byte order and with the floating point in the IEEE 754 4 byte format. Also, the Integration Service supports COMP-2 word storage in network byte order and with the floating point in the IEEE 754 8 byte format.

You may need to switch to byte storage for IBM VS COBOL and MicroFocus COBOL-related data files by selecting the IBM COMP option in the source definition. The Integration Service provides COMP word storage by default. COMP columns are 2, 4, and 8 bytes on IBM mainframes. COMP columns can be 1, 2, 3, 4, 5, 6, 7, and 8 bytes elsewhere, when derived through MicroFocus COBOL. Clear the IBM COMP option to use byte storage.

The following table describes the COBOL file properties that you can set on the Table tab:

Table Option	Description
Rename button	Use the Rename button to rename the source definition and enter a business name for the source definition.
Owner Name	Not applicable for COBOL files.
Description	Additional comments about the source definition.
Database Type	Source location or type. This must be set to VSAM.
IBM COMP	Indicates the storage type. If selected, the Integration Service uses word storage. Otherwise, it uses byte storage.
Flat File Type	Select Fixed-width.
Advanced button	Use the Advanced button to open a dialog box with fixed-width options.

Configuring Advanced Properties

Click Advanced on the Table tab to configure properties for fixed-width files.

When you import a COBOL file, you choose the code page of the COBOL file so the Designer can read the file correctly. After you import the COBOL file, you can change the code page to that of the source data so the Integration Service can read the data when you run a workflow. You can select the code page of the data file in Advanced Properties.

Related Topics:

- "Updating Fixed-Width File Properties" on page 73

Configuring the Columns Tab

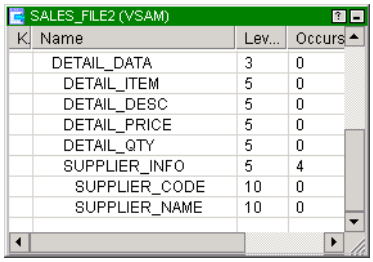
When you review the Columns tab of a COBOL source definition, you see several levels of columns in addition to multiple attributes for those columns. You might want to review and configure the following properties:

OCCURS

When you review the contents of the Columns tab in a COBOL source, you see several levels of columns. These levels represent the separate record sets contained within a single COBOL source.

For example, the following COBOL source contains a nested record set, HST_MTH. Each record set begins with a level 5 heading, indicating the beginning of the record. The columns within each record set must all be at the same level beneath the record heading. For example, the record set HST_MTH contains several columns, starting with HST_ACCR_REM. An OCCURS setting of 24 indicates that, when you review the data in this COBOL source, each record contains 24 nested records for HST_MTH.

The following figure shows a sample COBOL source definition with an OCCURS setting of 4:



SALES_FILE2 (VSAM)

K	Name	Lev...	Occurs
	DETAIL_DATA	3	0
	DETAIL_ITEM	5	0
	DETAIL_DESC	5	0
	DETAIL_PRICE	5	0
	DETAIL_QTY	5	0
	SUPPLIER_INFO	5	4
	SUPPLIER_CODE	10	0
	SUPPLIER_NAME	10	0

All of the columns in HST_MTH are at the same level, 7, in this COBOL source. The heading for the record HST_MTH is at level 5, two levels above the columns in that source.

FD Section

You may need to configure the source definition to group fields. Although the Designer creates a separate source definition for each FD entry in the COBOL file, each entry may represent the functional equivalent of multiple tables of data. When you configure the source definition, you create different levels of fields within the source to group data into separate pseudo-tables.

Field Attributes

When you review a COBOL source, you see several attributes for each field, the COBOL equivalent of a column, that represent how you can configure a field in a COBOL file.

Among these attributes, the picture clause is the most fundamental, since it shows how the COBOL file represents data. COBOL uses its own set of conventions for configuring how data is formatted within the column. For example, the picture X(32) indicates that text data in the field is 32 bytes long. The picture clause 9(7) indicates that the field contains numeric data of no more than 7 digits in length. The picture N(8), an Nstring datatype containing double-byte characters, indicates that the text data in the field is 16 bytes long.

You may need to adjust the definition of a field in the Source Analyzer, modifying the picture in the process. Since the Integration Service uses the source definition as a map for finding data in the source file, you need to be cautious when you make such adjustments.

The following table describes the attributes you can set in the Columns tab of a COBOL source definition:

Attribute	Description
Physical offsets (POffs)	Offset of the field in the file. The Designer calculates this read-only setting using the physical length, picture, usage, and REDEFINES settings for the field.
Physical length (PLen)	Number of bytes in this field.
Column name	Name of the field.
Level	An indicator used to identify all fields that provide data for the same record. If you want to group fields, you set all its columns to the same level. Using this feature, it is possible to create multiple record types, which are the equivalent of separate tables of data from the same COBOL source.
Occurs	A COBOL statement indicating that multiple instances of this field appear in the same record.
Datatype	Field datatype: String, Nstring, or Numeric.
Precision (Prec)	Precision of numeric values in the field.
Scale	Scale of numeric values in the field.
Picture	How the file represents data.
Usage	Storage format for data in the field. Different COBOL conventions exist, such as COMP-1 and COMP-X. All available conventions appear in the list of usages for each field.
Key Type	Type of key constraint to apply to this field. When you configure a field as a primary key, the Integration Service generates unique numeric IDs for this field when running a workflow using the COBOL file as a source.
Signed (S)	Indicates whether numeric values in the field are signed.
Trailing sign (T)	If selected, indicates that the sign (+ or -) exists in the last digit of the field. If not selected, the sign appears as the first character in the field.
Included sign (I)	Indicates whether the sign is included in any value appearing in the field.
Real decimal point (R)	For numeric values, specifies whether the decimal point is a period (.) or a V character.
Redefines	Indicates that the field uses a REDEFINES statement in COBOL to base its own field definition on that of another field.

Attribute	Description
Shift key	You can define the shift state for shift-sensitive COBOL files that do not contain shift keys. This attribute appears when you select User Defined Shift State in the Edit Flat File Information dialog box for fixed-width files. Choose Shift-In if the column contains single-byte characters. Choose Shift-Out if the column contains multibyte characters.
Business Name	Additional comments about the field.

Related Topics:

- "Working with Shift-Sensitive Flat Files" on page 85

Importing a Microsoft Excel Source Definition

PowerCenter treats a Microsoft Excel source as a relational database, not a flat file. Like relational sources, the Designer uses ODBC to import a Microsoft Excel source. You do not need database permissions to import Microsoft Excel sources.

Complete the following tasks before you import an Excel source definition:

1. Install the Microsoft Excel ODBC driver on the system.
2. Create a Microsoft Excel ODBC data source for each source file in the ODBC Data Source Administrator.
3. Prepare Microsoft Excel spreadsheets by defining ranges and formatting columns of numeric data.

Defining Ranges

The Designer creates source definitions based on ranges you define in Microsoft Excel. You can define one or more ranges in a Microsoft Excel sheet. If you have multiple sheets, define at least one range for each sheet. When you import sources in the Designer, each range displays as a relational source.

You must define a range in the Designer to import the Excel source definition.

To define a range:

1. Open the Microsoft Excel file.
2. Highlight the column or group of columns of data to import.
3. Click Insert > Name > Define.
4. Enter a name for the selected range and click OK.
5. If you have multiple sheets, select each sheet and repeat steps 2 to 4 to define ranges for each set of data.
6. Click File > Save.

Formatting Columns of Numeric Data

In Microsoft Excel, you can assign datatypes to columns of data. The Microsoft Excel datatypes are ODBC datatypes. PowerCenter supports ODBC datatypes and converts them to transformation datatypes. If you do

not assign datatypes in Microsoft Excel, the Designer imports each column as VARCHAR. If you want to perform numeric or aggregate calculations in a mapping, assign numeric datatypes in Microsoft Excel before importing the spreadsheet.

To format columns in Microsoft Excel:

1. Open the Microsoft Excel file.
2. Select the columns of data that consist of numeric data.
3. Click Format > Cells.
4. In the Number tab, select Number.
5. Specify the number of decimal places.
6. Click OK.
7. Click File > Save.

Steps to Import a Microsoft Excel Source Definition

After you define ranges and format cells, you can import the ranges in the Designer. Ranges display as source definitions when you import the source.

To import a Microsoft Excel source definition:

1. In the Designer, connect to the repository and open the folder for the source definition.
2. Open the Source Analyzer and click Sources > Import from Database.
3. Select Excel Files (Microsoft Excel Driver (*.xls)) for the data source.
4. Click the Browse button to open the ODBC Administrator.
5. In the User or System DSN tabs, depending on where you created the data source, double-click the Microsoft Excel driver.
6. Click Select Workbook and browse for the Microsoft Excel file, which is considered a relational database.
7. Click OK three times to return to the Import Tables dialog box.
8. Click Connect in the Import Tables dialog box.

   You do not need to enter a database user name and password. The ranges you defined in the Microsoft Excel file appear as table names. The database owner is No Owner because you are not required to enter a database user name.
9. Select the table you want to import.

   To select more than one table, hold down the Ctrl or Shift keys to highlight multiple tables.
10. Click OK.

    In the Navigator, the source definition appears in the Sources node, under the database name.

Manually Creating a Source Definition

You can manually create a source definition.

To create a source definition:

1. In the Source Analyzer, click Sources > Create.
2. Enter the name for the source, the database name, and database type.

3. Click Create.

   An empty table structure appears in the workspace. (It may be covered by the dialog box.) The new source table also appears within the Navigator window.

4. Click Done when you are finished creating source definitions.
5. Configure the source definition.

The source definition is saved to the repository. You can now use the source definition in a mapping. You can also create a source table based on this definition in the source database.

Troubleshooting Sources

I imported a source from a DB2 database and received an SQL0954C error message from the DB2 operating system.

If the value of the DB2 system variable APPLHEAPSZ is too small when you use the Designer to import sources from a DB2 database, the Designer reports an error accessing the repository. The Designer status bar displays the following message:

```
SQL Error:[IBM][CLI Driver][DB2]SQL0954C: Not enough storage is available in the
application heap to process the statement.
```

If you receive this error, increase the value of the APPLHEAPSZ variable for the DB2 operating system. APPLHEAPSZ is the application heap size in 4KB pages for each process using the database.

CHAPTER 3

Working with Flat Files

This chapter includes the following topics:

- Working with Flat Files Overview, 65
- Importing Flat Files, 65
- Editing Flat File Definitions, 71
- Formatting Flat File Columns, 79
- Working with File Lists, 85
- Working with Shift-Sensitive Flat Files, 85
- Working with Multibyte Data in Fixed-Width Targets, 87
- Troubleshooting Flat Files, 87

Working with Flat Files Overview

To use flat files as sources, targets, and lookups in a mapping, you must import or create the definitions in the repository. You can import or create flat file source definitions in the Source Analyzer. You can import or create flat file target definitions in the Target Designer. You can import flat file lookups or use existing file definitions in a Lookup transformation.

Note: Because source definitions must match the source, you should import file source definitions instead of creating them manually.

Creating Sessions with Flat File Sources and Targets

When you create sessions using file sources and targets, you can override some properties you define in the Designer. When you create a session with a file source, you can specify a source file location different from the location you use when you import the file source definition.

Importing Flat Files

You can import fixed-width and delimited flat file definitions that do not contain binary data. When importing the definition, the file must be in a directory local to the client machine. In addition, the PowerCenter Integration Service must be able to access all source files during the session.

When you create a file source, target, or lookup definition, you must define the properties of the file. The Flat File Wizard prompts you for the following file properties:

- File name and location
- File code page
- File type
- Column names and datatypes
- Number of header rows in the file
- Column size and null characters for fixed-width files
- Delimiter types, quote character, and escape character for delimited files

Special Character Handling

When you import a flat file in the Designer, the Flat File Wizard uses the file name as the name of the flat file definition by default. You can import a flat file with any valid file name through the Flat File Wizard. However, the Designer does not recognize some special characters in flat file source and target names.

When you import a flat file, the Flat File Wizard changes invalid characters and spaces into underscores (_). For example, you have the source file “sample prices+items.dat”. When you import this flat file in the Designer, the Flat File Wizard names the file definition sample_prices_items by default.

Related Topics:

- “Special Character Handling in Source Definitions” on page 48
- “Special Character Handling in Target Definitions” on page 89

Selecting Code Pages

When you import a flat file in the Designer, you can select the code page of the file. The Designer lists the five code pages you have most recently selected. Then it lists all remaining code pages in alphabetical order.

The code page represents the code page of the *data* contained in the file. When you configure the file definition, specify delimiters, null characters, and escape characters that are supported by the code page.

For delimited files, specify delimiters, optional quotes, and escape characters that are contained in the source file code page. For fixed-width files, specify null characters as either a binary value from 0 to 255 or a character from the selected code page.

When you configure the flat file definition, use delimiters, escape characters, and null characters that are valid in the code page necessary to run a workflow.

- **ASCII data movement mode.** Use characters from the code page you define for the flat file. Any 8-bit characters you specified in previous versions of PowerCenter are still valid.
- **Unicode data movement mode.** Use characters from the code page you define for the flat file.

Changing Display Fonts

You can use the Flat File Wizard to preview the data contained in a flat file. By default, the Flat File Wizard displays fonts based on the system locale. For example, the PowerCenter Client is installed on a Windows machine that uses the default Latin-1 system locale. By default, the font in the Flat File Wizard is set to Courier. The preview window displays Latin-1 characters in the Courier font.

To display a font that is not compatible with the Latin-1 locale, you must change the font for the Flat File Wizard. For example, if you want to see Japanese fonts in the preview window of the Flat File Wizard on the Latin-1 machine, you must change the font to a Japanese font, such as MS Gothic.

You can configure the font for the Flat File Wizard in the Designer.

If you configure the font and still encounter problems displaying text in the preview window, complete the following tasks:

1. Reset the formats for the Designer.
2. Ensure that the language of the data is installed on the machine. If necessary, install the language.

Resetting Designer Formats

You can reset Designer formats to default settings.

To change the Designer formats back to the default settings:

1. In the Designer, click Tools > Options > Format.
2. Click Reset All to reset to default.

Installing a Language

To install a language on a Windows machine:

- Click Control Panel > Regional Options.

Importing Fixed-Width Flat Files

Fixed-width flat files are byte-oriented, which means that the field lengths are measured in bytes. They can also be line sequential, which means each row ends with a newline character. You can import a fixed-width file that does not contain binary data or multibyte character data greater than two bytes per character.

When you import a fixed-width file, you can create, move, or delete column breaks using the Flat File Wizard. Incorrect positioning of column breaks can create misalignment errors when you run a session with a file source containing single-byte and multibyte characters. Misalignment of multibyte data in a file causes errors in a workflow.

To import a fixed-width flat file definition:

1. To import a source definition, open the Source Analyzer and click **Sources** > **Import from File**. To import a target definition, open the Target Designer and click **Targets** > **Import from File.**

   The **Open Flat File** dialog box appears.
2. Select the file you want to use.
3. Select a code page.

   When you import a flat file source definition, select a code page that matches the code page of the data in the file.
4. Click **OK.**

   The contents of the file appear in the window at the bottom of the Flat File Wizard.

5. Edit the following settings:

Fixed-Width Flat File Wizard, Step 1 of 3	Description
Flat File Type	File type. Select Fixed Width for a fixed-width file.
Enter a Name for This Source	Name of the source. This is the name of the source in the repository. You can use the file name or any other logical name.
Start Import At Row	Indicates the row number at which the Flat File Wizard starts reading when it imports the file. For example, if you specify to start at row 2, the Flat File Wizard skips 1 row before reading.
Import Field Names From First Line	If selected, the Designer uses data in the first row for column names. Select this option if column names appear in the first row. Invalid field names are preceded by "FIELD_."

6. Click **Next.**

   Follow the directions in the wizard to manipulate the column breaks in the file preview window. Move existing column breaks by dragging them. Double-click a column break to delete it.

   For shift-sensitive files, the Flat File Wizard displays single-byte shift characters as '.' in the window. Double-byte shift characters appear as '..' in the window so you can configure the column breaks accurately.

7. Click **Next.**

   Enter column information for each column in the file.

   To switch between columns, select a new column in the **Source Definition** or **Target Definition group**, or click the column heading in the file preview window.

Fixed-Width Flat File Wizard, Step 3 of 3	Description
Name	Port name that you want to appear for each column. If you select Import Field Names from First Line, the wizard reads the column names provided in the file.
Datatype	Column datatype. Select Text, Numeric, or Datetime, and then enter the appropriate Length/Precision, Scale, and Width. For numeric columns, Precision is the number of significant digits, and Width is the number of bytes to read from source files or to write to target files. For text columns, Precision is measured in bytes for fixed-width files and in characters for delimited files. By default, the Flat File Wizard enters the same value for both Precision and Width. You can change the value of the precision to enter the number of significant digits, but the width must be greater than or equal to the precision. **Note:** Only characters 0 to 9 are considered numeric. Columns that contain multibyte character set numbers, such as Japanese, are considered text.

8. Click **Finish.**

   Note: If the file size exceeds 256 KB or contains more than 16 KB of data for each row, verify that the Flat File Wizard imports the file with the correct field precision and width. If not, adjust the field precision and width in the Flat File Wizard or in the imported definition.

Importing Delimited Flat Files

Delimited flat files are always character-oriented and line sequential. The column precision is always measured in characters for string columns and in significant digits for numeric columns. Each row ends with a newline character. You can import a delimited file that does not contain binary data or multibyte character data greater than two bytes per character.

Complete the following steps to import a delimited file for either a source or target definition:

1. To import a source definition, open the Source Analyzer, and click **Sources** > **Import from File**. To import a target definition, open the Target Designer and click **Targets** > **Import from File**.

   The **Open Flat File** dialog box appears.
2. Select the file you want to use.
3. Select a code page.

   When you import a flat file source definition, select a code page that matches the code page of the data in the file.
4. Click **OK**.

   The contents of the file appear in the preview window at the bottom of the Flat File Wizard.
5. Edit the following settings:

Delimited Flat File Wizard, Step 1 of 3	Description
Flat File Type	File type. Select Delimited for a delimited file.
Enter a Name for This Source	Name of the source. This is the name of the source definition in the repository. You can use the file name or any other logical name.
Start Import At Row	Indicates the row number at which the Flat File Wizard starts reading when Flat File Wizard imports the file. For example, if you specify to start at row 2, the Flat File Wizard skips 1 row before reading.
Import Field Names From First Line	If selected, the Designer uses data in the first row for column names. Select this option if column names appear in the first row. Invalid field names are preceded by “FIELD_”.

6. Click **Next**.

   Any incorrectly parsed fields display in red text in the file preview window at the bottom of this screen.

7. Enter the following settings:

Delimited Flat File Wizard, Step 2 of 3	Description
Delimiters	Character used to separate columns of data. Use the Other field to enter a different delimiter. Delimiters must be printable characters and must be different from the escape character and the quote character if selected. You cannot select unprintable multibyte characters as delimiters.
Treat Consecutive Delimiters as One	If selected, the Flat File Wizard reads one or more consecutive column delimiters as one. Otherwise, the Flat File Wizard reads two consecutive delimiters as a null value.
Treat Multiple Delimiters as AND	If selected, the Flat File Wizard reads a specified set of delimiters as one.
Escape Character	Character immediately preceding a column delimiter character embedded in an unquoted string, or immediately preceding the quote character in a quoted string. When you specify an escape character, the Integration Service reads the delimiter character as a regular character. This is called *escaping* the delimiter or quote character.
Remove Escape Character From Data	This option is selected by default. Clear this option to include the escape character in the output string.
Use Default Text Length	If selected, the Flat File Wizard uses the entered default text length for all string datatypes.
Text Qualifier	Quote character that defines the boundaries of text strings. Choose No Quote, Single Quote, or Double Quotes. If you select a quote character, the Flat File Wizard ignores delimiters within pairs of quotes.

8. Click **Next**.
9. Enter column information for each column in the file.

To switch between columns, select a new column in the Source Definition pane or Target Definition pane, or click the column heading in the file preview.

Delimited Flat File Wizard, Step 3 of 3	Description
Name	Port name that you want to appear for each column. If you select Import Field Names from First Line, the wizard reads the column names provided in the file instead.
Datatype	Column datatype. Select Text, Numeric, or Datetime, and then enter the appropriate Length/Precision, Scale, and Width. For numeric columns, Precision is the number of significant digits. The Flat File Wizard ignores the width for numeric columns in delimited files. For Text columns, Precision is the maximum number of characters contained in the source field or the target field. The Flat File Wizard ignores the precision when reading text columns in or writing text columns to delimited files. By default, the Flat File Wizard enters the same value for both Precision and Width. You can change the value of the precision or width, but the Flat File Wizard only lets you define the precision to be greater than or equal to the width. **Note:** Only characters 0 to 9 are considered numeric. Columns that contain multibyte character set numbers, such as Japanese, are considered text.

10. Click **Finish**.

    Note: If the file size exceeds 256 KB or contains more than 16 KB of data for each line, verify that the Flat File Wizard imports the file with the correct field precision. If not, adjust the field precision in the Flat File Wizard or in the imported definition.

Related Topics:

- "Rules and Guidelines for Delimited File Settings" on page 78
- "Formatting Flat File Columns" on page 79

Editing Flat File Definitions

After you import a flat file source or target definition, you may need to add business names and configure file properties. Additionally, if the file definition changes, you might want to manually edit the definition.

You can edit source or target flat file definitions using the following definition tabs:

- **Table tab.** Edit properties such as table name, business name, and flat file properties.
- **Columns tab.** Edit column information such as column names, datatypes, precision, and formats.
- **Properties tab.** View the default numeric and datetime format properties in the Source Analyzer and the Target Designer. You can edit these properties for each source and target instance in a mapping in the Mapping Designer.
- **Metadata Extensions tab.** Extend the metadata stored in the repository by associating information with repository objects, such as flat file definitions.

Note: If the file structure for a source definition changes significantly, you may need to reimport the file source definition.

When you update a source or target definition, the Designer propagates the changes to any mapping using that source or target. Some changes to source and target definitions can invalidate mappings. If the changes invalidate the mapping, you must validate the mapping. You can validate mappings from the Query Results or View Dependencies window or from the Repository Navigator. You can also validate multiple objects without opening them in the workspace. If you cannot validate the mapping from these locations, you must open and edit the mapping.

When you create sessions using file source, target, or lookups, you can override some properties you define in the Designer. For example, when you create a session with a file source, you can specify a source file location different from the location you use when you import the file source definition.

Editing Table Options

You can edit the following options on the Table tab of a flat file source or target definition:

- **Business name.** Add a more descriptive name to the source or target definition.
- **Description.** Add a comment or link to business documentation. These display in Repository Manager for the source or target definition. Adding comments or business documentation links to a source or target is an easy way to document its purpose. You can add or modify comments to any existing source or target.

  You can enter up to (2,000 bytes)/K characters in the description, where K is the maximum number of bytes a character contains in the selected repository code page. For example, if the repository code page is a Japanese code page where K=2, each description and comment field can contain up to 1,000 characters.
- **Keywords.** Track flat file targets with key words. As development and maintenance work continues, the number of targets increases. While all of these targets may appear in the same folder, they may all serve different purposes. Keywords can help you find related targets. Keywords can include developer names, mappings, or the associated schema.

  Use keywords to perform searches in the Repository Manager.
- **Database type.** Define the source or target type. Choose Flat File for flat file sources and targets.
- **Flat file information.** When the database type is flat file, define the flat file properties by clicking the Advanced button.

To add options to a flat file source or target definition:

1. To add options to a source definition, in the Source Analyzer, double-click the title bar of the source definition. To add options to a target definition, in the Target Designer, double-click the title bar of the target definition.

   The Edit Tables dialog box appears.
2. Click the Rename button to edit the source or target name and the business name.
3. Choose Flat File in the Database Type field.
4. Click the Advanced button to edit the flat file properties.

   A different dialog box appears for fixed-width and delimited files.
5. To add a description, type a description in the Description field.
6. To add keywords for target definitions, click Edit Keywords.

   The Edit Keywords dialog box appears. Use the buttons to create and move keywords.

Related Topics:

- "Importing Fixed-Width Flat Files" on page 67
- "Importing Delimited Flat Files" on page 69

Editing Columns

You can edit the following information in the Columns tab of a flat file source or target definition:

- **Column Name.** The names of columns in the flat file source or target.
- **File name columns.** You can add a file name column to a flat file source or target. For sources, use the CurrentlyProcessedFileName column to return the names of the source files from which rows of data were read. Use this column if you configure the session to read data from a file list. For targets, use the FileName column to dynamically name flat file targets.
- **Datatype.** The datatype of the column. For flat files, you can choose bigint, datetime, double, int, nstring, number, or string.
- **Precision, Scale, and Format.** When importing file definitions, you often need to consider the precision, scale, field width, and format of values in each column. You can edit the field width and format by clicking in the Format column. Enter the precision, scale, and format.
- **Not Null.** Choose whether you want to allow null data in the source or target.
- **Key Type.** Choose NOT A KEY for flat file source and target definitions.
- **Shift Key.** You can define the shift state for shift-sensitive fixed-width flat file sources that do not contain shift keys. This attribute appears when you select User Defined Shift State in the Edit Flat File Information dialog box for fixed-width files.

  Choose Shift-In if the column contains single-byte characters. Choose Shift-Out if the column contains multibyte characters.
- **Business Name.** Optionally, you can add business names to each source or target field.

Note: If the file columns for a source or lookup definition change significantly, you may need to reimport the file.

To edit the columns of flat file source and target definitions:

1. To edit a source definition, in the Source Analyzer, double-click the title bar of the flat file source definition. To edit a flat file source definition, in the Target Designer, double-click the title bar of the flat file target definition.
2. Click the Columns tab.
3. Configure the options of the source or target definition as described above.
4. If you want to add columns, select a column and click Add.
5. Enter the name, datatype, and other characteristics of the column.

   Repeat these steps for each column you want to add to the source or target definition.
6. If you want to move a column, use the Up and Down buttons, or drag it within the scrolling list.

Updating Fixed-Width File Properties

After you import a fixed-width file, you can update the file properties. Double-click the title bar of the source or target definition. Edit the table and column information.

To edit file properties, click the Advanced button on the Table tab. The Edit Flat File Information - Fixed Width Files dialog box appears. The Edit Flat File Information - Fixed Width Files dialog box contains more options for file sources than for file targets. For example, it contains information that the Integration Service needs to read the file, such as the number of initial rows to skip, or the number of bytes between rows.

The following table describes the fixed-width file properties that you can configure for source, target, and lookup definitions:

Fixed-Width Advanced Setting	Description for Sources and Lookups	Description for Targets
Null Character	Character used in the source file to represent a null value. This can be any valid character in the file code page or any binary value from 0 to 255.	Character the Integration Service uses in the target file to represent a null value. This can be any valid character in the file code page.
Repeat Null Character	If selected, the Integration Service reads repeat null characters in a single field as a single null value. When you specify a multibyte null character and select Repeat Null Character, the field may contain extra trailing bytes if the field length is not evenly divisible by the number of bytes in the null character. In this case, the field is not null. You should always specify a single-byte null character.	If selected, the Integration Service writes as many null characters as possible into the target field. If you do not select this option, the Integration Service enters a single null character at the beginning of the field to represent a null value. If you specify a multibyte null character and there are extra bytes left after writing null characters, the Integration Service pads the column with single-byte spaces. If a column is not big enough to take a null character because it is smaller than the multibyte character specified as the null character, the session fails at initialization.
Code Page	Code page of the file definition. For source definitions, use a source code page that is a subset of the target code page. For lookup file definitions, use a code page that is a superset of the source code page and a subset of the target code page.	Code page of the file definition. Use a code page that is a superset of the source code page.
Line Sequential	If selected, the Integration Service reads a line feed or carriage return character in the last column as the end of the column. Use this option if the file uses line feeds or carriage returns to shorten the last column of each row.	n/a
Number of Initial Rows to Skip	Indicates the number of rows the Integration Service skips when it reads the file. Use this setting to skip blank or header rows. One row may contain multiple records. Enter any integer from 0 to 2,147,483,647.	n/a
Number of Bytes to Skip Between Records	Number of bytes between the last column of one row and the first column of the next. The Integration Service skips the entered number of bytes at the end of each row to avoid reading carriage return or line feed characters. Enter 1 for UNIX files and 2 for DOS files.	n/a

Fixed-Width Advanced Setting	Description for Sources and Lookups	Description for Targets
Strip Trailing Blanks	If selected, the Integration Service strips trailing blanks from string values.	n/a
User Defined Shift State	If selected, you can define the shift state for source columns on the Columns tab. Select User Defined Shift State when the source file contains both multibyte and single-byte data, but does not contain shift-in and shift-out keys. If a multibyte file source does not contain shift keys, you must define shift states for each column in the flat file source definition so the Integration Service can read each character correctly.	n/a

Null Character Handling

You can specify single-byte or multibyte null characters for fixed-width file sources. When reading a fixed-width source file, the Integration Service uses these characters to determine if a column is null. When writing to a fixed-width target file, the Integration Service uses these characters to represent null values.

The following table describes how the Integration Service uses the Null Character and Repeat Null Character properties to determine if a column is null:

Null Character	Repeat Null Character	Integration Service Behavior When Reading From Sources and Lookups	Integration Service Behavior When Writing To Targets
Binary	Disabled	A column is null if the first byte in the column is the binary null character. The Integration Service reads the rest of the column as text data to determine the column alignment and track the shift state for shift-sensitive code pages. If data in the column is misaligned, the Integration Service skips the row and writes the skipped row and a corresponding error message to the session log.	Integration Service enters a single binary null character at the beginning of the field to represent a null value. If there are extra bytes left after writing the null character, the Integration Service pads the column with single-byte spaces.
Non-binary	Disabled	A column is null if the first character in the column is the null character. The Integration Service reads the rest of the column to determine the column alignment and track the shift state for shift sensitive code pages. If data in the column is misaligned, the Integration Service skips the row and writes the skipped row and a corresponding error message to the session log.	Integration Service enters a single null character at the beginning of the field to represent a null value. If you specify a multibyte null character and there are extra bytes left after writing the null character, the Integration Service pads the column with single-byte spaces. If a column is not big enough to take a null character because it is smaller than the multibyte character specified as the null character, the session fails at initialization.

Null Character	Repeat Null Character	Integration Service Behavior When Reading From Sources and Lookups	Integration Service Behavior When Writing To Targets
Binary	Enabled	A column is null if it contains only the specified binary null character. The next column inherits the initial shift state of the code page.	Integration Service writes as many binary null characters as possible into the target field.
Non-binary	Enabled	A column is null if the repeating null character fits into the column exactly, with no bytes leftover. For example, a five-byte column is not null if you specify a two-byte repeating null character. In shift-sensitive code pages, shift bytes do not affect the null value of a column. If a column contains a shift byte at the beginning or end of the column and the repeating null character fits into the column with no bytes left over, the column is null. Specify a single-byte null character when you use repeating non-binary null characters. This ensures that repeating null characters fit into a column exactly.	Integration Service writes as many null characters as possible into the target field. If you specify a multibyte null character and there are extra bytes left after writing the null characters, the Integration Service pads the column with single-byte spaces. If a column is not big enough to take a null character because it is smaller than the multibyte character specified as the null character, the session fails at initialization.

Updating Delimited File Properties

After you import a delimited file, you can update the file properties. Double-click the title bar of the source or target definition. Edit the table and column information.

To edit file properties, click the Advanced button on the Table tab. The Edit Flat File Information - Delimited Files dialog box appears. The Edit Flat File Information - Delimited Files dialog box contains more options for file sources than for file targets. For example, it contains information that the Integration Service needs to read the file, such as the number of initial rows to skip or the escape character.

The following table describes the delimited file properties that you can configure:

Delimited File Advanced Settings	Description for Sources and Lookups	Description for Targets
Column Delimiters	Character used to separate columns of data. Delimiters can be either printable or single-byte unprintable characters, and must be different from the escape character and the quote character. You can enter a single-byte unprintable character by browsing the delimiter list in the Delimiters dialog box. You cannot select unprintable multibyte characters as delimiters. You cannot select the NULL character as the column delimiter for a flat file source.	Character used to separate columns of data. Delimiters can be either printable or single-byte unprintable characters and must be different from the quote character. You can enter a single-byte unprintable character by browsing the delimiter list in the Delimiters dialog box. You cannot select unprintable multibyte characters as delimiters. If you enter more than one delimiter, the Integration Service uses the first delimiter you specify.
Treat Consecutive Delimiters as One	If selected, the Integration Service treats one or more consecutive column delimiters as one. Otherwise, the Integration Service reads two consecutive delimiters as a null value.	n/a
Treat Multiple Delimiters as AND	If selected, the Integration Service treats a specified set of delimiters as one. For example, a source file contains the following record: abc~def|ghi~|~|jkl|~mno. By default, the Integration Service reads the record as nine columns separated by eight delimiters: abc, def, ghi, NULL, NULL, NULL, jkl, NULL, mno. If you select this option and specify the delimiter as (~ |), the Integration Service reads the record as three columns separated by two delimiters: abc~def|ghi, NULL, jkl|~mno.	n/a
Optional Quotes	Select No Quotes, Single Quotes, or Double Quotes. Quote character that defines the boundaries of text strings. Double Quotes is chosen by default. If selected, the Integration Service ignores column delimiters within the quote character. For example, a source file uses a comma as a delimiter and the Integration Service reads the following row from the source file: 342-3849, 'Smith, Jenna', 'Rockville, MD', 6. If you select the optional single quote character, the Integration Service ignores the commas within the quotes and reads the row as four fields. If you do not select the optional single quote, the Integration Service reads six separate fields. **Note:** You can improve session performance if the source file does not contain quotes or escape characters.	Select No Quotes, Single Quotes, or Double Quotes. Quote character that defines the boundaries of text strings. Double Quotes is chosen by default. If you select a quote character, the Integration Service does not treat column delimiter characters within the quote characters as a delimiter. For example, a target file uses a comma as a delimiter and the Integration Service writes the following row to the target file: 342-3849, 'Smith, Jenna', 'Rockville, MD', 6. If you select the optional single quote character, the Integration Service ignores the commas within the quotes and writes the row as four fields. If you do not select the optional single quote, the Integration Service writes six separate fields.

Delimited File Advanced Settings	Description for Sources and Lookups	Description for Targets
Code Page	Code page of the file definition. For source definitions, use a source code page that is a subset of the target code page. For lookup file definitions, use a code page that is a superset of the source code page and a subset of the target code page.	Code page of the file definition. Use a code page that is a superset of the source code page.
Row Delimiter	Specify a line break character. Select from the list or enter a character. Preface an octal code with a backslash (\). To use a single character, enter the character. The Integration Service uses only the first character when the entry is not preceded by a backslash. The character must be a single-byte character, and no other character in the code page can contain that byte. Default is line-feed, \012 LF (\n).	n/a
Escape Character	Character used to *escape* a delimiter character in an unquoted string if the delimiter is the next character after the escape character. If selected, the Integration Service reads the delimiter character as a regular character embedded in the string, called *escaping* the delimiter character. **Note:** You can improve session performance if the source file does not contain quotes or escape characters.	n/a
Remove Escape Character From Data	This option is selected by default. Clear this option to include the escape character in the output string.	n/a
Number of Initial Rows to Skip	Indicates the number of rows the Integration Service skips when it reads the file. Use this setting to skip blank or header rows.	n/a

Rules and Guidelines for Delimited File Settings

Delimited files are character-oriented and line sequential. Use the following rules and guidelines when you configure delimited files:

- The column and row delimiter character, quote character, and escape character must all be different for a source definition. These properties must also be contained in the source or target file code page.
- The escape character and delimiters must be valid in the code page of the source or target file.

Use the following rules and guidelines when you configure delimited file sources:

- In a quoted string, use the escape character to escape the quote character. If the escape character does not immediately precede a quote character, the Integration Service reads the escape character as an ordinary character.

- Use an escape character to escape the column *delimiter*. However, in a quoted string, you do not need to use an escape character to escape the *delimiter* since the quotes serve this purpose. If the escape character does not immediately precede a delimiter character, the Integration Service reads the escape character as an ordinary character.
- When two consecutive quote characters appear within a quoted string, the Integration Service reads them as one quote character. For example, the Integration Service reads the following quoted string as `I'm going tomorrow:`

  ```
  2353,'I''m going tomorrow'MD
  ```
- The Integration Service reads a string as a quoted string only if the quote character you select is the first character of the field.
- If the field length exceeds the column size defined in the Source Qualifier transformation, the Integration Service truncates the field.
- If the row of data exceeds the larger of the line sequential buffer length or the total row size defined in the Source Qualifier transformation, the Integration Service drops the row and writes it to the session log file. To determine the row size defined in the Source Qualifier transformation, add the column precision and the delimiters, and then multiply the total by the maximum bytes per character.

Formatting Flat File Columns

When you import or edit flat file definitions, you need to define the following column options:

- **Precision.** Precision is defined differently for different datatypes.
- **Scale.** Scale is the maximum number of digits after the decimal point for numeric values.
- **Field width.** Field width is the number of bytes the Integration Service reads from or writes to a file. Field width applies to fixed-width file definitions only. Field width must be greater than or equal to the precision.

The following table describes precision and field width for flat file definitions:

Datatype	Fixed-Width Flat Files	Delimited Flat Files
Number	Precision is the number of significant digits. Field width is the number of bytes the Integration Service reads from or writes to the file. By default, the field width equals the precision. Use the Format column to define the field width. When you configure the field width, accommodate characters such as thousand separators, decimal separators, and negative signs. For example, "-123,456" has a field width of 8.	Precision is the number of significant digits. The Integration Service ignores any field width formatting.
Datetime	You do not enter a precision value. Field width is the number of bytes the Integration Service reads from or writes to the file. The datetime format specified for the column determines the field width. For example, the default datetime format of MM/DD/YYYY HH24:MI:SS has a field width of 19. Use the Format column to define the format and field width.	You do not enter a precision value. The Integration Service ignores any field width formatting.
String	Precision is the number of bytes the Integration Service reads from or writes to the file. You do not enter a field width for string values. The precision is the total length of the source or target field. **Note:** If you plan to load multibyte data into a fixed-width file target, you need to configure the precision to accommodate the multibyte data.	Precision is the maximum number of characters the Integration Service reads from or writes to the file. You do not enter a field width.

- **Format.** Format defines the appearance of numeric and datetime values.

  For datetime values, you can choose to output only the date or time parts of the data. For example, you can configure the file definition for datetime data with the following format:

  ```
  03/2002
  ```

  For numeric values, you can choose thousands and decimal separators. For example, you can configure the file definition for numeric data with the following format:

  ```
  1.000.000,95
  ```

  Note: If the source file structure changes significantly, you may need to reimport the flat file source definition.

  You can define the appearance of number and datetime columns at the following locations:

  - **Source or target field.** You can define the format for individual columns on the Columns tab for the source in the Source Analyzer, or for the target in the Target Designer.
  - **Source or target instance in a mapping.** You can define the default datetime and numeric formats for a source or target instance in a mapping in the Mapping Designer. The Integration Service uses the default formats you define when you do not define a format for an individual column.

Formatting Numeric Columns

When you edit flat file definitions, you can define formats for numeric values with the Number datatype. Bigint, double, and integer datatypes have a default precision, scale, and format. You can change the precision for a column that is an integer or double datatype.

Use the Format column on the Columns tab of a flat file source or target definition to define the format for numeric values.

You can define the following formatting options for numeric values in the Column Format Settings dialog box:

- Numeric data
- Field width

When you define formatting options in the Column Format Settings dialog box, the Designer shows the options you define in the Format column on the Columns tab.

Numeric Data

You can define decimal and thousands separators in the numeric data area of the Column Format Settings dialog box. For the decimal separator, you can choose a comma or a period. The default is the period. For the thousands separator, you can choose no separator, a comma, or a period. The default is no separator.

To specify numeric separators, click Override Separators and choose a separator from the Decimal Separator and Thousands Separator fields. You can override one or both separators. When you override the separators, you must choose different options for each.

For example, the source data contains a numeric field with the following data:

```
9.999.999,00
5.000.000,00
```

Choose the period as the thousands separator, and choose the comma as the decimal separator in the flat file source definition.

For example, you want to output the data above to a file target with the following format:

```
9,999,999.00
```

Choose the comma as the thousands separator, and choose the period as the decimal separator in the flat file target definition.

Field Width

You can change the field width in the Column Format Settings dialog box by padding the width or by defining a fixed-width value in bytes. By default, the field width equals the precision.

To output numeric data to a fixed-width flat file, you must configure the field width for the target field to accommodate the total length of the target field. If the data for a target field is too long for the field width, the Integration Service rejects the row and writes a message to the session log. When you configure the field width for flat file target definitions, you must accommodate characters the Integration Service writes to the target file, such as decimals and negative signs.

To adjust the field width, select Adjust Width and enter the number of bytes in the Padding field. When you adjust the field width, the Integration Service defines the field width as the field precision plus the padding you enter. For example, when the precision of a field is 10 and you enter 5 in the Padding field, the Integration Service reads 15 bytes from a file source, and it writes 15 bytes to a file target.

To fix the field width, select Fixed Width and enter the number of bytes in the Field Width field. The Designer lets you enter a field width greater than or equal to the precision. When you enter 20 in the Field Width field, the Integration Service reads 20 bytes from a file source, and it writes 20 bytes to a file target.

For example, you have a target field with a precision of four, scale of zero. You want to add two blank spaces to the target field to make the target file easier to view. Select Adjust Width and enter 2 in the Padding field. Or, select Fixed Width and enter 6 in the Field Width field.

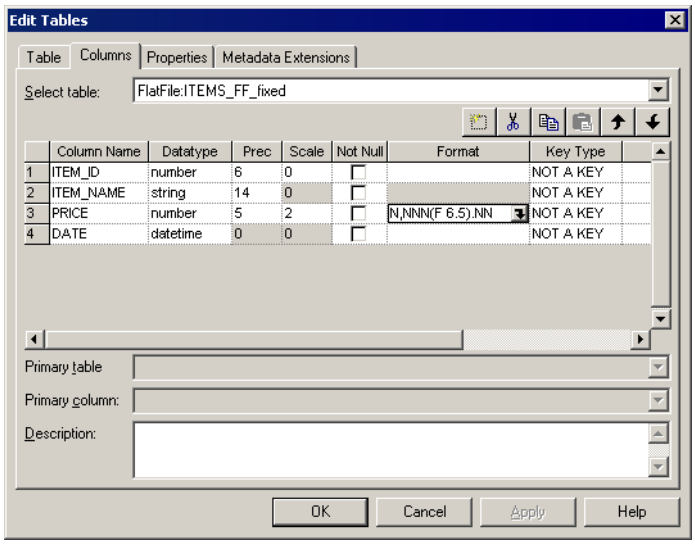
When padding numeric data in flat file targets, the Integration Service adds blank spaces to the left side of the target column.

Note: You might adjust the field width for a target field if you think another user might change the precision of the field. When the precision changes, the field width adjusts accordingly.

Format Column

When you override the numeric separators or define the field width, the Designer shows the options you define in the Format column on the Columns tab. For example, you have a numeric column with a precision of five. Click the Format column and define the options in the Column Format Settings dialog box.

The following figure shows the format options the Designer displays in the Format column:



The Designer displays `N,NNN(F 6.5).NN` in the Format column for the PRICE column. “N,NNN” displays the thousands separator you specified. “F” indicates a fixed field width. This displays “A” when you adjust the field width. “6.5” is the field width and precision in bytes. The first number is the field width, the second is the precision. “.NN” displays the decimal separator you specified.

Note: If you do not define decimal and thousands separators for a particular source or target field, the Integration Service uses the separators you specify in the source or target instance in the Mapping Designer.

Related Topics:

- “Defining Default Datetime and Numeric Formats” on page 84

Formatting Datetime Columns

When you edit flat file definitions, you can define formats for datetime values. Use the Format column on the Columns tab of a flat file source or target definition to define the format for datetime values.

You can define the following formatting options for datetime values in the Column Format Settings dialog box:

- Format string
- Field width

When you define formatting options in the Column Format Settings dialog box, the Designer shows the options you define in the Format column on the Columns tab.

Format String

You can enter any datetime format in the Format String field in the Column Format Settings dialog box. For example, you can specify the datetime format as MM/YYYY. Or, you can specify the time only, for example, HH24:MI.

To specify the datetime format, choose Format String and enter the format in the Format String field. You can choose a format from the list, or you can enter a format using the keyboard. The default format is MM/DD/YYYY HH24:MI:SS, which has a field width of 19.

For example, the source data contains a datetime field with the following data:

```
11/28/2002
10/15/2003
```

Enter the following format in the flat file source definition: MM/DD/YYYY.

For example, you want to output the data above to a file target with the following format:

```
28-11-2002
15-10-2003
```

Enter the following format in the flat file target definition: DD-MM-YYYY.

You can also enter any single-byte or multibyte string literal in the Format String field. To enter a string literal, enclose it in double quotes (“”). When you enter string literals in the format string, the Integration Service writes the strings to the file target when it runs the session. You might want to add string literals to describe the different date parts.

For example, you enter the following text in the Format String field:

```
“Month”MM/”Day”DD/”Year”YYYY
```

When you run the session and the Integration Service outputs the date October 21, 2002, it writes the following to the target file:

```
Month10/Day21/Year2002
```

Field Width

You can define the field width after you define a format string. You can change the field width by padding the width or by defining a fixed-width value in bytes. By default, the field width equals the precision.

To adjust the field width after you enter a format string, select Adjust Width and enter the number of bytes in the Padding field. When you adjust the field width, the Integration Service defines the field width as the number of bytes required for the datetime format plus the padding you enter. For example, when the datetime format is MM/YYYY and you enter 5 in the Padding field for a flat file source, the Integration Service reads 12 bytes from the file. When the datetime format is MM/YYYY and you enter 5 in the Padding field for a flat file target, the Integration Service writes 12 bytes to the file.

When you use Adjust Width, the Integration Service adjusts the field width based on the format string. You can change the format string without manually adjusting the field width.

To fix the field width after you enter a format string, select Fixed Width and enter the number of bytes in the Field Width field. You must specify a fixed-width value greater than or equal to the number of bytes required for the datetime format or the Integration Service truncates the data. For example, when the datetime format is MM/DD/YYYY HH24:MI:SS.NS, specify a fixed-width value greater than or equal to 29. When you enter 21 in the Field Width field, the Integration Service reads 21 bytes from a file source, and it writes 21 bytes to a file target.

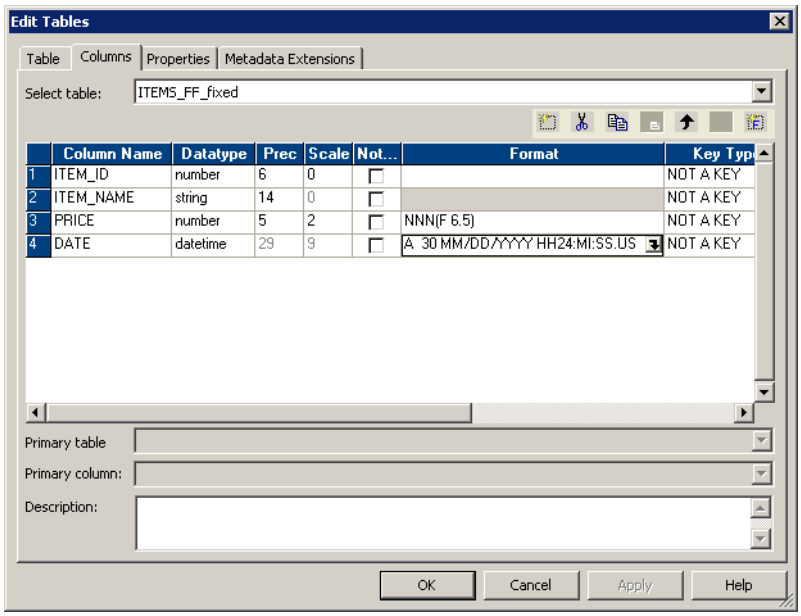
For example, you have a target field with a datetime format of MM/DD/YYYY, which requires 10 bytes. You want to add two blank spaces to the target field to make the target file easier to view. Select Adjust Width and enter 2 in the Padding field. Or, select Fixed Width and enter 12 in the Field Width field.

Note: When padding datetime data in flat file targets, the Integration Service adds blank spaces to the right side of the target column.

Format Column

When you choose a datetime format or define the field width, the Designer shows the options you define in the Format column on the Columns tab. For example, you define the options in the Column Format Settings dialog box.

The following figure shows the options the Designer displays in the Format column:



The Designer displays `A 30 MM/DD/YYYY HH24:MI:SS.US` in the Format column for the DATE port. The “A” indicates the field width is adjusted. The “30” is the field width in bytes: 26 bytes for precision to the microsecond, plus four bytes for padding.

Note: If you do not define a datetime format for a source or target field, the Integration Service uses the datetime format you specify in the source or target instance in the mapping.

Defining Default Datetime and Numeric Formats

When you use a flat file source or target definition in a mapping, you can define the default formats for datetime and number columns in the file. The Integration Service uses the default formats you define when you do not define the format for an individual column.

If you do not define the default formats, the Integration Service uses the following formats:

Format	Description
Datetime Format	Datetime format. Default is MM/DD/YYYY HH24:MI:SS.
Thousand separator	Thousand separator. Choose no separator, a comma, or a period. Default is no separator.
Decimal separator	Decimal separator. Choose a comma or a period. Default is a period.

Note: You can view the default formats for each source or target instance on the Mappings tab in the session properties.

Working with File Lists

You can create a session to read multiple source files for one source instance in a mapping. For example, if your organization collects data for multiple locations that you want to process through the same mapping, you can create a file list. A file list is a file that contains the names and directories of each source file you want the Integration Service to use.

You configure the session to read a file list. When you configure a session to read a file list, the Integration Service reads rows of data from the different source files in the file list. To configure the mapping to write the source file name to each target row, add the CurrentlyProcessedFileName port to the flat file source definition. The Integration Service uses this port to return the source file name.

You add the CurrentlyProcessedFileName port to a flat file source definition in the Source Analyzer.

To add the CurrentlyProcessedFileName port:

1. Open the flat file source definition in the Source Analyzer.
2. Click the Properties tab.
3. Select Add CurrentlyProcessedFlatFileName Port.

   The Designer adds the CurrentlyProcessedFileName port as the last column on the Columns tab. The CurrentlyProcessedFileName port is a string port with default precision of 256 characters.
4. Click the Columns tab to see your changes.

   You may change the precision of the CurrentlyProcessedFileName port if you wish.
5. To remove the CurrentlyProcessedFileName port, click the Properties tab and clear the Add Currently Processed Flat File Name Port check box.

Working with Shift-Sensitive Flat Files

You can import shift-sensitive flat files in the Flat File Wizard in both fixed-width mode and delimited mode.

A shift-sensitive file may contain both multibyte and single-byte characters. A file may or may not contain shift-in and shift-out keys to separate multibyte characters from single-byte characters.

Shift-in and shift-out keys separate multibyte characters so the Flat File Wizard and the Integration Service can read each character correctly. A shift-out key signals the beginning of a sequence of multibyte characters. A shift-in key signals the end of this sequence. If the file source does not contain shift keys, you need to define shift states for each column in the file so the Integration Service can read each character correctly.

Note: Use single-byte and double-byte shift keys.

Importing Flat Files with Shift Keys

Use the Flat File Wizard to import a file source that contains shift keys. You can import both fixed-width and delimited files that contain shift keys. The Flat File Wizard and the Integration Service use the shift keys in the file to determine the shift state of each column in the source.

The Flat File Wizard and the Integration Service can handle consecutive shift characters.

The following example is a valid row from a shift-sensitive flat file:

```
aaa-oAAA-i-oAAA-iaaaaa
```

The following table describes the notation used in this example:

Notation	Description
a	Single-byte character
A	Multibyte character
-o	Shift-out character
-i	Shift-in character

The Flat File Wizard displays single-byte shift characters as '.' in the window. Double-byte shift characters display as '..' in the window. Shift-in characters display on a green background. Shift-out characters display on a blue background.

Requirements for Shift-Sensitive Flat Files

The Designer returns an error if you analyze a shift-sensitive flat file that contains shift-in and shift-out characters, but does not meet the following requirements:

- A shift-out and a shift-in character must enclose all multibyte characters. Single-byte characters do not need to be enclosed with shift characters.
- The first shift character in a row must be a shift-out character.
- A file cannot contain nested shift characters. For example, you cannot have the following sequence:

  ```
  -oAA-oAA-iaaa
  ```
- A shift-in character must close a shift-out character in the same row.

The Flat File Wizard disables the fixed-width option if the file has a multibyte character that contains more than two bytes per character. Each row in a file must not exceed 16 KB.

The Flat File Wizard validates up to 500 rows or 256 KB of data, whichever comes first. If the file does not meet the above requirements, the Integration Service writes errors to the session log.

Importing Flat Files without Shift Keys

Use the Flat File Wizard to import a fixed-width file source that does not contain shift keys. However, after you import the source definition, you must define shift states for each column in the file source definition so the Integration Service can read each character correctly.

You can also import shift-sensitive COBOL files that do not contain shift keys. When you do, you must also define the shift states for each column in the COBOL source definition.

Note: When you create a session using a flat file source that contains user-defined shift states, verify that the code page in the Workflow Manager is the same as the code page you chose in the Designer. If you choose a different source code page in the Workflow Manager, the Integration Service does not use the shift keys you defined in the Designer.

To define shift states for fixed-width shift-sensitive files:

1. In the Designer, import the flat file source or COBOL file.
2. In the Source Analyzer, double-click the title bar of the file source definition.
3. In the Flat File Information section of the Table tab, select Fixed Width and click Advanced.

   The Edit Flat File Information - Fixed Width Files dialog box appears.
4. Select User Defined Shift State and click OK.
5. Click the Columns tab.

   The Shift Key column appears.
6. Choose a shift key for each column.

   Choose Shift-In if the column contains single-byte characters. Choose Shift-Out if the column contains multibyte characters.
7. Click OK.

Working with Multibyte Data in Fixed-Width Targets

To load multibyte data into a fixed-width flat file target, configure the precision to accommodate the multibyte data. Fixed-width files are byte-oriented, not character-oriented. When you configure the precision for a fixed-width target, you need to consider the number of bytes you load into the target, rather than the number of characters. The Integration Service writes the row to the reject file if the precision is not large enough to accommodate the multibyte data. When the Integration Service writes the row to the reject file, it writes a message in the session log.

Note: Delimited files are character-oriented, and you do not need to allow for additional precision for multibyte data.

Troubleshooting Flat Files

I ran a session for a flat file target that contains multibyte data. The data in the flat file target does not include some multibyte characters.

If the code page you select when you imported the flat file target using the Flat File Wizard is not a superset of the source code page, characters that are not encoded in the target code page may be lost. Select a code page that is a superset of the source code page when you import a flat file target using the Flat File Wizard.

CHAPTER 4

Working with Targets

This chapter includes the following topics:

- Working with Targets Overview, 88
- Importing a Target Definition, 91
- Creating a Target Definition from a Source Definition, 93
- Creating a Target Definition from a Transformation, 95
- Manually Creating a Target Definition, 99
- Maintaining Relational Target Definitions, 99
- Creating a Target Table, 104
- Troubleshooting Targets, 105

Working with Targets Overview

Before you create a mapping, you must define targets in the repository. Use the Target Designer to import and design target definitions or to create and maintain target definitions. Target definitions include properties such as column names and data types.

Creating Target Definitions

You can create the following types of target definitions in the Target Designer:

- **Relational.** Create a relational target for a particular database platform. Create a relational target definition when you want to use an external loader to the target database.
- **Flat file.** Create fixed-width and delimited flat file target definitions.
- **XML file.** Create an XML target definition to output data to an XML file.

You can create target definitions in the following ways:

- **Import the definition for an existing target.** Import the target definition from a relational target or a flat file. The Target Designer uses a Flat File Wizard to import flat files.
- **Create a target definition based on a source definition.** Drag a source definition into the Target Designer to make a target definition.
- **Create a target definition based on a transformation or mapplet.** Drag a transformation into the Target Designer to make a target definition.
- **Manually create a target definition.** Create a target definition in the Target Designer.

- **Design several related target definitions.** Create several related target definitions at the same time. You can create the overall relationship, called a *schema*, and the target definitions, through wizards in the Designer. The Cubes and Dimensions Wizards follow common principles of data warehouse design to simplify the process of designing related targets.

Maintaining Targets and Target Definitions

In addition to creating target definitions, you can complete the following tasks in the Target Designer:

- **Reimport target definitions.** When the target structure changes significantly, you can reimport the target definition to make sure it is correct.
- **Edit target definitions.** Edit target definitions to add comments or key relationships, or update them to reflect changed target definitions.
- **Create relational tables in the target database.** If the target tables do not exist in the target database, you can generate and execute the necessary SQL code to create the target table that matches the target definition.
- **Preview relational and flat file target data.** You can preview the data of relational and flat file target definitions in the Designer.
- **Compare target definitions.** You can compare two target definitions to identify differences between them.

Oracle Targets

You can import Oracle targets that use basic compression and OLTP compression. You can also manually create target definitions for Oracle targets that use basic compression and OLTP compression.

Target Code Pages

You can create targets with multibyte character sets. Target code pages must be a superset of source code pages when you run a session.

Special Character Handling in Target Definitions

You can import, create, and edit target definitions with table and field names containing special characters, such as the slash (/) character through the Designer. When you import, create, or edit a target definition using the Target Designer, the Designer retains special characters in table and field names when saving the target definition to the repository.

However, when you add a target definition with special characters to a mapping, the Designer either retains or replaces the character. Also, when you generate the target update override in target instance in the Target Designer, the Designer uses quotation marks around table and column names with some special characters. The Designer handles special characters differently for relational and non-relational targets.

The following table describes how the Designer handles special characters in relational targets:

Special Character	Target Designer Behavior	Mapping Designer Behavior
`@#$_`	Retains the character in the target definition table name. Retains the character in the target definition column names.	Retains the character in the target instance table name. Retains the character in the target instance column names. Does not use quotation marks around the table or column names in the target update override.
`/+-=~`!%^&*()[]{}';?,<>| <space>`	Retains the character in the target definition table name. Retains the character in the target definition column names.	Replaces the character in the target instance table name with the underscore character. Retains the character in the target instance column names. Delimits table and column names including special characters with quotation marks in the target update override.
`.":\t\r\n`	The Designer does not recognize these characters in relational target table and column names.	The Designer does not recognize these characters in relational target table and column names.

The following table describes how the Designer handles special characters in non-relational targets:

Special Character	Target Designer Behavior	Mapping Designer Behavior
`@#$_`	Retains the character in the target definition table name. Retains the character in the target definition column names. **Note:** You cannot use the @ character as the first character in a table or column name.	Retains the character in the target instance table name. Retains the character in the target instance column names. **Note:** You cannot use the @ character as the first character in a table or column name.
`/`	Retains the character in the target definition table name. Retains the character in the target definition column names.	Replaces the character in the target instance table name with the underscore character. Retains the character in the target instance column names.
`.+-=~`!%^&*()[]{}'" ;:?,<>| \t\r\n <space>`	The Designer does not recognize these characters in non-relational target table and column names.	The Designer does not recognize these characters in non-relational target table and column names.

Some databases require special configuration or commands to allow table and field names containing special characters. For more information, see the database documentation.

Importing a Target Definition

You can import the following target definitions:

- **Flat file.** The Target Designer uses the Flat File Wizard to import a target definition from a flat file that matches the structure of the flat file.
- **Relational table.** You can import a relational table to create a target definition that matches the structure of the relational table.
- **XML file.** You can import an XML target definition from an XML, DTD, or XML schema file.

To import a relational target definition, you need to configure connectivity between the target database and the PowerCenter Client.

Use a target definition in a mapping after you have added it to the repository.

Relational Target Definitions

When you import a target definition from a relational table, the Designer imports the following target details:

- **Target name.** The name of the target.
- **Database location.** You specify the database location when you import a relational source. You can specify a different location when you edit the target definition in the Target Designer and when you configure a session.
- **Column names.** The names of the columns.
- **Datatypes.** The Designer imports the native datatype for each column.
- **Key constraints.** The constraints in the target definition can be critical, since they may prevent you from moving data into the target if the Integration Service violates a constraint during a workflow. For example, if a column contains the NOT NULL constraint and you do not map data to this column, the Integration Service cannot insert new records into the target table.
- **Key relationships.** You can customize the Target Designer to create primary key-foreign key relationships. Click Tools > Options and select the Format tab. Select Import Primary and Foreign Keys.

  You can also create logical relationships in the repository. Key relationships do not have to exist in the database.

When you import target definitions, the Designer does not import the target indexes. You can change this default setting in powrmart.ini. You can find powrmart.ini in the root directory of the PowerCenter Client installation.

To import target definition indexes, add the following text to powrmart.ini [Main] section:

```
ImportIndexes=Yes
```

Note: Because views can include columns from more than one table, the Integration Service might encounter database errors when trying to insert, update, or delete data. If you import a target view, make sure that it is a view of a single table.

Connectivity for Relational Targets

To import a relational target definition, you must be able to connect to the database from the client machine using a properly configured ODBC data source or gateway. You may also require read permission on the database object.

When you create an ODBC data source, you must also specify the driver that the ODBC driver manager sends database calls to. The following table shows the recommended ODBC drivers to use with each database:

Database	ODBC Driver	Requires Database Client Software
IBM DB2	IBM ODBC driver	Yes
Informix	DataDirect ODBC Wire Protocol driver	No
Microsoft Access	Microsoft Access driver	Yes
Microsoft Excel	Microsoft Excel driver	Yes
Microsoft SQL Server	Microsoft SQL Server ODBC driver	No
Oracle	DataDirect 32-bit closed ODBC driver	No
Sybase ASE	DataDirect 32-bit closed ODBC driver	No
Teradata	Teradata ODBC driver	n/a

When you use a third-party ODBC data source to import a target definition, the Designer might display a message indicating that the third-party driver is not listed in powrmart.ini. The Designer attempts to import target definition metadata using a driver that is shipped with PowerCenter. If the third-party provides a driver to import metadata, configure powrmart.ini.

For example, if Vendor A provided a driver named `vendoraodbc.dll`, you can add an entry under the ODBCDLL heading based on the specified database:

```
Vendor A = pmodbc.dll
Vendor A = extodbc.dll
```

In the example, the Designer directly interacts with the system ODBC drivers to use the ODBC data source. The system ODBC drivers internally interacts with the third-party ODBC driver, `vendoraodbc.dll`.

The following table lists a sample database dependent entry to use with the PMODBC.ini system ODBC driver:

Database	Entry
MySQL	`MYSQL = PMODBC.DLL`
PowerExchange	`PWX = PMODBC.DLL`
dBase 4	`dBASE IV = PMODBC.DLL`
Visual FoxPro	`Visual FoxPro = PMODBC.DLL`
Sybase IQ	`Adaptive Server IQ = PMODBC.DLL`
Lotus Notes	`Lotus Notes = PMODBC.DLL`

Configuring a Third-Party ODBC Data Source

PowerCenter uses the powrmart.ini file to recognize the third-party ODBC driver, import the ODBC database object with third-party ODBC drivers, and to preview data in Designer. To import a target definition with an ODBC driver that is not included in the powrmart.ini file, configure the file on the PowerCenter Client machine.

1. Open powrmart.ini in the following directory:

   `<Informatica installation directory>\clients\PowerCenterClient\client\bin`

2. Add an entry to the file under the ODBCDLL section that includes the name of the ODBC data source.

   Ensure that the entry directly points to `pmodbc.dll` or `extodbc.dll`.

3. Save and close powrmart.ini.
4. Restart the PowerCenter Client and import the target definition.

Importing Relational Target Definitions

To create a relational target definition, use the Target Designer to import target metadata.

To import a relational target definition:

1. In the Target Designer, click Targets > Import from Database.
2. Select the ODBC data source used to connect to the target database.

   If you need to create or modify an ODBC data source first, click the Browse button to open the ODBC Administrator. After creating or modifying the ODBC source, continue with the following steps.

3. Enter the user name and password needed to open a connection to the database, and click Connect.

   If you are not the owner of the table you want to use as a target, specify the owner name.

4. Drill down through the list of database objects to view the available tables as targets.
5. Select the relational table or tables to import the definitions into the repository.

   You can hold down the Shift key to select a block of tables, or hold down the Ctrl key to make non-contiguous selections. You can also use the Select All and Select None buttons to select or clear all available targets.

6. Click OK.

   The selected target definitions now appear in the Navigator under the Targets icon.

Creating a Target Definition from a Source Definition

When you want to create a target definition that closely matches a source definition, you can use the source definition to create the target definition. You can also use a shortcut to a source definition to create the target definition. You can drag the following source definitions into the Target Designer to create target definitions:

- Relational sources
- Flat file sources
- COBOL sources
- XML sources

After you create the matching target definition, you can add or edit target properties and change the target type. When you create a relational target definition, you can generate the target table in the target database.

Creating a Target Definition from a Relational Source

When you drag a relational source definition into the Target Designer workspace, the Designer creates a relational target definition that matches the source definition.

You can edit the definition to change information, such as the description, columns, data types, and target type.

Creating a Target Definition from a Flat File Source

When you drag a flat file source definition into the Target Designer workspace, the Target Designer creates a flat file target definition by default that matches the source definition.

When you create a flat file target definition from a flat file source definition, the Designer uses the flat file source definition code page.

You can edit the definition to change information, such as the description, columns, datatypes, and target type.

Creating a Normalized Target from a COBOL Source

To create a target based on normalized COBOL sources, you first need to analyze the COBOL structure using the Source Analyzer.

When you drag a normalized COBOL source definition into the Target Designer workspace, the Target Designer creates relational target definitions based on the following rules:

- The number of tables that appears is one more than the number of OCCURS statements in the COBOL file.
- The target table name defaults to the record name.
- The generated key name for each table is GK_target_table_name.
- The number of generated key names is the number of OCCURS statements minus the number of primary keys.

The following figure shows a sample COBOL source definition with five OCCURS statements:

SCHOOL_REC (VSAM)

K	Name	Lev...	Oc...	Datatype
	SCHOOL_REC	1	0	
	SCHOOL_ID	2	0	number
	SCHOOL_NM	2	0	string
	CLASS_REC	2	2	
	CLASS_ID	3	0	number
	CLASS_NM	3	0	string
	STUDENT...	3	5	
	STUDE...	4	0	number
	STUDE...	4	0	string
	PARENT...	4	2	
	PARE...	5	0	number
	PARE...	5	0	string
	TEACHER...	3	3	
	TEACHE...	4	0	number
	TEACHE...	4	0	string
	SPORT_REC	2	2	
	SPORT_T...	3	0	string

When you drag the source into the Target Designer workspace, the Designer creates six target definitions.

Steps to Create a Target Definition from a Source Definition

Use the following procedure to create a target definition from a source definition.

To create a target definition based on a source definition:

1. With the Target Designer tool active, drag the source definition you want to use into the workspace. For XML sources, select the option to create relational targets or XML targets and click OK.

   The target definition appears.
2. To edit the target definition, double-click the title bar.
3. Enter a target name and select the target type. Add or edit columns or target properties, and then click OK.

You can now use the target definition in a mapping. You can also create a target tables in the target database based on relational target definitions.

Creating a Target Definition from a Transformation

To create a relational target definition that closely matches a transformation in the repository, you can create the target from the transformation. Drag a transformation from the Navigator to the Target Designer, or create a target from a transformation in the Mapping Designer workspace.

Create target definitions from the following types of transformations:

- **Single-group transformations.** Create a single target definition from a transformation with one output group.
- **Multiple-group transformations.** Create multiple target definitions from a transformation with multiple output groups.
- **Normalizer transformations.** Create a target definition from a source qualifier or pipeline Normalizer transformation.
- **Mapplets.** Create one or more target definitions from a mapplet instance in a mapping.

When you create a target definition from a transformation, the target database type is the same as the repository database by default. After you create the target definition in the repository, you can edit it. For example, you might want to change the target type.

If you need to create a target definition that contains columns from more than one transformation, you can copy the ports from each transformation to a transformation such as an Expression or Joiner. You can create the target definition from that transformation.

When you create a relational target definition, you must generate and execute the SQL to create the table in the target database.

Creating a Target from a Transformation with One Output Group

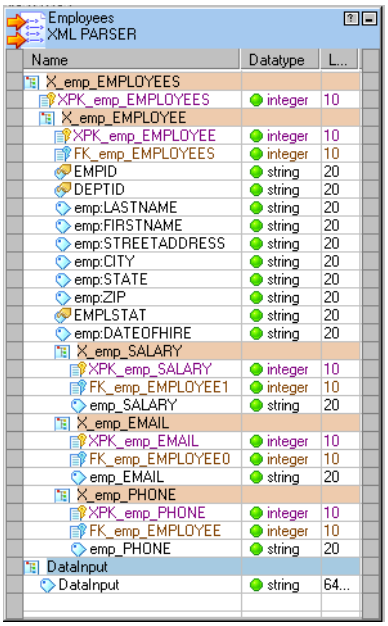
When you create a target from a transformation with one output group, the Designer creates one target. All the output ports become input ports in the target. The name of the target is the same as the transformation name.

Creating a Target from a Transformation with Multiple Output Groups

When you create targets from a transformation with more than one output group, the Designer creates one target for each output group in the transformation. When the transformation is a plug-in or Custom

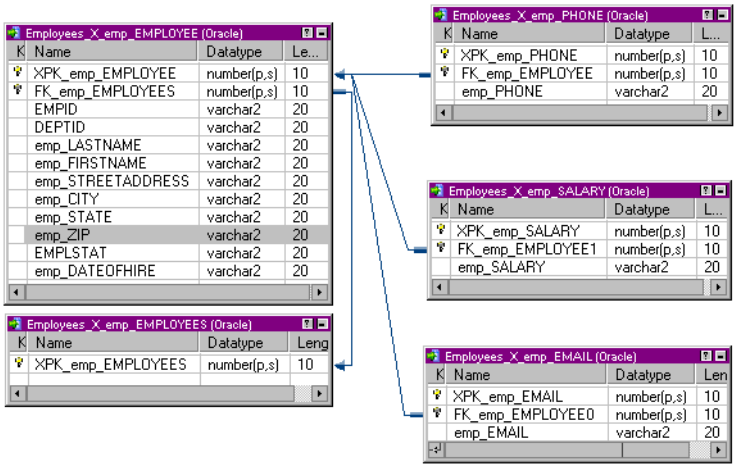
transformation, the Designer retains the primary key-foreign key relationships between the groups in the target definitions.

The following figure shows a transformation with multiple output groups:



The Employees XML Parser transformation has five output groups, from X_emp_EMPLOYEES to X_emp_PHONE. Each output group represents different types of employee information. DataInput is the input group.

When you create a target from this transformation, the Designer creates a separate target for each output group. Each target has the corresponding group name from the transformation group, as shown in the following figure:

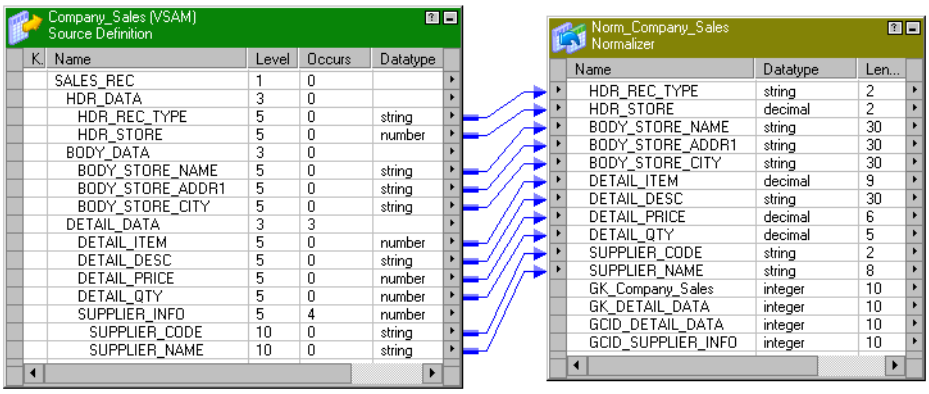


Creating a Target from a Normalizer Transformation

You can create a target from a source qualifier or pipeline Normalizer transformation. When you create a target from a Normalizer transformation, the Designer creates one target and includes all the columns from

the Normalizer. It does not create separate targets to represent the record hierarchies or multiple-occurring fields in the Normalizer transformation.

The following figure shows a Normalizer transformation in a mapping with a COBOL source definition:



The Normalizer transformation, Norm_Company_Sales, represents the hierarchical data structure of the Company_Sales COBOL source definition. When you create a target from the Norm_Company_Sales transformation, the Designer flattens the hierarchies in the transformation into one target. The target includes the generated keys from the Normalizer transformation and the generated column ID (GCID) for the multiple-occurring records, DETAIL_DATA and SUPPLIER_INFO.

Creating a Target from a Mapplet

You can create a target from a mapplet that is under a mapping Transformation Instances node. When you drag the mapplet instance to the Target Designer, the Designer creates a target for each output group in the mapplet.

Note: You cannot create a target when you drag a transformation instance from a mapplet to the Target Designer.

Transformation and Target Datatypes

The Designer creates the target datatypes based on the best match between the transformation datatypes and the datatypes for the repository database.

The following table describes the transformation datatypes and the corresponding datatypes for each database:

Transformation Datatype	IBM DB2	Microsoft SQL Server	Oracle	Sybase ASE	Teradata	Informix
bigint	bigint	bigint	number(19,0)	bigint	bigint	int8
binary	char for bit data	binary	raw	binary	byte/varbyte	byte
date/time	timestamp	datetime	timestamp	datetime	timestamp	datetime year to fraction
decimal	decimal	decimal	number(p,s)	decimal	decimal	decimal(p,s)

Transformation Datatype	IBM DB2	Microsoft SQL Server	Oracle	Sybase ASE	Teradata	Informix
double	float	float	number	float	float	float(p)
integer	integer	int	number(p,s)	int	integer	integer
nstring	vargraphic	nvarchar	nvarchar2	nvarchar	n/a	nvarchar
ntext	long vargraphic	ntext	nclob	nvarchar	n/a	n/a
real	float	real	number	real	n/a	smallfloat
small integer	smallint	smallint	smallint	smallint	smallint	smallint
string	varchar	varchar	varchar2	varchar	varchar	varchar(m,r)
text	long varchar	text	long	text	n/a	text

Steps to Create a Target

You can create a target by dragging one or more transformations from the Navigator to the Target Designer or you can create a target from a transformation instance in the Mapping Designer.

Steps to Create a Target in the Target Designer

You can create one or more targets in the Target Designer by selecting transformations in the Navigator and dragging them to the Target Designer workspace.

When you create a target from a transformation in a different folder, the Designer copies the transformation to the target folder and creates the target from the transformation. When you create a target from a transformation in a shared folder, the Designer creates a shortcut to the transformation in the target folder before it creates the transformation.

The following table describes the objects you can use to create a target in the Target Designer:

Object	Navigator Location
Transformation	Transformations node
Transformation Instance	Mappings node
Mapplet	Transformation Instances node for a mapping

To create a target definition in the Target Designer:

1. Open the Target Designer.
2. Drag a transformation from the Transformations node or the Transformation Instances node in the Navigator to the workspace.

   The target definition appears.

The Designer adds the new target to the Navigator and the Target Designer workspace. When you use version control in the repository, the new target is checked out by default.

If the new target name conflicts with an existing target name, the Designer prompts you to rename the new target or replace the existing target definition.

Steps to Create a Target in the Mapping Designer

You can create a target from a transformation instance in the Mapping Designer. When you create a target in the Mapping Designer, you create a target instance in the mapping. The Target Designer displays the target definition.

To create a target definition in the Mapping Designer:

1. Open a mapping in the Mapping Designer.
2. Right-click a transformation instance in the mapping.
3. Click Create and Add Target.

If the repository contains a target definition with the same name, you must create the target in the Target Designer instead of the Mapping Designer.

If there are no name conflicts, the Designer adds the new target to the Navigator and to the Mapping Designer workspace. You can link the transformation ports to the target.

Manually Creating a Target Definition

You can manually create a target definition rather than importing it or creating it from a source definition.

To manually create a target definition:

1. In the Target Designer, click Targets > Create.
2. Enter a name for the target and select the target type.

   If you create a relational definition, follow database-specific naming conventions.
3. Click Create.

   An empty definition appears in the workspace. It may be covered by the dialog box. The new target definition also appears within the Navigator window.
4. If you want to create another target definition, enter a new target name and target type and click Create. Repeat this step for each target you want to create.
5. Click Done when you finish creating target definitions.
6. Configure the target definition.

   The new target definition is saved to the repository. You can now use the target definition in a mapping.

You can also create a target table in the target database based on relational target definitions.

Note: You cannot manually create a target definition for XML files.

Maintaining Relational Target Definitions

You can maintain relational target definitions in the following ways:

- **Reimport the target definition.** Reimport a target definition rather than edit it if the target changes significantly.

- **Define primary key-foreign key relationships.** Define primary key-foreign key relationships between relational target tables.
- **Edit the target definition.** Edit a target definition to add comments or key relationships, or update it to reflect a changed target.

After you create a relational target definition, you can edit it using the following target definition tabs:

- **Table tab.** Edit properties such as constraints for relational targets and flat file properties for flat file targets.
- **Columns tab.** Edit column information such as datatype and precision.
- **Indexes tab.** Add index information for relational target definitions.
- **Metadata Extensions tab.** Extend the metadata stored in the repository by associating information with repository objects, such as target definitions.

When you change a target definition, the Designer propagates the changes to any mapping using that target. Some changes to target definitions can invalidate mappings.

The following table describes how you can impact mappings when you edit target definitions:

Modification	Result
Add a column.	Mapping not invalidated.
Change a column datatype.	Mapping may be invalidated. If the column is connected to an input port that uses a datatype that is incompatible with the new one (for example, Decimal to Date), the mapping is invalid.
Change a column name.	Mapping may be invalidated. If you change the column name for a column you just added, the mapping remains valid. If you change the column name for an existing column, the mapping is invalidated.
Delete a column.	Mapping may be invalidated if the mapping uses values from the deleted column.
Change the target definition type.	Mapping not invalidated.

When you add a new column to a target in the Target Designer, all mappings using the target definition remain valid. However, when you add a new column and change some of its properties, the Designer invalidates mappings using the target definition.

You can change the following properties for a newly added target column without invalidating a mapping:

- Name
- Datatype
- Format

If the changes invalidate the mapping, validate the mapping and any session using the mapping. You can validate objects from the Query Results or View Dependencies window or from the Repository Navigator. You can validate multiple objects from these locations without opening them in the workspace. If you cannot validate the mapping or session from one of these locations, open the object in the workspace and edit it.

Reimporting a Relational Target Definition

If a target table changes, such as when you change a column datatype, you can edit the definition or you can reimport the target definition. When you reimport the target, you can either replace the existing target definition or rename the new target definition to avoid a naming conflict with the existing target definition.

To reimport a target definition:

1. In the Target Designer, follow the same steps to import the target definition, and select the target to import.

   The Designer notifies you that a target definition with that name already exists in the repository. If you have multiple tables to import and replace, select Apply To All Tables.
2. Click Rename, Replace, Skip, or Compare.
3. If you click Rename, enter the name of the target definition and click OK.
4. If you have a relational target definition and click Replace, specify whether you want to retain primary key-foreign key information and target descriptions.

   The following table describes the options available in the Table Exists dialog box when reimporting and replacing a relational target definition:

Option	Description
Apply to all Tables	Select this option to apply rename, replace, or skip all tables in the folder.
Retain User-Defined PK-FK Relationships	Select this option to keep the primary key-foreign key relationships in the target definition being replaced. This option is disabled when the target definition is non-relational.
Retain User-Defined Descriptions	Select this option to retain the target description and column and port descriptions of the target definition being replaced.

Creating a Primary Key-Foreign Key Relationship

To create a relationship between two relational tables, choose the Link Column mode from the Layout menu. Drag from the Foreign Key column of one table to the Primary Key column of another table. The Designer may prompt you to remove the existing link to the default primary key table.

Editing Table Options

You can edit the following options on the Table tab of the target definition:

- **Business names.** Add a more descriptive name to the table using the Rename button.
- **Constraints.** SQL statements for table-level referential integrity constraints. Applies to relational targets only.
- **Creation options.** SQL statements for table storage options. Applies to relational targets only.
- **Description.** Add a comment or link to business documentation. These are displayed in Repository Manager for the target table. Adding comments or business documentation links to targets is an easy way to document the purpose of a target. You can add or modify comments to any existing target.

  You can enter up to 2,000 bytes/K in the description, where K is the maximum number of bytes a character contains in the selected repository code page. For example, if the repository code page is a Japanese code page where K=2, each description and comment field can contain up to 1,000 characters.

- **Keywords.** Keep track of target definitions with key words. As development and maintenance work continues, the number of targets increases. While all of these targets may appear in the same folder, they may all serve different purposes. Keywords can help you find related targets. Keywords can include developer names, mappings, or the associated schema.

  Use keywords to perform searches in the Repository Manager.

- **Database type.** Define the target type, either a relational database or a flat file. You can change a relational target definition to a flat file target definition and vice versa. When you change the target definition type, you lose some metadata when you save the changes to the repository.

  When you change the target definition type from relational to flat file, you lose indexes information, constraints information, and creation options information. The Workflow Manager invalidates all sessions using the target.

  When you change the target definition type from flat file to relational, you lose all flat file property information. If you change the target definition back to a flat file, the Designer uses default values for the flat file properties. The Workflow Manager invalidates all sessions using the target.

  Note: If you change the target type from flat file to relational, the Workflow Manager invalidates all sessions that use the target. However, you can change a target type from relational to flat file without invalidating sessions that use the target.

- **Flat file information.** When the database type is flat file, you can define the flat file properties by clicking the Advanced button.

To add options to a relational target definition:

1. In the Target Designer, double-click the title bar of the target definition.

   The Edit Tables dialog box appears.
2. Click the Rename button to edit the target name and the business name.
3. To change the target type, choose a different database in the Database Type field.

   To change the target type to a flat file target, choose flat file.
4. Edit the following properties for relational target definitions:
   - To add a constraint, type the SQL statement in the Constraints field.
   - To add a creation option, type the SQL statement in the Creation Options field.
5. To add a description, type a description in the Description field.
6. To add keywords, click Edit Keywords.

   The Edit Keywords dialog box appears.
7. Use the buttons to create and move keywords.
8. Click OK.

Editing Columns

You can edit the following information in the Columns tab of the target definition:

- **Column name.** The column names in the target. When editing a relational target definition, edit the column name if you are manually creating the relational target definition or if the actual target column name changed.
- **Datatype.** The datatypes that display in the target definition depend on the target type of the target definition.
- **Precision and scale.** When designing or importing relational targets, consider the precision and scale of values in each column. *Precision* is the maximum number of significant digits for numeric datatypes, or

the maximum number of characters for string datatypes. Precision includes scale. *Scale* is the maximum number of digits after the decimal point for numeric values. Therefore, the value 11.47 has a precision of 4 and a scale of 2. The string *Informatica* has a precision (or length) of 11.

All datatypes for relational targets have a maximum precision. For example, the Integer datatype has a maximum precision of 10 digits. Some numeric datatypes have a similar limit on the scale or do not allow you to set the scale beyond 0. Integers, for example, have a scale of 0, since by definition they never include decimal values.

You can change the precision and scale for some datatypes to values that differ from the values defined in the database. However, changing the precision or scale can cause numeric overflow on numeric columns, truncation on character columns, or insertion of zeroes on datetime columns when the Integration Service writes to the target column.

- **Not null.** Choose whether you want to allow null data in the target.
- **Key type.** Select Primary, Foreign, Primary-Foreign, or Not a Key. Applies to relational targets only.
- **Business name.** Optionally, you can add business names to each target column.

To edit the columns of a relational target definition:

1. In the Target Designer, double-click the title bar of a target definition.
2. Select the Columns tab.
3. Configure the options of the target definition as described above.
4. If you are creating a target definition and you want to add columns, select a column and click Add.
5. Enter the name, datatype, and other characteristics of the column.

   Repeat these steps for each column you want to add to the table.
6. If you want to move a column, use the Up and Down buttons, or drag it within the scrolling list.
7. Click OK.

Defining Indexes

Since indexes speed queries against tables, adding indexes to the target database is an important part of target table design. You can add index information to relational target definitions. Queries to the data warehouse determine which columns you should index. If you define indexes, select the option to create indexes when you create target tables.

To create indexes for a target table:

1. In the Target Designer, double-click the title bar of a relational target definition.
2. Select the Indexes tab.
3. To add an index, click the Add button in the Indexes section.
4. Enter a name for the index and press Enter.
5. To add a column to the index, click the Add button in the Columns section. Select a column name and click OK.
6. Repeat steps 3 to 5 for each column you want to assign.
7. Click OK.

Important: When you generate and execute the DDL to create the target table, choose to create an index.

Creating a Target Table

After you add a relational target definition to the repository, you can instruct the Designer to generate and execute the SQL code to create the target in a relational database. You cannot create a table in a relational database from an XML target definition or a flat file target definition.

The Designer generates the SQL script using characters in the UCS-2.

If the target already exists in that database, you can drop it and re-create it. The Designer writes the SQL code to an .SQL text file, so you can review and edit the DDL commands by opening this file.

To generate and execute the SQL code:

1. In the Target Designer, select the relational target definition you want to create in the database. If you want to create multiple tables, select all relevant table definitions.
2. Click Targets > Generate/Execute SQL.

   Click Connect and select the database where the target table should be created. Click OK to make the connection.

   Enter a file name and location for the SQL script you are about to generate and any options you want to include in the SQL DDL code. This text file exists on the local file system, not in the repository.

   Depending on the Generation options you select, the SQL script will contain all of the CREATE and DROP commands that match the selections. For example, if you created a target definition with primary keys, choose to generate the SQL with primary keys.
3. Click Generate SQL File if you want to create the SQL script, or Generate and Execute if you want to create the file, and then immediately run it.

   When you click Generate SQL file, the SQL generated for the selected table definitions is stored in the file you selected. If the file already exists, a dialog box appears prompting you to overwrite the existing file. The progress of copying the generated SQL file appears in the Output window in the Designer.

   After the file has been generated, you can click Edit SQL File, which opens a text editor so you can modify the SQL statements. When the Designer generates the SQL file for the target database, it encloses all table and field names containing the slash character in double quotes.

   You can click Execute SQL File to create the tables. When you click Generate and Execute, the SQL generated for the selected table definitions is stored in the file you selected and immediately executed.

   Note: As long as the Designer is open, it locks the SQL file you last opened and modified. If you want to unlock the file so that you can view it in a different application, open a different SQL file in the Designer, or exit the Designer.
4. Click Close.

   When you close this dialog box, the Designer maintains an open connection to the target database. If you reopen the dialog box, you do not need to reconnect to the target database.

SQL DDL Commands in the Designer

When the Designer generates the SQL code, it uses generic SQL, not the platform-specific version of the DDL code. The Designer passes these instructions to the ODBC Driver Manager, which converts the standard version of SQL into platform-specific commands. Do not try to run these SQL files through a different utility or use the syntax as an example of the native DDL syntax.

Dropping and Re-creating Indexes

After you insert significant amounts of data into a target, you normally need to drop and re-create indexes on that table to optimize query speed. You can drop and re-create indexes by either of the following methods:

- **Using pre- and post-session SQL commands.** The preferred method for dropping and re-creating indexes is to define a pre-session SQL statement in the Pre SQL property that drops indexes before loading data to the target. Use the Post SQL property to re-create the indexes after loading data to the target. Define pre- and post-session SQL for relational targets in the mapping target properties or on the Mappings tab in the session properties.
- **Using the Designer.** The same dialog box you use to generate and execute DDL code for table creation can drop and re-create indexes. Every time you run a workflow that modifies the target table, launch the Designer and use this feature when you use this method.
- **Stored procedures.** You can also use stored procedures to drop and re-create indexes.

Re-creating Targets

If you modify a relational target definition, use the Designer to drop and re-create the corresponding target table.

Note: When you drop a target table, the Designer deletes the table from the database. If you want to keep the target data, back it up before you drop the table.

To re-create the target table:

1. In the Target Designer, modify the relational target definition and select it.
2. Click Targets > Generate/Execute SQL.

   In the dialog box, connect to the appropriate target database. Select the DROP options checked for the table and any indexes on the table.
3. Click Generate and Execute.

   The Designer drops and re-creates the table, including any indexes assigned to it.

Troubleshooting Targets

When I modified a target definition and used the Designer to run the SQL DDL code, I lost all the data in the target table.

When you modify a target definition, the Designer can drop and re-create the table. It cannot issue an ALTER TABLE command to change or add columns. If you need to modify the table, back up the data in a temporary table before you drop and re-create the table. Alternatively, you can issue the ALTER TABLE command, but be careful to match the target definition now stored in the repository.

When I connect to a database to import target definitions, I do not see the tables, views, or synonyms I want to import.

Make sure you entered the correct owner name when connecting to the database. By default, the owner name the Designer uses to identify sources and targets for import is the same as the database user name you used to connect to the database. You may need to enter a different owner name to see the targets you want to import.

Instead of showing me a target when I drag it into a workspace, the Designer prompts me to copy it or make a shortcut.

Each workbook represents metadata from a single folder. To edit metadata from a different folder, move the focus in the Navigator window to the second folder and click Open. The Designer then opens another workbook, representing metadata from the second folder.

When I open a target definition appearing in a mapping, I cannot edit it.

Create mappings in the Mapping Designer. Create and modify source and target definitions in the Source Analyzer and Target Designer. The Designer divides the process of adding source definitions, target definitions, and mappings into separate modes of the Designer, to help you keep these processes distinct. To modify a target definition, switch to the Target Designer.

When I try to run a workflow that includes targets I have designed, the session log tells me that one or more target tables do not exist.

When you are designing a target, you are adding a target definition to the repository. To actually create the target table, run the necessary SQL DDL code in the database where you want the target to appear.

I imported a target from a DB2 database and received an SQL0954C error message from the DB2 operating system.

If the value of the DB2 system variable APPLHEAPSZ is too small when you use the Designer to import targets from a DB2 database, the Designer reports an error accessing the repository. The Designer status bar shows the following message:

```
SQL Error:[IBM][CLI Driver][DB2]SQL0954C: Not enough storage is available in the
application heap to process the statement.
```

If you receive this error, increase the value of the APPLHEAPSZ variable for the DB2 operating system. APPLHEAPSZ is the application heap size, in 4KB pages, for each process using the database.

CHAPTER 5

Mappings

This chapter includes the following topics:

- Mappings Overview, 107
- Working with Mappings, 109
- Connecting Mapping Objects , 114
- Linking Ports, 115
- Propagating Port Attributes, 118
- Working with Sources in a Mapping, 123
- Working with Relational Sources in a Mapping, 124
- Working with Transformations in a Mapping, 125
- Working with Mapplets in a Mapping, 125
- Working with Targets in a Mapping, 125
- Creating Target Files by Transaction, 127
- Working with Relational Targets in a Mapping, 129
- Validating a Mapping, 133
- Using the Workflow Generation Wizard, 136
- Troubleshooting Mappings, 137

Mappings Overview

A mapping is a set of source and target definitions linked by transformation objects that define the rules for data transformation. Mappings represent the data flow between sources and targets. When the Integration Service runs a session, it uses the instructions configured in the mapping to read, transform, and write data.

Every mapping must contain the following components:

- **Source definition.** Describes the characteristics of a source table or file.
- **Transformation.** Modifies data before writing it to targets. Use different transformation objects to perform different functions.
- **Target definition.** Defines the target table or file.
- **Links.** Connect sources, targets, and transformations so the Integration Service can move the data as it transforms it.

A mapping can also contain one or more mapplets. A mapplet is a set of transformations that you build in the Mapplet Designer and can use in multiple mappings.

When you add an object to a mapping, you configure the properties according to the way you want the Integration Service to transform the data. You also connect the mapping objects according to the way you want the Integration Service to move the data. You connect the objects through ports.

The Mapping Designer displays objects in three different views:

- **Iconized.** Shows an icon of the object with the object name.
- **Normal.** Shows the columns in the ports tab and the input and output port indicators. You can connect objects that are in the normal view.
- **Edit.** Shows the object properties. You can switch between the different tabs and configure the object in this view.

Object Dependency

Some objects in a mapping are also stored as independent objects in the repository:

- Sources
- Targets
- Reusable transformations
- Mapplets

The mapping is dependent on these objects. When this metadata changes, the Designer and other PowerCenter Client applications track the effects of these changes on mappings. In these cases, you may find that mappings become invalid even though you do not edit the mapping. When a mapping becomes invalid, the Integration Service cannot run it properly, and the Workflow Manager invalidates the session.

The only objects in a mapping that are not stored as independent repository objects are the non-reusable transformations that you build within the mapping. These non-reusable transformations are stored within the mapping only.

Developing a Mapping

Use the following steps as a guideline when you develop a mapping:

1. **Verify that all source, target, and reusable objects are created.** Create source and target definitions. If you want to use mapplets, you must create them also. You can create reusable transformations in the Transformation Developer, or you can create them while you develop a mapping.
2. **Create the mapping.** Create a mapping by dragging a source, target, mapplet, or reusable transformation into the Mapping Designer workspace, or you can click Mappings > Create from the menu.
3. **Add sources and targets.** Add sources and targets to the mapping.
4. **Add transformations and transformation logic.** Add transformations to the mapping and build transformation logic into the transformation properties.
5. **Connect the mapping.** Connect the 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.** Validate the mapping to identify connection or transformation errors.
7. **Save the mapping.** When you save the mapping, the Designer validates it, identifying any errors. The Designer displays validation messages in the Output window. A mapping with errors is invalid, and you cannot run a session against it until you validate it.

PowerCenter also provides a tool that you can use to create a template for a PowerCenter mapping and generate multiple mappings from the template. Mapping Architect for Visio provides an Informatica stencil for the Microsoft Office Visio software that contains shapes representing PowerCenter mapping objects. You can use the mapping object shapes to draw the mapping template on the Visio drawing window.

Working with Mappings

You can complete the following tasks with mappings:

- **Create a mapping.** When you create a mapping, you save the mapping name in the repository. You can then develop and save the mapping.
- **Open a mapping.** You can open one mapping at a time in a folder.
- **Copy a mapping.** You can copy a mapping within the same folder or to another folder.
- **Copy a mapping segment.** You can copy segments of mappings and mapplets when you want to reuse a portion of the mapping logic.
- **Copy objects in a mapping.** You can copy one or more objects in a mapping and paste them into another mapping or mapplet in the same folder.
- **Export a mapping.** You can export a mapping to an XML file.
- **Import a mapping.** You can import a mapping from an XML file that you exported in the Designer.
- **Edit a mapping.** You can add, modify, or delete objects in a mapping.
- **Save a mapping.** When you save a mapping in the repository, the Designer performs mapping validation.
- **Debug a mapping.** Run the Debugger in the Mapping Designer to test mapping logic.
- **Delete a mapping.** Delete a mapping from the repository if you do not want to use it again.
- **View link paths to a port.** You can view link paths to a port in a mapping. You can view the forward path, the backward path, or both.
- **View source column dependencies.** You can view from which source columns a target column receives data.
- **Connect objects in a mapping.** You can connect objects in a mapping to define the flow of data from sources to targets.
- **Link ports.** You can connect mapping objects by linking ports manually or automatically by name or position.
- **Propagate port attributes.** You can propagate port attributes in a mapping. You can propagate attributes forward, backward, or in both directions.

Creating a Mapping

The first step in the process of moving data between sources and targets is to create a mapping in the Mapping Designer.

To create a mapping:

1. Open the Mapping Designer.
2. Click Mappings > Create, or drag a repository object into the workspace.
3. Enter a name for the new mapping and click OK.

   The naming convention for mappings is m_*MappingName*, such as m_ResearchProjects.

Opening a Mapping

To open a mapping, drag it from the Navigator into the Mapping Designer workspace. If you have a mapping in the same folder already open, the Designer prompts you to close it before continuing. Click OK to close the current mapping and open the other one.

You can open one mapping at a time in a folder. If you open more than one folder at a time, you can open a mapping in each folder.

Tip: To open a mapping, you can also right-click a mapping in the Navigator and choose Open.

Copying a Mapping

You can copy mappings with the Designer:

- Within a folder
- To a folder in the same repository
- To another repository

The Designer provides a Copy Wizard that lets you copy objects in the repository. When you copy a mapping, the Copy Wizard creates a copy of each component in the mapping, if the component does not already exist. If any of the mapping components already exist, the Copy Wizard prompts you to rename, replace, or reuse those components. However, if the object is a shortcut, or if the destination folder already contains a shortcut with the same name, you cannot replace the object. You can only rename or reuse the object. If a mapping contains sources with primary key-foreign key relationships to sources not used in the mapping, the Copy Wizard prompts you to copy the related source.

Copying Mapping Segments

You can copy segments of mappings and mapplets when you want to reuse a portion of the mapping logic. A segment consists of one or more objects in a mapping or mapplet. A segment can include a source, target, transformation, mapplet, or shortcut. To copy mapping segments, select and copy the segments from the mapping designer and paste them into a target mapping or an empty mapping or mapplet workspace. You can copy segments across folders or repositories.

To copy a segment of a mapping or mapplet:

1. Open a mapping or mapplet.
2. Select a segment by highlighting each object you want to copy.

   You can select multiple objects. You can also select segments by dragging the pointer in a rectangle around objects in the workspace.
3. Click Edit > Copy or press Ctrl+C to copy the segment to the clipboard.
4. Open a target mapping or mapplet. You can also paste the segment into an empty workspace.
5. Click Edit > Paste or press Ctrl+V.

The Designer prompts you to rename, reuse, or replace objects with conflicts.

Using the Copy As Command

If you make changes to a mapping, but you do not want to overwrite the original mapping, you can make a copy of the changed mapping by clicking Mappings > Copy As. When you use Copy As, the copy of the mapping contains the changes and the original mapping does not reflect these changes.

You can only use Copy As within the same folder. When you use Copy As, the mapping must be open in the workspace.

To use the Copy As command to copy a mapping:

1. Open a mapping in the Mapping Designer workspace.
2. Click Mappings > Copy As.

3. Enter the new mapping name.
4. Click OK.

   You cannot use Copy As to copy shortcuts.

Copying Mapping Objects

The Designer lets you copy one or more objects in a mapping. You can paste the objects you copy into any other mapping or mapplet in the same folder. You might want to copy objects from a mapping and paste them into another mapping or mapplet to reuse transformation logic you created.

Exporting and Importing a Mapping

You export a mapping to an XML file and import a mapping from an XML file through the Designer. You might want to use the export and import feature to copy a mapping to the same repository, a connected repository, or a repository to which you cannot connect.

Editing a Mapping

After you create a mapping, you can edit it by adding, modifying, or deleting objects. Objects include source definitions, target definitions, mapplets, and transformations. Before the Designer deletes objects in the mapping, it displays the list of objects to delete. The Designer displays a validation message in the Output window when you save a mapping.

To see what sessions or shortcuts may be affected by changes you make to a mapping, select the mapping in the Navigator, right-click, and select View Dependencies. Or, click Mappings > View Dependencies.

Reverting to a Previously Saved Mapping

While editing a mapping, you can revert to a previously-saved mapping, undoing changes you have entered since the last save. To do this, click Edit > Revert to Saved. When you click Yes, the Designer removes all changes entered since the last time you saved the mapping.

Renaming and Adding Comments to a Mapping

You can rename, add comments, or specify links to business documentation for a mapping at any time. Adding comments or business documentation links is an easy way to document the purpose of a mapping. The Repository Manager and MX views include these comments to help you analyze the metadata.

To rename or add comments to a mapping:

1. Open a mapping in the Mapping Designer and click Mappings > Edit.
2. In the Edit Mapping dialog box, enter a new name for the mapping.
3. Add a description of the mapping in the comments box.

   You can enter up to 2,000 characters.
4. Click OK.

Invalidating Sessions

When you edit and save a mapping, some changes cause the session to be invalid even though the mapping remains valid. The Integration Service does not run invalid sessions. If you edit a mapping, the Designer invalidates sessions when you perform the following actions:

- Add or remove sources or targets.
- Remove mapplets or transformations.
- Replace a source, target, mapplet, or transformation when importing or copying objects.
- Add or remove Source Qualifiers or COBOL Normalizers, or change the list of associated sources for these transformations.
- Add or remove a Joiner or Update Strategy transformation.
- Add or remove transformations from a mapplet in the mapping.
- Change the database type for a source or target.

Debugging a Mapping

You can debug a valid mapping to gain troubleshooting information about data and error conditions. To debug a mapping, you configure and run the Debugger from within the Mapping Designer. When you run the Debugger, it pauses at breakpoints and you can view and edit transformation output data.

Deleting a Mapping

You may delete mappings that you no longer use. When you delete a mapping, you do not delete any sources, targets, mapplets, or reusable transformations defined outside the mapping.

Note: If you enable version control, a deleted mapping remains checked out until you check it in. To check in a deleted mapping, click Versioning > Find Checkouts. Select the deleted mapping and click Tools > Check In.

You can delete a mapping from the Navigator window, or you can delete the mapping currently displayed in the Mapping Designer workspace.

- To delete a mapping from the Navigator window, select the mapping and press the Delete key, or click Edit > Delete.
- To delete a mapping currently displayed in the Mapping Designer workspace, click Mappings > Delete.

Viewing Link Paths to a Port

When editing a mapping, you may want to view the forward and backward link paths to a particular port. Link paths allow you to see the flow of data from a column in a source, through ports in transformations, to a port in the target.

To view link paths, highlight a port and right-click it. Select the Select Link Path option. You can choose to view either the forward path, backward path, or both. The Designer displays all the connectors in the link path you select.

When displaying both link paths, the Designer traces the flow of data from one column in the source, in and out of each transformation, and into a single port in the target. For unconnected transformations, the Designer does not display a link path. For connected Lookup transformations, the Designer shows each output port dependent upon the input ports involved in the lookup condition. For Custom transformations, the Designer shows that an output port depends on all input ports by default. However, if you define port relationships in a Custom transformation, the Designer shows the dependent ports you define.

Note: You can configure the color the Designer uses to display connectors in a link path. When configuring the format options, choose the Link Selection option.

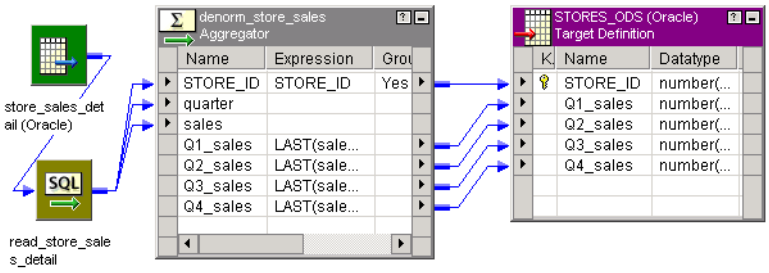
Viewing Source Column Dependencies

When editing a mapping, you can view source column dependencies for a target column. Viewing source column dependencies lets you see from which source columns a target column receives data.

To view column dependencies, right-click a target column in a mapping and choose Show Field Dependencies. The Designer displays the Field Dependencies dialog box which lists all source columns connected to the target column.

When you define a port expression that performs a calculation using multiple source columns, and then connect that port to a target column, the Field Dependencies dialog box lists all source columns you use in the expression.

For example, you have the following mapping:



Define the following expression in the Q3_sales port in the Aggregator transformation:

```
LAST(sales, quarter = 3)
```

Right-click the Q3_sales target column and choose Show Dependencies.

The following figure shows the Field Dependencies dialog box that appears:

Connecting Mapping Objects

After you add and configure source, target, and transformation objects in a mapping, complete the mapping by connecting the mapping objects. You connect mapping objects through the ports. Data passes into and out of a transformation through the following ports:

- **Input ports.** Receive data.
- **Output ports.** Pass data.
- **Input/output ports.** Receive data and pass it unchanged.

Every source instance, target instance, mapplet, and transformation contains a collection of ports. Each port represents a column of data:

- Sources provide data, so they contain only output ports.
- Targets receive data, so they contain only input ports.
- Mapplets contain only input ports and output ports.
- Transformations contain a mix of input, output, and input/output ports, depending on the transformation and its application.

To connect ports, you drag between ports in different mapping objects. The Designer validates the connection and creates the connection only when the connection meets link validation and concatenation requirements.

You can leave ports unconnected. The Integration Service ignores unconnected ports.

Options for Linking Ports

When you link transformations, you can link with one of the following options:

- **One to one.** Link one transformation or output group to one transformation, input group, or target only.
- **One to many.**
  - Link one port to multiple transformations, input groups, or targets.
  - Link multiple ports in one transformation or output group to multiple transformations, input groups, or targets.
- **Many to one.** Link many transformations to one transformation, input group, or target.

Linking One to Many

When you want to use the same data for different purposes, you can link the port providing that data to multiple ports in the mapping. For example, salary information might be used to calculate the average salary in a department through the Aggregator transformation, while the same information might appear in an Expression transformation configured to calculate monthly pay of each employee.

Linking Many to One

Frequently, you need to combine data from multiple transformations into a single transformation or target. For example, you might need to combine data from multiple Aggregator and Expression transformations into a single fact table.

Rules and Guidelines for Connecting Mapping Objects

Use the following rules and guidelines when you connect mapping objects:

- If the Designer detects an error when you try to link ports between two mapping objects, it displays a symbol indicating that you cannot link the ports.
- Follow logic of data flow in the mapping. You can link the following types of ports:
  - The receiving port must be an input or input/output port.
  - The originating port must be an output or input/output port.
  - You cannot link input ports to input ports or output ports to output ports.
- You must link at least one port of an input group to an upstream transformation.
- You must link at least one port of an output group to a downstream transformation.
- You can link ports from one active transformation or one output group of an active transformation to an input group of another transformation.
- You cannot connect an active transformation and a passive transformation to the same downstream transformation or transformation input group.
- You cannot connect more than one active transformation to the same downstream transformation or transformation input group.
- You can connect any number of passive transformations to the same downstream transformation, transformation input group, or target.
- You can link ports from two output groups in the same transformation to one Joiner transformation configured for sorted data if the data from both output groups is sorted.
- You can only link ports with compatible datatypes. The Designer verifies that it can map between the two datatypes before linking them. The Integration Service cannot transform data between ports with incompatible datatypes. While the datatypes do not have to be identical, they do have to be compatible, such as Char and Varchar.
- You must connect a source definition to a source qualifier only. You then link the source qualifier to targets or other transformations.
- You can link columns to a target definition in a mapping, but you cannot copy columns into a target definition in a mapping. Use the Target Designer to add columns to a target definition.
- The Designer marks some mappings invalid if the mapping violates data flow validation.

Linking Ports

You can manually link ports, or you can automatically link ports between transformations. When you link ports automatically, you can link by position or by name. When you link ports automatically by name, you can specify a prefix or suffix by which to link the ports. Use prefixes or suffixes to indicate where ports occur in a mapping. For example, a mapping includes a port in the Source Qualifier called Name and a corresponding port in a Filter transformation called FilName. ‘Fil’ is the prefix you specify when you automatically link ports between the Source Qualifier and Filter transformation.

Manually Linking Ports

To link ports manually, click Layout > Link Columns. When you drag from one port to another, the Designer creates a connection. When you drag a port into an empty port, the Designer copies the port and creates a connection.

You can also link multiple ports at the same time. Use the Ctrl or Shift key to select a range of ports to link to another transformation. The Designer links the ports, beginning with the top pair. It links all ports that meet the validation requirements.

Linking Ports by Position

When you link by position, the Designer links the first output port to the first input port, the second output port to the second input port, and so forth. Use this option when you create transformations with related ports in the same order. Use the following options to link by position:

- **Autolink dialog box.** To automatically link ports using the Autolink dialog box, click Layout > Autolink.
- **Autolink command.** To link ports by selecting the ports in the workspace, click Layout > Autolink by Position.

Linking Ports using the Autolink Dialog Box

To link ports by position using the Autolink dialog box:

1. Click Layout > Autolink.
2. Select the transformations and targets.

   You can select multiple transformations in the To Transformations list to link one transformation to multiple transformations. For objects that contain multiple input groups such as Custom transformations or XML targets, select the group name from the To Transformation list.

   You can also select the transformations in the workspace in the order you want to link them. Then, in the Autolink dialog box, select each From Transformation and the Designer selects the To Transformation based on the order in which you selected it in the workspace. Click Apply and then select the next From Transformation and click Apply.
3. Select Position.
4. Click OK.

   The Designer links the first output port to the first input port, the second output port to the second input port, and so forth.

Linking Ports Using the Autolink by Position Command

To link ports by position using the Autolink by Position command:

1. Click Layout > Autolink by Position.
2. Select the mapping object that you want to link by position and drag the selected ports to another mapping object.

   The Designer selects all of the ports in an object. To select only specific ports, use the Autolink dialog box.
3. The Designer links the first output port to the first input port, the second output port to the second input port, and so forth.
4. When you finish linking ports, click Layout > Link Columns.

Linking Ports by Name

You can link ports by name in the Designer. The Designer adds links between input and output ports that have the same name. Linking by name is not case sensitive. Link by name when you use the same port names across transformations. The Designer can link ports based on prefixes and suffixes that you define. Link by name and prefix or suffix when you use prefixes or suffixes in port names to distinguish where they occur in the mapping or mapplet. Use the following options to link by name:

- **Autolink dialog box.** To automatically link ports by name, link ports by name and prefix, and link ports by name and suffix using the Autolink dialog box, click Layout > Autolink.
- **Autolink command.** To link objects by selecting the ports in the workspace, click Layout > Autolink by Name.

Linking Ports by Name using the Autolink Dialog Box

To link ports by name using the Autolink dialog box:

1. Click Layout > Autolink.
2. Select the transformations and targets.

   You can select multiple transformations in the To Transformations list to link one transformation to multiple transformations. For objects that contain multiple input groups such as Custom transformations or XML targets, select the group name from the To Transformation list.
3. Select Name.
4. Click OK.

Linking Ports by Name and Prefix/Suffix using the Autolink Dialog Box

To link ports by name and prefix or suffix using the Autolink dialog box:

1. Click Layout > Autolink.
2. Select the transformations and targets.

   You can select multiple transformations in the To Transformations list to link one transformation to multiple transformations. For objects that contain input groups such as Custom transformations or XML targets, select the group name from the To Transformation list.
3. Select Name.
4. Click More to view the options for entering prefixes and suffixes.
5. In From Transformation, enter the prefix or suffix used in the ports from which you are linking.
6. In To Transformation, enter the prefix or suffix used in the ports to which you are linking.

   In this example, the Designer links ports in SQ_CUSTOMERS to ports in FIL_STATE where the port name in FIL_STATE is either the same as SQ_CUSTOMERS, or is the same and is preceded by the prefix “F_”.
7. Click OK.

Linking Ports by Name Using the Autolink by Name Command

To link ports by name using the Autolink by Name command:

1. Click Layout > Autolink by Name.
2. Select the mapping object that you want to link by name and drag the selected ports to another mapping object.

The Designer selects all of the ports in an object. To select only specific ports, use the Autolink dialog box.

3. The Designer adds links between input and output ports that have the same name, regardless of name case.
4. When you are finished linking ports, click Layout > Link Columns.

Propagating Port Attributes

When you edit a port name in a transformation, by default, the Designer propagates references to that port in the expressions, conditions, and other ports in that transformation. You can also propagate changed attributes throughout the mapping.

The Designer propagates ports, expressions, and conditions based on the following factors:

- **The direction that you propagate.** You can propagate changes forward, backward, or in both directions.
- **The attributes you choose to propagate.** You can propagate port name, datatype, precision, scale, and description.
- **The type of dependencies.** You can propagate changes to dependencies along a link path or to implicit dependencies within a transformation.

Understanding Dependency Types

When you propagate port attributes, the Designer can update the following dependencies:

- **Link path dependencies.** A link path dependency is a dependency between a propagated port and the ports in its link path. When you propagate link path dependencies, the Designer also performs default updates to references in expressions and conditions that depend on ports in the link path.
- **Implicit dependencies.** An implicit dependency is a dependency within a transformation between two ports based on an expression or condition.

  For example, when you change the datatype of a port that is used in a lookup condition, the Designer propagates the datatype change to the other port dependent on the condition.

Propagating Dependencies in a Link Path

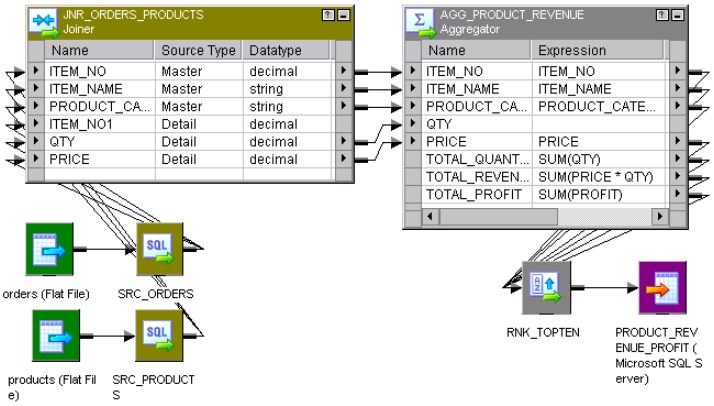
When you propagate dependencies in a link path, the Designer updates all the input and input/output ports in its forward link path and all the output and input/output ports in its backward link path. The Designer performs the following updates:

- Updates the port name, datatype, precision, scale, and description for all ports in the link path of the propagated port.
- Updates all expressions or conditions that reference the propagated port with the changed port name.
- Updates the associated port property in a dynamic Lookup transformation if the associated port name changes.
- Updates the port name of Custom transformation port dependencies.

Note: When you propagate a port name, the Designer appends “1” to the port name if a port with the same name exists in the transformation.

Example

In the following mapping, the QTY port in the Joiner transformation links to the QTY port in the Aggregator transformation. The Aggregator transformation references the QTY port in the expressions for TOTAL_QUANTITY and TOTAL_REVENUE:



You make the following changes to the QTY port in the Joiner transformation:

- You change the port name QTY to QUANTITY.
- You change the datatype from Decimal to Integer.

When you propagate forward the attributes, the Designer updates the following dependencies in the Aggregator transformation:

- The Designer updates QTY port name to QUANTITY.
- The Designer updates the reference to QTY port name in the expressions for the TOTAL_QUANTITY and TOTAL_REVENUE ports change to QUANTITY.
- The Designer updates the datatype of QTY port name to Integer.

Propagating Implicit Dependencies

You can propagate datatype, precision, scale, and description to ports with implicit dependencies. When you click Options in the Propagate Ports dialog box, you can choose to parse conditions and expressions to identify the implicit dependencies of the propagated port. All ports with implicit dependencies are output or input/output ports.

When you include conditions the Designer updates the following dependencies:

- Link path dependencies
- Output ports used in the same lookup condition as the propagated port
- Associated ports in dynamic Lookup transformations that are associated with the propagated port
- Output ports used by a Custom transformation to define a port relationship with one or more input or input/output ports
- Master ports used in the same join condition as the detail port

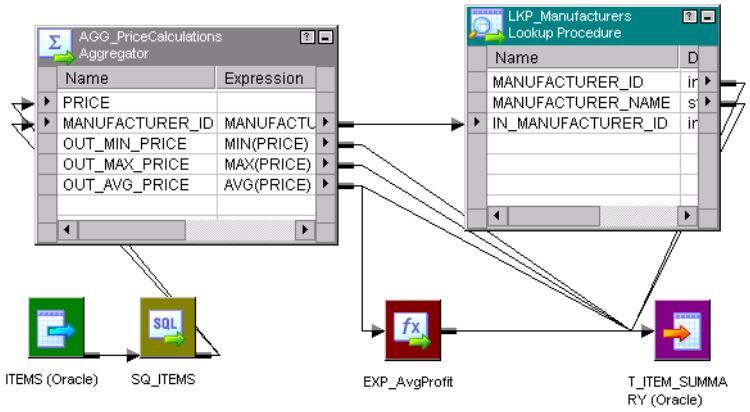
When you include expressions, the Designer updates the following dependencies:

- Link path dependencies
- Output ports containing an expression that uses the propagated port

The Designer does not propagate to implicit dependencies within the same transformation. You must propagate the changed attributes from another transformation. For example, when you change the datatype of a port that is used in a lookup condition and propagate that change from the Lookup transformation, the Designer does not propagate the change to the other port dependent on the condition in the same Lookup transformation.

Example

You have the following mapping:



The MANUFACTURER_ID port in the Aggregator transformation links to the IN_MANUFACTURER_ID port in the Lookup transformation. The Lookup transformation uses the following lookup condition:

```
MANUFACTURER_ID = IN_MANUFACTURER_ID
```

You change the datatype of the MANUFACTURER_ID port from integer to decimal in the Aggregator transformation. You choose to parse conditions to infer dependencies, and then propagate the datatype change. The Designer performs the following tasks:

- **Updates link path dependencies.** The Designer updates the ports in the link path, changing the datatype of the IN_MANUFACTURER_ID port in the Lookup transformation to decimal.
- **Identifies dependent ports.** The Designer parses the lookup condition and identifies the MANUFACTURER_ID port in the Lookup transformation as a dependent port.
- **Updates implicit dependencies.** The Designer changes the datatype of the MANUFACTURER_ID port in the Lookup transformation to decimal.

Propagated Attributes by Transformation

The following table describes the dependencies and attributes the Designer propagates for each transformation:

Transformation	Dependency	Propagated Attribute
Aggregator	- Ports in link path - Expression - Implicit dependencies	- Port name, datatype, precision, scale, description - Port name - Datatype, precision, scale
Application Source Qualifier	- Ports in link path	- Port name, datatype, precision, scale, description

Transformation	Dependency	Propagated Attribute
Custom	- Ports in link path - Port dependency - Implicit dependencies	- Port name, datatype, precision, scale, description - Port name - Datatype, precision, scale
Expression	- Ports in link path - Expression - Implicit dependencies	- Port name, datatype, precision, scale, description - Port name - Datatype, precision, scale, description
External Procedure	- Ports in link path	- Port name, datatype, precision, scale, description
Filter	- Ports in link path - Condition	- Port name, datatype, precision, scale, description - Port name
Input	- Ports in link path	- Port name, datatype, precision, scale, description
Joiner	- Ports in link path - Condition - Implicit dependencies	- Port name, datatype, precision, scale, description - Port name - Datatype, precision, scale, description
Lookup	- Ports in link path - Condition - Associated ports (dynamic Lookup) - Implicit dependencies	- Port name, datatype, precision, scale, description - Port name - Port name - Datatype, precision, scale, description
Normalizer in the Pipeline	- Ports in link path	- Port name, datatype, precision, scale, description
Normalizer Source Qualifier	- n/a	- none
Output	- Ports in link path	- Port name, datatype, precision, scale, description
Rank	- Ports in link path - Expression - Implicit dependencies	- Port name, datatype, precision, scale, description - Port name - Datatype, precision, scale, description
Router	- Ports in link path - Condition	- Port name, datatype, precision, scale, description - Port name
SDK Source Qualifier	- n/a	- none
Sequence Generator	- n/a	- none
Sorter	- Ports in link path	- Port name, datatype, precision, scale, description
Source Qualifier	- Ports in link path	- Port name, datatype, precision, scale, description
SQL	- Ports in link path	- Port name, datatype, precision, scale, description
Stored Procedure	- Ports in link path	- Port name, datatype, precision, scale, description

Transformation	Dependency	Propagated Attribute
Transaction Control	- Ports in link path - Condition	- Port name, datatype, precision, scale, description - Port name
Union	- Ports in link path - Implicit dependencies	- Port name, datatype, precision, scale, description - Datatype, precision, and scale, description
Update Strategy	- Ports in link path - Expression - Implicit dependencies	- Port name, datatype, precision, scale, description - Port name - Datatype, precision, scale, description
XML Generator	- Ports in link path	- Port name, datatype, precision, scale, description
XML Parser	- Ports in link path	- Port name, datatype, precision, scale, description
XML Source Qualifier	- n/a	- none

The Designer does not propagate changes to the following mapping objects:

- Unconnected transformations
- Reusable transformations
- Mapplets
- Source and target instances
- SDK Source Qualifier

Rules and Guidelines for Propagating Ports and Attributes

Use the following rules and guidelines when you propagate port attributes:

- The Designer does not propagate to implicit dependencies within the same transformation.
- When you propagate a port description, the Designer overwrites the description for the port in the other transformations in the mapping.
- When you propagate backward along the link path, verify that the change does not cause the Integration Service to fail the session. For example, if you propagate changes to a source qualifier, the Integration Service might generate invalid SQL when it runs the session. If you change the port name “CUST_ID” to “CUSTOMER_ID,” the Integration Service might generate SQL to select the wrong column name if the source table uses “CUST_ID.”
- When you propagate port attributes, verify that the change does not cause the Designer to invalidate the mapping. For example, when you change the datatype of a port from integer to string and propagate the datatype to other transformations, the Designer invalidates the mapping if a calculation uses one of the changed ports. Validate the mapping after you propagate ports. If the Designer invalidates the mapping, click Edit > Revert to Saved to revert to the last saved version of the mapping.
- When you propagate multiple ports, and an expression or condition depends on more than one propagated port, the Designer does not propagate attributes to implicit dependencies if the attributes do not match.

  For example, you have the following expression in an Expression transformation:

  ```
  Item_desc_out = Substr(ITEM_NAME, 0, 6) || Substr(ITEM_DESC, 0, 6)
  ```

  The precision of Item_desc_out is 12, ITEM_NAME is 10, and ITEM_DESC is 10. You change the precision of ITEM_DESC to 15. You select parse expressions to infer dependencies and propagate the port

attributes of ITEM_NAME and ITEM_DESC. The Designer does not update the precision of the Item_desc_out port in the Expression transformation since the ITEM_NAME and ITEM_DESC ports have different precisions.

Steps to Propagate Port Attributes

Complete the following steps to propagate port attributes:

1. Open a mapping in the Mapping Designer, and select one or more ports to propagate.
2. Click Mappings > Propagate Attributes, and choose Propagate Attributes. Or, right-click the port and choose Propagate Attributes.

   The Designer displays the Propagate Port Attributes dialog box.

   After you open the Propagate Port Attributes dialog box, you can select another port and propagate its attributes.
3. Select the direction and attributes you want to propagate.
4. Optionally, to infer dependencies, click Options.

   The following table describes the options on the Propagate Port Attributes dialog box:

Option	Description
Preview	Displays the links to the affected ports in green and the unaffected ports in red.
Propagate	Propagates the port attributes according to the options you specify in the dialog box.
Direction	Instructs the Designer to propagate attributes forward, backward, or in both directions.
Attributes to Propagate	Specifies the attributes that you want to propagate. The attributes include name, datatype, precision, scale, and description.
Options	Select to read conditions or expressions and propagate attributes to implicit dependencies. The Designer disables these options when you propagate the port name. These options are clear by default. When you select one of these options and click Preview, the Designer highlights the link paths to the dependent ports.

5. Click Preview to view the affected ports.

   The Designer displays the links to the affected ports in green and the unaffected ports in red.
6. Click Propagate.

   When you propagate port attributes, the Output window displays the propagated attributes and the affected ports.
7. Click Close.

Working with Sources in a Mapping

When you create a mapping, you must add one or more source definitions to it. When you drag a source into the Mapping Designer workspace, you add an instance of the source definition to the mapping.

For relational sources, you can edit the source definition and override the default source table name.

Every mapping requires at least one of the following source qualifiers that determines how the Integration Service reads the source data:

- **Source Qualifier transformation.** Represents data read from relational and flat file sources.
- **Normalizer transformation.** Represents data read from COBOL sources.
- **Application Source Qualifier transformation.** Represents data read from application sources.
- **Application Multi-Group Source Qualifier transformation.** Represents data read from multi-group application sources.
- **XML Source Qualifier transformation.** Represents data read from XML sources.

You can let the Designer create the source qualifier by default. Each time you drag a source instance into a mapping, the Designer adds a source qualifier and connects it to the source. Use the automatic source qualifier creation when you want to create one source qualifier for each source in the mapping. You can disable the automatic creation when you want to join data from different relational sources. You can then manually create and connect it to the source.

When you edit the source in the Source Analyzer, all instances of the source in mappings inherit the changes. Some changes might invalidate the mappings using the source.

However, you specify some properties for every source instance in a mapping. Double-click the source instance in the Mapping Designer and click the Properties tab. For relational sources, you can specify the table owner name. For flat file sources, you can specify the default datetime format, thousands separator, and decimal separator.

Note: When you add a source definition with some special characters in the table name to a mapping, the Designer replaces the character with an underscore in the source instance name for the mapping.

Working with Relational Sources in a Mapping

When you add a relational source to a mapping, you can view the table owner name and override the source table name. You can view this information on the Properties tab of a relational source instance in the Mapping Designer.

The table owner name displays the owner name of the source table in the database. For some databases, such as DB2, tables can have different owners. You can override the table owner name for each source instance in the session properties.

You can override the source table name for relational source instances on the Properties tab of the source instance. Override the source table name when you use a single mapping to read data from different source tables. Enter a table name in the source table name. You can also enter a parameter or variable. You can use mapping parameters, mapping variables, session parameters, workflow variables, or worklet variables in the source table name. For example, you can use a session parameter, $ParamSrcTable, as the source table name, and set $ParamSrcTable to the source table name in the parameter file.

Note: If you override the source table name on the Properties tab of the source instance, and you override the source table name using an SQL query, the Integration Service uses the source table name defined in the SQL query.

To override a source table name:

1. In the Designer, open the Mapping Designer tool.
2. Double-click the title bar of a relational source instance in the mapping.

3. On the Properties tab, enter the source table name. You can also enter a parameter or variable in the Source Table Name field.

   If you use a user-defined mapping parameter, mapping variable, workflow variable, or worklet variable, you must declare the parameter or variable.
4. Click OK.
5. If you use a parameter or variable for the source table name, define the parameter or variable in the appropriate section of the parameter file.

Working with Transformations in a Mapping

A transformation is a repository object that generates, modifies, or passes data. You configure logic in a transformation that the Integration Service uses to transform data. The Designer provides a set of transformations that perform specific functions. For example, an Aggregator transformation performs calculations on groups of data.

You can create transformations to use once in a mapping, or you can create reusable transformations to use in multiple mappings. When you add a reusable transformation to a mapping, you add an instance of the transformation. When you edit the reusable transformation in the Transformation Developer, all instances of the transformation in mappings inherit the changes. Some changes might invalidate the mappings using the reusable transformation.

Working with Mapplets in a Mapping

You can build mapplets in the Mapplet Designer when you want to use a standardized set of transformation logic in several mappings. When you use a mapplet in a mapping, the Designer creates an instance of the mapplet. The mapplet instance displays only the ports from the Input and Output transformations. These transformations display as groups. You can enter comments for the instance of the mapplet in the mapping, but you cannot edit other mapplet properties.

You can expand the mapplet in the Mapping Designer by selecting it and clicking Mappings > Expand from the menu. This expands the mapplet within the mapping for view. You can open or iconize all the transformations in the mapplet and mapping, but you cannot edit any of the properties.

When you edit the mapplet in the Mapplet Designer, all instances of the mapplet in mappings inherit the changes. Some changes might invalidate the mappings using the mapplet.

Working with Targets in a Mapping

When you create a mapping, you must add one or more target definitions to it. When you drag a target definition into the Mapping Designer workspace, you add an instance of the target definition.

When you add targets to a mapping, you can include different types of targets. You can include targets of the same database type, but different database connections. You can also include both relational and flat file targets in the same mapping.

When you edit the target in the Target Designer, all instances of the target in mappings inherit the changes. Some changes might invalidate the mappings using the target.

Note: When you add a target definition with some special characters in the table name to a mapping, the Designer replaces the character with an underscore in the target instance name for the mapping.

You can configure properties for relational, file, and XML targets in a mapping.

Configuring Relational Targets in a Mapping

For relational targets, you can configure the following properties within a mapping:

- **Reject truncated and overflow data.** Select this option in the target instance Properties tab when you want the Integration Service to write truncated data to the reject file.
- **Update override.** Override the default UPDATE statement using the SQL Editor in the target instance Properties tab.
- **Table name prefix.** Specify the owner of the target tables in the target instance Properties tab.
- **Pre- and post-session SQL.** Enter pre-session SQL commands for a target instance in a mapping to execute commands against the target database before the Integration Service reads the source. Enter post-session SQL commands to execute commands against the target database after the Integration Service writes to the target.
- **Target table name.** You can override the default target table name.

Related Topics:

- "Working with Relational Targets in a Mapping" on page 129

Configuring Flat File Targets in a Mapping

For flat file targets, you can configure the following properties within a mapping:

- **Datetime format.** Define the default datetime format to use for datetime values.
- **Thousands separator.** Define the default thousands separator to use for numeric values.
- **Decimal separator.** Define the default decimal separator to use for numeric values.

Related Topics:

- "Defining Default Datetime and Numeric Formats" on page 84

Configuring XML Targets in a Mapping

For XML targets, you can change the root element. However, if you change the root, you affect the target XML structure and you can invalidate the mapping.

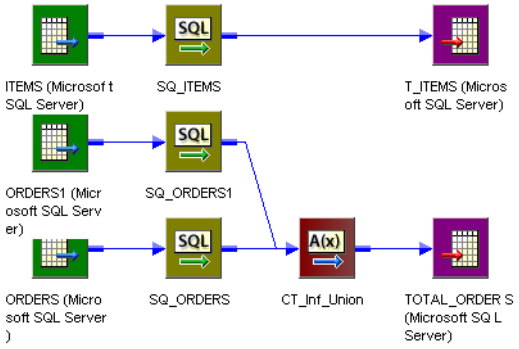
Setting the Target Load Order

You can configure the target load order for a mapping containing any type of target definition. In the Designer, you can set the order in which the Integration Service sends rows to targets in different target load order groups in a mapping. A target load order group is the collection of source qualifiers, transformations, and targets linked together in a mapping. You can set the target load order if you want to maintain referential integrity when inserting, deleting, or updating tables that have the primary key and foreign key constraints.

The Integration Service reads sources in a target load order group concurrently, and it processes target load order groups sequentially.

To specify the order in which the Integration Service sends data to targets, create one source qualifier for each target within a mapping. To set the target load order, you then determine in which order the Integration Service reads each source in the mapping.

The following figure shows two target load order groups in one mapping:



In this mapping, the first target load order group includes ITEMS, SQ_ITEMS, and T_ITEMS. The second target load order group includes all other objects in the mapping, including the TOTAL_ORDERS target. The Integration Service processes the first target load order group, and then the second target load order group.

When it processes the second target load order group, it reads data from both sources at the same time.

To set the target load order:

1. Create a mapping that contains multiple target load order groups.
2. Click Mappings > Target Load Plan.

   The Target Load Plan dialog box lists all Source Qualifier transformations in the mapping and the targets that receive data from each source qualifier.
3. Select a source qualifier from the list.
4. Click the Up and Down buttons to move the source qualifier within the load order.
5. Repeat steps 3 to 4 for other source qualifiers you want to reorder.
6. Click OK.

Creating Target Files by Transaction

You can generate a separate output file each time the Integration Service starts a new transaction. You can dynamically name each target flat file.

To generate a separate output file for each transaction, add a FileName port to the flat file target definition. When you connect the FileName port in the mapping, the Integration Service creates a separate target file at each commit. The Integration Service names the output file based on the FileName port value from the first row in each transaction. By default, the Integration Service writes output files to $PMTargetFileDir.

Configuring the Target

You add the FileName column to a flat file target definition in the Target Designer.

To add a FileName column:

1. Open the flat file target definition in the Target Designer.
2. Click the Columns tab.
3. Click Add FileName Column.

The Designer creates a string port called FileName. You can change the port precision.

Configuring the Mapping

You can generate output files from source-based or user-defined commits. You can use a source-based commit to commit data to a target file based on the number of rows from a source. For example, you might want to create a separate output file for each 1,000 rows of data. You can configure a user-defined commit when the mapping contains an effective transaction generator. For example, you might want to create a separate output file for each city.

In a mapping, connect the target FileName port to a transformation port that will contain a unique value at the start of each transaction. You can create an expression in the transformation to generate unique file names and pass them to the FileName port.

Running the Session

When you configure a FileName port, the Integration Service overrides the Output Filename session attribute with the value in the FileName column. If the FileName column value is NULL for a row, the row is an error and the Integration Service does not process it. If the FileName column is NULL after a transaction boundary, the Integration Service names the output file with the default output file name.

The FileName column must contain a unique value for each transaction. If the FileName column value does not change between transactions, the Integration Service overwrites the flat file target.

If you do not connect the FileName port in the mapping, the Integration Service generates one target file and uses the output file name configured in the session.

Rules and Guidelines for Creating Target Files by Transaction

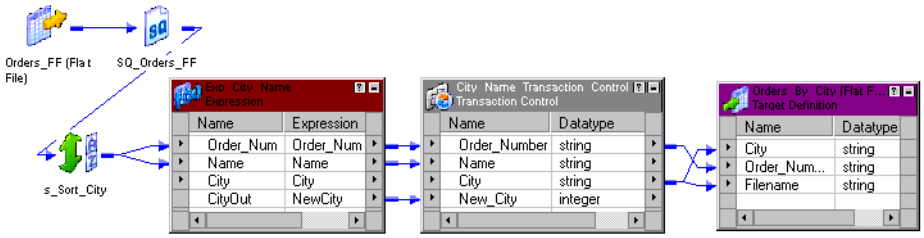
Use the following rules and guidelines when you create FileName columns:

- You can use a FileName column with flat file targets.
- You can add one FileName column to the flat file target definition.
- You can use a FileName column with data from real-time sources.
- A session fails if you use a FileName column with merge files, file lists, or FTP targets.
- If you pass the same file name to targets in multiple partitions, you might get unexpected results.
- When a transformation drops incoming transaction boundaries and does not generate commits, the Integration Service writes all rows into the same output file. The output file name is the initial value of the FileName port.

Example

A source contains orders for several cities. You want to write orders to separate output files based on the city.

The following figure shows a mapping to process the orders:



In addition to a source and Source Qualifier, the mapping has the following objects:

- **Sorter transformation**. Sorts the source data by city.
- **Expression transformation.** Determines when a new city appears in a row and passes an integer to the New City port of the Transaction Control transformation. Passes 0 by default and passes 1 when the row contains a new city.
- **Transaction Control transformation.** Evaluates the New City value from the Expression transformation. When New City is 1, the Transaction Control transformation commits all orders in the transaction to the target. The Transaction Control transformation passes the city and order number to the flat file target. It also passes the city to the FileName column in the target.
- **Flat file target.** Writes a new flat file for each transaction. The Integration Service names each target with the value of FileName.

The Integration Services passes a transaction for each city to the target. For this example, the data contains the following cities and order numbers:

```
Brisbane, 100
San Francisco, 101
San Francisco, 104
San Francisco, 105
San Francisco,107
Tiburon, 102
Tiburon, 106
Tiburon, 102
```

The Integration Service generates the following output files:

```
Brisbane
San Francisco
Tiburon
```

Working with Relational Targets in a Mapping

When you add a relational target to a mapping, you can configure the following properties:

- **Reject truncated and overflow data.** Select this option in the target instance Properties tab when you want the Integration Service to write truncated data to the reject file.
- **Update override.** Override the default UPDATE statement using the SQL Editor in the target instance Properties tab.
- **Table name prefix.** Specify the owner of the target tables in the target instance Properties tab.
- **Pre- and post-session SQL.** You can enter pre-session SQL commands for a target instance in a mapping to execute commands against the target database before the Integration Service reads the source. Enter post-session SQL commands to execute commands against the target database after the Integration Service writes to the target.

- **Target table name.** You can override the default target table name.

Note: You cannot configure these properties in the Target Designer.

Rejecting Truncated and Overflow Data

The Designer lets you convert data by passing it from port to port. Sometimes a conversion causes a numeric overflow (numeric data) or truncation (on character columns). For example, passing data from a Decimal (28, 2) to a Decimal (19, 2) port causes a numeric overflow. Likewise, if you pass data from a String(28) port to a String(10) port, the Integration Service truncates the strings to 10 characters. When a conversion causes an overflow, the Integration Service, by default, skips the row. The Integration Service does not write the data to the reject file. For strings, the Integration Service truncates the string and passes it to the next transformation.

The Designer provides an option to let you include all truncated and overflow data between the last transformation and target in the session reject file. If you select Reject Truncated/Overflow Rows, the Integration Service sends all truncated rows and any overflow rows to the session reject file or to the row error logs, depending on how you configure the session.

Configuring the Target Update Override

By default, the Integration Service updates target tables based on key values. However, you can override the default UPDATE statement for each target in a mapping. You might want to update the target based on non-key columns.

When the Integration Service executes SQL against a source, target, or lookup database, it searches the reserved words file stored in the Integration Service installation directory. It encloses matching reserved words in quotes. If you use target update override, you must manually put all reserved words in quotes.

For a mapping without an Update Strategy transformation or a Custom transformation with the update strategy property enabled, configure the session to mark source rows as update. The Target Update option only affects source rows marked as update. The Integration Service processes all rows marked as insert, delete, or reject normally. When you configure the session, mark source rows as data-driven. The Target Update Override only affects source rows marked as update by the Update Strategy or Custom transformation.

For example, a mapping passes the total sales for each salesperson to the T_SALES table.

The Designer generates the following default UPDATE statement for the target T_SALES:

```
UPDATE T_SALES SET EMP_NAME = :TU.EMP_NAME, DATE_SHIPPED = :TU.DATE_SHIPPED, TOTAL_SALES
= :TU.TOTAL_SALES WHERE EMP_ID = :TU.EMP_ID
```

Because the target ports must match the target column names, the update statement includes the keyword:TU to specify the ports in the target transformation. If you modify the UPDATE portion of the statement, be sure to use :TU to specify ports.

Overriding the WHERE Clause

You can override the WHERE clause to include non-key columns. For example, you might want to update records for employees named Mike Smith only. To do this, you edit the WHERE clause as follows:

```
UPDATE T_SALES SET DATE_SHIPPED = :TU.DATE_SHIPPED,

TOTAL_SALES = :TU.TOTAL_SALES WHERE :TU.EMP_NAME = EMP_NAME and

EMP_NAME = 'MIKE SMITH'
```

Rules and Guidelines for Configuring the Target Update Override

Use the following rules and guidelines when you enter target update queries:

- If you use target update override, you must manually put all database reserved words in quotes.
- You cannot override the default UPDATE statement if the target column name contains any of the following characters:

  ```
  ' , ( ) < > = + - * / \ t \ n \ 0 <space>
  ```

- You can use parameters and variables in the target update query. Use any parameter or variable type that you can define in the parameter file. You can enter a parameter or variable within the UPDATE statement, or you can use a parameter or variable as the update query. For example, you can enter a session parameter, $ParamMyOverride, as the update query, and set $ParamMyOverride to the UPDATE statement in a parameter file.
- When you save a mapping, the Designer verifies that you have referenced valid port names. It does not validate the SQL.
- If you update an individual row in the target table more than once, the database only has data from the last update. If the mapping does not define an order for the result data, different runs of the mapping on identical input data may result in different data in the target table.
- A WHERE clause that does not contain any column references updates all rows in the target table, or no rows in the target table, depending on the WHERE clause and the data from the mapping. For example, the following query sets the EMP_NAME to “MIKE SMITH” for *all* rows in the target table if any row of the transformation has EMP_ID > 100:

  ```
  UPDATE T_SALES set EMP_NAME = 'MIKE SMITH' WHERE :TU.EMP_ID > 100
  ```

- If the WHERE clause contains no port references, the mapping updates the same set of rows for each row of the mapping. For example, the following query updates all employees with EMP_ID > 100 to have the EMP_NAME from the last row in the mapping:

  ```
  UPDATE T_SALES set EMP_NAME = :TU.EMP_NAME WHERE EMP_ID > 100
  ```

- If the mapping includes an Update Strategy or Custom transformation, the Target Update statement only affects records marked for update.
- If you use the Target Update option, configure the session to mark all source records as update.

Steps to Enter a Target Update Statement

Use the following procedure to create an update statement:

1. Double-click the title bar of a target instance.
2. Click Properties.
3. Click the Open button in the Update Override field.

   The SQL Editor displays.
4. Select Generate SQL.

   The default UPDATE statement appears.
5. Modify the update statement.

   You can override the WHERE clause to include non-key columns.

   Enclose all reserved words in quotes.
6. Click OK.

   The Designer validates the SQL when you save the mapping.

Configuring the Table Name Prefix

The table name prefix is the owner of the target table. For some databases, such as DB2, target tables in a session can have different owners. If the database user specified in the database connection is not the owner of the target tables in a session, specify the table owner for each target instance. A session can fail if the database user is not the owner and you do not specify the table owner name.

You can specify the table owner name in the target instance or in the session properties. When you enter the table owner name in the session properties, you override the transformation properties.

Note: When you specify the table owner name and you set the sqlid for a DB2 database in the connection environment SQL, the Integration Service uses table owner name in the target instance. To use the table owner name specified in the SET sqlid statement, do not enter a name in the target name prefix.

To specify a target owner name at the target instance level:

1. In the Designer, open the Mapping Designer tool.
2. Double-click the title bar of a relational target instance in the mapping.
3. On the Properties tab, enter the table owner name or prefix in the Value field for Table Name Prefix.
4. Click OK.

Adding Pre- and Post-Session SQL Commands

You can enter pre- and post-session SQL commands on the Properties tab of the target instance in a mapping. You might want to run pre- and post-session SQL on the target to drop indexes before a session runs, and recreate them when the session completes.

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

You can override the SQL commands on the Mappings tab of the session properties. You can also configure the Integration Service to stop or continue when it encounters errors executing pre- or post-session SQL commands.

Rules and Guidelines for Adding Pre- and Post-Session SQL Commands

Use the following rules and guidelines when you enter pre- and post-session SQL commands in the target instance:

- Use any command that is valid for the database type. However, the Integration Service does not allow nested comments, even though the database might.
- You can use parameters and variables in the in the target pre- and post-session SQL commands. For example, you can enter a parameter or variable within the command. Or, you can use a session parameter, $ParamMyCommand, as the SQL command, and set $ParamMyCommand to the SQL statement in a parameter file.
- Use a semicolon (;) to separate multiple statements. The Integration Service issues a commit after each statement.
- The Integration Service ignores semicolons within /* ...*/.
- If you need to use a semicolon outside of comments, you can escape it with a backslash (\).
- The Designer does not validate the SQL.

Note: You can also enter pre- and post-session SQL commands on the Properties tab of the Source Qualifier transformation.

Overriding the Target Table Name

You can override the target table name in the target instance of a mapping. Override the target table name when you use a single mapping to load data to different target tables. Enter a table name in the target table name. You can also enter a parameter or variable. You can use mapping parameters, mapping variables, session parameters, workflow variables, or worklet variables in the target table name. For example, you can use a session parameter, $ParamTgtTable, as the target table name, and set $ParamTgtTable to the target table name in the parameter file.

To override a target table name:

1. In the Designer, open the Mapping Designer.
2. Double-click the title bar of a relational target instance in the mapping.
3. On the Properties tab, enter the target table name. Or, enter a parameter or variable name in the Target Table Name field.

   If you use a user-defined mapping parameter, mapping variable, workflow variable, or worklet variable, you must declare the parameter or variable.
4. Click OK.
5. If you use a parameter or variable for the target table name, define the parameter or variable in the appropriate section of the parameter file.

Validating a Mapping

When you develop a mapping, you must configure it so the Integration Service can read and process the entire mapping. The Designer marks a mapping invalid when it detects errors that will prevent the Integration Service from running sessions associated with the mapping.

The Designer marks a mapping valid for the following reasons:

- **Connection validation.** Required ports are connected and that all connections are valid.
- **Expression validation.** All expressions are valid.
- **Object validation.** The independent object definition matches the instance in the mapping.
- **Data flow validation.** The data must be able to flow from the sources to the targets without hanging at blocking transformations.

Connection Validation

The Designer performs connection validation each time you connect ports in a mapping and each time you validate or save a mapping. When you connect ports, the Designer verifies that you make valid connections. When you save or validate a mapping, the Designer verifies that the connections are valid and that all required ports are connected. When you save or validate a mapping, the Designer makes the following connection validations:

- **At least one source and one target must be connected.**
- **Source qualifiers must be mapped to a target.**
- **Mapplets must be connected.** At least one mapplet input port and output port is connected to the mapping. If the mapplet includes a source qualifier that uses an SQL override, the Designer prompts you to connect all mapplet output ports to the mapping.

- **Datatypes between ports must be compatible.** If you change a port datatype to one that is incompatible with the port it is connected to, the Designer generates an error and invalidates the mapping. For example, you have two Date/Time ports connected, and you change one port to a Decimal. The Designer invalidates the mapping. You can however, change the datatype if it remains compatible with the connected ports, such as Char and Varchar.

Related Topics:

- "Connecting Mapping Objects " on page 114

Expression Validation

You can validate an expression in a transformation while you are developing a mapping. If you did not correct the errors, the Designer writes the error messages in the Output window when you save or validate the mapping.

If you delete input ports used in an expression, the Designer marks the mapping as invalid.

Object Validation

When you validate or save a mapping, the Designer verifies that the definitions of the independent objects, such as sources 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 Designer and other PowerCenter Client applications track the effects of these changes on the mappings. The Repository Manager displays the status of mappings, so you can see if a mapping is valid or not. If you notice that a mapping is invalid, you can open the mapping and validate it to see the error messages in the Output window.

Data Flow Validation

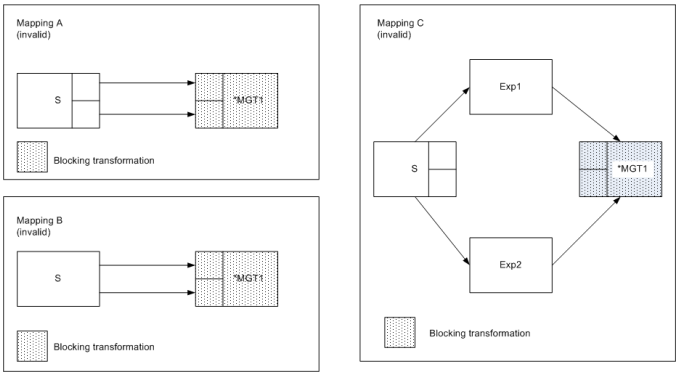
When you validate or save a mapping, the Designer verifies that the data can flow from all sources in a target load order group to the targets without the Integration Service blocking all sources.

Mappings that include blocking transformations might hang at runtime with any of the following mapping configurations:

- You connect one source pipeline to multiple input groups of the blocking transformation.
- You connect the sources and transformations in a target load order group in such a way that multiple blocking transformations could possibly block all source pipelines. Depending on the source data used in a session, a blocking transformation might block data from one source while it waits for a row from a different source.

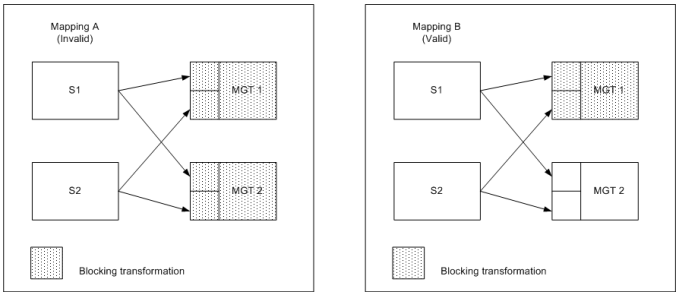
When you save or validate a mapping with one of these configurations, the Designer marks the mapping invalid. When the Designer marks a mapping invalid because the mapping violates data flow validation, you must configure the mapping differently, or use a non-blocking transformation where possible.

The following figure shows mappings that are invalid because one source provides data for multiple input groups of a blocking transformation:



To make the mappings valid, use a non-blocking transformation for MGT1 or create two instances of the same source and connect them to the blocking transformation.

The following figure shows two similar mappings, one which is valid, one which is invalid:



Mapping A contains two multigroup transformations that block data, MGT1 and MGT2. If you could run this session, MGT1 might block data from S1 while waiting for a row from S2. And MGT2 might block data from S2 while waiting for a row from S1. The blocking transformations would block both source pipelines and the session would hang. Therefore, the Designer marks the mapping invalid.

Mapping B contains one multigroup transformation that blocks data, MGT1. Blocking transformations can never block all input groups, so MGT1 might block either S1 or S2, but never both. MGT2 is not a blocking transformation, so it will never block data. Therefore, this session will not hang at runtime due to blocking. The Designer marks the mapping valid.

Steps to Validate a Mapping

You can validate a mapping while you are working on it through the Designer. Also, when you click Repository > Save, the Designer validates all mappings since the last time you saved. When you validate or save a mapping the results of the validation appear in the Output window. The Repository Manager also displays whether a mapping is valid.

To validate a mapping, check out and open the mapping, and click Mappings > Validate.

If the Output window is not open, click View > Output Window. Review any errors to determine how to fix the mapping.

Validating Multiple Mappings

You can validate multiple mappings without fetching them into the workspace. To validate multiple mappings you must select and validate the mappings from either a query results view or a view object dependencies list.

Note: If you use the Repository Manager, you can select and validate multiple mappings from the Navigator.

You can save and optionally check in mappings that change from invalid to valid status as a result of the validation.

To validate multiple mappings:

1. Select mappings from either a query or a view dependencies list.
2. Right-click one of the selected mappings and choose Validate.

   The Validate Objects dialog box displays.
3. Choose whether to save objects and check in objects that you validate.

Using the Workflow Generation Wizard

Use the Workflow Generation Wizard to generate sessions and workflows from a mapping. The mapping can use relational or flat file sources and targets.

Launch the Workflow Generation Wizard from one of the following locations, based on what you want to generate:

- **Menu options.** Use the Workflow Generation Wizard to create one workflow and session for one mapping.
- **Import Mapping Template Wizard.** Use the Workflow Generation Wizard to create workflows and sessions for multiple mappings.

Before you use the Workflow Generation Wizard, verify that the mapping is valid, and that the Integration Service and connection objects have been created. Verify that the mapping uses relational or flat file sources and targets.

The Workflow Generation Wizard allows you to complete the following steps:

1. Specify the type of session or workflow you want to generate.
2. Specify the Integration Service, connection settings, and workflow and session name prefixes. The values you specify in the second step apply to all sessions and workflows you configure.
3. Change the workflow name, session name, and Integration Service. You can also configure connection settings. The values you specify in the third step apply to the workflow and session you configure

The last page of the wizard displays a list of workflows or sessions that were generated and the status of the workflows and sessions.

Note: Use connection variables in a session that the Workflow Generation Wizard creates, edit the session properties in the Workflow Manager after the Workflow Generation Wizard creates the session in the repository.

Workflow Generation Wizard Steps

The options that appear in the wizard depend on the type of workflow or session you generate and the method you use to launch the Workflow Generation Wizard.

To use the Workflow Generation Wizard:

1. Open a mapping in the Mapping Designer workspace and select the mapping. Then, select Mappings > Generate Workflows.

   You can also launch the Workflow Generation Wizard from the Import Mapping Template Wizard.

2. Select the type of workflow or session you want to generate:
   - Reusable Session
   - Workflow with Reusable Session
   - Workflow with Non-Reusable Session
3. Click Next.
4. Specify the Integration Service, connection settings, and the workflow and session name prefix.
5. To change a connection object, click in the connection object field. Click the open button to open the Connection Browser and specify a connection object.

   To change the filename for a flat file, edit the file name in the connection object field.
6. Click Next.
7. Change the workflow name, session name, or Integration Service.
8. Click the configure button to configure connection settings.

   The Workflow Settings dialog displays the following tabs:

Tabs	Description
Connections	Configure source, target, and transformation connection information.
Properties	Configure source and target session properties.
Reader/Writer	Configure readers and writers for the source and target instances in the mapping.

9. Click Next.
10. Review the status and click Finish.

Troubleshooting Mappings

When I save a mapping, the Designer indicates that it contains errors.

When you save a mapping, the Designer checks it for errors, such as targets that do not receive data from any sources. While you can save an invalid mapping, you cannot run a session using it.

The Designer does not allow me to connect two ports in a mapping.

The Designer performs validation when you connect ports. You cannot connect an input port to an input port, or two ports with incompatible datatypes. Check the error message displayed when the Designer prevented you from making the connection.

I cannot connect multiple sources to one target.

This is not allowed. These are possible workarounds:

1. All targets are reusable. You can add the same target to the mapping multiple times. Then, connect each source qualifier to each target.

2. Join the sources in a Source Qualifier transformation. Then, remove the WHERE clause from the SQL query.
3. Join the sources in a Joiner transformation.

When I click and drag to create a connection between ports, the Designer copies the port instead.

Change the mode in which you are working by clicking Layout > Link Columns. Now, when you drag between columns, the Designer tries to link them instead of copying the selected port.

When I validate a mapping, I cannot see the results of the test.

Make sure that you have the Output window open when you validate the mapping. Click View > Output Window to see the results of the validation.

I entered a custom query, but it is not working when I run the workflow.

Be sure to test this setting for the source qualifier before you run the workflow. Return to the source qualifier and open the dialog box in which you entered the custom query. You can connect to a database and click the Validate button to test the SQL. The Designer shows you any errors exist that in the SQL you entered. Review the session logs if you need more information.

CHAPTER 6

Mapplets

This chapter includes the following topics:

- Mapplets Overview, 139
- Understanding Mapplet Input and Output , 140
- Using the Mapplet Designer, 142
- Using Mapplets in Mappings, 144
- Rules and Guidelines for Mapplets, 146
- Tips for Mapplets, 147

Mapplets Overview

A mapplet is a reusable object that you create in the Mapplet Designer. It contains a set of transformations and lets you reuse the transformation logic in multiple mappings.

For example, if you have several fact tables that require a series of dimension keys, you can create a mapplet containing a series of Lookup transformations to find each dimension key. You can then use the mapplet in each fact table mapping, rather than recreate the same lookup logic in each mapping.

When you use a mapplet in a mapping, you use an instance of the mapplet. Like a reusable transformation, any change made to the mapplet is inherited by all instances of the mapplet.

Mapplets help simplify mappings in the following ways:

- **Include source definitions.** Use multiple source definitions and source qualifiers to provide source data for a mapping.
- **Accept data from sources in a mapping.** If you want the mapplet to receive data from the mapping, use an Input transformation to receive source data.
- **Include multiple transformations.** A mapplet can contain as many transformations as you need.
- **Pass data to multiple transformations.** You can create a mapplet to feed data to multiple transformations. Each Output transformation in a mapplet represents one output group in a mapplet.
- **Contain unused ports.** You do not have to connect all mapplet input and output ports in a mapping.

Understanding Mapplet Input and Output

To use a mapplet in a mapping, you must configure it for input and output. In addition to transformation logic that you configure, a mapplet has the following components:

- **Mapplet input.** You can pass data into a mapplet using source definitions or Input transformations or both. When you use an Input transformation, you connect it to the source pipeline in the mapping.
- **Mapplet output.** Each mapplet must contain one or more Output transformations to pass data from the mapplet into the mapping.
- **Mapplet ports.** Mapplet ports display only in the Mapping Designer. Mapplet ports consist of input ports from Input transformations and output ports from Output transformations. If a mapplet uses source definitions rather than Input transformations for input, it does not contain any input ports in the mapping.

Mapplet Input

Mapplet input can originate from a source definition and/or from an Input transformation in the mapplet. You can create multiple pipelines in a mapplet. Use multiple source definitions and source qualifiers or Input transformations. You can also use a combination of source definitions and Input transformations.

Using Source Definitions for Mapplet Input

Use one or more source definitions in a mapplet to provide source data. When you use the mapplet in a mapping, it is the first object in the mapping pipeline and contains no input ports.

Using Input Transformations for Mapplet Input

Use an Input transformation in a mapplet when you want the mapplet to receive input from a source in a mapping. When you use the mapplet in a mapping, the Input transformation provides input ports so you can pass data through the mapplet. Each port in the Input transformation connected to another transformation in the mapplet becomes a mapplet input port. Input transformations can receive data from a single active source. Unconnected ports do not display in the Mapping Designer.

You can connect an Input transformation to multiple transformations in a mapplet. However, you cannot connect a single port in the Input transformation to multiple transformations in the mapplet.

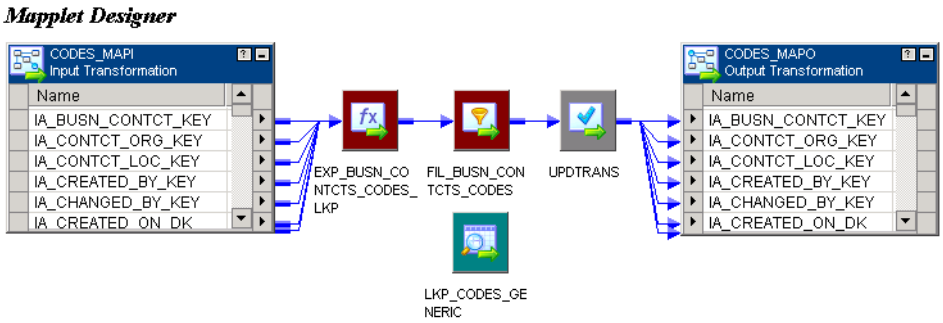
Mapplet Output

Use an Output transformation in a mapplet to pass data through the mapplet into a mapping. A mapplet must contain at least one Output transformation with at least one connected port in the mapplet. Each connected port in an Output transformation displays as a mapplet output port in a mapping. Each Output transformation in a mapplet displays as an output group in a mapping. An output group can pass data to multiple pipelines in a mapping.

Viewing Mapplet Input and Output

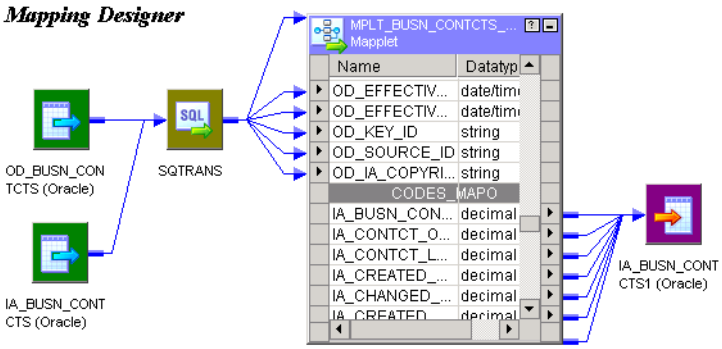
Mapplets and mapplet ports display differently in the Mapplet Designer and the Mapping Designer.

The following figure shows a mapplet with both an Input transformation and an Output transformation:



When you use the mapplet in a mapping, the mapplet object displays only the ports from the Input and Output transformations. These are referred to as the mapplet input and mapplet output ports.

The following figure shows the same mapplet in the Mapping Designer:

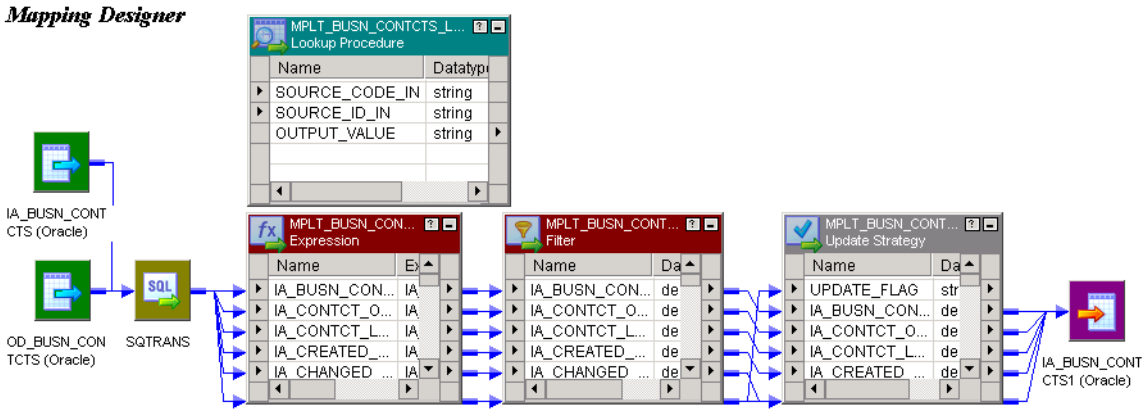


The mapplet displays the input ports from the Input transformation. The output ports from the CODES_MAP0 Output transformation appear below the input ports.

You can expand the mapplet in the Mapping Designer by selecting it and clicking Mappings > Expand. This expands the mapplet within the mapping for view. Transformation icons within an expanded mapplet display as shaded.

You can open or iconize all the transformations in the mapplet and mapping. You cannot edit any of the properties, navigate to other folders, or save the repository while the mapplet is expanded.

The following figure shows an expanded mapplet in the Mapping Designer:



In an expanded mapping, you do not see the Input and Output transformations.

Using the Mapplet Designer

After you create a mapplet, you can validate or edit the mapplet in the Mapplet Designer. You can also use the Designer to copy mapplets, export and import mapplets, view links between ports in a mapplet, create shortcuts to mapplets, and delete mapplets from the repository.

To create and configure a mapplet in the Mapplet Designer, complete the following steps:

1. Create a mapplet. Click Mapplets > Create from the menu in the Mapplet Designer. The recommended naming convention for mapplets is mplt_*MappletName*.
2. Create mapplet transformation logic. Create and link transformations in the same manner as in a mapping.
3. Create mapplet ports.

Creating a Mapplet

A mapplet can be active or passive depending on the transformations in the mapplet. Active mapplets contain one or more active transformations. Passive mapplets contain only passive transformations. When you use a mapplet in a mapping, all transformation rules apply to the mapplet depending on the mapplet type. For example, as with an active transformation, you cannot concatenate data from an active mapplet with a different pipeline.

Use the following rules and guidelines when you add transformations to a mapplet:

- If you use a Sequence Generator transformation, you must use a reusable Sequence Generator transformation.
- If you use a Stored Procedure transformation, you must configure the Stored Procedure Type to be *Normal*.
- You cannot include PowerMart 3.5-style LOOKUP functions in a mapplet.
- You cannot include the following objects in a mapplet:
  - Normalizer transformations
  - COBOL sources
  - XML Source Qualifier transformations
  - XML sources
  - Target definitions
  - Other mapplets

Although reusable transformations and shortcuts in a mapplet can be used, to protect the validity of the mapplet, use a copy of a transformation instead. Reusable transformations and shortcuts inherit changes to their original transformations. This might invalidate the mapplet and the mappings that use the mapplet.

Validating Mapplets

The Designer validates a mapplet when you save it. You can also validate a mapplet using the Mapplets > Validate menu command. When you validate a mapplet, the Designer writes all relevant messages about the mapplet in the Output window.

The Designer validates the mapplet pipeline in the same way it validates a mapping. The Designer also performs the following checks specific to mapplets:

- The mapplet can contain Input transformations and source definitions with at least one port connected to a transformation in the mapplet.
- The mapplet contains at least one Output transformation with at least one port connected to a transformation in the mapplet.

Editing Mapplets

You can edit a mapplet in the Mapplet Designer. The Designer validates the changes when you save the mapplet. When you save changes to a mapplet, all instances of the mapplet and all shortcuts to the mapplet inherit the changes. These changes might invalidate mappings that use the mapplet.

To see what mappings or shortcuts may be affected by changes you make to a mapplet, select the mapplet in the Navigator, right-click, and select Dependencies. Or, click Mapplets > Dependencies from the menu.

You can make the following changes to a mapplet *without* affecting the validity of existing mappings and sessions:

- Add input or output ports.
- Change port names or comments.
- Change Input or Output transformation names or comments.
- Change transformation names, comments, or properties.
- Change port default values for transformations in the mapplet.
- Add or remove transformations in the mapplet, providing you do not change the mapplet type from active to passive or from passive to active.

Use the following rules and guidelines when you edit a mapplet that is used by mappings:

- **Do not delete a port from the mapplet.** The Designer deletes mapplet ports in the mapping when you delete links to an Input or Output transformation or when you delete ports connected to an Input or Output transformation.
- **Do not change the datatype, precision, or scale of a mapplet port.** The datatype, precision, and scale of a mapplet port is defined by the transformation port to which it is connected in the mapplet. Therefore, if you edit a mapplet to change the datatype, precision, or scale of a port *connected* to a port in an Input or Output transformation, you change the mapplet port.
- **Do not change the mapplet type.** If you remove all active transformations from an active mapplet, the mapplet becomes passive. If you add an active transformation to a passive mapplet, the mapplet becomes active.

Mapplets and Mappings

The following mappings tasks can also be performed on mapplets:

- **Set tracing level.** You can set the tracing level on individual transformations within a mapplet in the same manner as in a mapping.
- **Copy mapplet.** You can copy a mapplet from one folder to another as you would any other repository object. After you copy the mapplet, it appears in the Mapplets node of the new folder.

  If you make changes to a mapplet, but you do not want to overwrite the original mapplet, you can make a copy of the mapplet by clicking Mapplets > Copy As.

- **Export and import mapplets.** You can export a mapplet to an XML file or import a mapplet from an XML file through the Designer. You might want to use the export and import feature to copy a mapplet to another repository.
- **Delete mapplets.** When you delete a mapplet, you delete all instances of the mapplet. This invalidates each mapping containing an instance of the mapplet or a shortcut to the mapplet.
- **Compare mapplets.** You can compare two mapplets to find differences between them. For example, if you have mapplets with the same name in different folders, you can compare them to see if they differ.
- **Compare instances within a mapplet.** You can compare instances in a mapplet to see if they contain similar attributes. For example, you can compare a source instance with another source instance, or a transformation with another transformation. You compare instances within a mapplet in the same way you compare instances within a mapping.
- **Create shortcuts to mapplets.** You can create a shortcut to a mapplet if the mapplet is in a shared folder. When you use a shortcut to a mapplet in a mapping, the shortcut inherits any changes you might make to the mapplet. However, these changes might not appear until the Integration Service runs the workflow using the shortcut. Therefore, only use a shortcut to a mapplet when you do not expect to edit the mapplet.
- **Add a description.** You can add a description to the mapplet in the Mapplet Designer in the same manner as in a mapping. You can also add a description to the mapplet instance in a mapping. When you add a description, you can also create links to documentation files. The links must be a valid URL or file path to reference the business documentation.
- **View links to a port.** You can view links to a port in a mapplet in the same way you would view links to a port in a mapping. You can view the forward path, the backward path, or both paths.
- **Propagate port attributes.** You can propagate port attributes in a mapplet in the same way you would propagate port attributes in a mapping. You can propagate attributes forward, backward, or in both directions.

Using Mapplets in Mappings

In a mapping, a mapplet has input and output ports that you can connect to other transformations in the mapping. You do not have to connect all mapplet ports in a mapping. However, if the mapplet contains an SQL override, you must connect all mapplet output ports in the mapping.

Like a reusable transformation, when you drag a mapplet into a mapping, the Designer creates an instance of the mapplet. You can enter comments for the instance of the mapplet in the mapping. You cannot otherwise edit the mapplet in the Mapping Designer.

If you edit the mapplet in the Mapplet Designer, each instance of the mapplet inherits the changes.

The PowerCenter Repository Reports has a Mapplets list report that you use to view all mappings using a particular mapplet.

To use a mapplet, complete the following steps:

1. Drag the mapplet into the mapping.
2. If the mapplet contains input ports, connect at least one mapplet input port to a transformation in the mapping.
3. Connect at least one mapplet output port to a transformation in the mapping.

Creating and Configuring Mapplet Ports

After creating transformation logic for a mapplet, you can create mapplet ports. Use an Input transformation to define mapplet input ports if the mapplet contains no source definitions. Use an Output transformation to create a group of output ports. Only connected ports in an Input or Output transformation become mapplet input or output ports in a mapping. Unconnected ports do not display when you use the mapplet in a mapping.

You can create a mapplet port in the following ways:

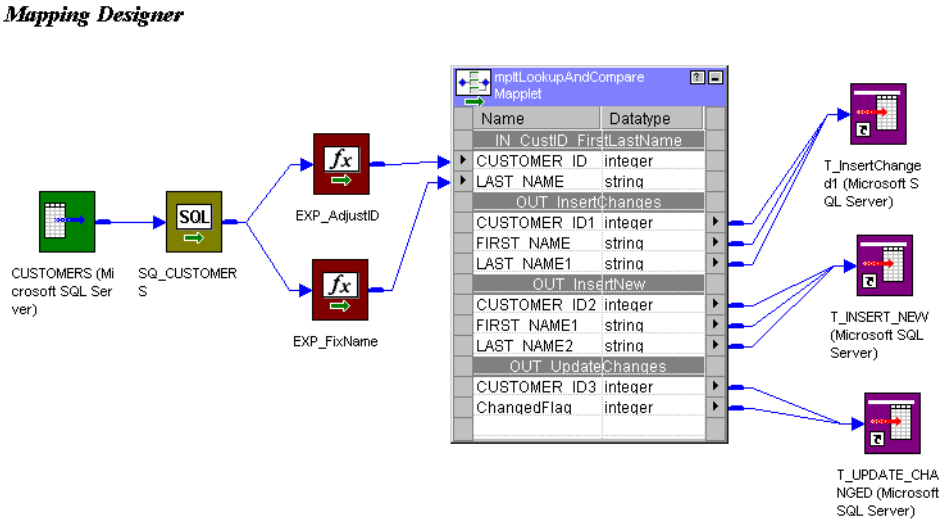
- **Manually create ports in the Input/Output transformation.** You can create port names in Input and Output transformations. You can also enter a description for each port name. The port has no defined datatype, precision, or scale until you connect it to a transformation in the mapplet.
- **Drag a port from another transformation.** You can create an input or output port by dragging a port from another transformation into the Input or Output transformation. The new port inherits the port name, description, datatype, and scale of the original port. You can edit the new port name and description in the transformation. If you change a port connection, the Designer updates the Input or Output transformation port to match the attributes of the new connection.

You can view the datatype, precision, and scale of available mapplet ports when you use the mapplet in a mapping.

Connecting to Mapplet Input Ports

When using a mapplet with input ports in a mapping, you connect the mapplet input ports to the mapping pipeline. You can pass data into a mapplet only when it originates from a single active transformation.

For example, in the following figure, the mapplet mpltLookupAndCompare accepts data from two Expression transformations because data from both transformations originate from a single source qualifier. The Source Qualifier SQ_CUSTOMERS is the active transformation providing mapplet source data:



Connecting to Mapplet Output Groups

Each Output transformation displays as an output group when you use a mapplet in a mapping. Connect the mapplet output ports to the mapping pipeline. Use Autolink to connect the ports.

Use the following rules and guidelines when you connect mapplet output ports in the mapping:

- When a mapplet contains a source qualifier that has an override for the default SQL query, you must connect all of the source qualifier output ports to the next transformation within the mapplet.
- If the mapplet contains more than one source qualifier, use a Joiner transformation to join the output into one pipeline.
- If the mapplet contains only one source qualifier, you must connect the mapplet output ports to separate pipelines. You cannot use a Joiner transformation to join the output.

  If you need to join the pipelines, you can create two mappings to perform this task:

  - Use the mapplet in the first mapping and write data in each pipeline to separate targets.
  - Use the targets as sources in the second mapping to join data, then perform any additional transformation necessary.

Viewing the Mapplet

When you use a mapplet in a mapping, the Designer displays the mapplet object, which contains only the input and output ports of the mapplet. However, you can expand the mapplet by clicking Mappings > Expand from the menu.

When the Designer expands the mapplet, it displays the entire mapping with the mapplet transformations. It does not display the Input and Output transformations. You can view the mapping in this expanded form, but you cannot edit it. To continue designing the mapping, click Mappings > Unexpand.

Setting the Target Load Plan

When you use a mapplet in a mapping, the Mapping Designer lets you set the target load plan for sources within the mapplet.

Pipeline Partitioning

If you have the partitioning option, you can increase the number of partitions in a pipeline to improve session performance. Increasing the number of partitions allows the Integration Service to create multiple connections to sources and process partitions of source data concurrently.

When you create a session, the Workflow Manager validates each pipeline in the mapping for partitioning. You can specify multiple partitions in a pipeline if the Integration Service can maintain data consistency when it processes the partitioned data.

Some partitioning restrictions apply to mapplets.

Rules and Guidelines for Mapplets

The following list summarizes the rules and guidelines that appear throughout this chapter:

- You can connect an Input transformation to multiple transformations in a mapplet. However, you cannot connect a single port in the Input transformation to multiple transformations in the mapplet.
- An Input transformation must receive data from a single active source.
- A mapplet must contain at least one Input transformation or source definition with at least one port connected to a transformation in the mapplet.

- A mapplet must contain at least one Output transformation with at least one port connected to another transformation in the mapping.
- When a mapplet contains a source qualifier that has an override for the default SQL query, you must connect all of the source qualifier output ports to the next transformation within the mapplet.
- If the mapplet contains more than one source qualifier, use a Joiner transformation to join the output into one pipeline. If the mapplet contains only one source qualifier, you must connect the mapplet output ports to separate pipelines. You cannot use a Joiner transformation to join the output.
- When you edit a mapplet, you might invalidate mappings if you change the mapplet type from passive to active.
- If you delete ports in the mapplet when the mapplet is used in a mapping, you can invalidate the mapping.
- Do not change the datatype, precision, or scale of a mapplet port when the mapplet is used by a mapping.
- If you use a Sequence Generator transformation, you must use a reusable Sequence Generator transformation.
- If you use a Stored Procedure transformation, you must configure the Stored Procedure Type to be *Normal*.
- You cannot include PowerMart 3.5-style LOOKUP functions in a mapplet.
- You cannot include the following objects in a mapplet:
  - Normalizer transformations
  - Cobol sources
  - XML Source Qualifier transformations
  - XML sources
  - Target definitions
  - Pre- and post- session stored procedures
  - Other mapplets

Tips for Mapplets

Enter comments for Input and Output transformations.

In a mapping, you can view comments associated with an Input or Output transformation by holding the pointer over the transformation name in the mapplet. You can clarify the use of a mapplet and its ports by creating descriptive comments or instructions in the Description field of Input and Output transformations. You can even include links to business documentation.

Create an Output transformation for each output group you need.

You can pass data from each mapplet output group to a different mapping pipeline. Create an Output transformation for each output group you need.

To create mapplets from existing mappings, copy the objects from the mapping into the Mapplet Designer.

Configure Sequence Generator caches appropriately.

You can include a reusable Sequence Generator transformation in a mapplet. If you use the mapplet in several mappings and each mapping expends a large number of values in a session, you might want to configure the cache size for reusable Sequence Generator to limit the number of unused values.

To keep existing mappings valid when you edit a mapplet used in a mapping:

- Do not delete connected ports in an Input or Output transformation.
- Do not change the datatype, precision, or scale of connected ports in an Input or Output transformation.
- Do not change a passive mapplet to an active mapplet or an active mapplet to a passive mapplet.

CHAPTER 7

Mapping Parameters and Variables

This chapter includes the following topics:

- Mapping Parameters and Variables Overview, 149
- Mapping Parameters, 153
- Mapping Variables, 155
- Defining Expression Strings in Parameter Files, 161
- Tips for Mapping Parameters and Variables, 161
- Troubleshooting Mapping Parameters and Variables, 162

Mapping Parameters and Variables Overview

In the Designer, use mapping parameters and variables to make mappings more flexible. Mapping parameters and variables represent values in mappings and mapplets.

If you declare mapping parameters and variables in a mapping, you can reuse a mapping by altering the parameter or variable values of the mapping in the session. Reuse a mapping to reduce the overhead of creating multiple mappings when only certain attributes of a mapping need to be changed.

When you use a mapping parameter or variable in a mapping, you first declare the mapping parameter or variable for use in each mapplet or mapping. Then, you define a value for the mapping parameter or variable before you run the session.

You can declare mapping parameters with the same name for use in a mapplet and a mapping. If the mapping parameter in a mapplet does not have a value, the parameter takes the value of the mapping parameter with the same name in the mapping.

Use mapping parameters and variables in a mapping to incrementally extract data. Use mapping parameters or variables in the source filter of a Source Qualifier transformation to determine the beginning time stamp and end time stamp for incrementally extracting data.

For example, you can create a user-defined mapping variable $$LastUpdateDateTime to save the time stamp of the last row that the Integration Service read in the previous session. In the source filter, use $ $LastUpdateDateTime for the beginning time stamp and the built-in variable $$$SessStartTime for the end

time stamp. Use the following filter to incrementally extract data based on the SALES.sales_datetime column in the source:

```
SALES.sales_datetime > TO_DATE ('$$LastUpdateDateTime') AND SALES.sales_datetime <
TO_DATE ('$$$SessStartTime')
```

Mapping Parameters

A mapping parameter represents a constant value that you can define before running a session. A mapping parameter retains the same value throughout the entire session.

When you use a mapping parameter, you declare and use the parameter in a mapping or mapplet. Then define the value of the parameter in a parameter file. The Integration Service evaluates all references to the parameter to that value.

For example, you want to use the same session to extract transaction records for each of the customers individually. Instead of creating a separate mapping for each customer account, you can create a mapping parameter to represent a single customer account. Then use the parameter in a source filter to extract only data for that customer account. Before running the session, you enter the value of the parameter in the parameter file.

To reuse the same mapping to extract records for other customer accounts, you can enter a new value for the parameter in the parameter file and run the session. Or, you can create a parameter file for each customer account and start the session with a different parameter file each time using *pmcmd*. By using a parameter file, you reduce the overhead of creating multiple mappings and sessions to extract transaction records for different customer accounts.

When you want to use the same value for a mapping parameter each time you run the session, use the same parameter file for each session run. When you want to change the value of a mapping parameter between sessions you can perform one of the following tasks:

- Update the parameter file between sessions.
- Create a different parameter file and configure the session to use the new file.
- Remove the parameter file from the session properties. The Integration Service uses the parameter value in the pre-session variable assignment. If there is no pre-session variable assignment, the Integration Service uses the configured initial value of the parameter in the mapping.

Mapping Variables

Unlike a mapping parameter, a mapping variable represents a value that can change through the session. The Integration Service saves the value of a mapping variable to the repository at the end of each successful session run and uses that value the next time you run the session.

When you use a mapping variable, you declare the variable in the mapping or mapplet, and then use a variable function in the mapping to change the value of the variable. At the beginning of a session, the Integration Service evaluates references to a variable to determine the start value. At the end of a successful session, the Integration Service saves the final value of the variable to the repository. The next time you run the session, the Integration Service evaluates references to the variable to the saved value. To override the saved value, define the start value of the variable in a parameter file or assign a value in the pre-session variable assignment in the session properties.

Use mapping variables to perform incremental reads of a source. For example, the customer accounts in the mapping parameter example above are numbered from 001 to 065, incremented by one. Instead of creating a mapping parameter, you can create a mapping variable with an initial value of 001. In the mapping, use a variable function to increase the variable value by one. The first time the Integration Service runs the session, it extracts the records for customer account 001. At the end of the session, it increments the variable by one and saves that value to the repository. The next time the Integration Service runs the session, it extracts the

data for the next customer account, 002. It also increments the variable value so the next session extracts and looks up data for customer account 003.

Using Mapping Parameters and Variables

You can create mapping parameters and variables in the Mapping Designer or Mapplet Designer. Once created, mapping parameters and variables appear on the Variables tab of the Expression Editor. Use them in any expression in the mapplet or mapping. The Designer validates mapping parameters and variables in the Expression Editor of mapplets and mappings.

Use mapping parameters and variables in a source qualifier in a mapplet or mapping. When you use mapping parameters and variables in a Source Qualifier transformation, the Designer expands them before passing the query to the source database for validation. This allows the source database to validate the query.

When you create a reusable transformation in the Transformation Developer, use any mapping parameter or variable. Since a reusable transformation is not contained within any mapplet or mapping, the Designer validates the usage of any mapping parameter or variable in the expressions of reusable transformation. When you use the reusable transformation in a mapplet or mapping, the Designer validates the expression again. If the parameter or variable is not defined in the mapplet or mapping, or if it is used incorrectly in the reusable transformation, the Designer logs an error when you validate the mapplet or mapping.

When the Designer validates a mapping variable in a reusable transformation, it treats the variable as an Integer datatype.

You cannot use mapping parameters and variables interchangeably between a mapplet and a mapping. Mapping parameters and variables declared for a mapping cannot be used within a mapplet. Similarly, you cannot use a mapping parameter or variable declared for a mapplet in a mapping.

Initial and Default Values

When you declare a mapping parameter or variable in a mapping or a mapplet, you can enter an initial value. The Integration Service uses the configured initial value for a mapping parameter when the parameter is not defined in the parameter file. Similarly, the Integration Service uses the configured initial value for a mapping variable when the variable value is not defined in the parameter file, and there is no saved variable value in the repository.

When the Integration Service needs an initial value, and you did not declare an initial value for the parameter or variable, the Integration Service uses a default value based on the datatype of the parameter or variable.

The following table lists the default values the Integration Service uses for different types of data:

Data	Default Value
String	Empty string.
Numeric	0
Datetime	1/1/1753 A.D. or 1/1/1 when the Integration Service is configured for compatibility with 4.0.

For example, you create a new mapping using an Integer mapping variable, $$MiscellaneousExpenses. You do not configure an initial value for the variable or define it in a parameter file. The first time you run a session with the mapping, the Integration Service uses the default value for numeric datatypes, 0.

Or, if you create a mapping parameter $$MiscellaneousCosts to represent additional expenses that might become relevant in the future, but do not currently exist in source data. You configure the parameter for a

Decimal datatype. Since you want $$MiscellaneousCosts to evaluate to 0 when you do not have additional expenses, you set the initial value to 0.

As long as you do not define the parameter value in the parameter file, the Integration Service replaces $ $MiscellaneousCosts with 0. When you want to include miscellaneous expenses in mapping calculations, set $$MiscellaneousCosts to that value in the parameter file.

Using String Parameters and Variables

When you enter mapping parameters and variables of a string datatype in a Source Qualifier transformation, use a string identifier appropriate for the source database. When the Integration Service expands a parameter or variable in a Source Qualifier transformation, the Integration Service replaces it with its start value, and then passes the expanded query to the source database. Most databases require single quotation marks around string values.

When you enter string parameters or variables using the PowerCenter transformation language, do not use additional quotes. The Integration Service recognizes mapping parameter and variable naming syntax in the PowerCenter transformation language. For example, you might use a parameter named $$State in the filter for a Source Qualifier transformation to extract rows for a particular state:

```
STATE = '$$State'
```

During the session, the Integration Service replaces the parameter with a string. If $$State is defined as MD in the parameter file, the Integration Service replaces the parameter as follows:

```
STATE = 'MD'
```

You can perform a similar filter in the Filter transformation using the PowerCenter transformation language as follows:

```
STATE = $$State
```

If you enclose the parameter in single quotes in the Filter transformation, the Integration Service reads it as the string literal “$$State” instead of replacing the parameter with “MD.”

Using Datetime Parameters and Variables

When you use a datetime parameter or variable in the Source Qualifier transformation, you might need to change the date format to the format used in the source.

Code Page Relaxation

You can configure the Integration Service to relax code page validation when you run the Integration Service in Unicode data movement mode. However, you might get unexpected results in the following situations:

- The mapping variable value that the Integration Service saves in the repository is not compatible with the repository code page.

  For example, the repository uses the ISO 8859-1 Latin1 code page and you configure the Integration Service to relax code page validation. If the mapping variable value contains Japanese character data, such as JapanEUC, the saved mapping variable value in the repository could be incorrect. There could be data loss converting from the JapanEUC code page to the Latin1 code page. Make sure the saved mapping variable value is two-way compatible with the repository code page.

  To ensure the Integration Service can write all metadata to the repository, use 7-bit ASCII characters for all repository metadata or use UTF-8 for the repository.

- The parameter file contains characters that are not compatible with the Integration Service code page.

  The Integration Service interprets the data in the parameter file using the Integration Service code page. For example, the Integration Service uses the ISO 8859-1 Latin1 code page and you configure the Integration Service to relax code page validation. If you create a parameter file and use Greek character data, such as ISO 8859-7, the value the Integration Service reads from the file could be incorrect. There could be data loss converting from the ISO 8859-7 code page to the Latin1 code page. Make sure the characters in the parameter file are a subset of the Integration Service code page.

Mapping Parameters

In the Designer, you can create a mapping parameter in a mapplet or mapping. After you create a parameter, it appears in the Expression Editor. You can then use the parameter in any expression in the mapplet or mapping. You can also use parameters in a source qualifier filter, user-defined join, or extract override, and in the Expression Editor of reusable transformations.

Before you run a session, define the mapping parameter value in a parameter file for the session. Use any constant value. During the session, the Integration Service evaluates all references to the parameter to the specified value. If the parameter is not defined in the parameter file, the Integration Service uses the user-defined initial value for the parameter. If the initial value is not defined, the Integration Service uses a default value based on the datatype of the mapping parameter.

You can change the value of a mapping parameter between sessions by editing the parameter file or by changing the parameter file used by the session.

You might use a mapping parameter instead of a database lookup. For example, you want to perform calculations using monthly gross earnings. Instead of using a Lookup transformation to connect to a database table for that information, you can create a gross earnings mapping parameter and update its value in the parameter file each month to reflect current earnings.

You might also use a mapping parameter in conjunction with a session parameter to reuse a mapping and session. For example, you have transactional data from different states stored in the same table in different databases, and you want to perform the same calculations on all data, while changing the state sales tax accordingly. Instead of creating a separate mapping and session for each state, you can create one mapping with a sales tax mapping parameter and a session using a source database connection session parameter. You can then create a different parameter file for each state. Before running the session, you can change the parameter file the Integration Service uses by entering a different parameter file name from *pmcmd* or by editing the session in the Workflow Manager.

To use a mapping parameter, complete the following steps:

1. Create a mapping parameter.
2. Use the mapping parameter.
3. Define the parameter value.

Step 1. Create a Mapping Parameter

You can create mapping parameters for any mapping or mapplet. You can declare as many mapping parameters as you need. Once declared, use the parameter in the mapping or mapplet.

To create a mapping parameter:

1. In the Mapping Designer, click Mappings > Parameters and Variables. Or, in the Mapplet Designer, click Mapplet > Parameters and Variables.
2. Click the Add button.
3. Enter the following information and click OK:

Field	Description
Name	Parameter name. Name parameters $$*ParameterName*. The syntax for the parameter name must be $$ followed by any alphanumeric or underscore characters.
Type	Variable or parameter. Select Parameter.
Datatype	Datatype of the parameter. Select a valid transformation datatype. Use any datatype except Binary or Raw.
Precision or Scale	Precision and scale of the parameter.
Aggregation	Use for variables.
IsExprVar	Determines how the Integration Service expands the parameter in an expression string. If true, the Integration Service expands the parameter before parsing the expression. If false, the Integration Service expands the parameter after parsing the expression. Default is false. **Note:** If you set this field to true, you must set the parameter datatype to String, or the Integration Service fails the session.
Initial Value	Initial value of the parameter. If you do not set a value for the parameter in the parameter file, the Integration Service uses this value for the parameter during sessions. If this value is also undefined, the Integration Service uses a default value based on the datatype of the mapping variable. Use any of the following formats for initial values for Date/Time parameters: - MM/DD/RR - MM/DD/RR HH24:MI:SS - MM/DD/YYYY - MM/DD/YYYY HH24:MI:SS.US
Description	Description associated with the parameter.

Step 2. Use a Mapping Parameter

After you create a parameter, use it in the Expression Editor of any transformation in a mapping or mapplet. You can also use it in Source Qualifier transformations and reusable transformations.

In a Source Qualifier transformation, mapping parameters appear on the Variables tab in the SQL Editor. Use the following rules and guidelines when you use mapping parameters in a Source Qualifier transformation:

- Enclose string parameters in string identifiers appropriate to the source system.
- When necessary, change the format of the datetime parameter to match the format in the source.

You can also use mapping parameters in the Expression Editor. When using mapping parameters in the Expression Editor, do not enclose string parameters in string identifiers. The Integration Service handles parameters just like any other port identifiers.

Use mapping parameters in reusable transformations.

You can also use mapping parameters in transformation overrides in the session properties in the Workflow Manager. You can override properties such as a filter or user-defined join in a Source Qualifier transformation.

Step 3. Define a Parameter Value

Before you run a session, define values for mapping parameters in the parameter file. When you do not define a parameter in the parameter file, the Integration Service gets the parameter value from another place. The Integration Service looks for the value in the following order:

1. Value in parameter file
2. Value in pre-session variable assignment
3. Initial value saved in the repository
4. Datatype default value

Mapping Variables

In the Designer, you can create mapping variables in a mapping or mapplet. After you create a mapping variable, it appears in the Expression Editor. You can then use it in any expression in the mapping or mapplet. You can also use mapping variables in a source qualifier filter, user-defined join, or extract override, and in the Expression Editor of reusable transformations.

Unlike mapping parameters, mapping variables are values that can change between sessions. The Integration Service saves the latest value of a mapping variable to the repository at the end of each successful session. During the next session run, it evaluates all references to the mapping variable to the saved value. You can override a saved value with the parameter file. You can also clear all saved values for the session in the Workflow Manager.

You might use a mapping variable to perform an incremental read of the source. For example, you have a source table containing timestamped transactions and you want to evaluate the transactions on a daily basis. Instead of manually entering a session override to filter source data each time you run the session, you can create a mapping variable, $$IncludeDateTime. In the source qualifier, create a filter to read only rows whose transaction date equals $$IncludeDateTime, such as:

```
TIMESTAMP = $$IncludeDateTime
```

In the mapping, use a variable function to set the variable value to increment one day each time the session runs. If you set the initial value of $$IncludeDateTime to 8/1/2004, the first time the Integration Service runs the session, it reads only rows dated 8/1/2004. During the session, the Integration Service sets $ $IncludeDateTime to 8/2/2004. It saves 8/2/2004 to the repository at the end of the session. The next time it runs the session, it reads only rows from August 2, 2004.

Variable Values

The Integration Service holds two different values for a mapping variable during a session run:

- Start value of a mapping variable
- Current value of a mapping variable

The current value of a mapping variable changes as the session progresses. To use the current value of a mapping variable within the mapping or in another transformation, create the following expression with the SETVARIABLE function:

```
SETVARIABLE($$MAPVAR,NULL)
```

At the end of a successful session, the Integration Service saves the final current value of a mapping variable to the repository.

Start Value

The start value is the value of the variable at the start of the session. The start value could be a value defined in the parameter file for the variable, a value assigned in the pre-session variable assignment, a value saved in the repository from the previous run of the session, a user defined initial value for the variable, or the default value based on the variable datatype. The Integration Service looks for the start value in the following order:

1. Value in parameter file
2. Value in pre-session variable assignment
3. Value saved in the repository
4. Initial value
5. Datatype default value

For example, you create a mapping variable in a mapping or mapplet and enter an initial value, but you do not define a value for the variable in a parameter file. The first time the Integration Service runs the session, it evaluates the start value of the variable to the configured initial value. The next time the session runs, the Integration Service evaluates the start value of the variable to the value saved in the repository. If you want to override the value saved in the repository before running a session, you need to define a value for the variable in a parameter file. When you define a mapping variable in the parameter file, the Integration Service uses this value instead of the value saved in the repository or the configured initial value for the variable. When you use a mapping variable ('$$MAPVAR') in an expression, the expression always returns the start value of the mapping variable. If the start value of MAPVAR is 0, then $$MAPVAR returns 0.

Current Value

The current value is the value of the variable as the session progresses. When a session starts, the current value of a variable is the same as the start value. As the session progresses, the Integration Service calculates the current value using a variable function that you set for the variable. Unlike the start value of a mapping variable, the current value can change as the Integration Service evaluates the current value of a variable as each row passes through the mapping. The final current value for a variable is saved to the repository at the end of a successful session. When a session fails to complete, the Integration Service does not update the value of the variable in the repository. The Integration Service states the value saved to the repository for each mapping variable in the session log.

Variable Datatype and Aggregation Type

When you declare a mapping variable in a mapping, you need to configure the datatype and aggregation type for the variable.

The datatype you choose for a mapping variable allows the Integration Service to pick an appropriate default value for the mapping variable. The default is used as the start value of a mapping variable when there is no value defined for a variable in the parameter file, in the repository, and there is no user defined initial value.

The Integration Service uses the aggregate type of a mapping variable to determine the final current value of the mapping variable. When you have a pipeline with multiple partitions, the Integration Service combines the variable value from each partition and saves the final current variable value into the repository.

You can create a variable with the following aggregation types:

- Count
- Max
- Min

You can configure a mapping variable for a Count aggregation type when it is an Integer or Small Integer. You can configure mapping variables of any datatype for Max or Min aggregation types.

To keep the variable value consistent throughout the session run, the Designer limits the variable functions you use with a variable based on aggregation type. For example, use the SetMaxVariable function for a variable with a Max aggregation type, but not with a variable with a Min aggregation type.

The following table describes the available variable functions and the aggregation types and datatypes you use with each function:

Variable Function	Valid Aggregation Types	Valid Datatype
SetVariable	Max or Min	All transformation datatypes except binary datatype.
SetMaxVariable	Max only	All transformation datatypes except binary datatype.
SetMinVariable	Min only	All transformation datatypes except binary datatype.
SetCountVariable	Count only	Integer and small integer datatypes only.

For multiple target load order groups, the mapping variable value in a target load order group depends on the variable aggregation type and the variable value in the previous target load order group. After every target load order group runs, the Integration Service calculates the mapping variable values to use in the next target load order group based on the variable aggregation type.

For example, a session contains two target load order groups. You have set Max as the aggregation type of the mapping variable.

When the first target load order group runs, you have set the following different values using the SetVariable function for the mapping variable, $$MAPVAR:

1. SetVariable($$MAPVAR,20)
2. SetVariable($$MAPVAR,10)
3. SetVariable($$MAPVAR,60)
4. SetVariable($$MAPVAR,30)

At the end of the first target load order group run, the Integration Service calculates the Max of all the four values of $$MAPVAR. As the Max of the four values is 60, so the Integration Service uses 60 as the initial value of the mapping variable, $$MAPVAR, in the next target load order group.

Variable Functions

Variable functions determine how the Integration Service calculates the current value of a mapping variable in a pipeline. Use variable functions in an expression to set the value of a mapping variable for the next session run. The transformation language provides the following variable functions to use in a mapping:

- **SetMaxVariable.** Sets the variable to the maximum value of a group of values. It ignores rows marked for update, delete, or reject. To use the SetMaxVariable with a mapping variable, the aggregation type of the mapping variable must be set to Max.
- **SetMinVariable.** Sets the variable to the minimum value of a group of values. It ignores rows marked for update, delete, or reject. To use the SetMinVariable with a mapping variable, the aggregation type of the mapping variable must be set to Min.
- **SetCountVariable.** Increments the variable value by one. In other words, it adds one to the variable value when a row is marked for insertion, and subtracts one when the row is marked for deletion. It ignores rows marked for update or reject. To use the SetCountVariable with a mapping variable, the aggregation type of the mapping variable must be set to Count.
- **SetVariable.** Sets the variable to the configured value. At the end of a session, it compares the final current value of the variable to the start value of the variable. Based on the aggregate type of the variable, it saves a final value to the repository. To use the SetVariable function with a mapping variable, the aggregation type of the mapping variable must be set to Max or Min. The SetVariable function ignores rows marked for delete or reject.

Use variable functions only once for each mapping variable in a pipeline. The Integration Service processes variable functions as it encounters them in the mapping. The order in which the Integration Service encounters variable functions in the mapping may not be the same for every session run. This may cause inconsistent results when you use the same variable function multiple times in a mapping.

The Integration Service does not save the final current value of a mapping variable to the repository when any of the following conditions are true:

- The session fails to complete.
- The session is configured for a test load.
- The session is a debug session.
- The session runs in debug mode and is configured to discard session output.

Mapping Variables in Mapplets

When you declare a mapping variable for a mapplet and use the mapplet multiple times within the same mapping, the same mapping variable value is shared across all mapplet instances.

Using Mapping Variables

To use mapping variables, complete the following steps:

1. Create a mapping variable.
2. Use the variable and set the variable value.
3. Override or clear the variable value.

Step 1. Create a Mapping Variable

You can create a mapping variable for any mapping or mapplet. You can create as many variables as you need. Once created, use the variable in the mapping or mapplet.

To create a mapping variable:

1. In the Mapping Designer, click Mappings > Parameters and Variables. Or, in the Mapplet Designer, click Mapplet > Parameters and Variables.
2. Click the Add button.
3. Specify the variable information.

   The following table describes the options on the Declare Parameters and Variables dialog box:

Field	Required/ Optional	Description
Name	Required	Variable name. Name variables $$*VariableName*. The syntax for the variable name must be $$ followed by any alphanumeric or underscore characters.
Type	Required	Variable or parameter. Select Variable.
Datatype	Required	Datatype of the variable. Select a valid transformation datatype. Use any datatype except Binary. The datatype you select can affect the Aggregation type you can select. For example, when you create a String variable, you cannot configure it with a Count aggregation type.
Precision or Scale	Required	Precision and scale of the variable.
Aggregation	Required	Aggregation type of the variable. Determines the type of calculation you can perform with the variable. - Set the aggregation to Count if you want to use the mapping variable to count number of rows read from source. - Set the aggregation to Max if you want to use the mapping variable to determine a maximum value from a group of values. - Set the aggregation to Min if you want to use the mapping variable to determine a minimum value from a group of values.
IsExprVar	Required	Determines how the Integration Service expands the variable in an expression string. If true, the Integration Service expands the variable before parsing the expression. If false, the Integration Service expands the variable after parsing the expression. Default is false. **Note:** If you set this field to true, you must set the variable datatype to String, or the Integration Service fails the session.

Field	Required/ Optional	Description
Initial Value	Optional	Initial value of the variable. The Integration Service uses this value for the variable when the variable value is not saved in the repository or defined in the parameter file. If this value is also undefined, the Integration Service uses a default value based on the datatype of the mapping variable. Use any of the following formats for initial values for datetime variables: - MM/DD/RR - MM/DD/RR HH24:MI:SS - MM/DD/YYYY - MM/DD/YYYY HH24:MI:SS.US
Description	Optional	Description associated with the variable.

4. Click OK.

Step 2. Set a Mapping Variable Value

After you declare a variable, use it in any expression in the mapping or mapplet. You can also use a mapping variable in a Source Qualifier transformation or reusable transformation.

In a Source Qualifier transformation, mapping variables appear on the Variables tab in the SQL Editor. When using mapping variables in a Source Qualifier transformation follow these rules:

- Enclose string variables in string identifiers, such as single quotation marks, to indicate the variable is a string.
- When necessary, change the format of the datetime variable to match the format in the source. The Integration Service converts dates from the PowerCenter default date format to the default date format of the source system.

In other transformations in a mapplet or mapping, mapping variables appear in the Expression Editor. When you write expressions that use mapping variables, you do not need string identifiers for string variables.

Use mapping variables in reusable transformations. When you validate the expression, the Designer treats the variable as an Integer datatype.

You can also use mapping variables in transformation overrides in the session properties. You can override properties such as a filter or user-defined join in a Source Qualifier transformation.

When you use a mapping variable, you have to determine how to set the value of the mapping variable. Use a variable function to set a variable value. Use a variable function in any of the following transformations:

- Expression
- Filter
- Router
- Update Strategy

Step 3. Override or Clear Saved Values

After a session completes successfully, the Integration Service saves the final value of each variable in the repository. When you do not want to use that value the next time you run the session, you can override the value in the parameter file or the pre-session variable assignment in the session properties.

When you do not want to use any of the variable values saved for a session, you can clear all saved values. You can clear variable values for a session using the Workflow Manager. After you clear variables values from the repository, the Integration Service runs the session as if for the first time.

Defining Expression Strings in Parameter Files

The Integration Service expands mapping parameters and variables when you run a session. If you use a mapping parameter or variable in an expression, the Integration Service expands the parameter or variable after it parses the expression. You might want the Integration Service to expand a parameter or variable before it parses the expression when you create expressions to represent business rules that change frequently. Define the expressions in a parameter file so you do not have to change the mappings every time the business rules change.

For example, you create an expression that generates a color name based on an ID string as follows:

```
IIF(color='A0587','white')
```

The next month, you modify the expression as follows:

```
IIF(color='A0587','white',IIF(color='A0588','off white'))
```

Instead of updating the mappings that use this expression every time the business rule changes, you can define the expression in a parameter file and update the file when the expression changes.

To define an expression in a parameter file, set up the mapping and workflow as follows:

1. Create a mapping parameter or variable to store the color name expression. For example, create a mapping parameter, $$ExpColor.
2. For mapping parameter $$ExpColor, set the IsExprVar property to true. You must also set the datatype for the parameter to String or the Integration Service fails the session.
3. In the Expression transformation, set the output port to the following expression:

   ```
   $$ExpColor
   ```

4. Configure the session or workflow to use a parameter file.
5. In the parameter file, set $$ExpColor to the correct expression. For example:

   ```
   $$ExpColor=IIF(color='A0587','white')
   ```

Because IsExprVar for mapping parameter $$ExpColor is set to true, the Integration Service expands the parameter before it parses the expression. Rows with color ID “A0587” return the string “white.” If IsExprVar is set to false, the Integration Service expands the parameter after it parses the expression. Therefore, all rows return the string “IIF(color=‘A0587’,‘white’).”

When the color name expression changes, you can update the value of the mapping parameter in the parameter file. You do not need to modify the mapping.

Tips for Mapping Parameters and Variables

Enter initial values for mapping parameters and variables.

When you know a logical default value for a mapping parameter or variable, use it as the initial value when you create the parameter or variable. This allows the Integration Service to use the default value instead of a datatype-based default value.

Enclose string and datetime parameters and variables in quotes in the SQL Editor.

When you use a string parameter or variable in a Source Qualifier transformation, enclose it in string identifiers recognized by the source system, such as single quotation marks.

Save all parameter files in one of the process variable directories.

If you keep all parameter files in one of the process variable directories, such as $SourceFileDir, use the process variable in the session property sheet. If you need to move the source and parameter files at a later date, you can update all sessions by changing the process variable to point to the new directory.

Create separate parameter files for reusable sets of parameters.

When you plan to rotate through sets of parameters for a session, create a parameter file for each set of parameters. Use *pmcmd* to start the session and specify the name of the parameter file you want to use, or edit the session to use a different parameter file.

Troubleshooting Mapping Parameters and Variables

I created a mapping variable for a mapping, but each time I run the session, the value of the variable stays the same.

You might not have included a variable function in the mapping. To change the value of a variable from session to session, use a variable function.

Or, you might have configured a value for the variable in the parameter file. When you define a value for a mapping variable in the parameter file, the Integration Service uses that value for the variable. See the session log to determine what start value the Integration Service uses for each variable.

In the parameter file, I configured values for parameters in a mapplet, but they are not being used in the session.

Mapping parameter and variable values in mapplets must be preceded by the mapplet name in the parameter file, as follows:

```
mappletname.parameter=value
mappletname.variable=value
```

I cannot find the variable functions in the Expression Editor of the Rank or Aggregator transformation.

You cannot use variable functions in the Rank or Aggregator transformation. Use a different transformation for variable functions.

CHAPTER 8

Working with User-Defined Functions

This chapter includes the following topics:

- Working with User-Defined Functions Overview, 163
- Creating User-Defined Functions, 163
- Managing User-Defined Functions, 165
- Creating Expressions with User-Defined Functions, 167

Working with User-Defined Functions Overview

User-defined functions extend the PowerCenter transformation language. You can create and manage user-defined functions with the PowerCenter transformation language in the Designer. You can add them to expressions in the Designer or Workflow Manger to reuse expression logic and build complex expressions. User-defined functions are available to other users in a repository.

Example

You want to remove leading and trailing spaces from last names. You can create a user-defined function named RemoveSpaces to perform the LTRIM and RTRIM functions. When you configure the user-defined function, you enter the following expression:

```
LTRIM( RTRIM( name))
```

After you create the function, you can create the following expression in an Expression transformation to remove leading and trailing spaces from last names:

```
:UDF.REMOVESPACES(LAST_NAME)
```

The user-defined function name is prefaced by :UDF. The port name is LAST_NAME. It is the argument of the expression.

Creating User-Defined Functions

You create user-defined functions in the Designer.

The following table shows the properties you configure when you create a user-defined function:

Property	Description
Name	Name of the function. The name must begin with a letter and can contain letters, numbers, and the following special characters: _ @ $ # The name cannot exceed 80 characters and cannot contain spaces.
Type	Public if the function is callable from any expression. Private if the function is only callable from another user-defined function.
Description	Description of the function.
Return Type	Datatype of the values the function returns. The Designer determines the datatype when you create a valid function.
Arguments	Arguments you want to include in the function. Specify the argument name, transformation datatype, precision, and scale to determines the datatype, precision, and scale of the function input values.
Expression	Expression that defines the function. Configure syntax in the Expression Editor. Use the arguments you configured for the function. You can also use transformation language functions, custom functions, or other user-defined functions. Follow the PowerCenter transformation language rules and guidelines. **Note:** If you enter the syntax in the Expression window, validate the function from the Tools menu.

Configuring the Function Type

You can place user-defined functions in other user-defined functions. You can also configure a user-defined function to be callable from expressions. Callable means that you can place user-defined functions in an expression.

Select one of the following options when you configure a user-defined function:

- **Public.** Callable from any user-defined function, transformation expression, link condition expression, or task expression.
- **Private.** Callable from another user-defined function. Create a private function when you want the function to be part of a more complex function. The simple function may not be usable independently of the complex function.

After you create a public user-defined function, you cannot change the function type to private.

Although you can place a user-defined function in another user-defined function, a function cannot refer to itself. For example, the user-defined function RemoveSpaces includes a user-defined function TrimLeadingandTrailingSpaces. TrimLeadingandTrailingSpaces cannot include RemoveSpaces. Otherwise, RemoveSpaces is invalid.

Configuring Public Functions that Contain Private Functions

When you include ports as arguments in a private user-defined function, you must also include the ports as arguments in any public function that contains the private function. Use the same datatype and precision for the arguments in the private and public function.

For example, you define a function to modify order IDs to include 'INFA' and the customer ID. You first create the following private function called ConcatCust that concatenates 'INFA' with the port CUST_ID:

```
CONCAT ('INFA', CUST_ID)
```

After you create the private function, you create a public function called ConcatOrder that contains ConcatCust:

```
CONCAT (:UDF.CONCATCUST( CUST_ID), ORDER_ID)
```

When you add ConcatCust to ConcatOrder, you add the argument CUST_ID with the same datatype and precision to the public function.

Note: If you enter a user-defined function when you manually define the public function syntax, you must prefix the user-defined function with :UDF.

Steps to Create a User-Defined Function

Use the following procedure to create a user-defined function:

1. Click Tools > User-Defined Functions > New.

   The Edit User-Defined Function dialog box appears.
2. Enter a function name.
3. Select a function type.
4. Optionally, enter a description of the user-defined function.

   You can enter up to 2,000 characters.
5. Create arguments for the user-defined function.

   When you create arguments, configure the argument name, transformation datatype, precision, and scale.
6. Click Launch Editor to create an expression that contains the arguments you defined in step 5.
7. Click OK.

   The Designer assigns the datatype of the data the expression returns.
8. Click OK.

Managing User-Defined Functions

You can perform the following tasks to manage user-defined functions:

- Edit a user-defined function.
- Validate a user-defined function.
- View properties of a user-defined function.
- View user-defined function dependencies.
- Export a user-defined function.
- Manage versioning for a user-defined function.

The following table describes the user-defined function management tasks and lists where you can perform each task:

Task	Area to Perform the Task
Edit a user-defined function	- Click Tools > User-Defined Functions. - Right-click a user-defined function.
Validate a user-defined function	- Click Tools > User-Defined Functions. - Right-click a user-defined function in the Navigator, Query Results window, or View History window.
Delete a user-defined function	- Click Tools > User-Defined Functions.
View properties of a user-defined function	- Click Tools > User-Defined Functions. - Right-click a user-defined function.
View user-defined function dependencies	- Click Tools > User-Defined Functions. - Right-click a user-defined function.
Export a user-defined function	- Click Tools > User-Defined Functions. - Right-click a user-defined function.
Manage versioning for the user-defined function	- Click Tools > User-Defined Functions. - Right-click a user-defined function.

Editing User-Defined Functions

You can edit a user-defined function to change the function properties. The changes propagate to all user-defined functions and expressions that use the function.

Use the following rules and guidelines when you edit a user-defined function:

- If you change the user-defined function name, the Designer does not propagate the name change to expressions within the object. Mappings and workflows that use an expression with the user-defined function may be invalid.
- If you change the expression of a user-defined function, the Designer may change the datatype of the values the function returns when used in an expression.
- You cannot change a user-defined function from public to private.
- If a user-defined function is invalid, mappings and workflows that use the user-defined function may also be invalid. Validate the mappings or workflows.

Deleting User-Defined Functions

When you delete a user-defined function, the Repository Service deletes the function from other user-defined functions and expressions that use it. As a result, these user-defined functions and expressions may be invalid. Mappings and workflows that use the user-defined function or an expression with the user-defined function may also be invalid. Validate the mappings or workflows.

Exporting User-Defined Functions

You can export a user-defined function to XML. You can then import it into other repositories or repository folders.

If you export a mapping or workflow with a user-defined function, the PowerCenter Client exports the user-defined function.

Validating User-Defined Functions

You can validate a user-defined function from the following areas:

- Expression Editor when you create or edit a user-defined function
- Tools menu
- Query Results window
- View History window

When you validate a user-defined function, the PowerCenter Client does not validate other user-defined functions and expressions that use the function. If a user-defined function is invalid, any user-defined function and expression that uses the function is also invalid. Similarly, mappings and workflows that use the user-defined function are invalid.

Copying and Deploying User-Defined Functions

When you copy an object that contains a user-defined function, the Copy Wizard also copies the user-defined function.

When you deploy a static deployment group that contains user-defined functions, the Copy Wizard also copies the functions. When you deploy a dynamic deployment group that contains user-defined functions, the Copy Wizard does not deploy the functions that are not included in an expression.

Creating Expressions with User-Defined Functions

You can add a user-defined function to an expression. If you enter a user-defined function when you manually create an expression, you must prefix the user-defined function with :UDF. When you create an expression with the Expression Editor, valid user-defined functions display on the User-Defined Functions tab. Use user-defined functions as you would any other function.

If you create a user-defined function, which is only usable in the Designer, the function only displays in the Designer.

When you select a user-defined function, the Expression Editor shows the function syntax in the following format:

```
<return datatype> <function name> (<argument 1> as <datatype>,
<argument N> as <datatype>)
```

For example:

```
NSTRING RemoveSpaces(NAMES as string)
```

When you add the function to the Formula window, the function includes the prefix :UDF, as in the following example:

```
:UDF.RemoveSpaces( )
```

When you validate the expression, PowerCenter does not validate the user-defined function. It only validates the expression.

CHAPTER 9

Using the Debugger

This chapter includes the following topics:

- Using the Debugger Overview, 168
- Creating Breakpoints, 170
- Configuring the Debugger, 176
- Running the Debugger, 179
- Monitoring the Debugger, 182
- Modifying Data, 187
- Evaluating Expressions, 188
- Copying Breakpoint Information and Configuration, 189
- Troubleshooting the Debugger, 190

Using the Debugger Overview

You can debug a valid mapping to gain troubleshooting information about data and error conditions. To debug a mapping, you configure and run the Debugger from within the Mapping Designer. The Debugger uses a session to run the mapping on the Integration Service. When you run the Debugger, it pauses at breakpoints and you can view and edit transformation output data.

You might want to run the Debugger in the following situations:

- **Before you run a session.** After you save a mapping, you can run some initial tests with a debug session before you create and configure a session in the Workflow Manager.
- **After you run a session.** If a session fails or if you receive unexpected results in the target, you can run the Debugger against the session. You might also want to run the Debugger against a session if you want to debug the mapping using the configured session properties.

Debugger Session Types

You can select three different debugger session types when you configure the Debugger. The Debugger runs a workflow for each session type. You can choose from the following Debugger session types when you configure the Debugger:

- **Use an existing non-reusable session.** The Debugger uses existing source, target, and session configuration properties. When you run the Debugger, the Integration Service runs the non-reusable session and the existing workflow. The Debugger does not suspend on error.

- **Use an existing reusable session.** The Debugger uses existing source, target, and session configuration properties. When you run the Debugger, the Integration Service runs a debug instance of the reusable session and creates and runs a debug workflow for the session.
- **Create a debug session instance.** You can configure source, target, and session configuration properties through the Debugger Wizard. When you run the Debugger, the Integration Service runs a debug instance of the debug workflow and creates and runs a debug workflow for the session.

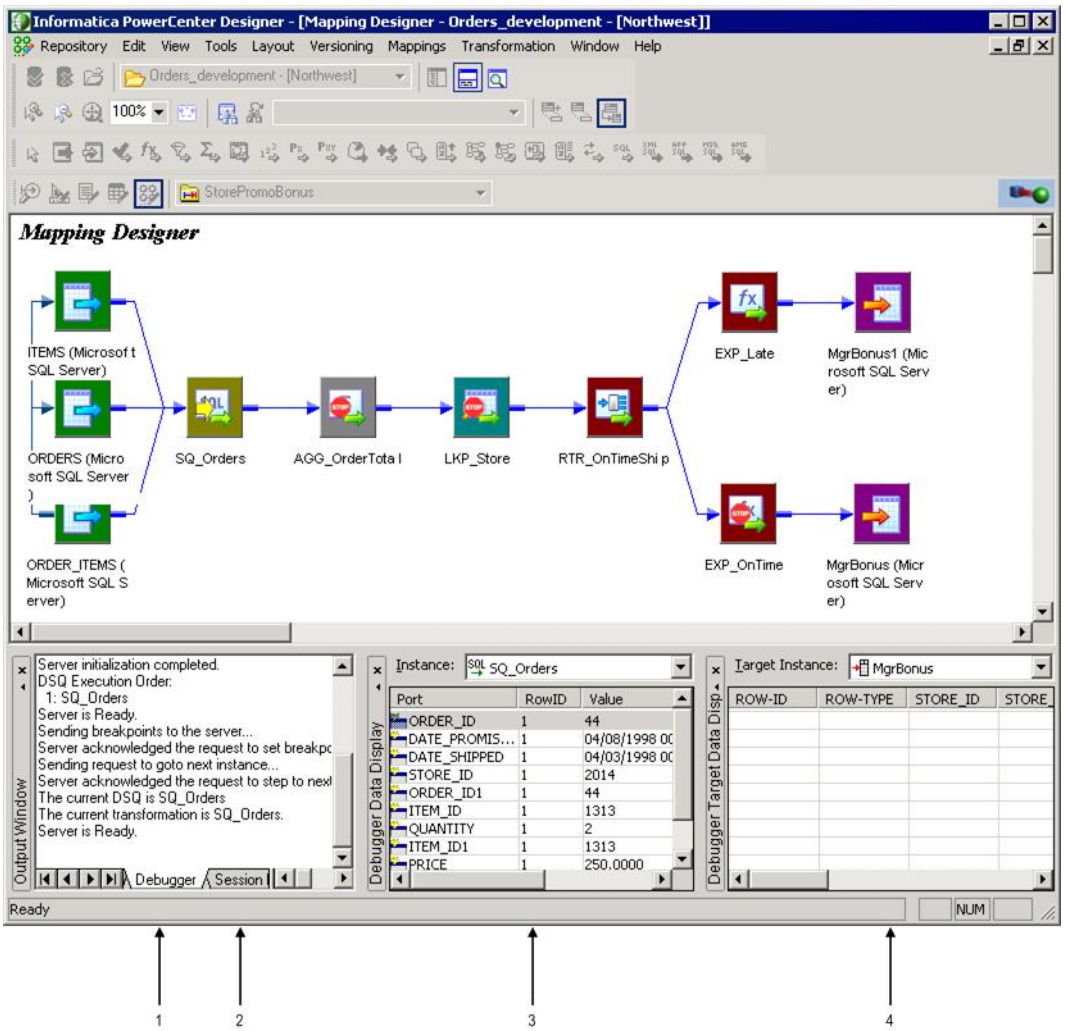
Debug Process

To debug a mapping, complete the following steps:

1. **Create breakpoints.** Create breakpoints in a mapping where you want the Integration Service to evaluate data and error conditions.
2. **Configure the Debugger.** Use the Debugger Wizard to configure the Debugger for the mapping. Select the session type the Integration Service uses when it runs the Debugger. When you create a debug session, you configure a subset of session properties within the Debugger Wizard, such as source and target location. You can also choose to load or discard target data.
3. **Run the Debugger.** Run the Debugger from within the Mapping Designer. When you run the Debugger, the Designer connects to the Integration Service. The Integration Service initializes the Debugger and runs the debugging session and workflow. The Integration Service reads the breakpoints and pauses the Debugger when the breakpoints evaluate to true.
4. **Monitor the Debugger.** While you run the Debugger, you can monitor the target data, transformation and mapplet output data, the debug log, and the session log. When you run the Debugger, the Designer displays the following windows:
   - **Debug log.** View messages from the Debugger.
   - **Target window.** View target data.
   - **Instance window.** View transformation data.
5. **Modify data and breakpoints.** When the Debugger pauses, you can modify data and see the effect on transformations, mapplets, and targets as the data moves through the pipeline. You can also modify breakpoint information.

The Designer saves mapping breakpoint and Debugger information in the workspace files. You can copy breakpoint information and the Debugger configuration to another mapping. If you want to run the Debugger from another PowerCenter Client machine, you can copy the breakpoint information and the Debugger configuration to the other PowerCenter Client machine.

The following figure shows the windows in the Mapping Designer that appears when you run the Debugger:



1. Debugger log.
2. Session log.
3. Instance window.
4. Target window.

Creating Breakpoints

Before you run the Debugger, use the Breakpoint Editor in the Mapping Designer to create breakpoint conditions in a mapping. You can create data or error breakpoints for transformations or for global conditions. When you run the Debugger, the Integration Service pauses the Debugger when a breakpoint evaluates to true. You can review and modify transformation data, and then continue the session.

Note: You cannot create breakpoints for mapplet Input and Output transformations.

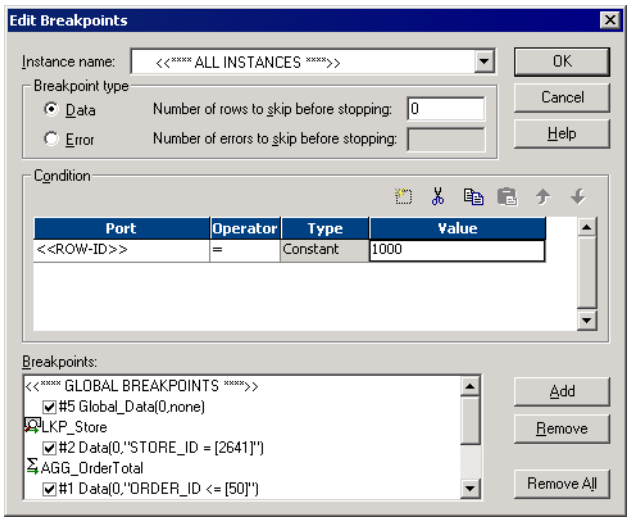
A breakpoint can consist of an instance name, a breakpoint type, and a condition. When you enter breakpoints, set breakpoint parameters in the following order:

1. **Select the instance name.** Choose to create a global breakpoint or a breakpoint against a single transformation within the mapping or mapplet.
2. **Select the breakpoint type.** Choose to evaluate error or data conditions. You can also choose the number of rows to skip for the breakpoint type.
3. **Enter the condition.** Enter data conditions for global or transformation data breakpoints. You cannot enter conditions for error breakpoints.

After you set the instance name, breakpoint type, and optional data condition, you can view each parameter in the Breakpoints section of the Breakpoint Editor:

- **Instance name.** From the Instance Name parameter.
- **Breakpoint enabled flag.** Enable in the View Breakpoints section.
- **Breakpoint number.** The Designer assigns a number based on the quantity of breakpoints you create in the mapping. These numbers are sequential. If you remove a breakpoint, the Designer renumbers them when you close and reopen the Breakpoint Editor.
- **Breakpoint type.** From the Breakpoint Type section.
- **Number of rows to skip.** From the Breakpoint Type section.
- **Data condition.** From the Condition section.

The following figure shows the parameters in the Breakpoints section:



For the Lookup transformation, LKP_Store, the check indicates the breakpoint is enabled. “#2” is the sequential breakpoint number, “Data” indicates the breakpoint type, “0” indicates the number of breakpoint rows to skip, and “STORE_ID=[2641]” is the data condition.

Selecting the Instance Name

When you select the instance name, you can create a breakpoint for an individual transformation, or you can create a global breakpoint:

- **Transformation instance.** Select a transformation from a mapping or mapplet to configure a breakpoint condition that the Integration Service evaluates when it processes that transformation. The naming convention for transformations within mapplets is *MappletName.TransformationName*.

- **Global instance.** Select Global to configure a breakpoint condition that the Integration Service evaluates when it processes each transformation in the mapping.

Creating Error Breakpoints

When you create an error breakpoint, the Debugger pauses when the Integration Service encounters error conditions such as a transformation error or calls to the ERROR function.

You also set the number of errors to skip for each breakpoint before the Debugger pauses:

- If you want the Debugger to pause at every error, set the number of errors to zero.
- If you want the Debugger to pause after a specified number of errors, set the number of errors greater than zero. For example, if you set the number of errors to five, the Debugger skips five errors and pauses at every sixth error.

Evaluating Error Breakpoints

The Integration Service uses the following rules to evaluate error breakpoints:

- The Integration Service encounters a null input value, and the port contains a user-defined default value of ERROR ( ).
- The Integration Service encounters an output transformation error, and the port contains the system default value of ERROR ( ). The following errors are considered transformation errors:
  - Data conversion errors, such as passing a string to date
  - Expression evaluation errors, such as dividing by zero
  - Calls to the ERROR function in an expression
- The Integration Service encounters a null input value or transformation output error, and the port contains a user-defined default value of ABORT ( ).
- The Integration Service encounters a fatal error.
- If the Integration Service encounters an error when it processes a transformation, it evaluates error breakpoints for that transformation and global error breakpoints. If any error breakpoint evaluates to true, the Debugger pauses and does not evaluate data breakpoints.

Note: When the Integration Service encounters a fatal error or an ABORT, the Debugger breaks even if it has not reached the configured number of errors to skip.

Creating Data Breakpoints

When you create a data breakpoint, the Debugger pauses when the data breakpoint evaluates to true. You can set the number of rows to skip or a data condition or both. You have the following options when you set data breakpoints:

- If you want the Debugger to pause at every row, set the number of rows to zero and do not set a data condition.
- If you want the Debugger to pause after a specified number of rows passes through the transformation, set the number of rows greater than zero. For example, if you set the number of rows to three, the Debugger skips three rows of data and pauses every fourth row.
- If you want the Debugger to pause each time the data meets a data condition, enter the data condition and set the number of rows set to zero.

- If you want the Debugger to pause at designated intervals when a row meets the data condition, you can enter the data condition and set the number of rows greater than zero. For example, if you set the number of rows to three, the Debugger skips three breakpoint rows that meet the condition and pauses at every fourth row.

The following table summarizes the options when you set data breakpoint types and conditions:

Number of Rows	Data Condition	Debugger Behavior
0	No	Pauses at every row.
>0	No	Pauses after every *n* times the number of rows passes through the transformation.
0	Yes	Pauses each time the row meets the data condition.
>0	Yes	Pauses after every *n* times the row meets the data condition.

Entering the Data Breakpoint Condition

When you create a data condition, you enter a global parameter or a parameter for a single transformation. You have different options depending on the selected instance and the type of condition you create.

Use the following syntax when you evaluate a transformation against a port or constant value:

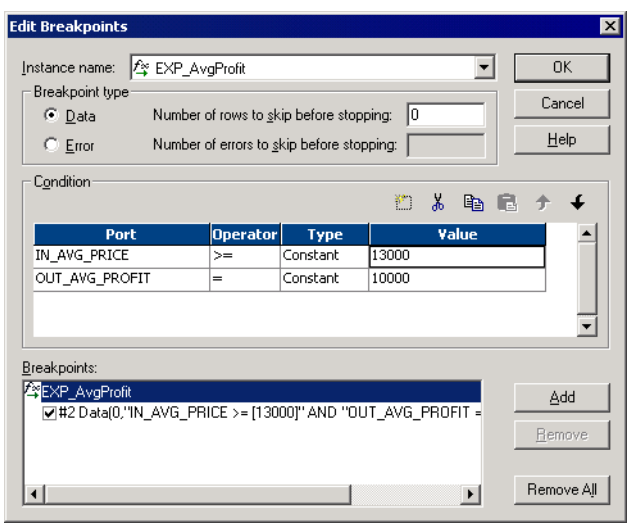
```
<port> <operator> <type> <value>
```

Use the following syntax when you check for null and default values, either globally or in a transformation:

```
<port> <operator>
```

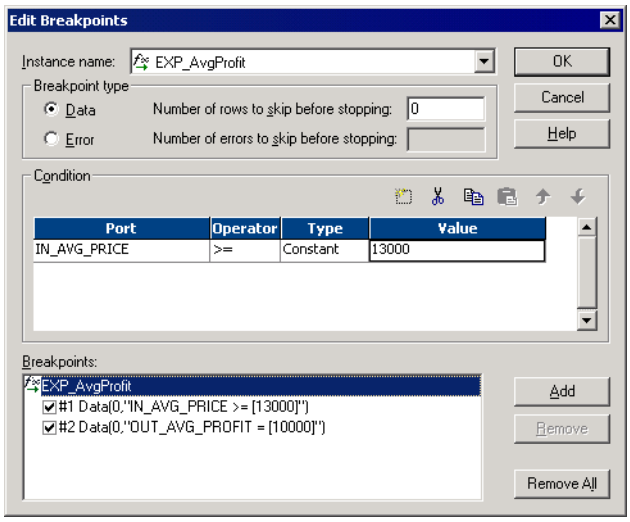
For a single transformation, you can enter multiple conditions within one breakpoint if you want the Debugger to pause when all the conditions are true. You can also enter multiple breakpoints for a single transformation if you want the Debugger to pause when at least one condition is true.

The following figure shows multiple conditions within a single breakpoint:



In this example, the Debugger pauses when both conditions are true.

The following figure shows multiple breakpoints within a single transformation:



In this example, the Debugger pauses when either condition is true.

The following table describes the condition parameters for transformation and global data breakpoints:

Parameter	Transformation Options	Global Options
Port	Choose port name, <<ROW-ID>>, <<ROW-TYPE>>, or <<ANY-PORT>>.	<ANY-PORT>. You can evaluate this condition against each port in each transformation and mapplet.
Operator	<, <=, =, >, >=, !=, ISNULL, ISDEFAULT.	ISNULL, ISDEFAULT.
Type	Constant, Port. Evaluate against a constant value or another port in the transformation.	Not available.
Value	Enter the value or select the port name.	Not available.

Entering a Transformation Data Condition

When you enter a transformation data condition, you can select a specific port in the transformation, or you can select from the following port parameters:

- **Row type.** Insert, update, delete, reject, or filtered.
- **Row ID.** Row number passing through the transformation.
- **Any port.** When you choose <<ANY-PORT>>, you can enter a condition that the Integration Service evaluates for every port in the transformation. Use the ISNULL or ISDEFAULT operator to test for null or error default values.

Note: For best results with port-to-port comparison, use ports that have the same datatype. When you compare ports that have different datatypes, the Integration Service converts the datatype of the upstream port to the datatype of the downstream port before it compares the ports, which might result in an invalid comparison.

Entering a Global Data Condition

When you enter a global data condition, enter a condition that the Integration Service evaluates null and error default values in every port in every transformation. You can choose <ANY-PORT> <ISNULL> or <ANY-PORT> <ISDEFAULT>.

Using ISNULL and ISDEFAULT

You can create ISNULL and ISDEFAULT conditions in transformation and global data breakpoints. When you use the ISNULL or ISDEFAULT operator, you cannot use the type or value in the condition.

When you create an ISNULL condition, the Debugger pauses when the Integration Service encounters a null input value, and the port contains the system default value.

When you create an ISDEFAULT condition, the Debugger pauses in the following circumstances:

- The Integration Service encounters an output transformation error, and the port contains a user-defined default value of a constant value or constant expression.
- The Integration Service encounters a null input value, and the port contains a user-defined default value of a constant value or constant expression.

Steps to Enter Breakpoints

Complete the following steps to enter breakpoints:

1. In a mapping, open the Breakpoint Editor in one of the following ways:
   - Press Alt+F9.
   - Click Mappings > Debugger > Edit Breakpoints.
   - Right-click a transformation or mapplet and choose Edit Breakpoint.

   Note: You can right-click any portion of an iconized transformation or mapplet. If the transformation or mapplet is in normal view, right-click the title bar.
2. Select the instance name in the Instance Name section.
3. Click Add to add a new breakpoint.

   If you select <<ALL INSTANCES>> and click Add, the Designer prompts you to choose a specific transformation or a global instance.

   You can click OK to create an unconditional data breakpoint.
4. Select the Breakpoint Type.
5. If you select a data breakpoint type, you can enter the condition. Enter multiple conditions if you want the Integration Service to evaluate multiple conditions.
6. Repeat steps 2 to 5 for each breakpoint.
7. Click OK.

Editing a Breakpoint

You might want to review breakpoints before or during a Debugger session. You can edit, disable, or remove breakpoints within the Breakpoint Editor.

To edit breakpoints:

1. In a mapping, open the Breakpoint Editor in one of the following ways:
   - Press Alt+F9.
   - Click Mappings > Debugger > Edit Breakpoints.
   - Right-click a transformation and choose Edit Breakpoint.

   Note: You can right-click over any portion of an iconized transformation or mapplet. If the transformation is in normal view, right-click over the title bar.
2. Select an instance name in the Instance Name section.

   Tip: Select <<ALL INSTANCES>> in the Instance Name list to view all breakpoints for a mapping.
3. Select a breakpoint in the Breakpoints section.
4. Change the condition or breakpoint type.
5. Clear the check box by a breakpoint to disable it. You can enable the breakpoint again by selecting the check box.
6. Create a new breakpoint using the instructions above for creating breakpoints.
7. Click the Remove button to remove a selected breakpoint.
8. Click the Remove All button to remove all breakpoints for the selected instance. If you want to remove all breakpoints in the mapping, select <<ALL INSTANCES>>.

Configuring the Debugger

In addition to setting breakpoints, you must also configure the Debugger. Use the Debugger Wizard in the Mapping Designer to configure the Debugger against a saved mapping. When you configure the Debugger, enter parameters such as the Integration Service, source and target type, and memory parameters. You must enable debugging in the Integration Service configuration before you use it to run the Debugger.

The Debugger Wizard has the following pages:

1. **Introduction.** This page gives you an overview of the wizard. You do not configure any properties on this page.
2. **Integration Service and session type.** Choose the Integration Service to run the session. You can also choose to run the Debugger against an existing non-reusable session, an existing reusable session, or create a debug session instance. When you run the Debugger against an existing session, the Integration Service runs the session in debug mode. When you create a debug session, you configure a subset of session properties within the Debugger Wizard, such as source and target location.
3. **Session information.** If you run the Debugger against an existing session, select the session name. If you create a debug session, configure session parameters.
4. **Session configuration.** If you create a debug session, set the session configuration.
5. **Target options.** Choose to load or discard target data.

Step 1. Debugger Introduction

The first page of the Debugger Wizard provides an overview of the wizard.

Step 2. Select Integration Service and Session Type

On the second page, you can select the following options:

- Integration Service to run the session. The list displays all Integration Services associated with the repository. Choose an Integration Service that has debugging enabled.
- Run the Debugger against an existing non-reusable session, an existing reusable session, or create a debug session instance. If you want to debug a mapping that uses session properties, such as incremental aggregation, FTP, or pre- or post-session commands, you must run an existing session in debug mode.

When you click Next, the Designer tests the connection to the Integration Service and verifies that debugging is enabled for the Integration Service. If the connection fails, the Designer prompts you to select another Integration Service.

Step 3. Select Session Information

The page that the wizard displays for session information depends on whether you choose to run a debug session or an existing session in debug mode.

Running an Existing Session in Debug Mode

If you choose to run an existing session in debug mode, the Debugger Wizard displays a list of all sessions in the current folder that use the mapping. Select the session you want to use.

You cannot run the Debugger against a session configured with multiple partitions or a session configured to run on a grid. You must either change the properties of the session or choose to create a debug session for the mapping.

Running a Debug Session

If you choose to run a debug session, you can specify some session parameters. The Debugger uses default session parameters for all other parameters that you cannot configure with a debug session. The Debugger Wizard displays a session page that contains the following tabs:

- **Connections.** Select source and target connection information.

  For relational sources and targets, you can select a database connection that is configured in the Workflow Manager. For relational targets, you can choose to truncate the target tables.

  For file sources and targets, you can override the file name. The default locations are $SourceFileDir and $TargetFileDir. To override the default location for a source or target file, use the Session Parameters tab. Do not enter the direct path in the file name field.
- **Properties.** Configure source and target properties.
- **Reader/Writer.** Configure readers and writers for the source and target instances in the mapping.

When you run a debug session, the Integration Service uses default session properties for all properties that you do not configure in the Debugger Wizard.

You can select source and target connections for each source and target instance in the debug session on the Connections tab.

You can choose the source reader type and target writer type for each source and target instance in the debug session on the Reader/Writer tab.

The Properties tab displays different source and target properties depending on the type of reader or writer you select on the Reader/Writer tab. The Integration Service uses default values for all session properties you do not configure on the Properties tab.

You can configure session properties for the Debug session.

Step 4. Set Session Configuration

When you set the debug session configuration, you configure information such as the location of files, row type, and memory.

The following table describes the session parameters for a debug session:

Session Parameter	Description
Row Type	Insert, Delete, Update, or Driven. The default row type for session using an Update Strategy is Update.
DTM Buffer Size	Default memory allocation is 12 MB. You might want to increase this to 24 MB if you run a session with large amounts of character data against an Integration Service configured to run in Unicode mode.
Parameter File	Name and location of the parameter file if the mapping uses any mapping parameters or variables. The current working directory for the Integration Service is the default location for the parameter file.
Enable High Precision	When selected, the Integration Service processes the Decimal datatype to a precision of 28. If a session does not use the Decimal datatype, do not enable high precision.
$Source Connection Value	Database connection you want the Integration Service to use for the $Source variable.
$Target Connection Value	Database connection you want the Integration Service to use for the $Target variable.
Constraint Based Load Ordering	Uses constraint-based loading.

Step 5. Set Target Options

On the last page of the Debugger Wizard, you can select the following target options:

- **Discard target data.** You can choose to load or discard target data when you run the Debugger. If you discard target data, the Integration Service does not connect to the target.
- **Display target data.** You can select the target instances you want to display in the Target window while you run a debug session.

When you click Finish, if the mapping includes mapplets, the Debugger displays the mapplet instance dialog box. Select the mapplets from this dialog box that you want to debug. To clear a selected mapplet, press the Ctrl key and select the mapplet.

When you select a mapplet to debug, the Designer expands it to display the individual transformations when the Debugger runs.

When you do not select a mapplet to debug, the Designer does not expand it in the workspace. You cannot complete the following tasks for transformations in the mapplet:

- Monitor or modify transformation data.
- Evaluate expressions.
- Edit breakpoints.

- Step to a transformation instance.

Running the Debugger

When you complete the Debugger Wizard, the Integration Service starts the session and initializes the Debugger. After initialization, the Debugger moves in and out of running and paused states based on breakpoints and commands that you issue from the Mapping Designer. The Debugger can be in one of the following states:

- **Initializing.** The Designer connects to the Integration Service.
- **Running.** The Integration Service processes the data.
- **Paused.** The Integration Service encounters a break and pauses the Debugger.

Note: To enable multiple users to debug the same mapping at the same time, each user must configure different port numbers in the Tools > Options > Debug tab.

The Debugger does not use the high availability functionality.

Initializing State

When you run the Debugger, the Designer connects to the Integration Service, and the Integration Service initializes the session. During initialization, the Designer closes the Navigator window and disables functions, such as switching to another tool, saving the repository, or opening a folder. These functions remain disabled until the Debugger stops.

Running State

When initialization completes, the Debugger moves to the paused state and waits for the command to continue processing. When you continue, the Debugger moves to the running state. The Integration Service transforms data and evaluates data against breakpoint conditions. The Debugger remains in the running state until the Integration Service encounters a breakpoint, you issue a break command, or the session ends.

Paused State

The Debugger pauses when the Integration Service encounters a break. The following break conditions cause the Debugger to pause:

- The Integration Service encounters a configured breakpoint condition.
- You instruct the Integration Service to continue to an instance that does not have an associated breakpoint.
- You issue a manual break command.
- The Integration Service encounters a fatal error.
- The Integration Service completes evaluation of all data. The Debugger tab displays a message that the session is complete. When you continue the session, the Debugger clears the Target and Transformation windows.

While the Debugger pauses, you can review and modify transformation output data. The Debugger remains paused until you continue or end the session.

Debugger Tasks

You can perform multiple tasks when you run the Debugger. The type of information that you monitor and the tasks that you perform can vary depending on the Debugger state. For example, you can monitor logs in all three Debugger states, but you can only modify data when the Debugger is in the paused state. You can complete the following types of tasks:

- **Monitor the session.** While the Integration Service runs the Debugger, the Mapping Designer displays indicators and windows to help you monitor the session.
- **Modify data and breakpoints.** When the Debugger pauses, you can modify output data, row indicators, and breakpoint conditions.
- **Evaluate expressions.** When the Debugger pauses, you can invoke the Expression Editor and evaluate an expression against the current data in a transformation. The Debugger returns the result of the expression in a message box. You can enter expressions using ports in the selected transformation. You can also evaluate mapping variables.
- **Issue commands to the Integration Service.** You can issue commands to the Integration Service, such as break, continue, or stop.

The following table describes the different tasks you can perform in each of the Debugger states:

Task	Description	Debugger State	Access
Monitor logs	Monitor the session log and the debug log in the Output window.	Initializing Running Paused	- View Output Window
Monitor target data	When the Debugger moves from the initializing state to the running state, the Designer displays the Target window. You can view updated target data as the Integration Service processes the target data.	Running Paused	- View Target Instance Window - View Target Data Command
Monitor debug indicators	Debug indicators display on mapping objects that help you monitor the breakpoints and data flow.	Running Paused	- View Mapping Objects
Monitor transformation data	When the Debugger moves from the initializing state to the running state, the Designer displays the Instance window. When the Debugger pauses, the Instance window displays data of the transformation that caused the break. You can also select other transformations to view.	Paused	- View Instance Window - Show Current Instance Command
Modify data	While the Debugger pauses, you can modify output data and row parameters.	Paused	- View Instance Window
Evaluate expressions	While the Debugger pauses, use the Expression Editor to evaluate mapping variables and expressions in a transformation.	Paused	- Evaluate Expression Command
Issue a manual break command	Issue a manual break command when you want to view transformation data.	Running	- Break Now Command

Task	Description	Debugger State	Access
Edit breakpoints	The Integration Service begins evaluating the modified breakpoints immediately.	Running Paused	- Edit Breakpoints Command
Refresh data	After you modify data, you can refresh the data. When you refresh data, the Integration Service returns the result of the modification. If you enter data that is not valid, you can edit it again before you continue processing.	Paused	- Refresh Data Command from Instance Window
Continue processing	When you finish viewing and modifying data, you can continue the Debugger. You have several options for continuing the session.	Paused	- Next Instance Command - Step to Instance Command - Continue Command
Stop the Debugger	Stop the Debugger.	Initializing Running Paused	- Stop Debugger Command
Status request	When you request the Debugger status, the Integration Service displays the status of the Debugger in the Output window.	Running Paused	- Status Request Command

Working with Persisted Values

When you run the Debugger against mappings with sequence generators and mapping variables, the Integration Service might save or discard persisted values:

- **Discard persisted values.** The Integration Service does not save final values of generated sequence numbers or mapping variables to the repository when you run a debug session or you run a session in debug mode and discard target data.
- **Save persisted values.** The Integration Service saves final values of generated sequence numbers and mapping variables to the repository when you run a session in debug mode and do not discard target data. You can view the final value for Sequence Generator and Normalizer transformations in the transformation properties.

Designer Behavior

When the Debugger starts, you cannot perform the following tasks:

- Close the folder or open another folder.
- Use the Navigator.
- Perform repository functions, such as Save.
- Edit or close the mapping.
- Switch to another tool in the Designer, such as Target Designer.
- Close the Designer.

Note: Dynamic partitioning is disabled during debugging.

You can perform these tasks after the Debugger stops.

Monitoring the Debugger

When you run the Debugger, you can monitor the following information:

- **Session status.** Monitor the status of the session.
- **Data movement.** Monitor data as it moves through transformations.
- **Breakpoints.** Monitor data that meets breakpoint conditions.
- **Target data.** Monitor target data on a row-by-row basis.

The Mapping Designer displays windows and debug indicators that help you monitor the session:

- **Debug indicators.** Debug indicators on transformations help you follow breakpoints and data flow.
- **Instance window.** When the Debugger pauses, you can view transformation data and row information in the Instance window.
- **Target window.** View target data for each target in the mapping.
- **Output window.** The Integration Service writes messages to the following tabs in the Output window:
  - **Debugger tab.** The debug log displays in the Debugger tab.
  - **Session Log tab.** The session log displays in the Session Log tab.
  - **Notifications tab.** Displays messages from the Repository Service.

While you monitor the Debugger, you might want to change the transformation output data to see the effect on subsequent transformations or targets in the data flow. You might also want to edit or add more breakpoint information to monitor the session more closely.

Monitoring Debug Indicators

During a session, the mapping displays the indicators in the top left corner of transformations that help you monitor breakpoints, data flow, and Debugger status:

- **Breakpoint indicator.** After the Integration Service completes initialization, it reads the breakpoints in the mapping and displays a stop sign on each transformation to indicate breakpoint conditions. You can view breakpoint indicators while the Debugger is in the running or paused state. The Mapping Designer does not display global breakpoint indicators.
- **Current source qualifier indicator.** When the Integration Service runs the Debugger, it processes data from each source qualifier in a target load order group concurrently by default. The Mapping Designer displays a flashing arrow on all current source qualifiers.

  Note: You can configure the Integration Service to read sources connected to Joiner transformations sequentially.
- **Current transformation indicator.** The Debugger displays a solid yellow arrow that indicates the transformation or mapplet that the Integration Service was processing when it encountered a break. This transformation is called the current transformation. The Debugger displays a solid blue arrow at all other transformations that the Integration Service was processing when it encountered a break at the current transformation.
- **Debugger status.** When you start the Debugger, the Mapping Designer displays a Debugger icon by the tool bars that indicates the Debugger state. When the Debugger is initializing or running, the icon rotates. When the Debugger is paused, the icon stops.

Monitoring Transformation Data

When the Debugger pauses, it displays the current row of data in the Instance window for the transformation in the mapping or mapplet where the Integration Service encountered the break. This is the current transformation. You can view the data of any transformation at the point where the Debugger pauses by selecting another transformation from the list in the Instance window.

If you did not select any mapplet to debug after completing the Debugger Wizard, you cannot monitor or view transformations inside the mapplet when the Debugger runs.

Click Tools > Options > Debug tab to select columns to display in the Debugger Data Display window. For more information about displaying columns, see "Configuring Debug Options" on page 24.

Note: When the mapping contains a Custom transformation, the Instance window displays information for ports in all input and output groups.

You can display the following information in the Instance window:

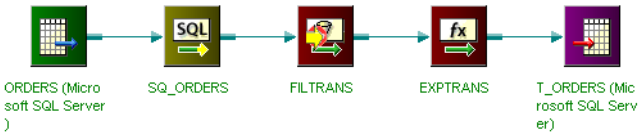
- **Port name.** Displays all ports that are connected to another transformation or target.
- **RowID.** Displays row number passing through the transformation.
- **Value.** Displays the value for each port. If the port contains binary data, this column displays <raw data>. If the port does not contain data, or if it is NULL, this column displays <no data available>. For Router transformations, the values display for input ports and output ports that meet the group condition. Output ports that do not meet the group condition display <no data available> in the value column.
- **Datatype.** Displays the datatype of the port.
- **Length/precision and scale.** Displays the length/precision and scale of the port.
- **Null indicator.** If a column contains a null value, the null indicator column is selected, and the value displays <no data available>.
- **Row type.** Insert, update, delete, reject, filtered, or not applicable. <<ROW TYPE>> displays in the Port column, and the row type value displays in the Value column. The row type does not display if it is not applicable, such as with Sequence Generator transformations.
- **Port indicator.** Displays one of the following indicators next to the port name:
  - **Current row.** Indicates the port is in the current row.
  - **Previous row.** Indicates the port is in the previous row.
  - **Current row, default value.** Indicates the value for the column in the current row is the default value.
  - **Previous row, default value.** Indicates the value for the column in the previous row is the default value.
  - **Current row, modifiable.** Indicates an output port in the current row that you can modify.

  Tip: Move the pointer across the port indicator to display tooltips about the indicator.

When the Instance window displays the current transformation or any transformation in the pipeline before the current transformation, the current rows display. If you view a transformation in the Instance window that appears in the pipeline after the current instance, the previous rows display. An indicator also displays beside the port name to designate the current or previous row.

For example, in the following mapping, FILTRANS is the current transformation, as shown by the current transformation indicator. When you view the Instance window for FILTRANS or SQ_ORDERS, you see the current row. If you switch to EXPTRANS, you see the previous row because the Integration Service has not processed the current row through EXPTRANS.

The following figure shows transformations with port processing indicators:



Current and previous row indicators can also show when the Instance window displays the default value. The Debugger uses the same rules to display the default value indicator in the Instance window that it uses to evaluate default values for breakpoints.

Note: The Debugger does not immediately drop rows flagged to drop. This includes invalid, error, and filtered rows. The Integration Service sets the ROW_FLAG to NULL and designates the row type by a negative number, such as -3 (error). You see the error indicators in the session logs with verbose data tracing only. The Integration Service drops the rows later in the pipeline.

Continuing the Debugger

After you review or modify data, you can continue the Debugger in the following ways:

- **Continue to the next break.** To continue to the next break, click Continue on the toolbar or from the Mapping > Debugger menu option. The Debugger continues running until it encounters the next break.
- **Continue to the next instance.** To continue to the next instance, click Next Instance on the toolbar or from the Mapping > Debugger menu option. The Debugger continues running until it reaches the next transformation or until it encounters a break. If the current instance has output going to more than one transformation instance, the Debugger stops at the first instance it processes. If you did not select a mapplet to debug when you completed the Debugger Wizard steps, the Debugger continues to the instance after the mapplet.
- **Step to a specified instance.** To continue to a specified instance, select the transformation instance in the mapping, then click Step to Instance on the toolbar or from the Mapping > Debugger menu option. The Debugger continues running until it reaches the selected transformation in the mapping or until it encounters a break.

  You can step to connected transformations in the mapping, even if they do not have an associated breakpoint. You cannot step to the following instances:

  - Sources
  - Targets
  - Unconnected transformations
  - Mapplets not selected for debugging

Monitoring Target Data

When the Debugger runs, the Designer caches the target data. You can view the cached data in the Target window while the Debugger is running or paused. You can view cached target data even if you configure the Debugger Wizard to discard target data.

You can view the following information for each row of target data:

- **Row ID.** Row number of the cached target data.
- **Row type.** Insert, update, delete, reject, or filtered.
- **Column name and value.** Column name and value for each row in the cache.

If the mapping has more than one target, you can choose the target you want to display. Select a target to view from the list in the Debugger Target Display window. The targets that display in the list are the targets you selected in the Debugger Wizard. If you want to view a target that you did not select in the wizard, click Mappings > Debugger > View Target Data.

The Target window displays up to 1,000 rows. After it reaches 1,000 rows, the Designer begins overwriting data from the first row. You can clear the cache and the Debugger Target Display window by right-clicking in the Debugger Target Display window, and selecting Clear Data.

Monitoring the Debug Log

When the Debugger initializes, the Integration Service begins writing the debug log in the Debugger tab of the Output window. The Integration Service writes the following types of messages to the debug log:

- Session initialization
- Acknowledge user request
- The Integration Service encounters a break
- Debug errors
- Debugger status
- Session stops

When the bottom line in the Output window reads that the Integration Service is ready, you know the Debugger is paused. The Integration Service remains paused until you issue a command, such as Continue, Step to Next, or Stop.

Note: If the Debugger does not start or if it abruptly stops, and the Integration Service does not write any message to the Output window, you can look in the Administration Console Domain page for more information.

The following table shows samples of debug log messages in the Output window:

Debugger Action	Sample Debug Log Messages

The Debugger initializes.	`Establishing a connection to the Integration Service dwilliam...`
	`Integration Service dwilliam acknowledged its ability to debug a mapping.`
	`Initializing debugger...`
	`Sending request to Integration Service dwilliam to debug mapping Customers...`
	`Waiting for Integration Service to initialize...`
	`Establishing communication with the DTM...`
	`Established communication with the DTM.`
	`Waiting for DTM initialization to complete...`
	`Integration Service is running in ASCII mode.`
	`Integration Service is running with High Precision disabled.`
	`Integration Service initialization completed.`

After initialization, the Integration Service acknowledges breakpoints in the mapping.	`DSQ Execution Order:`
	`Integration Service is Ready.`
	`Sending breakpoints to the Integration Service...`
	`Integration Service acknowledged the request to set breakpoints.`

Debugger Action	Sample Debug Log Messages

The Integration Service encounters a breakpoint.	`Integration Service has stopped at a data breakpoint.`
	`List of breakpoints that the Integration Service hit:`
	`  #3 Data(2,"CUSTOMER_ID > [14]")`
	`The current transformation is Exp_Customers.`
	`Integration Service is Ready.`

Issue the Continue command.	`Sending request to continue...` `Integration Service acknowledged the request to continue.`

Issue the Step to Next Instance command. The Integration Service acknowledges and executes the request.	`Sending request to go to next instance...`
	`Integration Service acknowledged the request to step to next instance.`
	`The current transformation is Exp_Customers.`
	`Integration Service is Ready.`

Issue the Step to Instance command. The Integration Service acknowledges and executes the request.	`Sending request to step to instance...`
	`Integration Service acknowledged the request to step to instance.`
	`The current transformation is SQ_CUSTOMERS.`
	`Integration Service is Ready.`

Modify data in the Instance window.	`Sending request to modify data for transformation Exp_Customers, port NEW_CUST_ID, to value [1234566...]`
	`Integration Service acknowledged the request to modify data.`
	`Sending request to continue...`
	`Integration Service acknowledged the request to continue.`

Session is complete. The Integration Service waits for the final command to shut down the Debugger.	`Response from the Integration Service: Session completed <successfully/with failure>. Last chance to look at target data. Debugger will shut down when we continue.` `Integration Service is Ready.`

The Debugger completes.	`Integration Service is Shutting Down...`
	`Integration Service execution completed.`
	`Debugger shutdown complete.`

To save the log when the session completes, right-click inside the Debugger tab, and choose Save Output As from the menu.

Using the Workflow Monitor

You can view running and completed workflows and sessions in the Workflow Monitor. The Integration Service uses a debug workflow to run the debug session. The Workflow Monitor displays the debug workflow and session names differently depending on the session type you chose in the Debugger Wizard.

The following table describes the workflow and session names the Workflow Monitor displays for each debug session type:

Session Type	Workflow Name	Session Name
Existing reusable session	DebugWorkflow_mappingname	DebugInst_sessionname
Existing non-reusable session	workflowname	sessionname
Debug session	DebugWorkflow_mappingname	DebugInst_DebugSession_mappingname

The Workflow Monitor displays the debug run mode in the Run Type column for all debugger sessions and workflows. You can also access the logs through the Workflow Monitor.

Note: You cannot use the Workflow Monitor to restart, abort, or stop a debug session, debug workflow, or a session run in debug mode.

Modifying Data

When the Debugger pauses, the current instance displays in the Instance window, and the current instance indicator displays on the transformation in the mapping. You can make the following modifications to the current instance when the Debugger pauses on a data breakpoint:

- **Modify output data.** You can modify output data of the current transformation. When you continue the session, the Integration Service validates the data. It performs the same validation it performs when it passes data from port to port in a regular session.
- **Change null data to not-null.** Clear the null column, and enter a value in the value column to change null data to not-null.
- **Change not-null to null.** Select the null column to change not-null data to null. The Designer prompts you to confirm that you want to make this change.
- **Modify row types.** Modify Update Strategy, Filter, or Router transformation row types.

  For Router transformations, you can change the row type to override the group condition evaluation for user-defined groups. For example, if the group condition evaluates to false, the rows are not passed through the output ports to the next transformation or target. The Instance window displays <no data available>, and the row type is filtered. If you want to pass the filtered row to the next transformation or target, you can change the row type to Insert. Likewise, for a group that meets the group condition, you can change the row type from insert to filtered.

After you change data, you can refresh the cache before you continue the session. When you issue the Refresh command, the Designer processes the request for the current transformation, and you can see if the data you enter is valid. You can change the data again before you continue the session.

Restrictions

You cannot change data for the following output ports:

- **Normalizer transformation.** Generated Keys and Generated Column ID ports.
- **Rank transformation.** RANKINDEX port.
- **Router transformation.** All output ports.

- **Sequence Generator transformation.** CURRVAL and NEXTVAL ports.
- **Lookup transformation.** NewLookupRow port for a Lookup transformation configured to use a dynamic cache.
- **Custom transformation.** Ports in output groups other than the current output group.
- **Java transformation.** Ports in output groups other than the current output group.

Additionally, you cannot change data associated with the following:

- Mapplets that are not selected for debugging
- Input or input/output ports
- Output ports when the Debugger pauses on an error breakpoint

Evaluating Expressions

When the Debugger pauses, use the Expression Editor to evaluate expressions using mapping variables and ports in a selected transformation.

You can evaluate expressions in the following transformations:

- Aggregator transformation
- Expression transformation
- Filter transformation
- Rank transformation
- Router transformation
- Update Strategy transformation

When you create an expression, you use references to ports within the transformation. The Expression Editor does not display ports for other transformations in the mapping. When you evaluate the expression, the Debugger returns the result of the expression in a message box.

Evaluating Expressions Using Mapping Variables

If you define variables in the mapping, you can evaluate the start value or current value of the mapping variable. To find the start value of a mapping variable, evaluate the mapping variable only.

The Debugger uses the following process to determine the start value of a mapping variable:

1. **Parameter file.** If you use a parameter file, the Debugger returns the value in the parameter file.
2. **Repository.** If you do not use a parameter file, the Debugger returns the value in the repository.
3. **Initial value.** If the repository does not have a stored value for the variable, the Debugger returns the initial value configured in the mapping variable.
4. **Default value.** If you do not configure an initial value for the mapping variable, the Debugger returns the default value based on datatypes.

To find the current value of a mapping variable, evaluate the mapping variable with one of the variable functions, such as SetMaxVariable or SetCountVariable. The Designer displays only the mapping variables associated with the transformations in the mapping. To view mapping variables associated with a mapplet, select the transformation within the expanded mapplet when the Debugger is running.

Steps to Evaluate Expressions

To evaluate an expression while the Debugger pauses, complete the following steps:

To evaluate an expression while the Debugger pauses:

1. Select the transformation for which you want to evaluate an expression.
2. Click Mappings > Debugger > Evaluate Expression.
3. Enter an expression in the Expression Editor that references ports in the selected transformation.
4. To validate the expression, click Validate.
5. To evaluate the expression, click Evaluate.

   If the expression is valid, the Integration Service evaluates it. The Integration Service returns the result of the expression in a message box.

Copying Breakpoint Information and Configuration

The Designer stores breakpoint information and Debugger configuration in the workspace files. If you copy a mapping or create a shortcut to another folder, or if you want to debug the mapping from another PowerCenter Client machine, you can transfer the breakpoints and configuration using the Save Configuration and Load Configuration options on the Debugger menu.

To save breakpoints and configuration associated with a mapping:

1. Open the mapping with breakpoints and configuration that you want to save.
2. Click Mappings > Debugger > Save Configuration.
3. Save the file with a .dcf extension.
4. When you choose an existing .dcf file, the Designer appends the configuration information to the file.
5. If the file contains configuration information for the mapping, the Designer prompts you to verify that you want to overwrite the configuration.

Transferring Breakpoints and Configuration

You can transfer breakpoints and configuration information saved from one mapping to a different mapping.

To load breakpoints and configuration to another mapping:

1. Make the .dcf file accessible to the PowerCenter Client machine where you want to load configurations.
2. Open the mapping you want to load configuration.
3. Click Mapping > Debugger > Load Configuration.

   The Open DCF File dialog box appears.
4. Select the .dcf configuration file and click Open.

   If there are multiple configurations in the file, or if the Designer does not find a matching mapping name, the Designer displays a list of mapping configurations in the file.
5. Select a mapping and click OK.

   The Designer loads all breakpoint and configuration for matching mapping objects. It loads all configuration that is not dependent on mapping objects, such as the Integration Service and the option to discard target data.

   Note: You cannot load breakpoints while the Debugger is active.

Troubleshooting the Debugger

I am trying to run a Debugger session and I get the following error:

```
Establishing a connection to the Integration Service <name>...
Integration Service <name> acknowledged its ability to debug a mapping.
Initializing debugger...
bind: (WSAEADDRINUSE)Address already in use
```

This error occurs when the port that the Debugger is using is already in use by another application. Assign a different port for the Debugger to use.

To assign the Debugger a different port:

1. Click Tools > Options > Debug.
2. Assign a new port number in the Port field or select Pick a Port Number Between Min and Max Automatically and assign a range of ports for the Debugger to use.

   You can select a port between 5001 and 32000 or set a range between 5001 and 32000.

CHAPTER 10

Viewing Data Lineage

This chapter includes the following topics:

- Viewing Data Lineage Overview, 191
- Configuring Data Lineage Access, 191
- Running Data Lineage from the Designer, 192

Viewing Data Lineage Overview

You can use Metadata Manager to trace the flow of data in PowerCenter from the sources to targets. With the Metadata Manager data lineage, you can analyze where data originates, how the data is transformed and what objects transform it, and where the data ends. You can analyze the data flow within a single PowerCenter repository or across multiple PowerCenter repositories.

You can display the data lineage for PowerCenter objects in the Designer. When you display data lineage for a PowerCenter object, the Designer connects to the Metadata Manager application and extracts the data lineage information from the Metadata Manager warehouse. The data lineage diagram appears in a browser window.

Data lineage analysis on a PowerCenter repository displays one or more of the following PowerCenter objects:

- **Repositories.** You can load objects from multiple PowerCenter repositories into Metadata Manager. In the data lineage, Metadata Manager displays the repository for each metadata object.
- **Data structures.** Data lineage on a PowerCenter object includes the source definition, target definition, and transformation data structures.
- **Fields.** Fields are objects within data structures that store the metadata. Data lineage on a PowerCenter object includes the source, target, and transformation ports.

Configuring Data Lineage Access

To configure PowerCenter and Metadata Manager to display lineage for PowerCenter objects, complete the following tasks:

1. **Make sure Metadata Manager is running.** Create a Metadata Manager Service in the Informatica Administrator or verify that an enabled Metadata Manager Service exists in the domain that contains the PowerCenter repository you want to run data lineage analysis on.

2. **Load the PowerCenter repository metadata.** Create a resource for the PowerCenter repository in Metadata Manager and load the PowerCenter repository metadata into the Metadata Manager warehouse.
3. **Set up the connection between the PowerCenter repository and Metadata Manager.** Configure the Metadata Manager Service and the resource name for the PowerCenter repository in the Administrator tool.
4. **Configure the web browser.** When you run data lineage analysis on a PowerCenter object, the Designer launches data lineage analysis in the default browser configured for your system. Data lineage requires the Flash 9 viewer installed on the browser.

Running Data Lineage from the Designer

You can access data lineage from any PowerCenter Designer tool. You can display data lineage on source and target definitions, transformations, mapplets, and mappings.

To run data lineage from the PowerCenter Designer:

1. In the Designer, open the PowerCenter folder containing the object on which you want to run data lineage.
2. Open the object in the applicable workspace area.

   For example, if you want to run data lineage on a transformation, open the transformation in the Transformation Developer.
3. In the workspace, right-click and select **Data Lineage**.
4. In the Informatica Metadata Manager Business Glossary browser, enter the user credentials.

   The data lineage for the object appears in the browser window.

   Note: The performance of the data lineage analysis depends on the amount of available system resources on the machine running the Metadata Manager application. Running multiple data lineage analysis on multiple objects simultaneously can impact performance.

After you access data lineage, you can view details about each object in the data lineage diagram. You can export the data lineage to a PNG image file or print the diagram. You can also email the data lineage to other users.

CHAPTER 11

Comparing Objects

This chapter includes the following topics:

- Comparing Objects Overview, 193
- Comparing Sources, Targets, and Transformations , 194
- Comparing Mappings and Mapplets , 195

Comparing Objects Overview

You can compare two repository objects of the same type to identify differences between the objects through the Designer. For example, you may want to use a target definition in a mapping, but you have two target definitions that are similar. You can compare the target definitions to see which one contains the columns you need. When you compare two objects, the Designer displays their attributes side-by-side.

You can compare the following types of objects:

- **Sources.** You can compare two sources, two source shortcuts, or a source with a source shortcut.
- **Targets.** You can compare two targets, two target shortcuts, or a target with a target shortcut.
- **Transformations.** You can compare two transformations, two transformation shortcuts, or a transformation with a transformation shortcut.
- **Mappings and mapplets.** You can compare two mappings, two mapping shortcuts, or a mapping with a mapping shortcut. You can also compare two mapplets, two mapplet shortcuts, or a mapplet with a mapplet shortcut.
- **Instances.** Within mappings and mapplets, you can compare two source instances, two target instances, or two transformations. Within mappings, you can also compare two mapplet instances.
- **Folders.** You can compare two folders in the Repository Manager.
- **Object shortcut versions.** You can compare two versions of an object in the Repository Manager.

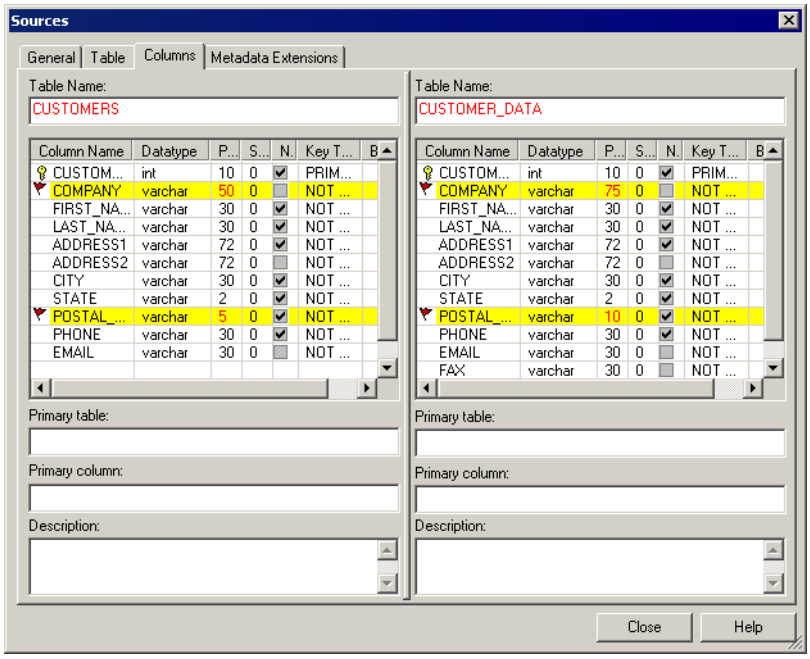
You can compare objects under the following circumstances:

- **When both objects exist in open folders.** You can compare an object in any open folder with another object of the same type in any open folder. The folders can be in different repositories. You cannot compare objects of different types.
- **When importing an object.** When you import an object into a folder that contains an object with the same name, the Designer lets you compare the objects to determine whether you want to overwrite or use the existing object.

- **When copying an object.** If you copy an object into a folder that contains an object of that type with the same name, the Designer lets you compare the two objects to determine whether you want to overwrite or use the existing object.
- **When copying or deploying objects to another repository**. If you copy or deploy an object into a folder in another repository, the Copy or Deployment Wizard compares the objects in the target folder with the objects you deploy or copy.
- **When you view an object that has multiple versions**. You can compare the previous version of an object with the current version of the object to see the changes made to the latest version.
- **When you view an object in the Results View window**. When you view an object in the Results View window, you can compare a previous versions of the object with the current version of the object to view the differences between the different versions.

You can compare objects across folders and repositories through the Designer. To do this, you must have both folders open. When you compare objects, a dialog box displays the results. Each dialog box contains different tabs for different types of objects.

The following figure shows a comparison of two relational sources:



Each tab in this dialog box contains two columns. The left column lists the attributes in the first object you compare, and the right column lists the attributes in the second object. A divider separates the two columns. Slide the divider left or right to resize the columns. The Designer highlights differences between the objects. Ports or columns common to both objects appear in the same row.

Note: You cannot change the metadata displayed in the results.

Comparing Sources, Targets, and Transformations

You can compare individual objects of the same type to identify differences between the objects through the Designer.

The following table describes the methods you can use to compare sources, targets or transformations:

To Compare...	Open the following tool...	Next step...
Sources	Source Analyzer	Click Sources > Compare.
Targets	Target Designer	Click Targets > Compare.
Transformations	Transformation Developer	Click Transformation > Compare.
Instances within a mapping	Mapping Designer	Select an object and click Transformation > Compare.
Instances within a mapplet	Mapplet Designer	Select an object and click Transformation > Compare.

Note: If one of the fields in this dialog box is blank or displays the name of an object you do not want to compare, click Browse to select objects to compare.

Comparing Mappings and Mapplets

You can compare two mappings or two mapplets. For example, if you have two mappings with the same name in different folders, you can compare them to see if they differ.

When you compare two mappings or two mapplets, a dialog box displays the results. Use the tabs on this dialog box to display the following information:

- **Summary information.** View a summary of the differences between each mapping or mapplet on the Summary tab. You can save this information to a text file.
- **General information.** Compare general information about each object such as name, description and validity on the Mappings or Mapplets tab. You can also compare general information about shortcut.
- **Instances.** Compare the source, target, and transformation instances in each mapping or mapplet on the Instances tab. You can compare source, target, transformation, and mapplet instances of the same type. Instances with the same name appear on the same line. To compare the instances in detail, select an instance in each mapping or mapplet and click Compare Instances.
- **Links.** Compare differences in incoming and outgoing links and the ports that connect the instance to other transformation instances using the Instances tab. To compare links, first compare instances in two mappings or mapplets. Then, click on an instance on the Instances tab and click Compare Links. To compare incoming links, click the Incoming Links tab. To compare outgoing links, click the Outgoing Links tab. To compare ports for each link, click on an link. If a port exists in both instances, it appears on the same line.
- **Parameters and variables.** Compare the parameters and variables in each object on the Variables tab.
- **Target load orders.** Compare the target load order in each mapping on the Target Load Order tab. The Designer does not display this tab when you compare mapplets.
- **Metadata extensions.** Compare the metadata extensions in each mapping or mapplet on the Metadata Extensions tab.

The following table describes the method to compare mappings, mapplets, or instances in two mappings or mapplets:

To Compare...	Open the following tool...	Select...
Mappings	Mapping Designer	Click Mappings > Compare.
Mapplets	Mapplet Designer	Click Mapplets > Compare.
Instances in two mappings	Mapping Designer	Click Mappings > Compare.
Instances in two mapplets	Mapplet Designer	Click Mapplets > Compare.

Note: If one of the fields in this dialog box is blank or displays the name of an object you do not want to compare, click Browse to select an object.

CHAPTER 12

Managing Business Components

This chapter includes the following topics:

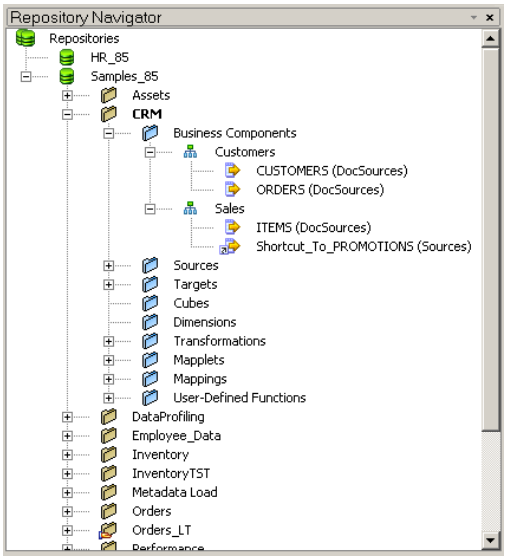
- Managing Business Components Overview, 197
- Managing Business Components and Directories, 198

Managing Business Components Overview

Use business components to group and display sources and mapplets in a repository folder. A business component node appears in each repository folder in the Repository Manager and the Designer. You can add an unlimited number of directories to the business component tree.

A business component can refer a source, a mapplet, or a shortcut to a source or mapplet.

The following figure shows directories in which you group business components:



You can use the same source or mapplet multiple times in the business component tree.

If the business component tree is in a shared folder, all directories in the business component tree are shared.

Business Component Locking

The Designer locks the business component tree while its contents are being edited. You cannot copy a business component when the business component tree is locked.

Creating Links to Business Component Documentation

You can create and edit links that users click to view business documentation that you develop for business components and directories. Business documentation provides details about a particular repository object or transformation expression.

Managing Business Components and Directories

Use the Designer to perform the following tasks:

- Create a directory.
- Create a business component.
- Edit a directory.
- Delete a directory or business component.
- Copy a directory or business component.

To move a business component or directory, drag a single business component or an entire directory within the same business component tree. If you drag a single business component or an entire directory to or from a different repository folder or repository, the Designer prompts you to create a copy of the original object in the destination location.

Creating and Editing a Directory

Use directories to organize repository objects within the business component tree. You can enter a name for each directory in the tree and provide comments to identify it. When you create a directory, the directory location appears as one of its properties. You can create a directory in a different location by selecting a different directory node when you create the directory.

To create a business components directory:

1. In the Designer, connect to a repository and open a folder.
2. Click Repository > Business Components > New Directory.

   Note: To create a business components directory under another directory, select the directory in the Navigator and click Repository > Business Components > New Directory.
3. Enter a name and description for the business components directory.

Editing a Directory

To edit a business components directory:

1. In the Designer, connect to a repository and open a folder.
2. Select the business components directory you want to edit.
3. Click Repository > Business Components > Edit Properties.

Creating a Business Component

Create business components to view the sources and mapplets using directories. Edit the object from its original location or from the business components directory. You can reference a source or mapplet multiple times in the business component tree.

You can also create business components with objects from other repository folders or repositories by using shortcuts or by copying business components.

To create a business component:

1. In the Designer, connect to a repository.
2. Open the repository folder in which you want to create a business component.
3. From the Navigator in this repository folder, select the source or mapplet to which you want to create a reference.
4. Drag the source or mapplet into the directory in the business component tree.

Creating Business Components from a Copy

To create a copy of a business component that is not shared, drag the object from a repository folder into a directory in the business component tree. The Designer prompts you to create a copy of the object in the business component tree. Copies of objects from other repository folders do not inherit changes you make to the original object.

Creating Business Components using Shortcuts

To create a business component of an object in a shared folder within a repository, drag the object into a directory in the business component tree. The Designer prompts you to create a local shortcut. A local shortcut references a source or mapplet in a shared folder within a repository.

To create a business component of an object in a shared folder of a global repository, connect to the global repository and drag the object into a directory in the business component tree. The Designer prompts you to create a global shortcut. A global shortcut is a shortcut in a local repository that references an object in a shared folder within the global repository.

Deleting a Directory or Business Component

You can delete a directory or business component from the business component tree. Deleting a directory deletes all subdirectories and business components in the directory. Deleting a directory or business component does not delete the original repository objects.

To delete a business components directory:

1. In the Designer, connect to a repository and open a folder.
2. Select a business components directory and click Edit > Delete.

   Tip: You can delete a directory by selecting it in the Navigator and pressing the Delete key.

Deleting a Business Component

To delete a business component:

1. In the Designer, connect to a repository and open a folder.
2. Select a business component and click Edit > Delete.

   Tip: You can delete a business component by selecting it in the Navigator and pressing the Delete key.

Copying a Directory or Business Component

You can copy a business components directory to a different location in the same tree. You can also copy a business components directory to a different repository folder or repository. Copies of objects from other repository folders do not inherit changes you make to the original object.

To copy a directory or business component:

1. In the Designer, connect to the source database and open the folder from which you want to copy the directory or business component.
2. In the Navigator, select the object you want to copy.
3. Click Edit > Copy, or press Ctrl+C.
4. If you want to copy the object to a different repository, connect to the target repository.
5. If you want to copy the object to a different folder, open the destination folder.
6. In the Navigator, select the business components directory location where you want to paste the directory or business component.
7. Click Edit > Paste, or press Ctrl+V.

   Tip: You can copy a directory or business component by connecting to a repository and pressing the control key while you drag the directory or business component into the business components node in the destination folder.

CHAPTER 13

Creating Cubes and Dimensions

This chapter includes the following topics:

- Creating Cubes and Dimensions Overview, 201
- Creating a Dimension, 203
- Creating a Cube, 204
- Editing a Cube, 205
- Editing a Dimension, 205
- Deleting a Cube or Dimension, 205
- Opening and Closing a Cube, 206
- Viewing Metadata for Cubes and Dimensions, 206
- Tips for Cubes and Dimensions, 206

Creating Cubes and Dimensions Overview

The Target Designer provides an interface to enable you to create and edit cubes and dimensions. Multi-dimensional metadata refers to the logical organization of data used for analysis in online analytical processing (OLAP) applications. This logical organization is generally specialized for the most efficient data representation and access by end users of the OLAP application. The following sections provide an overview of the concepts relevant to the multi-dimensional features of PowerCenter.

Understanding Multi-Dimensional Metadata

The multi-dimensional model is a key aspect of data warehouse design. A well-designed dimensional model can help you organize large amounts of data. The dimensional model was originally created for the retail industry, where analysts view business data by simple dimensions, such as products and geographies. This dimensional model consists of a large central fact table and smaller dimension tables. The fact table contains the measurable facts, such as total sales and units sold, and disjoint dimensions represent the attributes pertaining to various business segments of the industry. The central fact table is the only table in the schema with multiple joins connecting it to the dimension tables. The dimension tables in turn each have a single join connecting them to the central fact table.

There are different types of multi-dimensional models depending on the degree of redundancy in the logical schema. More redundancy can improve the efficiency of data access but represents a less normalized view of the logical schema. The most common type of a multi-dimensional schema is called a star schema. A star schema is a normalized multi-dimensional model where each of its disjoint dimensions is represented in a single table.

Another type of a normalized multi-dimensional model is a snowflake schema. A snowflake schema is logically similar to a star-schema except that at least one dimension is represented in two or more hierarchically-related tables. The star schema can become a snowflake schema if the product dimension is represented by means of multiple tables. For example, you could add one dimension table for the main product attributes, one for the brand attributes, and one for a specific brand attributes.

Non-normalized multi-dimensional models have duplicate attributes in tables that are associated with a dimension. You can quickly retrieve various attributes of a dimension without having to perform multiple joins between tables in the dimension.

Key Elements of Multi-Dimensional Metadata

The following table describes key elements of multi-dimensional metadata:

Term	Definition
Aggregate	Pre-stored summary of data or grouping of detailed data which satisfies a specific business rule. Example rules: sum, min, count, or combinations of them.
Level	A specific property of a dimension. Examples: size, type, and color.
Cube	A set of related factual measures, aggregates, and dimensions for a specific dimensional analysis problem. Example: regional product sales.
Dimension	A set of level properties that describe a specific aspect of a business, used for analyzing the factual measures of one or more cubes which use that dimension. Examples: geography, time, customer, and product.
Drilling	Drilling is the term used for navigating through a cube. This navigation is usually performed to access a summary level of information or to provide more detailed properties of a dimension in a hierarchy.
Fact	A fact is a time variant measurement of quantitative data in a cube; for example, units sold, sales dollars, or total profit.
Hierarchy	Hierarchy concept refers to the level of granularity represented by the data in a particular dimension of a cube. For example, state, county, district, and city represent different granularity in the hierarchy of the geography dimension.
Measure	Means for representing quantitative data in facts or aggregates. Example measures are total sales or units sold per year.
Normalization	A process used for reducing redundancies and removing anomalies in related dimension tables in various hierarchies.
Redundancy	Term used for referring to duplication of data among related tables for the sake of improving the speed of query processing.
Star Schema	A normalized multi-dimensional model in which each disjoint dimension is represented by a single table.
Snow Flake Schema	A normalized multi-dimensional model in which at least one dimension is represented by two or more hierarchically related tables.

Creating a Dimension

Before you can create a cube, you need to create dimensions. Complete the following steps to create a dimension:

1. Enter a dimension description.
2. Add levels to the dimension.
3. Add hierarchies to the dimension.
4. Add level instances to the hierarchies.

Step 1. Create a Dimension

To create a dimension:

1. In the Target Designer, click Targets > Create/Edit Dimension.

   The Dimension Editor displays.
2. Click Add Dimension.
3. Enter the following information:
   - **Name.** Dimension names must be unique in a folder.
   - **Description.** Enter a description for the dimension. This description appears in the Repository Manager.
   - **Database type.** The database type of a dimension must match the database type of the cube.

   Note: You cannot change the database type after you create the dimension.
4. Click OK.

Step 2. Add Levels to the Dimension

After you create the dimension, add the needed levels. Levels hold the properties necessary to create target tables.

To add a level to a dimension:

1. In the Dimension Editor, select Levels and click Add Level.
2. Enter a name and description for the level.

   The Level name must be unique within the dimension.
3. Click Level Properties.
4. Click the Import from Source Fields button.

   The name of a level property must be unique within the dimension.
5. Select a source table from which you want to copy columns to the level.

   The columns display in the Source Fields section.
6. Select the columns you want to add to the level.
7. Click the Copy Columns button to add source columns to the level.
8. Click the Add Columns button to add a new column to the level.
9. Click OK after you add all the columns.

   The Dimension Editor displays the new level.

Step 3. Add Hierarchies to the Dimension

To add a hierarchy to a dimension:

1. In the Dimension Editor, select Hierarchies.
2. Click Add Hierarchy.
3. Enter a hierarchy name, description, and select Normalized or Non-normalized.

   Normalized cubes restrict redundant data. Non-normalized cubes allow for redundant data, which increases speed for retrieving data.

Step 4. Add Levels to the Hierarchy

After you create a hierarchy, you add levels to it. You can have only one root level in a hierarchy.

To add a level to a hierarchy:

1. From the Dimension Editor, drill down to view the levels in the dimension.
2. Drag the level you want to define as the root level in the hierarchy.

   The root level is the level of finest granularity.
3. Enter a target table name and description of the target table.
4. Click OK.

   A window displays a listing of all the objects affected by the new level.
5. Click OK.

   The new level displays under the hierarchy.

Creating a Cube

To create a cube:

1. From the Target Designer, click Targets > Create Cube.
2. Enter the following information:
   - **Cube name.** The cube name must be unique in a folder.
   - **Cube type.** Choose Normalized or Non-normalized. Normalized dimensions must have a normalized cube. Similarly, non-normalized dimensions must have a non-normalized cube.
   - **Database type.** The database type for the cube must match the database type for the dimensions in the cube.
3. Click Next.
4. Specify the dimensions and hierarchies to include in the cube.
5. Click Next.
6. Add measures to the cube.

   You can copy columns from source tables or add new columns.

   Measure names must be unique within a fact. Level names must be unique within each cube.
7. Add a name for the fact table.
8. Click Finish.

   The Designer adds the cube and fact tables to the workspace.

Editing a Cube

You can edit a cube in the Target Designer. You cannot edit a fact table or dimension table directly. To edit a column in a fact table or dimension table, you need to edit the cube or dimension.

To edit a cube:

1. Click Targets > Edit Cube.
2. You can modify any settings in the dimension except the following:
   - Database type
   - Dimension type (normalized or non-normalized)
3. Click Close.

Editing a Dimension

You can edit dimensions in the Target Designer. You cannot, however, change the database type after you create a dimension.

When you edit a dimension, the Designer marks all mappings with the dimension invalid.

To edit a dimension:

1. Click Targets > Create/Edit Dimension.
2. Optionally, modify any setting in the dimension except database type or dimension type.
3. Click Close.

Deleting a Cube or Dimension

You can delete a cube or dimension from the Navigator in the Designer. Unlike target tables, you cannot delete cubes and dimensions in the Target Designer.

When you delete a cube, you delete all fact tables associated with the cube. When you delete a dimension, you delete all dimension tables and references to the dimension.

To delete a cube or dimension:

1. In the Designer, open the repository.
2. In the Navigator, select the cube or dimension you want to delete.
3. Click Delete.

   A message prompts you to verify if you want to delete the cube or dimension.
4. Click OK to delete the cube or dimension.

Opening and Closing a Cube

You can open a cube in the Target Designer.

To open a cube:

1. Open a repository and open a folder.
2. Open the Target Designer.
3. Select the cube and drag it into the workspace.

   A message prompts you to clear the workspace.
4. Click OK to open the cube.

   The Designer clears the workspace and displays all fact and dimension tables associated with the cube.

To close a cube in the Target Designer, click Targets > Close Cube. The Designer closes the cube, saving the layout of all the tables.

Viewing Metadata for Cubes and Dimensions

You can view the metadata for cubes and dimensions in the Repository Manager.

To view cube or dimension metadata:

1. In the Repository Manager, open a folder.
2. Drill down to the cube or dimension you want to analyze.

   The Repository Manager displays the metadata for each object.

Tips for Cubes and Dimensions

Consider the following tips when working with cubes and dimensions:

- If you want to copy a cube, you need to copy the folder that stores the cube.
- To view the levels of a cube or dimension, you can either edit the cube or dimension or use the Navigator in the Repository Manager.
- You cannot revert to previous versions of cubes or dimensions.
- You can delete a cube or dimension from the Navigator.
- You can delete a dimension using Targets > Create/Edit Dimension.
- You cannot delete cubes and dimensions from the Target Designer workspace.
- If you want to change a column in a fact table or dimension table, you need to edit the cube or dimension. You cannot edit a fact table or dimension table directly.
- If you delete a level, the Designer deletes the associated level instances in hierarchies. The Designer also removes the level instance from any associated cubes.
- A primary key is generated for each fact and dimension table. The format is GK_*TABLE_NAME*.
- A foreign key is added to the appropriate fact table when you associate a dimension level instance to the fact table.

- You can drag a cube to the workspace and then edit the cube and cardinality by double-clicking the links.
- You cannot delete or create a link between fact and dimension tables in the workspace. You can only delete and create links in the Cube and Dimension Editors. You can create a graph in the hierarchy.

CHAPTER 14

Using the Mapping Wizards

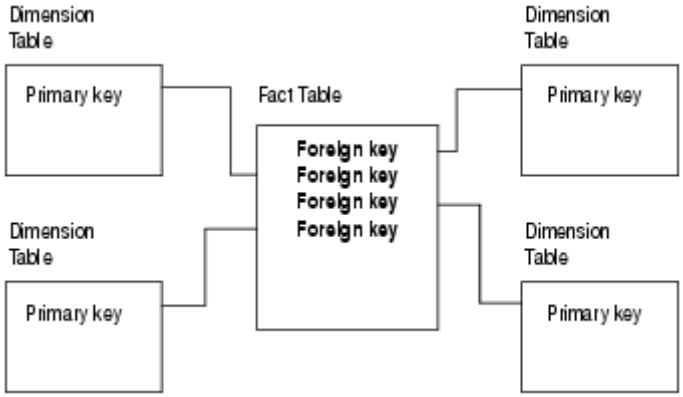
This chapter includes the following topics:

- Maintaining Star Schemas, 208
- Understanding the Mapping Wizards, 210
- Creating a Pass-Through Mapping, 212
- Creating a Slowly Growing Target Mapping, 213
- Creating a Type 1 Dimension Mapping, 216
- Creating a Type 2 Dimension/Version Data Mapping, 220
- Creating a Type 2 Dimension/Flag Current Mapping, 226
- Creating a Type 2 Dimension/Effective Date Range Mapping, 232
- Creating a Type 3 Dimension Mapping, 238
- Creating Targets in the Target Database, 244
- Scheduling Sessions and Workflows, 244
- Creating a Mapping from the Informatica Mapping Templates, 245

Maintaining Star Schemas

When you design a star schema, you create a fact table for information collected over time, such as purchases or transactions. You then build separate dimension tables for related lists of information, such as inventory or shipping methods. Each dimension table has a logical primary or generated composite key to enable access to dimensional data. For example, you might use an inventory part number as a primary key, or create a composite key using a part number and a current flag.

The following image shows a star schema with four dimension tables and one fact table:



When you implement a star schema, you decide how to handle updates to the fact and dimension tables. Fact tables change regularly as new information collects. Do you need to keep all existing data in the fact table, or do you want only the most recent version or snapshot?

If you do not need any historical fact information, you can drop or truncate the existing fact table before using a new session in a workflow. To keep historical information in a fact table, you usually append the latest snapshot to the existing table, using a flag such as a load date or session number to identify the most recent snapshot.

Although dimension tables are typically static lists, most dimension tables do change over time. For example, you might need to update an inventory dimension once a month to reflect new or changed part numbers. Since these changes are smaller in magnitude compared to changes in fact tables, these dimensions are known as slowly growing or slowly changing dimensions.

Slowly growing dimensions are dimension tables that have slowing increasing dimension data, without updates to existing dimensions. You maintain slowly growing dimensions by appending new data to the existing table.

Slowly changing dimensions are dimension tables that have slowly increasing dimension data and updates to existing dimensions. When updating existing dimensions, you decide whether to keep all historical dimension data, no historical data, or just the current and previous versions of dimension data.

When you do not need historical information in a slowly growing or slowly changing dimension table, you can drop or truncate the existing table before using a new session in a workflow. However, in some cases, inserting new dimensions and updating existing dimensions can be more efficient than reloading the entire table.

When you need historical information in a dimension table, you decide how to differentiate between current and historical data in the target:

- To keep a full history, you might version new data by:
  - Creating a version number and versioning the primary key.
  - Creating a composite key using a current version flag.
  - Creating an effective date range.
- To keep a partial history, you might keep a current version and a previous version in a single row. You can also enter a timestamp to indicate the most recent update.

Understanding the Mapping Wizards

The Designer provides two mapping wizards to help you create mappings quickly and easily. Both wizards are designed to create mappings for loading and maintaining star schemas, a series of dimensions related to a central fact table. You can, however, use the generated mappings to load other types of targets.

You choose a different wizard and different options in each wizard based on the type of target you want to load and the way you want to handle historical data in the target:

- **Getting Started Wizard.** Creates mappings to load static fact and dimension tables and slowly growing dimension tables.
- **Slowly Changing Dimensions Wizard.** Creates mappings to load slowly changing dimension tables based on the amount of historical dimension data you want to keep and the method you choose to handle historical dimension data.

After using a mapping wizard, you can edit the generated mapping to further customize it.

Using the Getting Started Wizard

The Getting Started Wizard creates mappings to load static fact and dimension tables and slowly growing dimension tables.

The Getting Started Wizard can create two types of mappings:

- **Pass through.** Loads a static fact or dimension table by inserting all rows. Use this mapping when you want to drop all existing data from the table before loading new data.
- **Slowly growing target.** Loads a slowly growing fact or dimension table by inserting new rows. Use this mapping to load new data when existing data does not require updates.

The following table describes the getting started mapping types:

Getting Started Mapping Type	Target Table Type	History	Data Handling
Pass through	Static Fact or Dimension	None	Inserts all source rows. Use the truncate target table option in the session properties, or use a pre-session shell command to drop or truncate the target before each session run.
Slowly growing target	Slowly Growing Fact or Dimension	Full	Flags and inserts new rows to the existing target.

Using the Slowly Changing Dimensions Wizard

The Slowly Changing Dimensions Wizard creates mappings to load slowly changing dimension tables:

- **Type 1 Dimension mapping.** Loads a slowly changing dimension table by inserting new dimensions and overwriting existing dimensions. Use this mapping when you do not want a history of previous dimension data.
- **Type 2 Dimension/Version Data mapping.** Loads a slowly changing dimension table by inserting new and changed dimensions using a version number and incremented primary key to track changes. Use this mapping when you want to keep a full history of dimension data and to track the progression of changes.
- **Type 2 Dimension/Flag Current mapping.** Loads a slowly changing dimension table by inserting new and changed dimensions using a flag to mark current dimension data and an incremented primary key to track

changes. Use this mapping when you want to keep a full history of dimension data, tracking the progression of changes while flagging only the current dimension.

- **Type 2 Dimension/Effective Date Range mapping.** Loads a slowly changing dimension table by inserting new and changed dimensions using a date range to define current dimension data. Use this mapping when you want to keep a full history of dimension data, tracking changes with an effective date range.
- **Type 3 Dimension mapping.** Loads a slowly changing dimension table by inserting new dimensions and updating values in existing dimensions. Use this mapping when you want to keep the current and previous dimension values in the dimension table.

The following table describes the slowly changing dimension mapping types:

Slowly Changing Dimensions Mapping	Target Table	History	Data Handling
Type 1 Dimension	Slowly Changing Dimension	None	Inserts new dimensions. Overwrites existing dimensions with changed dimensions.
Type 2 Dimension/ Version Data	Slowly Changing Dimension	Full	Inserts new and changed dimensions. Creates a version number and increments the primary key to track changes.
Type 2 Dimension/Flag Current	Slowly Changing Dimension	Full	Inserts new and changed dimensions. Flags the current version and increments the primary key to track changes.
Type 2 Dimension/ Effective Date Range	Slowly Changing Dimension	Full	Inserts new and changed dimensions. Creates an effective date range to track changes.
Type 3 Dimension	Slowly Changing Dimension	Partial	Inserts new dimensions. Updates changed values in existing dimensions. Optionally uses the load date to track changes.

Choosing Sources for the Mappings

Use the following sources with a mapping wizard:

- Flat file
- Relational
- Application
- Shortcut to a flat file, relational, or Application sources

You cannot use COBOL or XML sources with the mapping wizards.

When you select a source for a mapping, the mapping wizards display all available sources by source name. The Designer can optionally display sources and target definitions by business name, rather than source and target name. Even when this option is selected, mapping wizards display sources by source name.

The Mapping Wizard cannot import a source when a column name uses an SQL keyword. When the Mapping Wizard encounters an SQL keyword used as a column name, it prompts you to choose another source. If you want to use a source with a column using an SQL keyword, create the mapping in the Mapping Designer.

The mapping wizards also complete the following based on the selected source:

- Use the source name to name the source qualifier as follows: SQ_*SourceName* for file or relational sources, or ASQ_*SourceName* for Application sources.
- Create port names based on source column names.

Creating a Pass-Through Mapping

The pass-through mapping inserts all source rows. Use the pass-through mapping to load tables when you do not need to keep historical data in the target table. If source rows already exist in the target, truncate or drop the existing target before running the workflow. In the pass-through mapping, all rows are current.

You might use the pass-through mapping to load a fact or dimension table if the table remains static for a period of time, and then changes dramatically.

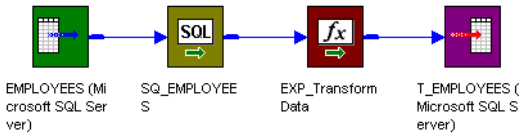
For example, you might have a vendor dimension table that remains the same for a year. At the end of the year, you reload the table to reflect new vendor contracts and contact information. If this information changes dramatically and you do not want to keep historical information, you can drop the existing dimension table and use the pass-through mapping to reload the entire table. If the information changes only incrementally, you might prefer to update the existing table using the Type 1 Dimension mapping created by the Slowly Changing Dimensions Wizard.

Understanding the Mapping

The pass-through mapping performs the following tasks:

- Selects all source rows
- Inserts all rows into the target

The following figure shows a mapping that the Getting Started Wizard creates when you create a pass-through mapping:



A single data flow passes from a source definition, through a source qualifier and an Expression transformation, to the target. By default, the Expression transformation passes data directly to the target without changes.

Understanding the Transformations

The following table describes the function of each transformation in the pass-through mapping:

Transformation Name	Transformation Type	Description
SQ_*SourceName*	Source Qualifier or Application Source Qualifier	Selects all rows from the source you choose in the Mapping Wizard.
EXP_TransformData	Expression	Passes all source data to the target without changes.
T_*TargetName*	Target Definition	Target definition allowing source data to be inserted into the target.

Steps to Create a Pass-Through Mapping

To create a pass-through mapping:

1. In the Mapping Designer, click Mappings > Wizards > Getting Started.
2. Enter a mapping name and select Simple Pass Through, and click Next.

   The naming convention for mapping names is m_*MappingName*.
3. Select a source definition to use in the mapping.

   All available source definitions appear in the Select Source Table list. This list can include shortcuts, flat file, relational, and Application sources.
4. Enter a name for the mapping target table and click Finish.

   The new mapping appears in the workspace. The naming convention for target definitions is T_*TARGET_NAME*.

   Make necessary edits to the mapping.

Create the target table in the target database before running a workflow.

Customizing the Mapping

After the wizard creates the mapping, you can configure the Expression transformation, EXP_TransformData. You can also add any other transformations to customize the mapping.

If you want data to pass directly from source to target without any other transformations, delete the Expression transformation. You can optimize performance for pass-through mappings by connecting the source qualifier directly to the target.

Configuring a Pass-Through Session

The pass-through mapping inserts source rows into the target. To prevent primary key errors from duplicate rows, drop or truncate the target table before running the workflow. Use the truncate target table option in the session properties, or use a pre-session shell command to perform this task automatically.

Creating a Slowly Growing Target Mapping

The slowly growing target mapping filters source rows based on user-defined comparisons, and then inserts only those found to be *new* to the target. Use the slowly growing target mapping to determine which source rows are new and to load them to an existing target table. In the slowly growing target mapping, all rows are current.

Use the slowly growing target mapping to load a slowly growing fact or dimension table, one in which existing data does not require updates.

For example, you have a site code dimension table that contains only a store name and a corresponding site code that you update only after the company opens a new store. Although listed stores might close, you want to keep the store code and name in the dimension for historical analysis. With the slowly growing target mapping, you can load new source rows to the site code dimension table without deleting historical sites.

Handling Keys

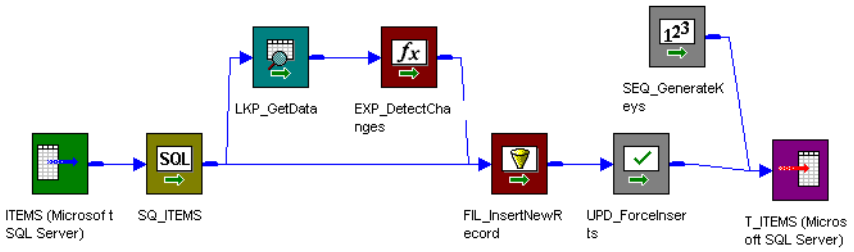
When you create a slowly growing target mapping, the Designer creates an additional column in the mapping target, PM_PRIMARYKEY. In this column, the Integration Service generates a primary key for each row written to the target, incrementing new key values by 1.

Understanding the Mapping

The slowly growing target mapping performs the following tasks:

- Selects all rows
- Caches the existing target as a lookup table
- Compares logical key columns in the source against corresponding columns in the target lookup table
- Filters out existing rows
- Generates a primary key for new rows
- Inserts new rows to the target

The following figure shows a mapping that the Getting Started Wizard creates when you create a slowly growing target mapping:



The slowly growing target mapping uses a Lookup and an Expression transformation to compare source data against existing target data. When you step through the Getting Started Wizard you enter the logical key columns in the source to compare against the existing target. When the Expression transformation detects source rows without matching key columns in the target, it flags the row new.

A Filter transformation passes only new rows to the Update Strategy transformation. The Update Strategy transformation marks new rows for insert and a Sequence Generator creates a new primary key value for each row written to the target.

Understanding the Transformations

The following table describes the function of each transformation in the slowly growing target mapping:

Transformation Name	Transformation Type	Description
SQ_*SourceName*	Source Qualifier or Application Source Qualifier	Selects all rows from the source you choose in the Mapping Wizard.
LKP_GetData	Lookup	Caches the existing target table. Compares a logical key column in the source against the corresponding key column in the target.

Transformation Name	Transformation Type	Description
EXP_DetectChanges	Expression	Uses the following expression to flag source rows that have no matching key in the target (indicating they are new): `IIF(ISNULL(PM_PRIMARYKEY),TRUE,FALSE)` Populates the NewFlag field with the results. Passes all rows to FIL_InsertNewRecord.
FIL_InsertNewRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked new (TRUE): `NewFlag`. Passes new rows to UPD_ForceInserts.
UPD_ForceInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.
SEQ_GenerateKeys	Sequence Generator	Generates a value for each new row written to the target, incrementing values by 1. Passes values to the target to populate the PM_PRIMARYKEY column.
T_*TargetName*	Target Definition	Instance of the target definition for new rows to be inserted into the target.

Steps to Create a Slowly Growing Target Mapping

To create a slowly growing target mapping:

1. In the Mapping Designer, click Mappings > Wizards > Getting Started.
2. Enter a mapping name and select Slowly Growing Target, and click Next.

   The naming convention for mapping names is m_*MappingName*.
3. Select a source definition to be used in the mapping.

   All available source definitions appear in the Select Source Table list. This list includes shortcuts, flat file, relational, and Application sources.
4. Enter a name for the mapping target table. Click Next.

   The naming convention for target definitions is T_*TARGET_NAME*.
5. Select the column or columns from the Target Table Fields list that you want the Integration Service to use to look up data in the target table. Click Add.

   The wizard adds selected columns to the Logical Key Fields list.

   Tip: The columns you select should be a key column in the source.

   When you run a workflow containing the session, the Integration Service performs a lookup on existing target data. The Integration Service returns target data when Logical Key Fields columns match corresponding target columns.

   To remove a column from Logical Key Fields, select the column and click Remove.

   Note: You cannot add a port using the name, FILLER, to the Logical Key field list.
6. Click Finish.

   The new mapping appears in the workspace. Make necessary edits to the mapping.

   Note: The Fields to Compare for Changes field is disabled for the slowly growing target mapping.

Configuring a Slowly Growing Target Session

The slowly growing target mapping flags new source rows, and then inserts them to the target with a new primary key. The mapping uses an Update Strategy transformation to indicate new rows must be inserted. Therefore, when you create a session for the mapping, configure the session as follows:

1. In the session properties, click the General Options settings on the Properties tab. Set Treat Source Rows As to Data Driven.
2. In the session properties, click the Target Properties settings on the Mappings tab. To verify that the Integration Service loads rows to the target properly, select Insert for each relational target.

Creating a Type 1 Dimension Mapping

The Type 1 Dimension mapping filters source rows based on user-defined comparisons and inserts only those found to be new dimensions to the target. Rows containing changes to existing dimensions are updated in the target by overwriting the existing dimension. In the Type 1 Dimension mapping, all rows contain current dimension data.

Use the Type 1 Dimension mapping to update a slowly changing dimension table when you do not need to keep any previous versions of dimensions in the table.

For example, you might have a site dimension table with store code, location, and overhead that you update after the company opens a new store. This dimension is used for sales and overhead calculations. Since you do not need to know the previous address of the same store or the store overhead from the previous year, you do not need previous dimension data in the table. With the Type 1 Dimension mapping, you can keep current data without a historical log.

Handling Keys

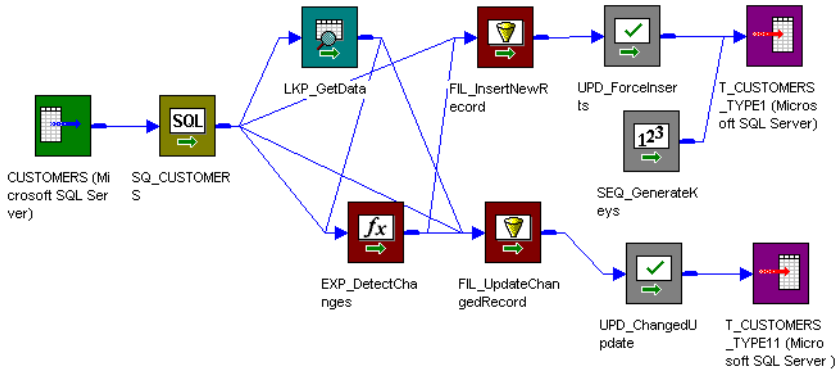
When you use the Type 1 Dimension option, the Designer creates an additional column in the mapping target, PM_PRIMARYKEY. In this column, the Integration Service generates a primary key for each row written to the target, incrementing new key values by 1.

Understanding the Mapping

The Type 1 Dimension mapping performs the following tasks:

- Selects all rows.
- Caches the existing target as a lookup table.
- Compares logical key columns in the source against corresponding columns in the target lookup table.
- Compares source columns against corresponding target columns if key columns match.
- Flags new rows and changed rows.
- Creates two data flows: one for new rows, one for changed rows.
- Generates a primary key for new rows.
- Inserts new rows to the target.
- Updates changed rows in the target, overwriting existing rows.

The following figure shows a mapping that the Slowly Changing Dimensions Wizard creates when you select the Type 1 Dimension option:



The Type 1 Dimension mapping uses a Lookup and an Expression transformation to compare source data against existing target data. When you step through the Slowly Changing Dimensions Wizard, you enter the lookup conditions (source key columns) and source columns that you want the Integration Service to compare against the existing target.

For each source row without a matching primary key in the target, the Expression transformation marks the row new. For each source row with a matching primary key in the target, the Expression compares user-defined source and target columns. If those columns do not match, the Expression marks the row changed. The mapping then splits into two separate data flows.

The first data flow uses the Filter transformation, FIL_InsertNewRecord, to filter out existing rows. The Filter transformation passes only new rows to the UPD_ForceInserts Update Strategy transformation. UPD_ForceInserts inserts new rows to the target, and a Sequence Generator creates a primary key for each row.

In the second data flow, the FIL_UpdateChangedRecord Filter transformation allows only changed rows to pass to the Update Strategy transformation, UPD_ChangedUpdate. UPD_ChangedUpdate replaces existing rows in the target with the updated source rows.

Understanding the Transformations

The following table describes the function of each transformation in the Type 1 Dimension mapping:

Transformation Name	Transformation Type	Description
SQ_*SourceName*	Source Qualifier or Application Source Qualifier	Selects all rows from the source you choose in the Mapping Wizard.
LKP_GetData	Lookup	Caches the existing target table. Compares key columns in the source against corresponding key columns in the target. When matching keys exist, LKP_GetData returns additional column data from the target for comparison. Passes all rows to EXP_DetectChanges.
EXP_DetectChanges	Expression	Uses the following expression to flag source rows that *do not* have matching keys in the target. The expression returns TRUE if a matching key does not exist in the target, indicating the row is new: `IIF(ISNULL(PM_PRIMARYKEY),TRUE,FALSE)` Populates the NewFlag port with the results. Uses the following expression to flag source rows that *have* a matching key in the target and contain changes in the specified columns. The expression returns TRUE only if a matching key exists in the target (indicating the row is not new) and if it detects a difference between source and target columns: `IIF(ISNULL(PM_PRIMARYKEY) AND (SourceColumnName<>PM_PREV_TargetColumnName) AND (other comparisons) TRUE,FALSE)` Populates the ChangedFlag port with the results. Passes all rows to FIL_InsertNewRecord and FIL_UpdateChangedRecord.

Data Flow for New Rows

For each new row in the source, this data flow creates a primary key, sets the beginning of the effective date range, and inserts the row into the target.

The following table describes the data flow for new rows:

Transformation Name	Transformation Type	Description
FIL_InsertNewRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked new (TRUE): `NewFlag`. Passes new rows to UPD_ForceInserts.
UPD_ForceInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.

Transformation Name	Transformation Type	Description
SEQ_GenerateKeys	Sequence Generator	Generates a value for each new row written to the target, incrementing values by 1. Passes values to the target to populate the PM_PRIMARYKEY column.
T_*TargetName*	Target Definition	Instance of the target definition for new rows to be inserted into the target.

Data Flow for Changed Rows

For each changed row in the source, this data flow marks the row for update and overwrites the corresponding row in the target.

The following table describes the data flow for changed rows:

Transformation Name	Transformation Type	Description
FIL_UpdateChangedRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked changed (TRUE): `ChangedFlag`. Passes changed rows to UPD_ChangedUpdate.
UPD_ChangedUpdate	Update Strategy	Uses `DD_UPDATE` to overwrite corresponding rows in the target.
T_*TargetName*1	Target Definition	Instance of the target definition allowing changed rows to be overwritten in the target.

Steps to Create a Type 1 Dimension Mapping

To create a slowly growing target mapping, complete the following steps:

1. In the Mapping Designer, click Mappings > Wizards > Slowly Changing Dimension.
2. Enter a mapping name and select Type 1 Dimension, and click Next.

   The naming convention for mappings is m_*MappingName*.
3. Select a source definition to be used by the mapping.

   All available source definitions appear in the Select Source Table list. This list includes shortcuts, flat file, relational, and Application sources.
4. Enter a name for the mapping target table. Click Next.

   The naming convention for target definitions is T_*TARGET_NAME*.
5. Select the column or columns you want to use as a lookup condition from the Target Table Fields list and click Add.

   The wizard adds selected columns to the Logical Key Fields list.

   Tip: The columns you select should be a key column in the source.

When you run the workflow containing the session, the Integration Service performs a lookup on existing target data. The Integration Service returns target data when Logical Key Fields columns match corresponding target columns.

To remove a column from Logical Key Fields, select the column and click Remove.

6. Select the column or columns you want the Integration Service to compare for changes, and click Add.

   The wizard adds selected columns to the Fields to Compare for Changes list.

   When you run the workflow containing the session, the Integration Service compares the columns in the Fields to Compare for Changes list between source rows and the corresponding target (lookup) rows. If the Integration Service detects a change, it marks the row changed.

   To remove a column from the list, select the column and click Remove.

7. Click Finish.

   The new mapping appears in the workspace. Make the necessary edits to the mappings.

   Note: In the Type 1 Dimension mapping, the Designer uses two instances of the same target definition to enable inserting and updating data in the same target table. Generate only one target table in the target database.

Configuring a Type 1 Dimension Session

The Type 1 Dimension mapping inserts new rows with a new primary key and updates existing rows. When you create a session for the mapping, configure the session as follows:

1. In the session properties, click the General Options settings on the Properties tab. Set Treat Source Rows As to Data Driven.
2. In the session properties, click the Target Properties settings on the Mappings tab. To verify that the Integration Service loads rows to the target properly, select Insert and Update as Update for each relational target.

Creating a Type 2 Dimension/Version Data Mapping

The Type 2 Dimension/Version Data mapping filters source rows based on user-defined comparisons and inserts both new and changed dimensions into the target. Changes are tracked in the target table by versioning the primary key and creating a version number for each dimension in the table. In the Type 2 Dimension/Version Data target, the current version of a dimension has the highest version number and the highest incremented primary key of the dimension.

Use the Type 2 Dimension/Version Data mapping to update a slowly changing dimension table when you want to keep a full history of dimension data in the table. Version numbers and versioned primary keys track the order of changes to each dimension.

When you use this option, the Designer creates two additional fields in the target:

- **PM_PRIMARYKEY.** The Integration Service generates a primary key for each row written to the target.
- **PM_VERSION_NUMBER.** The Integration Service generates a version number for each row written to the target.

Handling Keys

In a Type 2 Dimension/Version Data mapping, the Integration Service generates a new primary key value for each new dimension it inserts into the target. An Expression transformation increments key values by 1,000 for new dimensions.

When updating a dimension, the Integration Service increments the existing primary key by 1.

For example, the Integration Service inserts the following new row with a key value of 65,000 since this is the sixty-fifth dimension in the table.

PM_PRIMARYKEY	ITEM	STYLES
65000	Sandal	5

The next time you run the workflow containing the session, the same item has a different number of styles. The Integration Service creates a new row with updated style information and increases the existing key by 1 to create a new key of 65,001. Both rows exist in the target, but the row with the higher key version contains current dimension data.

PM_PRIMARYKEY	ITEM	STYLES
65000	Sandal	5
65001	Sandal	14

When you run the workflow again, the Integration Service again increments the key. The highest key version contains current dimension data. The target keeps a full history of the item and the order in which the versions occurred.

PM_PRIMARYKEY	ITEM	STYLES
65000	Sandal	5
65001	Sandal	14
65002	Sandal	17

Numbering Versions

In addition to versioning the primary key, the Integration Service generates a matching version number for each row inserted into the target. Version numbers correspond to the final digit in the primary key. New dimensions have a version number of 0.

For example, in the data below, the versions are 0, 1, and 2. The highest version number contains the current dimension data.

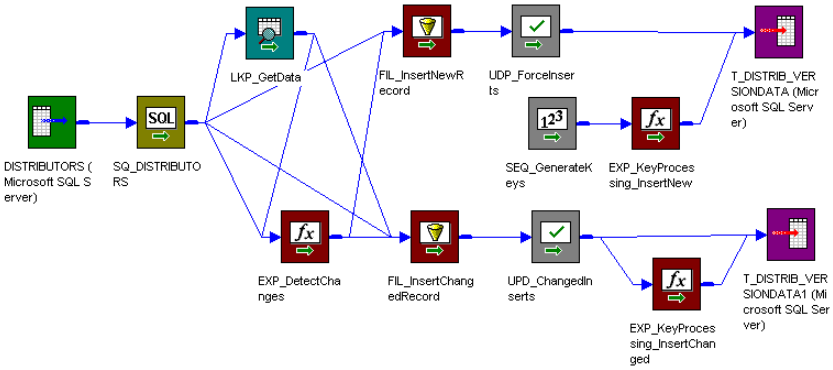
PM_PRIMARYKEY	ITEM	STYLES	PM_VERSION_NUMBER
65000	Sandal	5	0
65001	Sandal	14	1
65002	Sandal	17	2

Understanding the Mapping

The Type 2 Dimension/Version Data mapping performs the following tasks:

- Selects all rows.
- Caches the existing target as a lookup table.
- Compares logical key columns in the source against corresponding columns in the target lookup table.
- Compares source columns against corresponding target columns if key columns match.
- Flags new rows and changed rows.
- Creates two data flows: one for new rows, one for changed rows.
- Generates a primary key and version number for new rows.
- Inserts new rows to the target.
- Increments the primary key and version number for changed rows.
- Inserts changed rows in the target.

The following figure shows a mapping that the Slowly Changing Dimensions Wizard creates when you select the Type 2 Dimension/Version Data option:



The Type 2 Dimension/Version Data mapping uses a Lookup and an Expression transformation to compare source data against existing target data. When you step through the Slowly Changing Dimensions Wizard, you enter the lookup conditions (source key columns) and source columns that you want the Integration Service to compare against the existing target.

For each source row without a matching primary key in the target, the Expression transformation marks the row new. For each source row with a matching primary key in the target, the Expression compares user-defined source and target columns. If those columns do not match, the Expression marks the row changed. The mapping then splits into two data flows.

The first data flow uses the Filter transformation, FIL_InsertNewRecord, to filter out existing rows. The Filter transformation passes only new rows to the UPD_ForceInserts Update Strategy transformation. UPD_ForceInserts inserts new rows to the target. A Sequence Generator creates a primary key for each row. The Expression transformation, EXP_KeyProcessing_InsertNew, increases the increment between keys by 1,000 and creates a version number of 0 for each new row.

In the second data flow, the FIL_InsertChangedRecord Filter transformation allows only changed rows to pass to the Update Strategy transformation, UPD_ChangedInserts. UPD_ChangedInserts inserts changed rows to the target. The Expression transformation, EXP_KeyProcessing_InsertChanged, increments both the key and the version number by one.

Understanding the Transformations

The following table describes the function of each transformation in the Type 2 Dimension/Version Data mapping:

Transformation Name	Transformation Type	Description
SQ_*SourceName*	Source Qualifier or Application Source Qualifier	Selects all rows from the source you choose in the Mapping Wizard.
LKP_GetData	Lookup	Caches the existing target table. Compares key columns in the source against corresponding key columns in the target. When matching keys exist, LKP_GetData returns additional column data from the target for comparison. Passes all rows to EXP_DetectChanges.
EXP_DetectChanges	Expression	Uses the following expression to flag source rows that *do not* have matching keys in the target. The expression returns TRUE if a matching key does not exist in the target, indicating the row is new: `IIF(ISNULL(PM_PRIMARYKEY),TRUE,FALSE)` Populates the NewFlag port with the results. Uses the following expression to flag source rows that *have* a matching key in the target and contain changes in the specified columns. The expression returns TRUE only if a matching key exists in the target (indicating the row is not new) and if it detects a difference between source and target columns: `IIF(ISNULL(PM_PRIMARYKEY) AND (SourceColumnName<>PM_PREV_TargetColumnName) AND (other comparisons) TRUE,FALSE)` Populates the ChangedFlag port with the results. Passes all rows to FIL_InsertNewRecord and FIL_InsertChangedRecord.

Data Flow for New Rows

The following table describes the data flow for new rows. For each new row in the source, this data flow creates a primary key and version number, and then inserts it into the target:

Transformation Name	Transformation Type	Description
FIL_InsertNewRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked new (TRUE): `NewFlag`. Passes new rows to UPD_ForceInserts.
UPD_ForceInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.
SEQ_GenerateKeys	Sequence Generator	Generates a value for each new row written to the target, incrementing values by 1. Passes values to EXP_KeyProcessing_InsertNew.

Transformation Name	Transformation Type	Description
EXP_KeyProcessing_InsertNew	Expression	Uses the following expression to increment generated values by 1,000: `NEXTVAL * 1,000`. Then passes incremented values to the target to populate the PM_PRIMARYKEY column. Creates a version number of 0 for each row to populate the PM_VERSION_NUMBER column in the target.
T_*TargetName*	Target Definition	Instance of the target definition for new rows to be inserted into the target.

Data Flow for Changed Rows

The following table describes the data flow for changed rows. For each changed row in the source, this data flow increments the existing primary key by one, creates a corresponding version number, and inserts the row into the target:

Transformation Name	Transformation Type	Description
FIL_InsertChangedRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked changed (TRUE): `ChangedFlag`. Passes changed rows to UPD_ChangedInserts.
UPD_ChangedInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.
EXP_KeyProcessing_InsertChanged	Expression	Uses the following expression to increment the existing primary key by one: `PM_PRIMARYKEY + 1`. Uses the following expression to increment the existing version number by one: `(PM_PRIMARYKEY + 1) % 1,000`.
T_*TargetName*1	Target Definition	Instance of the target definition allowing changed rows to be inserted into the target.

Steps to Create a Type 2 Dimension/Version Data Mapping

To create a Type 2 Dimension/Version Data mapping:

1. In the Mapping Designer, click Mappings > Wizards > Slowly Changing Dimensions.
2. Enter a mapping name and select Type 2 Dimension. Click Next.
3. Select a source definition to be used by the mapping.

   All available source definitions appear in the Select Source Table list. The list includes shortcuts, flat file, relational, and Application sources.
4. Enter a name for the mapping target table. Click Next.

   The naming convention for target definitions is T_*TARGET_NAME*.
5. Select the column or columns you want to use as a lookup condition from the Target Table Fields list and click Add.

   The wizard adds selected columns to the Logical Key Fields list.

Tip: The columns you select should be a key column in the source.

When you run the workflow containing the session, the Integration Service performs a lookup on existing target data. The Integration Service returns target data when Logical Key Fields columns match corresponding target columns.

To remove a column from Logical Key Fields, select the column and click Remove.

6. Select the column or columns you want the Integration Service to compare for changes, and click Add.

   The wizard adds selected columns to the Fields to Compare for Changes list.

   When you run the workflow containing the session, the Integration Service compares the columns in the Fields to Compare for Changes list between source rows and the corresponding target rows. If the Integration Service detects a change, it marks the row changed.

   To remove a column from the list, select the column and click Remove.

7. Click Next.
8. Select Keep the 'Version' Number in Separate Column. Click Finish.

   Note: In the Type 2 Dimension/Version Data mapping, the Designer uses two instances of the same target definition to enable the two separate data flows to write to the same target table. Generate only one target table in the target database.

Customizing the Mapping

Depending on the number of versions you expect the dimension data to have, you might reduce or increase the increment the Integration Service creates between generated keys. By default, the Integration Service increments the sequence by 1,000. This allows for 1,000 versions of a single dimension.

If appropriate, you might reduce or increase the increment. To do this, you edit the Expression transformation, EXP_KeyProcessing_InsertNew, after creating the mapping.

To change the primary key increment:

1. Double-click the title bar of the Expression transformation, EXP_KeyProcessing_InsertNew.
2. Click the Ports tab.
3. Click in the far right corner of the Expression field of the PM_PRIMARYKEY port.

   The Expression Editor appears.

4. Delete the existing value, 1000, and enter the value you want the Integration Service to use when incrementing the primary key. Click Validate.

Configuring a Type 2 Dimension/Version Data Session

The Type 2 Dimension/Version Data mapping inserts both new and updated rows with a unique primary key. When you configure a session for the mapping, complete the following steps:

1. In the session properties, click the General Options settings on the Properties tab. Set Treat Source Rows As to Data Driven.
2. In the session properties, click the Target Properties settings on the Mappings tab. To verify that the Integration Service loads rows to the target properly, select Insert for each relational target.

Creating a Type 2 Dimension/Flag Current Mapping

The Type 2 Dimension/Flag Current mapping filters source rows based on user-defined comparisons and inserts both new and changed dimensions into the target. Changes are tracked in the target table by flagging the current version of each dimension and versioning the primary key. In the Type 2 Dimension/Flag Current target, the current version of a dimension has a current flag set to 1 and the highest incremented primary key.

Use the Type 2 Dimension/Flag Current mapping to update a slowly changing dimension table when you want to keep a full history of dimension data in the table, with the most current data flagged. Versioned primary keys track the order of changes to each dimension.

When you use this option, the Designer creates two additional fields in the target:

- **PM_CURRENT_FLAG.** The Integration Service flags the current row “1” and all previous versions “0.”
- **PM_PRIMARYKEY.** The Integration Service generates a primary key for each row written to the target.

Flagging the Current Value

The Integration Service generates a current flag of 1 for each row written to the target. This flag indicates the dimension is new or newly updated. If the row is an update to an existing dimension, the Integration Service resets the existing dimension current flag to 0.

As a result, all current versions of a dimension appear in the target with a current flag of 1. All previous versions have a current flag of 0.

For example, the following dimension data is current because the current flag is set to 1:

ITEM	STYLES	PM_CURRENT_FLAG
Sandal	5	1
Boot	25	1

When these dimensions change, the Integration Service inserts the updated versions with the current flag of 1. The Integration Service also updates the existing rows in the target. It finds the previous current version (where the current flag is set to 1 and updates the current flag to 0:

ITEM	STYLES	PM_CURRENT_FLAG
Sandal	5	0
Boot	25	0
Sandal	12	1
Boot	15	1

Handling Keys

When you use the Flag Current option, the Integration Service generates a primary key value for each row written to the target, incrementing key values by one. An Expression transformation increments key values by 1,000 for new dimensions.

When updating an existing dimension, the Integration Service increments the existing primary key by 1.

For example, the following dimensions are current dimension data (current flags set to one). Their primary keys are multiples of 1,000. This indicates they are both the first version of a dimension:

PM_PRIMARYKEY	ITEM	STYLES	PM_CURRENT_FLAG
3000	Sandal	5	1
4000	Boot	25	1

When the Integration Service inserts updated versions of these dimensions into the target, it sets the current flag to 1. It also creates a new primary key for the updated row by incrementing key of the existing dimension by one. The Integration Service indicates the existing dimension is no longer current by resetting its current flag to 0:

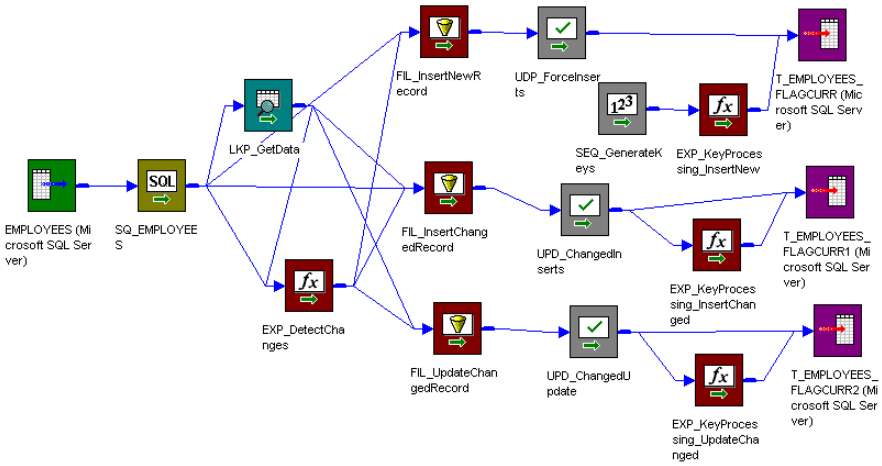
PM_PRIMARYKEY	ITEM	STYLES	PM_CURRENT_FLAG
3000	Sandal	5	0
4000	Boot	25	0
3001	Sandal	12	1
4001	Boot	15	1

Understanding the Mapping

The Type 2 Dimension/Flag Current mapping performs the following tasks:

- Selects all rows.
- Caches the existing target as a lookup table.
- Compares logical key columns in the source against corresponding columns in the target lookup table.
- Compares source columns against corresponding target columns if key columns match.
- Flags new rows and changed rows.
- Creates two data flows: one for new rows, one for changed rows.
- Generates a primary key and current flag for new rows.
- Inserts new rows to the target.
- Increments the existing primary key and sets the current flag for changed rows.
- Inserts changed rows in the target.
- Updates existing versions of the changed rows in the target, resetting the current flag to indicate the row is no longer current.

The following figure shows a mapping that the Type 2 Dimension/Flag Current option in the Slowly Changing Dimensions Wizard creates:



The Type 2 Dimension/Flag Current mapping uses a Lookup and an Expression transformation to compare source data against existing target data. When you step through the Slowly Changing Dimensions Wizard, you enter the lookup conditions (source key columns) and source columns that you want the Integration Service to compare against the existing target.

For each source row without a matching primary key in the target, the Expression transformation marks the row new. For each source row with a matching primary key in the target, the Expression compares user-defined source and target columns. If those columns do not match, the Expression marks the row changed. The mapping then splits into three data flows.

The first data flow uses the Filter transformation, FIL_InsertNewRecord, to filter out existing rows. The Filter transformation passes only new rows to the UPD_ForceInserts Update Strategy transformation. UPD_ForceInserts inserts new rows to the target. A Sequence Generator creates a primary key for each new row. The Expression transformation, EXP_KeyProcessing_InsertNew, increases the increment between keys by 1,000 and creates a current flag of 1 for each new row.

In the second data flow, the FIL_InsertChangedRecord Filter transformation allows only changed rows to pass to the Update Strategy transformation, UPD_ChangedInserts. UPD_ChangedInserts inserts changed rows to the target. The Expression transformation, EXP_KeyProcessing_InsertChanged, increments the primary key by one and creates a current flag of 1 to indicate the updated row contains current dimension data.

In the third data flow, for each changed row written to the target, the Filter transformation, FIL_UpdateChangedRecord, passes the primary key of the previous version to the Update Strategy transformation, UPD_ChangedUpdate. UPD_ChangedUpdate updates dimensions in the target. The Expression transformation, EXP_UpdateChanged, sets the current flag to 0. This changes the status of the previous dimension from current to not-current.

Understanding the Transformations

The following table describes the function of each transformation in the Type 2 Dimension/Flag Current mapping:

Transformation Name	Transformation Type	Description
SQ_*SourceName*	Source Qualifier or Application Source Qualifier	Selects all rows from the source you choose in the Mapping Wizard.
LKP_GetData	Lookup	Caches the existing target table. Compares key columns in the source against corresponding key columns in the target. When matching keys exist, LKP_GetData returns additional column data from the target for comparison. Passes all rows to EXP_DetectChanges.
EXP_DetectChanges	Expression	Uses the following expression to flag source rows that *do not* have matching keys in the target. The expression returns TRUE if a matching key does not exist in the target, indicating the row is new: `IIF(ISNULL(PM_PRIMARYKEY),TRUE,FALSE)` Populates the NewFlag port with the results. Uses the following expression to flag source rows that *have* a matching key in the target and contain changes in the specified columns. The expression returns TRUE only if a matching key exists in the target, which indicates the row is not new, and if it detects a difference between source and target columns: `IIF(ISNULL(PM_PRIMARYKEY) AND (SourceColumnName<>PM_PREV_TargetColumnName) AND (other comparisons) TRUE,FALSE)` Populates the ChangedFlag port with the results. Passes all rows to FIL_InsertNewRecord, FIL_InsertChangedRecord, and FIL_UpdateChangedRecord.

Data Flow for New Rows

The following table describes the data flow for new rows. For each new row in the source, this data flow creates a primary key and increments it by 1,000. It also flags the row current and inserts it into the target:

Transformation Name	Transformation Type	Description
FIL_InsertNewRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked new (TRUE): `NewFlag`. Passes new rows to UPD_ForceInserts.
UPD_ForceInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.
SEQ_GenerateKeys	Sequence Generator	Generates a value for each new row written to the target, incrementing values by 1. Passes values to EXP_KeyProcessing_InsertNew.

Transformation Name	Transformation Type	Description
EXP_KeyProcessing_InsertNew	Expression	Uses the following expression to increment generated values by 1,000: `NEXTVAL * 1,000`. Then passes incremented values to the target to populate the PM_PRIMARYKEY column. Creates a current flag of 1 for each row to populate the PM_CURRENT_FLAG column in the target.
T_*TargetName*	Target Definition	Instance of the target definition for new rows to be inserted into the target.

Data Flow for Changed Rows

The following table describes the data flow for changed rows. For each changed row in the source, this data flow increments the existing primary key by 1, flags the row current, and inserts it into the target:

Transformation Name	Transformation Type	Description
FIL_InsertChangedRecord	Filter	Uses the following filter condition to filter out any rows that are not marked changed: `ChangedFlag`. Passes changed rows to UPD_ChangedInserts.
UPD_ChangedInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.
EXP_KeyProcessing_InsertChanged	Expression	Uses the following expression to increment the existing primary key by one: `PM_PRIMARYKEY + 1`. Creates a current flag of 1 for each row to populate the PM_CURRENT_FLAG column in the target.
T_*TargetName*2	Target Definition	Instance of the target definition allowing changed rows to be inserted into the target.

Data Flow to Update Existing Rows

The following table describes the data flow for existing rows. For each changed row in the source, this data flow updates the end date of the corresponding row in the target to indicate the existing row is no longer current:

Transformation Name	Transformation Type	Description
FIL_UpdateChangedRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked changed (TRUE): `ChangedFlag`. For each changed row, passes the primary key of the previous version to UPD_ChangedUpdate.
UPD_ChangedUpdate	Update Strategy	Uses `DD_UPDATE` to update existing rows in the target.

Transformation Name	Transformation Type	Description
EXP_KeyProcessing_UpdateChanged	Expression	For each changed row, sets PM_CURRENT_FLAG to 0 for the previous version in the target, indicating that version is no longer current.
T_*TargetName*3	Target Definition	Instance of the target definition allowing updates to existing rows in the target.

Steps to Create a Type 2 Dimension/Flag Current Mapping

To create a Type 2 Dimension/Flag Current mapping:

1. In the Mapping Designer, click Mappings > Wizards > Slowly Changing Dimensions.
2. Enter a mapping name and select Type 2 Dimension. Click Next.

   The naming convention for mappings is m_*MappingName*.
3. Select a source definition to be used by the mapping.

   All available source definitions appear in the Select Source Table list. This list includes shortcuts, flat file, relational, and Application sources.
4. Enter a name for the mapping target table. Click Next.

   The naming convention for target definitions is T_*TARGET_NAME*.
5. Select the column or columns you want to use as a lookup condition from the Target Table Fields list and click Add.

   The wizard adds selected columns to the Logical Key Fields list.

   Tip: The columns you select should be a key column in the source.

   When you run the session, the Integration Service performs a lookup on existing target data. The Integration Service returns target data when Logical Key Fields columns match corresponding target columns.

   To remove a column from Logical Key Fields, select the column and click Remove.
6. Select the column or columns you want the Integration Service to compare for changes, and click Add.

   The wizard adds selected columns to the Fields to Compare for Changes list.

   When you run the session, the Integration Service compares the columns in the Fields to Compare for Changes list between source rows and the corresponding target (lookup) rows. If the Integration Service detects a change, it marks the row changed.

   To remove a column from the list, select the column and click Remove.
7. Click Next. Select Mark the 'Current' Dimension Record with a Flag.
8. Click Finish.

   The new mapping appears in the workspace. Make the necessary edits to the mappings.

   Note: In the Type 2 Dimension/Flag Current mapping, the Designer uses three instances of the same target definition to enable the three separate data flows to write to the same target table. Generate only one target table in the target database.

Configuring a Type 2 Dimension/Flag Current Session

The Type 2 Dimension/Flag Current mapping inserts both new and updated rows with a unique primary key. It also updates existing rows in the target. When you configure a session for the mapping, complete the following steps:

1. In the session properties, click the General Options settings on the Properties tab. Set Treat Source Rows As to Data Driven.
2. In the session properties, click the Target Properties settings on the Mappings tab. To verify that the Integration Service loads rows to the target properly, select Insert and Update as Update for each relational target.

Creating a Type 2 Dimension/Effective Date Range Mapping

The Type 2 Dimension/Effective Date Range mapping filters source rows based on user-defined comparisons and inserts both new and changed dimensions into the target. Changes are tracked in the target table by maintaining an effective date range for each version of each dimension in the target. In the Type 2 Dimension/Effective Date Range target, the current version of a dimension has a begin date with no corresponding end date.

Use the Type 2 Dimension/Effective Date Range mapping to update a slowly changing dimension table when you want to keep a full history of dimension data in the table. An effective date range tracks the chronological history of changes for each dimension.

When you use this option, the Designer creates the following additional fields in the target:

- **PM_BEGIN_DATE.** For each new and changed dimension written to the target, the Integration Service uses the system date to indicate the start of the effective date range for the dimension.
- **PM_END_DATE.** For each dimension being updated, the Integration Service uses the system date to indicate the end of the effective date range for the dimension.
- **PM_PRIMARYKEY.** The Integration Service generates a primary key for each row written to the target.

Maintaining the Effective Date Range

The Integration Service generates a begin date for each new and changed dimension it inserts into the target, using the current system date. The end date for these dimensions is NULL.

Each time the Integration Service inserts a changed dimension, it updates the previous version of the dimension in the target, using the current system date to fill the previously null end date column.

As a result, all current dimension data in the Type 2 Dimension/Effective Date Range target have null values in the PM_END_DATE column. All previous versions of dimension data have a system date in PM_END_DATE to indicate the end of the effective date range for each version.

For example, the following dimensions are current dimension data since their end date columns are null:

PM_PRIMARYKEY	ITEM	STYLES	PM_BEGIN_DATE	PM_END_DATE
4325	Sock	13	9/1/98	-
5401	Boot	20	10/1/98	-

When the Integration Service finds updated versions of these dimensions in the source, it inserts them into the target, using the system date to indicate the beginning of their effective date ranges and leaving the end dates null.

The Integration Service also updates the existing versions in the target, entering the system date to end the effective date range:

PM_PRIMARYKEY	ITEM	STYLES	PM_BEGIN_DATE	PM_END_DATE
4325	Sock	13	9/1/98	6/1/99
5401	Boot	20	10/1/98	6/1/99
6345	Sock	18	6/1/99	-
6346	Boot	25	6/1/99	-

Handling Keys

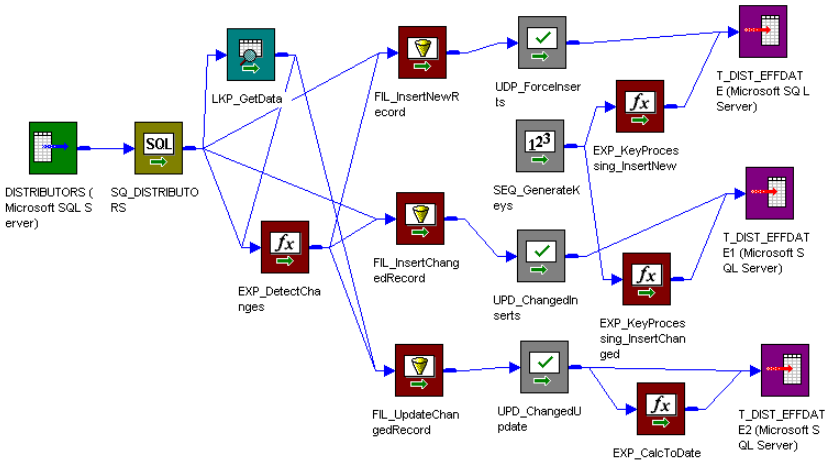
When you use the Effective Date Range option, the Integration Service generates a primary key value for each row written to the target, incrementing key values by one.

Understanding the Mapping

The Type 2 Dimension/Effective Date Range mapping performs the following tasks:

- Selects all rows.
- Caches the existing target as a lookup table.
- Compares logical key columns in the source against corresponding columns in the target lookup table.
- Compares source columns against corresponding target columns if key columns match.
- Flags new rows and changed rows.
- Creates three data flows: one for new rows, one for changed rows, one for updating existing rows.
- Generates a primary key and beginning of the effective date range for new rows.
- Inserts new rows to the target.
- Generates a primary key and beginning of the effective date range for changed rows.
- Inserts changed rows in the target.
- Updates existing versions of the changed rows in the target, generating the end of the effective date range to indicate the row is no longer current.

The following figure shows a mapping that the Type 2 Dimension/Effective Date Range option in the Slowly Changing Dimensions Wizard creates:



The Type 2 Dimension/Effective Date Range mapping uses a Lookup and an Expression transformation to compare source data against existing target data. When you step through the Slowly Changing Dimensions Wizard, you enter the lookup conditions (source key columns) and source columns that you want the Integration Service to compare against the existing target.

For each source row without a matching primary key in the target, the Expression transformation marks the row new. For each source row with a matching primary key in the target, the Expression compares user-defined source and target columns. If those columns do not match, the Expression marks the row changed. The mapping then splits into three data flows.

The first data flow uses the Filter transformation, FIL_InsertNewRecord, to filter out existing rows. The Filter transformation passes only new rows to the UPD_ForceInserts Update Strategy transformation. UPD_ForceInserts inserts new rows to the target. A Sequence Generator creates a primary key for each row. The Expression transformation, EXP_KeyProcessing_InsertNew, uses the system date to indicate the start of the effective date range. The transformation leaves the end date null, which indicates the new row contains current dimension data.

In the second data flow, the FIL_InsertChangedRecord Filter transformation allows only changed rows to pass to the Update Strategy transformation, UPD_ChangedInserts. UPD_ChangedInserts inserts changed rows to the target. The Expression transformation, EXP_KeyProcessing_InsertChanged, uses the system date to indicate the start of the effective date range. The transformation leaves the end date null, which indicates the changed row contains current dimension data.

In the third data flow, for each changed row written to the target, the Filter transformation, FIL_UpdateChangedRecord, passes the primary key of the previous version to the Update Strategy transformation, UPD_ChangedUpdate. UPD_ChangedUpdate updates rows in the target. The Expression transformation, EXP_UpdateChanged, updates the end date column with the system date. This changes the status of the dimension from the current version to a previous version.

Understanding the Transformations

The following table describes the function of each transformation in the Type 2 Dimension/Effective Date Range mapping:

Transformation Name	Transformation Type	Description
SQ_*SourceName*	Source Qualifier or Application Source Qualifier	Selects all rows from the source you choose in the Mapping Wizard.
LKP_GetData	Lookup	Caches the existing target table. Compares key columns in the source against corresponding key columns in the target. When matching keys exist, LKP_GetData returns additional column data from the target for comparison. Passes all rows to EXP_DetectChanges.
EXP_DetectChanges	Expression	Uses the following expression to flag source rows that *do not* have matching keys in the target. The expression returns TRUE if a matching key does not exist in the target, indicating the row is new: `IIF(ISNULL(PM_PRIMARYKEY),TRUE,FALSE)` Populates the NewFlag port with the results. Uses the following expression to flag source rows that *have* a matching key in the target and contain changes in the specified columns. The expression returns TRUE only if a matching key exists in the target (indicating the row is not new) and if it detects a difference between source and target columns: `IIF(ISNULL(PM_PRIMARYKEY) AND (SourceColumnName<>PM_PREV_TargetColumnName) AND (other comparisons) TRUE,FALSE)` Populates the ChangedFlag port with the results. Passes all rows to FIL_InsertNewRecord, FIL_InsertChangedRecord, and FIL_UpdateChangedRecord.

Data Flow for New Rows

The following table describes the data flow for new rows. For each new row in the source, this data flow creates a primary key, sets the beginning of the effective date range, and inserts the row into the target:

Transformation Name	Transformation Type	Description
FIL_InsertNewRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked new (TRUE): `NewFlag`. Passes new rows to UPD_ForceInserts.
UPD_ForceInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.
SEQ_GenerateKeys	Sequence Generator	Generates a value for each new row written to the target, incrementing values by 1. Passes values to EXP_KeyProcessing_InsertNew.

Transformation Name	Transformation Type	Description
EXP_KeyProcessing_InsertNew	Expression	Passes generated values to the target to populate the PM_PRIMARYKEY column in the target. Uses `SYSDATE` to populate the PM_BEGIN_DATE column in the target, marking the beginning of the effective date range.
T_*TargetName*	Target Definition	Instance of the target definition for new rows to be inserted into the target.

Data Flow for Changed Rows

The following table describes the data flow for changed rows. For each changed row in the source, this data flow creates a new primary key, sets the beginning of the effective date range, and inserts the row into the target:

Transformation Name	Transformation Type	Description
FIL_InsertChangedRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked changed (TRUE): `ChangedFlag`. Passes changed rows to UPD_ChangedInserts.
UPD_ChangedInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.
SEQ_GenerateKeys (same Sequence Generator as above)	Sequence Generator	Generates a value for each changed row written to the target, incrementing values by 1. Passes values to EXP_KeyProcessing_InsertChanged.
EXP_KeyProcessing_InsertChanged	Expression	Passes generated values to the target to populate the PM_PRIMARYKEY column in the target. Uses `SYSDATE` to populate the PM_BEGIN_DATE column in the target, marking the beginning of the effective date range.
T_*TargetName*2	Target Definition	Instance of the target definition allowing changed rows to be inserted into the target.

Data Flow to Update Existing Rows

The following table describes the data flow for existing rows. For each changed row in the source, this data flow updates the end date of the corresponding row in the target to indicate the existing row is no longer current:

Transformation Name	Transformation Type	Description
FIL_UpdateChangedRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked changed (TRUE): `ChangedFlag`. For each changed row, passes the primary key of the previous version to UPD_ChangedUpdate.
UPD_ChangedUpdate	Update Strategy	Uses `DD_UPDATE` to update existing rows in the target.
EXP_CalcToDate	Expression	Uses `SYSDATE` to update the PM_END_DATE column in the existing target row, marking the end of the effective date range.
T_*TargetName*3	Target Definition	Instance of the target definition allowing updates to existing rows in the target.

Steps to Create a Type 2 Dimension/Effective Date Range Mapping

To create a Type 2 Dimension/Effective Date Range mapping:

1. In the Mapping Designer, click Mappings > Wizards > Slowly Changing Dimensions.
2. Enter a mapping name and select Type 2 Dimension. Click Next.

   The naming convention for mappings is m_*MappingName*.
3. Select a source definition to be used by the mapping.

   All available source definitions appear in the Select Source Table list. This list includes shortcuts, flat file, relational, and Application sources.
4. Enter a name for the mapping target table. Click Next.

   The naming convention for target definitions is T_*TARGET_NAME*.
5. Select the column or columns you want to use as a lookup condition from the Target Table Fields list and click Add.

   The wizard adds selected columns to the Logical Key Fields list.

   Tip: The columns you select should be a key column in the source.

   When you run the session, the Integration Service performs a lookup on existing target data. The Integration Service returns target data when Logical Key Fields columns match corresponding target columns.

   To remove a column from Logical Key Fields, select the column and click Remove.
6. Select the column or columns you want the Integration Service to compare for changes, and click Add.

   The wizard adds selected columns to the Fields to Compare for Changes list.

   When you run the session, the Integration Service compares the columns in the Fields to Compare for Changes list between source rows and the corresponding target rows. If the Integration Service detects a change, it marks the row changed.

   To remove a column from the list, select the column and click Remove.

7. Click Next.
8. Select Mark the Dimension Records with their Effective Date Range. Click Finish.

   The new mapping appears in the workspace. Make the necessary edits to the mappings.

   Note: In the Type 2 Dimension/Effective Date Range mapping, the Designer uses three instances of the same target definition to enable the three separate data flows to write to the same target table. Generate only one target table in the target database.

Configuring a Type 2 Dimension/Effective Date Range Session

The Type 2 Dimension/Effective Date Range mapping inserts both new and updated rows with a unique primary key. It also updates existing rows in the target. When you configure a session for the mapping, complete the following steps:

1. In the session properties, click the General Options settings on the Properties tab. Set Treat Source Rows As to Data Driven.
2. In the session properties, click the Target Properties settings on the Mappings tab. To verify that the Integration Service loads rows to the target properly, select Insert and Update as Update for each relational target.

Creating a Type 3 Dimension Mapping

The Type 3 Dimension mapping filters source rows based on user-defined comparisons and inserts only those found to be new dimensions to the target. Rows containing changes to existing dimensions are updated in the target. When updating an existing dimension, the Integration Service saves existing data in different columns of the same row and replaces the existing data with the updates. The Integration Service optionally enters the system date as a timestamp for each row it inserts or updates. In the Type 3 Dimension target, each dimension contains current dimension data.

Use the Type 3 Dimension mapping to update a slowly changing dimension table when you want to keep only current and previous versions of column data in the table. Both versions of the specified column or columns are saved in the same row.

When you use this option, the Designer creates additional fields in the target:

- **PM_PREV_*ColumnName*.** The Designer generates a *previous* column corresponding to each column for which you want historical data. The Integration Service keeps the previous version of dimension data in these columns.
- **PM_PRIMARYKEY.** The Integration Service generates a primary key for each row written to the target.
- **PM_EFFECT_DATE.** An optional field. The Integration Service uses the system date to indicate when it creates or updates a dimension.

Saving Previous Values

When you step through the Slowly Changing Dimensions Wizard, you choose the columns in which you want to detect changes. The Designer creates an additional column for each column you select and names the additional column after the original, PM_PREV_*ColumnName*. The Integration Service uses these columns to store previous dimension values.

When the Integration Service writes a new dimension to the target, the *previous* columns remain null. Each time the Integration Service updates a dimension, it writes existing data into the corresponding *previous*

column, and then writes updates into the original column. As a result, each row in a Type 3 Dimension target contains current dimension data. Each row also contains the previous versions of dimension data, if the dimension has changed.

For example, the first time the Integration Service writes the following dimensions to the target, the *previous* column, PM_PREV_STYLES remains null:

PM_PRIMARYKEY	ITEM	STYLES	PM_PREV_STYLES
6345	Sock	20	-
6346	Boot	25	-

When the Integration Service updates these rows, it writes the values in the STYLES column (20 and 25) into PM_PREV_STYLES, and then replaces the style data with new style data (14 and 31):

PM_PRIMARYKEY	ITEM	STYLES	PM_PREV_STYLES
6345	Sock	14	20
6346	Boot	31	25

Handling Keys

In the Type 3 Dimension mapping, the Integration Service generates a primary key value for each new row written to the target, incrementing key values by one. Updated rows retain their original key values.

Marking the Effective Date

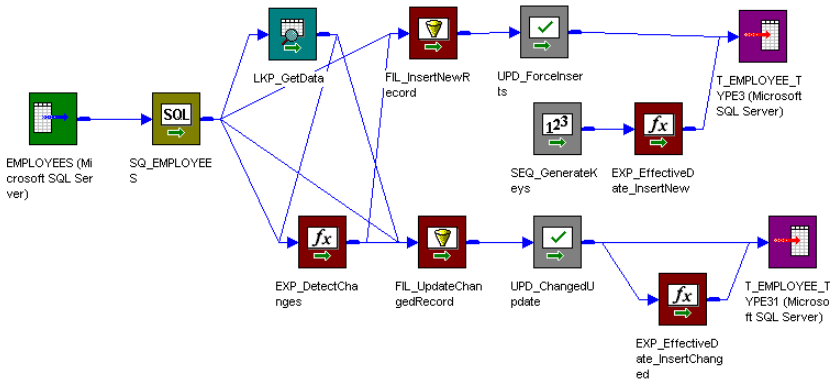
The Type 3 Dimension mapping can optionally note the date on which the Integration Service creates or updates a dimension. If you choose this option, the Designer creates the PM_EFFECT_DATE column. The Integration Service enters the system date in this column each time it creates a new row or updates a row.

Understanding the Mapping

The Type 3 Dimension mapping performs the following tasks:

- Selects all rows.
- Caches the existing target as a lookup table.
- Compares logical key columns in the source against corresponding columns in the target lookup table.
- Compares source columns against corresponding target columns if key columns match.
- Flags new rows and changed rows.
- Creates two data flows: one for new rows, one for updating changed rows.
- Generates a primary key and optionally notes the effective date for new rows.
- Inserts new rows to the target.
- Writes previous values for each changed row into *previous* columns and replaces previous values with updated values.
- Optionally uses the system date to note the effective date for inserted and updated values.
- Updates changed rows in the target.

The following figure shows a mapping that the Type 3 Dimension option in the Slowly Changing Dimensions Wizard creates:



The Type 3 Dimension mapping uses a Lookup and an Expression transformation to compare source data against existing target data. When you step through the Slowly Changing Dimensions Wizard, you enter the lookup conditions (source key columns) and source columns that you want the Integration Service to compare against the existing target. The Designer creates additional columns for the change columns to hold historic data.

For each source row without a matching primary key in the target, the Expression transformation marks the row new. For each source row with a matching primary key in the target, the Expression compares user-defined source and target columns. If those columns do not match, the Expression marks the row changed. The mapping then splits into two data flows.

The first data flow uses the Filter transformation, FIL_InsertNewRecord, to filter out rows. The Filter transformation passes only new rows to the UPD_ForceInserts Update Strategy transformation. UPD_ForceInserts inserts new rows to the target. A Sequence Generator creates a primary key for each row. If you select the Effective Date option in the mapping wizard, the Designer creates an Expression transformation, EXP_EffectiveDate_InsertNew. The Integration Service uses the system date to indicate when it creates a new row.

In the second data flow, the FIL_UpdateChangedRecord Filter transformation allows only changed rows to pass to the Update Strategy transformation UPD_ChangedInserts. In addition, the Filter transformation updates the changed row: it takes the new versions of data from the source qualifier, and uses existing versions of dimension data (passed from the Lookup transformation) to populate the *previous* column fields. UPD_ChangedInserts inserts changed rows to the target. If you select the Effective Date option in the mapping wizard, the Designer creates an Expression transformation, EXP_EffectiveDate_InsertChanged. The Integration Service uses the system date to indicate when it updates a row.

Understanding the Transformations

The following table describes the function of each transformation in the Type 3 Dimension mapping:

Transformation Name	Transformation Type	Description
SQ_*SourceName*	Source Qualifier or Application Source Qualifier	Selects all rows from the source you choose in the Mapping Wizard.
LKP_GetData	Lookup	Caches the existing target table. Compares key columns in the source against corresponding key columns in the target. When matching keys exist, LKP_GetData returns additional column data from the target for comparison. Passes all rows to EXP_DetectChanges.
EXP_DetectChanges	Expression	Uses the following expression to flag source rows that *do not* have matching keys in the target. The expression returns TRUE if a matching key does not exist in the target, indicating the row is new: `IIF(ISNULL(PM_PRIMARYKEY),TRUE,FALSE)` Populates the NewFlag port with the results. Uses the following expression to flag source rows that *have* a matching key in the target and contain changes in the specified columns. The expression returns TRUE only if a matching key exists in the target (indicating the row is not new) and if it detects a difference between source and target columns: `IIF(ISNULL(PM_PRIMARYKEY) AND (SourceColumnName<>PM_PREV_TargetColumnName) AND (other comparisons) TRUE,FALSE)` Populates the ChangedFlag port with the results. Passes all rows to FIL_InsertNewRecord and FIL_UpdateChangedRecord.

Data Flow for New Rows

The following table describes the data flow for new rows. For each new row in the source, this data flow creates a primary key, optionally notes the load date, and inserts the row into the target:

Transformation Name	Transformation Type	Description
FIL_InsertNewRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked new (TRUE): `NewFlag`. Passes new rows to UPD_ForceInserts.
UPD_ForceInserts	Update Strategy	Uses `DD_INSERT` to insert rows to the target.
SEQ_GenerateKeys	Sequence Generator	Generates a value for each new row written to the target, incrementing values by 1. Passes values to EXP_KeyProcessing_InsertNew, if you selected the Mapping Wizard Effective Date option. Otherwise, passes values to the target to populate the PM_PRIMARYKEY column.

Transformation Name	Transformation Type	Description
EXP_KeyProcessing_InsertNew	Expression	Created only when you select the Effective Date option in the Mapping Wizard. Passes generated values to the target to populate the PM_PRIMARYKEY column in the target. Uses `SYSDATE` to populate the PM_EFFECT_DATE column in the target, indicating when the row is created.
T_*TargetName*	Target Definition	Instance of the target definition for new rows to be inserted into the target.

Data Flow to Update Existing Rows

The following table describes the data flow for existing rows. For each changed row in the source, this data flow updates the corresponding row in the target, copying existing data into *previous* columns, updating new data, and optionally notes the date the row changed:

Transformation Name	Transformation Type	Description
FIL_UpdateChangedRecord	Filter	Uses the following filter condition to filter out any rows from EXP_DetectChanges that are not marked changed (TRUE): `ChangedFlag`. Uses values returned from LKP_GetData to populate *previous* columns. Passes changed rows to UPD_ChangedUpdate.
UPD_ChangedUpdate	Update Strategy	Uses `DD_UPDATE` to update existing rows in the target. Passes updated rows to EXP_EffectiveDate_InsertChanged, if you selected the Mapping Wizard Effective Date option. Otherwise, passes updated rows to the target.
EXP_EffectiveDate_InsertChanged	Expression	Created only when you select the Effective Date option in the Mapping Wizard. Uses `SYSDATE` to populate the PM_EFFECT_DATE column in the target, indicating when the row is updated.
T_*TargetName*2	Target Definition	Instance of the target definition allowing updates to existing rows in the target.

Steps to Create a Type 3 Dimension Mapping

To create a Type 3 Dimension mapping:

1. In the Mapping Designer, click Mappings > Wizards > Slowly Changing Dimensions.
2. Enter a mapping name and select Type 3 Dimension. Click Next.

   The naming convention for mappings is m_*MappingName*.
3. Select a source definition to be used by the mapping.

All available source definitions appear in the Select Source Table list. This list includes shortcuts, flat file, relational, and Application sources.

4. Enter a name for the mapping target table. Click Next.

   The naming convention for target definitions is T_*TARGET_NAME*.

5. Select the column or columns you want to use as a lookup condition from the Target Table Fields list and click Add.

   The wizard adds selected columns to the Logical Key Fields list.

   Tip: The columns you select should be a key column in the source.

   When you run the session, the Integration Service performs a lookup on existing target data. The Integration Service returns target data when Logical Key Fields columns match corresponding target columns.

   To remove a column from Logical Key Fields, select the column and click Remove.

6. Select the column or columns you want the Integration Service to compare for changes, and click Add.

   The wizard adds selected columns to the Fields to Compare for Changes list.

   When you run the session, the Integration Service compares the columns in the Fields to Compare for Changes list between source rows and the corresponding target (lookup) rows. If the Integration Service detects a change, it marks the row changed.

   Note: Select the columns for which you want to keep previous values. To keep previous values in the target, the Designer creates an additional column for each column in this list. It names the columns PM_PREV_*ColumnName*.

   To remove a column from the list, select the column and click Remove.

7. Click Next.

8. If you want the Integration Service to timestamp new and changed rows, select Effective Date.

   The wizard displays the columns the Integration Service compares and the name of the column to hold historic values.

9. Click Finish.

   Note: In the Type 3 Dimension mapping, the Designer uses two instances of the same target definition to enable the two separate data flows to write to the same target table. Generate only one target table in the target database.

Configuring a Type 3 Dimension Session

The Type 3 Dimension mapping inserts new rows and updates existing rows in the target. When you configure a session, complete the following steps:

1. Click the General Options settings on the Properties tab. Set Treat Source Rows As to Data Driven.
2. Click the Target Properties settings on the Mappings tab. To verify that the Integration Service loads rows to the target properly, select Insert and Update as Update for each relational target.

Creating Targets in the Target Database

The mapping wizards create a new target definition based on columns in the source definition and the options you select. This ensures the resulting target table contains the necessary columns for the mapping. By default, the resulting target definition has:

- The same database type as the repository.
- The same column names, definitions, descriptions, and key restrictions as the source definition.
- Any additional columns necessary to track or update changes.

After a wizard creates a target definition for the mapping, use the Target Designer to create the target table in the target database before running the session.

If the repository and target database types differ, be sure to change the database type of the target definition to match the target database before generating the target.

Scheduling Sessions and Workflows

After you create mappings to load the star schemas, create sessions and workflows to run the mappings. To verify that the Integration Service loads each table appropriately, configure each session according to the mapping type you created.

To verify that the fact tables contain the most recent data, refresh all dimension tables before loading the fact table. You can accomplish this by creating a workflow that executes the dimension sessions before executing the fact session.

To verify that all dimension sessions successfully complete before the Integration Service runs a fact session, use the following workflow logic:

1. Place all dimension sessions in the workflow before the fact session.
2. Link all dimension sessions to a Decision task.
3. Define the following decision condition:

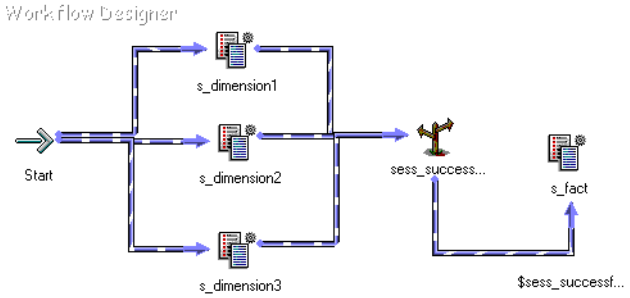
   ```
   $<session_name1>.Status = SUCCEEDED AND $<session_name2>.Status = SUCCEEDED AND ...
   $<session_nameN>.Status = SUCCEEDED
   ```

4. Place the fact session in the workflow after the Decision task, and link the Decision task to the fact session.
5. Configure the following link condition from the Decision task to the fact session:

   ```
   $<Decision_name>.Condition = TRUE
   ```

For example, you create three sessions that load data to dimension tables and one session that loads data to a fact table.

The following figure shows the workflow you can create:



Define the following decision condition for the sess_successful Decision task:

```
$s_dimension1.Status = SUCCEEDED AND $s_dimension2.Status = SUCCEEDED AND
$s_dimension3.Status = SUCCEEDED
```

Define the following link condition for the link from the Decision task to the s_fact fact session:

```
$sess_successful.Condition = TRUE
```

To improve workflow performance, run the dimension sessions concurrently. Arrange the sessions in the workflow to start at the same time. If the Integration Service cannot start and run all dimension sessions concurrently due to the existing session load, you can run the dimension sessions sequentially by arranging the dimension sessions sequentially in the workflow.

If you have a dimension table that does not need to reload each time you run the workflow, you can disable that session.

Creating a Mapping from the Informatica Mapping Templates

Informatica mapping templates are predefined mapping templates that cover common data warehousing patterns. Slowly Changing Dimensions, Remove Duplicates, and Incremental Load are the data warehousing patterns for which Informatica provides mapping templates. These templates provide solutions for common issues in the data warehousing designs. You can use the predefined mapping templates to document methods to map or process data.

Use the Import Mapping Template Wizard to specify a mapping name, description, and parameter values for each mapping that you want to create from the Informatica mapping template.

Note: When you reuse a parameter file, verify that the parameter file and the mapping template are of the same parameter type.

To create a mapping from the mapping template wizards, complete the following tasks:

1. Select the mapping template.
2. Specify the mapping details and the parameter values.
3. Create the mappings and save the parameter values.
4. Import the mappings into the repository.

Step 1. Select the Mapping Template

In the first step of the Mapping Template Wizard, select the mapping template that you want to use.

To start the Mapping Template Wizard:

1. Open the Designer and connect to the repository.
2. Open the folder that contains the source and target objects or a shortcut to the source and target objects that you want to use in the imported mappings.
3. Select Mapping > Mapping Template Wizards and select the mapping template that you want to use.

   The Import Mapping Template Wizard appears.
4. To use an existing mapping template parameter file, click Use Existing. Navigate to the location where the parameter file is stored, and then select the file.

   By default, the parameter files are stored in the following directory:

   ```
   C:\Documents and Settings\<current user>\Local Settings\Temp
   ```

Step 2. Specify Mapping Details and Parameter Values

You can specify the parameter values for each mapping that you want to generate.

To specify the mapping details and the parameter values:

1. In the first page of the Import Mapping Template Wizard, click the Add button.
2. In the MappingName field, enter a mapping name.
3. Optionally, enter a description.
4. Click the Open button in the Parameters/Values field.
5. Enter the following information in the Parameters for Mapping dialog box:

Field	Description
Source Table	All available source definitions of the current working directory.
Incremental Extract Condition	Incremental extract condition. For example, enter UPDATE-TS>SYSDATE-1 as incremental extract condition.
Create Target Table	Creates a target table instead of using an existing target table. The target database type will be the same as the source database type.
Target Table	All available target definitions of the repository folder.
Surrogate Key	Primary key of the target table.
Logical Key Fields	Columns of the source table that identify a particular entity.
Comparison Key Fields	Set of fields that identify a changed row between the source and target tables.

   The Database Types field shows the type of the database that you selected. You cannot modify this option.
6. If you use an existing target table, set the field association. To set the field association, click Set Fields Association.
7. In the Field Association dialog box, click the Add button.
8. Select the source field from the Source Fields list.
9. Select the target field from the Target Fields list.

10. In the Field Association dialog box, click OK.
11. In the Parameters for Mapping dialog box, click OK.
12. In the Import Mapping Template Wizard, click Next.

    The second page of the Import Mapping Template Wizard displays the list of mappings you configured.

Step 3. Create Mappings and Save Parameter Values

You can select the mappings that you want to generate. To save parameter values for the selected mapping, reuse the mapping template parameter file and create more mappings. The wizard saves the parameter values for each mapping that you selected in the mapping template parameter file.

Note: Save the parameter values to a mapping template parameter file. If you save the parameter file for a mapping and an error occurs while you generate that mapping, you can retrieve the parameter value settings.

To create the mappings and save the parameter values:

1. From the list of mappings that appear, select the mappings that you want to generate.
2. To create one mapping with multiple pipelines, click Create multiple pipelines in a single mapping, and then enter a name for the mapping.
3. Click Save parameter values for the selected mappings, and then click Browse to navigate to the folder where you want to save the parameter file.

   By default, the Publish Template function creates a mapping template parameter file in the same location as the mapping template file. You can choose to overwrite the existing file or create a mapping template parameter file.
4. Click Next.

   The wizard prompts you to export table definitions.
5. Click Yes.

   The Export Object dialog box appears, and the table definitions are exported to the default location.
6. Click Close in the Export Objects dialog box.

Step 4. Import Mappings into the Repository

You can import mappings into the Repository using these steps.

To import the mappings into the repository:

1. Review the list of mappings to verify that the wizard generated the correct number of mappings.

   By default, the option to create workflows and sessions for the mappings is selected.
2. To launch the Workflow Generation Wizard, click Next. For more information about the Workflow Generation Wizard, see "Using the Workflow Generation Wizard" on page 136.

   To disable the option to create workflows and sessions for the mappings, click Finish.

   The generated mappings appear in the mappings node of the selected repository folder.

APPENDIX A

Datatype Reference

This appendix includes the following topics:

- Datatype Reference Overview, 248
- Transformation Data Types, 249
- IBM DB2 and Transformation Datatypes, 254
- Informix and Transformation Datatypes, 255
- Microsoft SQL Server and Transformation Datatypes, 257
- Oracle and Transformation Datatypes, 259
- SAP HANA and Transformation Datatypes, 260
- Sybase and Transformation Datatypes, 261
- Teradata and Transformation Datatypes, 264
- ODBC and Transformation Datatypes, 265
- COBOL and Transformation Datatypes, 266
- Flat File and Transformation Datatypes, 266
- XML and Transformation Datatypes, 267
- Converting Data, 267

Datatype Reference Overview

When you create a mapping, you create a set of instructions for the Integration Service to read data from the source table, transform it, and write it to a target table. The Integration Service transforms data based on dataflow in the mapping, starting at the first transformation in the mapping, and the datatype assigned to each port in a mapping.

The Designer displays two types of datatypes:

- **Native datatypes.** Specific to the source and target databases, flat files, or ERP systems. Native datatypes appear in the Source Analyzer and Target Designer. They also appear in source and target definitions in the Mapping Designer and in source definitions in the Mapplet Designer.
- **Transformation datatypes.** Set of datatypes that appear in the transformations. They are internal datatypes based on ANSI SQL-92 generic datatypes, which the Integration Service uses to move data across platforms. Because the transformation datatypes are generic, you can use different source and target platforms. For example, you can read information from an Oracle source and write it to a Sybase target. Likewise, you can read information from a flat file and write it to a Microsoft SQL Server database. The transformation datatypes appear in all transformations in a mapping.

When the Integration Service reads source data, it converts the native datatypes to the comparable transformation datatypes before transforming the data. When the Integration Service writes to a target, it converts the transformation datatypes to the comparable native datatypes.

When you specify a multibyte character set, the datatypes allocate additional space in the database to store characters of up to three bytes.

Transformation Data Types

The following table describes the transformation data types:

Data Type	Size in Bytes	Description
Bigint	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision of 19, scale of 0 Integer value.
Binary	Precision	1 to 104,857,600 bytes You cannot use binary data for COBOL or flat file sources.
Date/Time	16 bytes	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. Precision of 29, scale of 9 (precision to the nanosecond) Combined date/time value.
Decimal	8 bytes (if high precision is off or precision is greater than 28) 16 bytes (if precision <= 18 and high precision is on) 20 bytes (if precision > 18 and <= 28)	Decimal value with declared precision and scale. Scale must be less than or equal to precision. Precision 1 to 28 digits, scale 0 to 28
Double	8 bytes	Precision of 15 digits Double-precision floating-point numeric value.
Integer	4 bytes	-2,147,483,648 to 2,147,483,647 Precision of 10, scale of 0 Integer value.
Nstring	Unicode mode: (precision + 1) * 2 ASCII mode: precision + 1	1 to 104,857,600 characters Fixed-length or varying-length string.
Ntext	Unicode mode: (precision + 1) * 2 ASCII mode: precision + 1	1 to 104,857,600 characters Fixed-length or varying-length string.
Real	8 bytes	Precision of 7, scale of 0 Double-precision floating-point numeric value.

Data Type	Size in Bytes	Description
Small Integer	4 bytes	-32,768 and 32,767 Precision of 5, scale of 0 Integer value.
String	Unicode mode: (precision + 1) * 2 ASCII mode: precision + 1	1 to 104,857,600 characters Fixed-length or varying-length string.
Text	Unicode mode: (precision + 1) * 2 ASCII mode: precision + 1	1 to 104,857,600 characters Fixed-length or varying-length string.

Integer Datatypes

You can pass integer data from sources to targets and perform transformations on integer data. The transformation language supports Bigint, Integer, and Small Integer datatypes.

The transformation integer datatypes represent exact values.

Integer Values in Calculations

When you use integer values in calculations, the PowerCenter Integration Service sometimes converts integer values to floating-point numbers before it performs the calculation. For example, to evaluate MOD(12.00, 5), the PowerCenter Integration Service converts the integer value “5” to a floating-point number before it performs the division operation. The PowerCenter Integration Service converts integer values to double or decimal values depending on whether you enable high precision.

The PowerCenter Integration Service converts integer values in the following arithmetic operations:

Arithmetic Operation	High Precision Disabled	High Precision Enabled
Functions and calculations that cannot introduce decimal points. For example, integer addition, subtraction, and multiplication, and functions such as CUME, MOVINGSUM, and SUM.	No conversion[1]	Decimal
Non-scientific functions and calculations that can introduce decimal points. For example, integer division, and functions such as AVG, MEDIAN, and PERCENTILE.	Double	Decimal
All scientific functions and the EXP, LN, LOG, POWER, and SQRT functions.	Double	Double

1. If the calculation produces a result that is out of range, the Integration Service writes a row error.

The transformation Double data type supports precision of up to 15 digits, while the Bigint data type supports precision of up to 19 digits. Therefore, precision loss can occur in calculations that produce Bigint values with precision of more than 15 digits.

For example, an expression transformation contains the following calculation:

```
POWER( BIGINTVAL, EXPVAL )
```

Before it performs the calculation, the PowerCenter Integration Service converts the inputs to the POWER function to double values. If the BIGINTVAL port contains the Bigint value 9223372036854775807, the PowerCenter Integration Service converts this value to 9.22337203685478e+18, losing the last 4 digits of precision. If the EXPVAL port contains the value 1.0 and the result port is a Bigint, this calculation produces a row error since the result, 9223372036854780000, exceeds the maximum bigint value.

When you use an Integer data type in a calculation that can produce decimal values and you enable high precision, the PowerCenter Integration Service converts the integer values to decimal values.

For transformations that support the Decimal data type with precision up to 28 digits, precision loss does not occur in a calculation unless the result produces a value with precision greater than 28 digits in high precision mode. In this case, the PowerCenter Integration Service stores the result as a double. If the port precision is less than or equal to 28 digits and the result produces a value greater than 28 digits in high precision mode, the PowerCenter Integration Service rejects the row.

Integer Constants in Expressions

The Integration Service interprets constants in an expression as floating-point values, even if the calculation produces an integer result. For example, in the expression INTVALUE + 1000, the Integration Service converts the integer value “1000” to a double value if high precision is not enabled. It converts the value 1000 to a decimal value if high precision is enabled. To process the value 1000 as an integer value, create a variable port with an Integer datatype to hold the constant and modify the expression to add the two ports.

NaN Values

NaN (Not a Number) is a value that is usually returned as the result of an operation on invalid input operands, especially in floating-point calculations. For example, when an operation attempts to divide zero by zero, it returns a NaN result.

Operating systems and programming languages may represent NaN differently. For example the following list shows valid string representations of NaN:

```
nan
NaN
NaN%
NAN
NaNQ
NaNS
qNaN
sNaN
1.#SNAN
1.#QNAN
```

The Integration Service converts QNAN values to 1.#QNAN on Win64EMT platforms. 1.#QNAN is a valid representation of NaN.

Convert String Values to Integer Values

When the Integration Service performs implicit conversion of a string value to an integer value, it truncates the data at the first non-numeric character. For example, you link a string port that contains the value “9,000,000,000,000,000,000.777” to a Bigint port. The Integration Service converts the string to the bigint value 9,000,000,000,000,000,000.

Write Integer Values to Flat Files

When writing integer values to a fixed-width flat file, the file writer does not verify that the data is within range. For example, the file writer writes the result 3,000,000,000 to a target Integer column if the field width of the target column is at least 13. The file writer does not reject the row because the result is outside the valid range for Integer values.

Binary Datatype

If a mapping includes binary data, set the precision for the transformation binary datatype so that the Integration Service can allocate enough memory to move the data from source to target.

You cannot use binary datatypes for COBOL or flat file sources.

Date/Time Datatype

The Date/Time datatype handles years from 1 A.D. to 9999 A.D. in the Gregorian calendar system. Years beyond 9999 A.D. cause an error.

The Date/Time datatype supports dates with precision to the nanosecond. The datatype has a precision of 29 and a scale of 9. Some native datatypes have a smaller precision. When you import a source that contains datetime values, the import process imports the correct precision from the source column. For example, the Microsoft SQL Server Datetime datatype has a precision of 23 and a scale of 3. When you import a Microsoft SQL Server source that contains Datetime values, the Datetime columns in the mapping source have a precision of 23 and a scale of 3.

The Integration Service reads datetime values from the source to the precision specified in the mapping source. When the Integration Service transforms the datetime values, it supports precision up to 29 digits. For example, if you import a datetime value with precision to the millisecond, you can use the ADD_TO_DATE function in an Expression transformation to add nanoseconds to the date.

If you write a Date/Time value to a target column that supports a smaller precision, the Integration Service truncates the value to the precision of the target column. If you write a Date/Time value to a target column that supports a larger precision, the Integration Service inserts zeroes in the unsupported portion of the datetime value.

Decimal and Double Data Types

You can pass decimal and double data from sources to targets and perform transformations on decimal and double data.

The transformation language supports the following data types:

Decimal

Precision 1 to 28 digits, scale 0 to 28 . You cannot use decimal values with scale greater than precision or a negative precision. Transformations display any range you assign to a Decimal data type, but the PowerCenter Integration Service supports precision only up to 28 digits.

When you enable high precision and the port precision is greater than 28 digits, the PowerCenter Integration Service stores the result as a double.

Double

Precision of 15.

Decimal and Double Values in Calculations

Precision loss can occur with Decimal and Double data types in a calculation when the result produces a value with a precision greater than the maximum.

If you disable high precision, the PowerCenter Integration Service converts decimal values to double. Precision loss occurs if the decimal value has a precision greater than 15 digits. For example, you have a mapping with Decimal (20,0) that passes the number 40012030304957666903. If you disable high precision, the PowerCenter Integration Service converts the decimal value to double and passes 4.00120303049577 x 10^{19}.

For transformations that support Decimal data type of precision up to 28 digits, use the Decimal data type and enable high precision to ensure precision of up to 28 digits.

Precision loss does not occur in a calculation unless the result produces a value with precision greater than the maximum allowed digits. In this case, the PowerCenter Integration Service stores the result as a double.

Do not use the Double data type for data that you use in an equality condition, such as a lookup or join condition.

The following table lists how the PowerCenter Integration Service handles decimal values based on high precision configuration:

Port Data Type	Precision	High Precision Off	High Precision On
Decimal	0 to 15	Decimal	Decimal
Decimal	15 to 28	Double	Decimal
Decimal	Over 28	Double	Double

When you enable high precision, the PowerCenter Integration Service converts numeric constants in any expression function to Decimal. If you do not enable high precision, the PowerCenter Integration Service converts numeric constants to Double.

You can ensure the maximum precision for numeric values greater than 28 or 38 digits depending on the transformation. Before you perform any calculations or transformations with the transformation functions, truncate or round any large numbers.

Rounding Methods for Double Values

Due to differences in system run-time libraries and the computer system where the database processes double datatype calculations, the results may not be as expected. The double datatype conforms to the IEEE 794 standard. Changes to database client library, different versions of a database or changes to a system run-time library affect the binary representation of mathematically equivalent values. Also, many system run-time libraries implement the round-to-even or the symmetric arithmetic method. The round-to-even method states that if a number falls midway between the next higher or lower number it round to the nearest value with an even least significant bit. For example, with the round-to-even method, 0.125 is rounded to 0.12. The symmetric arithmetic method rounds the number to next higher digit when the last digit is 5 or greater. For example, with the symmetric arithmetic method 0.125 is rounded to 0.13 and 0.124 is rounded to 0.12.

To provide calculation results that are less susceptible to platform differences, the Integration Service stores the 15 significant digits of double datatype values. For example, if a calculation on Windows returns the number 1234567890.1234567890, and the same calculation on UNIX returns 1234567890.1234569999, the Integration Service converts this number to 1234567890.1234600000.

String Datatypes

The transformation datatypes include the following string datatypes:

- Nstring
- Ntext
- String
- Text

Although the Nstring, Ntext, String, and Text datatypes support the same precision up to 104,857,600 characters, the Integration Service uses String to move string data from source to target and Text to move text data from source to target. Because some databases store text data differently than string data, the Integration Service needs to distinguish between the two types of character data. If the source qualifier displays String, set the target column to String. Likewise, if the source qualifier displays Text, set the target column to Text, Long, or Long Varchar, depending on the source database.

In general, the smaller string datatypes, such as Char and Varchar, display as String in the Source Qualifier, Lookup, and Stored Procedure transformations, while the larger text datatypes, such as Text, Long, and Long Varchar, display as Text in the Source Qualifier.

Use Nstring, Ntext, String, and Text interchangeably within transformations. However, in the Source Qualifier, Lookup, and Stored Procedure transformations, the target datatypes must match. The database drivers need to match the string datatypes with the transformation datatypes, so that the data passes accurately. For example, Nchar in a lookup table must match Nstring in the Lookup transformation.

IBM DB2 and Transformation Datatypes

The following table compares IBM DB2 datatypes and transformation datatypes:

Datatype	Range	Transformation	Range
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Blob	1 to 2,147,483,647 bytes	Binary	1 to 104,857,600 bytes
Char(L)	1 to 254 characters	String	1 to 104,857,600 characters
Char for bit data	1 to 254 bytes	Binary	1 to 104,857,600 bytes
Clob	1 to 2,447,483,647 bytes	Text	1 to 104,857,600 characters
Date	0001 to 9999 A.D. Precision 19; scale 0 (precision to the day)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Dbclob	Up to 1 GB	Ntext	1 to 104,857,600 characters
Decimal(P,S)	Precision 1 to 31, scale 0 to 31	Decimal	Precision 1 to 28, scale 0 to 28
Float	Precision 1 to 15	Double	Precision 15

Datatype	Range	Transformation	Range
Graphic	1 to 127 bytes	Nstring	1 to 104,857,600 characters
Integer	-2,147,483,648 to 2,147,483,647	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Long Varchar	1 to 32,700 characters	String	1 to 104,857,600 characters
Long Vargraphic	Up to 16,350 bytes	Ntext	1 to 104,857,600 characters
Numeric(P,S)	Precision 1 to 31, scale 0 to 31	Decimal	Precision 1 to 28, scale 0 to 28
Smallint	-32,768 to 32,767	Small Integer	Precision 5, scale 0
Time	24-hour time period Precision 19, scale 0 (precision to the second)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Timestamp	26 bytes Precision 26, scale 6 (precision to the microsecond)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Varchar	Up to 4,000 characters	String	1 to 104,857,600 characters
Varchar for bit data	Up to 4,000 bytes	Binary	1 to 104,857,600 bytes
Vargraphic	Up to 16,336 bytes	String	1 to 104,857,600 characters

Informix and Transformation Datatypes

The following table compares Informix datatypes to the transformation datatypes that display in the Source Qualifier, Lookup, and Stored Procedure transformations. You can change the transformation datatype within the mapping, and the Integration Service converts the data. However, the datatype in the Source Qualifier, Lookup, and Stored Procedure transformations must match the source datatype. If they do not match, the mapping is invalid.

Note: Only the Informix Text datatype appears as the transformation Text datatype in Source Qualifier, Lookup, and Stored Procedure transformations. However, use the transformation datatypes Text and String interchangeably.

Informix	Range	Transformation	Range
Byte	Minimum is 1 byte. No maximum size.	Binary	1 to 104,857,600 bytes
Char(L)	1 to 32,767 characters	String	1 to 104,857,600 characters

Informix	Range	Transformation	Range
Date	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. Precision 19, scale 0 (precision to the day)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Datetime year to fraction	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. Precision 21 to 25, scale 1 to 5 (precision to the millisecond)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Decimal(P,S)	Precision 1 to 32, scale 0 to 32	Decimal	Precision 1 to 28, scale 0 to 28
Float(P)	Precision 1 to 14	Double	Precision 15
Int8	-9,223,372,036,854,775,807 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Integer	-2,147,483,647 to 2,147,483,647	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Money(P,S)	Precision 1 to 32, scale is 2	Decimal	Precision 1 to 28, scale 0 to 28
Nvarchar	1 to 255 characters	Nstring	1 to 104,857,600 characters
Serial(L)	-2,147,483,647 to 2,147,483,647	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Smallfloat	Precision 1 to 7, scale 0	Real	Precision 7, scale 0
Smallint	-32,767 to 32,767	Small Integer	Precision 5, scale 0
Text	Maximum length is 2e31	Text	1 to 104,857,600 characters
Varchar(M,R)	1 to 255 characters	String	1 to 104,857,600 characters

Datatype Synonyms

The following table compares Informix synonyms to transformation datatypes:

Synonym	Transformation
Bigint (Extended Parallel Server)	Bigint
Character(L)	String
Character Varying(m,r)	String
Dec	Decimal
Double Precision	Double

Synonym	Transformation
Int	Integer
Numeric	Decimal
Real	Real

Microsoft SQL Server and Transformation Datatypes

The following table compares Microsoft SQL Server datatypes and transformation datatypes:

Microsoft SQL	Range	Transformation	Range
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Binary(L)	1 to 8,000 bytes	Binary	1 to 104,857,600 bytes
Bit	1 bit	String	1 to 104,857,600 characters
Char(L)	1 to 8,000 characters	String	1 to 104,857,600 characters
Datetime	Jan 1, 1753 A.D. to Dec 31, 9999 A.D. Precision 23, scale 3 (precision to 3.33 milliseconds)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Datetime2	Jan 1,1 A.D. to Dec 31, 9999 A.D. Precision 27, scale 7 (precision to 100 nanoseconds)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Decimal(P,S)	Precision 1 to 38, scale 0 to 38	Decimal	Precision 1 to 28, scale 0 to 28
Float	-1.79E+308 to 1.79E+308	Double	Precision 15
Image	1 to 2,147,483,647 bytes	Binary	1 to 104,857,600 bytes
Int	-2,147,483,648 to 2,147,483,647	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Money	-922,337,203,685,477.5807 to 922,337,203,685,477.5807	Decimal	Precision 1 to 28, scale 0 to 28
Nchar	1 to 4,000 characters	Nstring	1 to 104,857,600 characters
Ntext	1 to 1,073,741,823 characters	Ntext	1 to 104,857,600 characters
Numeric(P,S)	Precision 1 to 38, scale 0 to 38	Decimal	Precision 1 to 28, scale 0 to 28

Microsoft SQL	Range	Transformation	Range
Numeric Identity	1 to 9,999	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Nvarchar	1 to 4,000 characters	Nstring	1 to 104,857,600 characters
Real	-3.40E+38 to 3.40E+38	Real	Precision 7, scale 0
Smalldatetime	Jan 1, 1900, to June 6, 2079 Precision 19, scale 0 (precision to the minute)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Smallint	-32,768 to 32,768	Small Integer	Precision 5, scale 0
Smallmoney	-214,748.3648 to 214,748.3647	Decimal	Precision 1 to 28, scale 0 to 28
Sysname	1 to 128 characters	Nstring	1 to 104,857,600 characters
Text	1 to 2,147,483,647 characters	Text	1 to 104,857,600 characters
Timestamp	8 bytes	Binary	1 to 104,857,600 bytes
Tinyint	0 to 255	Small Integer	Precision 5, scale 0
Varbinary(L)	1 to 8,000 bytes	Binary	1 to 104,857,600 bytes
Varchar(L)	1 to 8,000 characters	String	1 to 104,857,600 characters

Datatype Synonyms

The following table compares Microsoft SQL Server synonyms to transformation datatypes:

Synonym	Transformation
Binary Varying	Binary
Character	String
Character Varying	String
Dec	Decimal
Double Precision	Double
Integer	Integer

Uniqueidentifier Datatype

PowerCenter imports the Microsoft SQL Server uniqueidentifier datatype as a Microsoft SQL Server Varchar datatype of 38 characters.

Oracle and Transformation Datatypes

The following table compares Oracle datatypes and transformation datatypes:

Oracle	Range	Transformation	Range
Blob	Up to 4 GB	Binary	1 to 104,857,600 bytes
Char(L)	1 to 2,000 bytes	String	1 to 104,857,600 characters
Clob	Up to 4 GB	Text	1 to 104,857,600 characters
Date	Jan. 1, 4712 B.C. to Dec. 31, 4712 A.D. Precision 19, scale 0	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Long	Up to 2 GB	Text	1 to 104,857,600 characters If you include Long data in a mapping, the Integration Service converts it to the transformation String datatype, and truncates it to 104,857,600 characters.
Long Raw	Up to 2 GB	Binary	1 to 104,857,600 bytes
Nchar	1 to 2,000 bytes	Nstring	1 to 104,857,600 characters
Nclob	Up to 4 GB	Ntext	1 to 104,857,600 characters
Number	Precision of 1 to 38	Double	Precision of 15
Number(P,S)	Precision of 1 to 38, scale of 0 to 38	Decimal	Precision of 1 to 28, scale of 0 to 28
Nvarchar2	1 to 4,000 bytes	Nstring	1 to 104,857,600 characters
Raw(L)	1 to 2,000 bytes	Binary	1 to 104,857,600 bytes
Timestamp	Jan. 1, 4712 B.C. to Dec. 31, 9999 A.D. Precision 19 to 29, scale 0 to 9 (precision to the nanosecond)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Varchar(L)	1 to 4,000 bytes	String	1 to 104,857,600 characters
Varchar2(L)	1 to 4,000 bytes	String	1 to 104,857,600 characters
XMLType based CLOB storage	Up to 4 GB	Text	1 to 104,857,600 characters

Number(P,S) Datatype

PowerCenter supports Oracle Number(P,S) values with negative scale. However, it does not support Number(P,S) values with scale greater than precision 28 or a negative precision.

If you import a table with an Oracle Number with a negative scale, the Mapping Designer displays it as a Decimal datatype, but the Integration Service converts it to a double.

Char, Varchar, Clob Datatypes

When the Integration Service uses the Unicode data movement mode, the Integration Service reads the precision of Char, Varchar, and Clob columns based on the length semantics you set for columns in the Oracle database. If you use the byte semantics to determine column length, the Integration Service reads the precision as the number of bytes. If you use the char semantics, the Integration Service reads the precision as the number of characters.

SAP HANA and Transformation Datatypes

The following table compares SAP HANA datatypes and transformation datatypes:

SAP HANA Datatype	Range	Transformation Datatype	Range
Alphanum	Precision 1 to 127	Nstring	1 to 104,857,600 characters
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Binary	Used to store bytes of binary data	Binary	1 to 104,857,600 bytes
Blob	Up to 2 GB	Binary	1 to 104,857,600 bytes
Clob	Up to 2 GB	Text	1 to 104,857,600 characters
Date	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. Precision 10, scale 0	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Decimal (precision, scale) or Dec (p, s)	Precision 1 to 34	Decimal	Precision 1 to 28, scale 0 to 28
Double	Specifies a single-precision 64-bit floating-point number	Double	Precision 15
Float	Precision 1 to 53	Double	Precision 15
Integer	-2,147,483,648 to 2,147,483,647	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
NClob	Up to 2 GB	Ntext	1 to 104,857,600 characters
Nvarchar	Precision 1 to 5000	Nstring	1 to 104,857,600 characters

SAP HANA Datatype	Range	Transformation Datatype	Range
Real	Specifies a single-precision 32-bit floating-point number	Real	Precision 7, scale 0
Seconddate	0001-01-01 00:00:01 to 9999-12-31 24:00:00	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Shorttext	Specifies a variable-length character string, which supports text search and string search features	Nstring	1 to 104,857,600 characters
Smalldecimal	Precision 1 to 16	Decimal	Precision 1 to 28, scale 0 to 28
Smallint	-32,768 to 32,767	Small Integer	Precision 5, scale 0
Text	Specifies a variable-length character string, which supports text search features	Text	1 to 104,857,600 characters
Time	24-hour time period	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Timestamp	0001-01-01 00:00:00.0000000 to 9999-12-31 23:59:59.9999999	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Tinyint	0 to 255	Small Integer	Precision 5, scale 0
Varchar	Precision 1 to 5000	String	1 to 104,857,600 characters
Varbinary	1 to 5000 bytes	Binary	1 to 104,857,600 bytes

Sybase and Transformation Datatypes

The following table compares Sybase ASE and IQ datatypes and transformation datatypes:

Sybase	Range	Transformation	Range
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Binary (n)	- Sybase ASE: server logical page size - Sybase IQ: 1 to 255 bytes	Binary	1 to 104,857,600 bytes
Bit	0 or 1	String	1 to 104,857,600 characters
Char (n)	ASE server logical page size	String	1 to 104,857,600 characters

Sybase	Range	Transformation	Range
Date	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. Precision 10, scale 0	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Datetime	Jan 1, 1753 A.D. to Dec 31, 9999 A.D. Precision 23, scale 3 (precision to 1/300 second)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Decimal (P,S)	Precision 1 to 38, scale 0 to 38	Decimal	Precision 1 to 28, scale 0 to 28
Float	Machine dependent	Double	Precision 15
Image	1 to 2,147,483,647 bytes	Binary	1 to 104,857,600 bytes
Int	-2,147,483,648 to 2,147,483,647	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Money	-922,337,203,685,477.5808 to 922,337,203,685,477.5807	Decimal	Precision 1 to 28, scale 0 to 28
Nchar (n)	ASE server logical page size	Nstring	1 to 104,857,600 characters
Numeric (P,S)	Precision 1 to 38, scale 0 to 38	Decimal	Precision 1 to 28, scale 0 to 28
Nvarchar (n)	ASE server logical page size	Nstring	1 to 104,857,600 characters
Real	Machine dependent	Real	Precision 7, scale 0
Smalldatetime	Jan 1, 1900, to June 6, 2079 Precision 19, scale 0 (precision to the minute)	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Smallint	-32,768 to 32,767	Small Integer	Precision 5, scale 0
Smallmoney	-214,748.3648 to 214,748.3647	Decimal	Precision 1 to 28, scale 0 to 28
Text	1 to 2,147,483,647 characters	Text	1 to 104,857,600 characters
Time	12:00:00AM to 11:59:59:999PM Precision 8; scale 0	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Timestamp	8 bytes	Binary	1 to 104,857,600 bytes
Timestamp (Sybase IQ)	0001-01-01 00:00:00.000000 to 9999-12-31 23:59:59.999999 Precision 26, scale 6	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Tinyint	0 to 255	Small Integer	Precision 5, scale 0

Sybase	Range	Transformation	Range
Unichar	ASE server logical page size	Nstring	1 to 104,857,600 characters
Univarchar	ASE server logical page size	Nstring	1 to 104,857,600 characters
Varbinary (n)	- Sybase ASE: server logical page size - Sybase IQ: 1 to 32,767 bytes	Binary	1 to 104,857,600 bytes
Varchar (n)	ASE server logical page size	String	1 to 104,857,600 characters

The following datatypes are not supported by Sybase IQ:

- Image
- Nchar (n)
- Nvarchar (n)
- Text
- Unichar
- Univarchar

Datatype Synonyms

The following table compares Sybase synonyms to Transformation datatypes:

Synonym	Transformation
Char Varying	String
Character(L)	String
Character Varying	String
Dec	Decimal
Integer	Small Integer
National Char	Nstring
National Char Varying	Nstring
National Character	Nstring
National Character Varying	Nstring
Nchar Varying	Nstring

Binary and Varbinary Datatypes for Sybase IQ

Sybase IQ supports the Binary and Varbinary datatypes when the ASE_Binary_Display database option is set to OFF in Sybase IQ. By default, the option is ON. For more information, see the Sybase documentation.

Teradata and Transformation Datatypes

The following table compares Teradata datatypes and transformation datatypes:

Teradata	Range	Transformation	Range
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Byte	1 to 64,000 bytes	Binary	1 to 104,857,600 bytes
Byteint	-128 to 127	Small Integer	Precision 5, scale 0
Char	1 to 64,000 bytes	String	1 to 104,857,600 characters
Date	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. Precision 19, scale 0	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Decimal	Precision 1 to 18, scale 0 to 18	Decimal	Precision 1 to 28, scale 0 to 28
Float	-2.226E+308 to 1.797E+308	Double	Precision 15
Integer	-2,147,483,648 to 2,147,483,647	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Smallint	-32768 to 32768	Small Integer	Precision 5, scale 0
Time	00:00:00.000000 to 23:59:61.999999 Precision 8, scale 0	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Timestamp	1 to 19 characters Precision 19 to 26, scale 0 to 6	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Varbyte	1 to 64,000 bytes	Binary	1 to 104,857,600 bytes
Varchar	1 to 64,000 bytes	String	1 to 104,857,600 characters

Datatype Synonyms

The following table compares Teradata synonyms to transformation datatypes:

Synonym	Transformation
Double Precision	Double
Numeric	Decimal
Real	Double

ODBC and Transformation Datatypes

The following table compares ODBC datatypes, such as Microsoft Access or Excel, to transformation datatypes:

Datatype	Transformation	Range
Bigint	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Binary	Binary	1 to 104,857,600 bytes
Bit	String	1 to 104,857,600 characters
Char(L)	String	1 to 104,857,600 characters
Date	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Decimal(P, S)	Decimal	Precision 1 to 28, scale 0 to 28
Double	Double	Precision 15
Float	Double	Precision 15
Integer	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Long Varbinary	Binary	1 to 104,857,600 bytes
Nchar	Nstring	1 to 104,857,600 characters
Nvarchar	Nstring	1 to 104,857,600 characters
Ntext	Ntext	1 to 104,857,600 characters
Numeric	Decimal	Precision 1 to 28, scale 0 to 28
Real	Real	Precision 7, scale 0
Smallint	Smallint	Precision 5, scale 0
Text	Text	1 to 104,857,600 characters
Time	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Timestamp	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Tinyint	Small Integer	Precision 5, scale 0

Datatype	Transformation	Range
Varbinary	Binary	1 to 104,857,600 bytes
Varchar(L)	String	1 to 104,857,600 characters

Note: When the Integration Service runs in Unicode data movement mode, the column precision that you specify for ODBC datatypes determines the number of characters the Integration Service reads and writes.

COBOL and Transformation Datatypes

The following table compares COBOL datatypes to transformation datatypes:

COBOL	Transformation	Range
Nstring	String	1 to 104,857,600 characters
Number	Decimal	Precision 1 to 28; scale 0 to 28
String	String	1 to 104,857,600 characters

Flat File and Transformation Datatypes

The following table compares flat file datatypes to transformation datatypes:

Flat File	Transformation	Range
Bigint	Bigint	Precision of 19 digits, scale of 0
Datetime	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Double	Double	Precision of 15 digits
Integer	Integer	-2,147,483,648 to 2,147,483,647
Nstring	Nstring	1 to 104,857,600 characters
Number	Decimal	Precision 1 to 28, scale 0 to 28
String	String	1 to 104,857,600 characters

When the Integration Service reads non-numeric data in a numeric column from a flat file, it drops the row and writes a message in the session log. Also, when the Integration Service reads non-datetime data in a datetime column from a flat file, it drops the row and writes a message in the session log.

Number Datatype

For flat files, the Integration Service reads data with a Number datatype as it appears in the file. If you want to specify decimal precision and scale, use an Expression transformation to convert number ports to decimal ports. Also enable high precision in the session.

For example, if the flat file contains a 5-digit numeric field called Price, and you want to output the Price as a number with two decimal places, you would set up the following expression:

```
PRICE / 100
```

This expression moves the decimal place two digits to the left as shown in the following example:

Number	Return Value
34500	345.00
12340	123.40
23450	234.50

XML and Transformation Datatypes

PowerCenter supports all XML datatypes specified in the W3C May 2, 2001 Recommendation. For more information about XML datatypes, see the W3C specifications for XML datatypes at http://www.w3.org/TR/xmlschema-2.

When you pass data to the target, make sure that it is in the correct format so that the Integration Service writes the data correctly in the target XML file.

You can change XML datatypes in the source and target definitions. You can change midstream XML datatypes in the Transformation Developer. You cannot change XML datatypes when you import them from an XML schema, and you cannot change the transformation datatypes for XML sources within a mapping.

Converting Data

You can convert data from one datatype to another by using the following methods:

- Passing data between ports with different datatypes (port-to-port conversion).
- Using transformation functions to convert data.
- Using transformation arithmetic operators to convert data.

Port-to-Port Data Conversion

The Integration Service converts data based on the datatype of the port. Each time data passes through a port, the Integration Service looks at the datatype assigned to the port and converts the data if necessary.

When you pass data between ports of the same numeric datatype and the data is transferred between transformations, the Integration Service does not convert the data to the scale and precision of the port that the data is passed to. For example, you transfer data between two transformations in a mapping. If you pass data from a decimal port with a precision of 5 to a decimal port with a precision of 4, the Integration Service stores the value internally and does not truncate the data.

You can convert data by passing data between ports with different datatypes. For example, you can convert a string to a number by passing it to an Integer port.

The Integration Service performs port-to-port conversions between transformations and between the last transformation in a dataflow and a target.

The following table describes the port-to-port conversions the Integration Service performs:

Datatype	Bigint	Integer, Small Integer	Decimal	Double, Real	String, Text	Nstring, Ntext	Date/ Time	Binary
Bigint	No	Yes	Yes	Yes	Yes	Yes	No	No
Integer, Small Integer	Yes	No	Yes	Yes	Yes	Yes	No	No
Decimal	Yes	Yes	No	Yes	Yes	Yes	No	No
Double, Real	Yes	Yes	Yes	No	Yes	Yes	No	No
String, Text	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Nstring, Ntext	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Date/Time	No	No	No	No	Yes	Yes	Yes	No
Binary	No	No	No	No	No	No	No	Yes

Converting Strings to Dates

You can convert strings to datetime values by passing the strings to a Date/Time port. However, the strings must be stored in the date format defined by the DateTime Format String session property.

Converting Strings to Numbers

The Integration Service converts strings to numbers based on the ASCII numeric format.

The following table provides examples of how the Integration Service converts strings to numbers when you pass strings from a string port to a numeric port:

String	Converts to Decimal	Double, Real	Bigint, Integer, Small Integer
‘18e-3’	18	0.018	18
‘18-e-3abc’	18	0.018	18
‘123abc’	123	123	123
‘A123cb’	0	0	0
‘abc’	0	0	0

APPENDIX B

Configure the Web Browser

This appendix includes the following topic:

- Configure the Web Browser, 269

Configure the Web Browser

Some PowerCenter Client browser functions require you to configure the following options in Internet Explorer:

Adobe Flash Player Plug-In

Metadata Manager requires the Adobe Flash Player plug-in version 9 or later to upload PowerCenter parameter files and to display data lineage. To upload PowerCenter parameter files or to run data lineage analysis in Metadata Manager or from the Designer, download and install the Flash Player plug-in on the web browser. Get the Flash Player plug-in from the following web site: www.adobe.com/products/flashplayer

Scripting and ActiveX

Enable the following controls on Microsoft Internet Explorer:

- Active scripting
- Allow programmatic clipboard access
- Run ActiveX controls and plug-ins
- Script ActiveX controls marked safe for scripting

To configure the controls, click **Tools** > **Internet options** > **Security** > **Custom level**.

Note: Configure the scripting and ActiveX controls to display the Start Page for the PowerCenter Client.

INDEX

A

active transformations
  connecting 115
  mapplets 142
adding
  repositories 33
  sources to mappings 123
  target update statements 131
aggregate
  definition (metadata) 202
Application Source Qualifier transformation
  description 123
arguments
  for user-defined functions 163
attributes
  propagated 120
autolink
  by name 117
  by position 116

B

bigint
  constants in expressions 251
  high precision handling 250
  using in calculations 250
  writing to flat files 252
binary datatypes
  overview 252
blocking transformations
  data flow validation 134
breakpoints
  .dcf files 189
  copying 189
  creating 168, 170
  data conditions 172, 173
  Debugger 168
  error conditions 172
  global 171
  global data conditions 175
  instance name 171
  ISDEFAULT 175
  ISNULL 175
  steps to enter 175
  transformation 171
  transformation data conditions 174
business components
  adding shortcuts 199
  copying 200
  creating 199
  definition 197
  deleting 199
  directories 197
  linking to documentation 198
business components *(continued)*
  local vs. global shortcuts 199
  locking 198
  moving 198
  overview 197
business components directory
  copying 200
  creating 199
  deleting 199
  editing 198
business documentation
  adding links to mappings 53, 111
  creating links 45
  for expressions 44
  for repository objects 44
  root 44
  viewing 45
business names
  adding to sources 43
  adding to target definitions 72, 101
  adding to targets 43
  as source column names 44
  as target column names 44
  display sources 19
  display targets 19
  displaying in Navigator 43
  in Source Qualifier 44
  renaming imported sources 50
  using 43

C

character loss
  code pages 87
  flat files 87
character sets
  Flat File Wizard 66
checking in
  versioned objects 35
checking out
  versioned objects 35
COBOL
  comparing to transformation datatypes 266
  copybook 57
  creating normalized targets 94
  tab handling 57
COBOL file properties
  configuring Advanced properties 60
  configuring Columns tab 60
  configuring Table tab 59
COBOL source definitions
  code pages 56
  components 58
  configuring properties 59
  copybooks 57

COBOL source definitions *(continued)*
FD section 58, 60
field attributes 60
fields 58
importing 56
OCCURS statement 58, 60
PICTURE clauses 60
REDEFINES statement 58
table options 59
word storage 59
COBOL sources
tab handling 57
code pages
character loss 87
COBOL sources 56
delimited source 76
fixed-width source 73
flat files 87
mapping parameters 152
mapping variables 152
relational targets 89
color themes
applying 23
colors
configuring 22
format options 22
column description
retaining when re-importing a source 54
columns
adding to dimensions 203
adding to targets 73
delimited file settings 69
relational targets 102
comments
adding to mappings 111
adding to source definitions 53
COMP columns
COBOL word storage 59
comparing
mapping and mapplet dependencies 195
mapping and mapplet links 195
mapping parameters and variables 195
mappings and mapplets 195
mapplet instances 195
metadata extensions 195
reusable transformations 194
source instances 195
sources 194
target instances 195
target load orders 195
targets 194
transformation instances 195
configuring
breakpoints 170
custom tools 28
data lineage 191
Debugger 176
mapplet ports 145
connecting
mapplet output groups 145
mapplets to mappings 145
mapplets with SQL override 145
multiple transformations 114
objects in mappings 114
ports 115
rules for connecting objects 115
sources to targets, overview 114
transformations 115
connections
validating 115, 133
Constraint Based Load Ordering
debug session parameter 178
constraints
adding to target definitions 101
converting
datatypes 266
strings to dates 268
strings to numbers 266, 268
copy as
mappings 110
copying
business components 200
business components directory 200
mappings 110
mapplets 143
user-defined functions 167
creating
breakpoints 170
business components 199
cubes 204
dimensions 203
directory for business components 199
flat file source definitions 73
flat file target definitions 73
keys for targets 101
mappings 109
mapplet ports 145
mapplets 142
target definitions 88, 99
target tables 244
Type 1 Dimension mapping 216, 219
Type 2 Dimension/Effective Date Range mapping 232, 237
Type 2 Dimension/Flag Current mapping 226, 231
Type 2 Dimension/Version Data mapping 220, 224
Type 3 Dimension mapping 238, 242
user-defined functions 165
Cube Wizard
overview 204
cubes
adding fact tables 204
creating 204
database type 204
definition (metadata) 202
deleting 205
editing 205
opening and closing 206
overview 201
tips 206
viewing metadata 206
wizard for creating 204
current value
mapping variables 156
CurrentlyProcessedFileName port
adding to file sources 85
custom tools
configuring 28

D

data
converting port-to-port 266
converting strings to numbers 266
converting types of 266
handling truncated and overflow 130
previewing 39

data breakpoint conditions
global 175
overview 173
transformations 174
data breakpoints
global options 173
transformation options 173
data flow validation
overview 134
data lineage
configuring 191
description 191
displaying 192
data movement mode
datatypes 248
data preview
relational targets 39
data types
decimal 252
double 252
transformation 249
database type
for cubes 204
for dimensions 203
DataDirect ODBC drivers
platform-specific drivers required 51, 91
datatypes
Bigint 250
binary 252
COBOL 266
conversion 266
Date/Time 252
flat file 266
for user-defined function arguments 163
IBM DB2 254
Informix 255
Integer 250
mapping variables 156
Microsoft SQL Server 257
native datatypes 248
ODBC 265
Oracle 259
overview 248
reading flat file sources 266
Small Integer 250
string 254
Sybase ASE 261
Teradata 264
transformation 248
transformation datatypes in targets 97
XML 267
Date/Time datatypes
overview 252
dates
display format 34
debug indicators
monitoring 182
debug log
monitoring 185
sample 185
debug session
session parameters 177
Debugger
ANY-PORT 173
configuring 176
continuing 184
copying configuration 189
creating breakpoints 168, 170
Debugger *(continued)*
creating debug session 177
data breakpoints 172
data conditions 173
debug indicators 182
Debugger Wizard 176
Designer behavior 181
error breakpoints 172
evaluating expressions 188
initializing state 179
Instance Data window 18
Instance window 182
modifying data 187
monitoring target data 184
next instance 184
options 24
overview 168
paused state 179
persisted values 181
restrictions for modifying data 187
ROW-ID 173
ROW-TYPE 173
running 179
running in debug mode 177
running state 179
session type 177
states 179
Target Data window 18
target options 178
Target window 182
tasks 180
toolbar 26
Debugger tab
monitoring debug log 185
Output window 182
Debugger Wizard
configuring 176
debugging
mappings 112, 168
mapplets 178
decimal
high precision handling 250, 252
decimal data types
overview 252
default options
Designer, customizing 19
default values
mapping parameters 151
mapping variables 151
defining
mapping parameters 155
deleting
business components 199
business components directory 199
cubes 205
dimensions 205
mappings 112
mapplets 143
user-defined functions 166
delimited flat files
advanced settings 76
column settings 69
importing source definitions 76
row settings 76
rules 78
dependencies
implicit 118, 119
link path 118

dependencies *(continued)*
propagated 120
viewing 113
deployment groups
user-defined functions, adding 167
description fields
maximum precision 72, 101
Designer
adding mappings 109
behavior during Debugger sessions 181
business documentation 44
checking out and in versioned objects 35
configuring data lineage 191
copying objects 36
creating links to business documentation 45
creating shortcuts 34
creating toolbars 28
customizing options 19
customizing toolbars 28
date display format 34
displaying data lineage 192
exporting objects 37
importing objects 37
Instance Data window 18
Mapping Designer 17
Mapplet Designer 17
Navigator 18
opening and closing a folder 34
output window 18
overview 17
port validation 115
printing mappings 34
searching 29
searching for versioned objects 35
shortcut keys 38
Source Analyzer 17
SQL DDL commands 104
status bar 18
Target Data window 18
Target Designer 17
tasks 33
time display format 34
toolbars 26
tools 17
viewing reports 46
windows 18
working with business components 197
workspace window 18
zooming 32
dimension tables
levels 203
metadata 201
non-normalized models 201
reasons for 208
sources 203
dimensions
creating 203
database type for 203
definition (metadata) 202
deleting 205
editing 205
overview 201
slowly changing 208
slowly growing 208
tips 206
viewing metadata 206
directories
grouping business components 197
displaying
data lineage 192
date 34
icons 31
time 34
toolbars 27
zooming 32
documentation
path 44
domains
metadata extensions 41
double
high precision handling 252
double data types
overview 252
drilling
definition (metadata) 202
DTM Buffer Size
debug session parameter 178

E

editing
business components directory 198
cubes 205
dimensions 205
mappings 111
mapplets 143
relational source definitions 53
user-defined functions 166
Enable High Precision
debug session parameter 178
entering
repository object descriptions 35
error handling
misalignment, flat file 67
errors
port 115
validated during mappings 133
Excel
datatypes 265
exporting
mappings 111
mapplets 143
objects 37
user-defined functions 166
Expression Editor
mapping parameters 154
mapping variables 160
public and private functions 164
using with user-defined functions 167
expression strings
defining in parameter files 161
Expression transformation
example with Transaction Control transformation 128
expression variables
mapping parameters and variables 161
expressions
creating with user-defined functions 164, 167
validating 134

F

fact tables
adding to cubes 204
definition 208

fact tables *(continued)*
definition (metadata) 202
with metadata 201
FD Section
COBOL source definitions 58
field attributes
COBOL source definitions 60
field width
description 79
fields
COBOL source definitions 58
deleting 37
moving 37
file list
CurrentlyProcessedFileName port 85
returning source file names with data rows 85
FileName column
flat file example 128
flat file targets 73
generating flat file targets 127
running a session with flat file targets 128
Find in Workspace tool
overview 30
Find Next tool
overview 29
fixed-width files
advanced settings 73
importing sources 73
flat file definitions
generating flat files by transaction 127
Flat File Wizard
delimited file column settings 69
delimited file options 69
flat file column settings 67
flat file options 67
importing source definitions 65
flat files
code page 66
code pages 87
column settings 67
comparing to transformation datatypes 266
consecutive shift characters 86
creating by transaction 127
datatypes, reading 266
delimited 78
editing source definition properties 71
editing target definition properties 71
formatting columns 79
importing delimited 76
importing fixed-width 73
importing, overview 65
multibyte data 87
precision, targets 87
rules for delimited 78
shift-sensitive 85
shift-sensitive flat file requirements 86
special character handling 66
supported character sets 66
troubleshooting 87
updating ASCII source definitions 71
wizard for importing 65
folders
copy as 110
opening and closing 34
fonts
configuring 22
format options
configuring in the Designer 22
full screen
view 33
function type
description 164
functions
in mapping variables 158
user-defined functions 163

G

general options
Designer, configuring 19
Getting Started Wizard
creating targets 244
description 210
pass-through mapping 210, 212
scheduling sessions 244
slowly growing target mapping 210, 213
sources 211
global repositories
business components 199
globalization
sources 47

H

hierarchies
adding to dimensions 204
definition (metadata) 202
schema example 201
high precision
Bigint data type 250
Decimal data type 250
historical data
differentiating from current 208
maintaining 208

I

IBM DB2
comparing to transformation datatypes 255
icons
displaying 31
implicit dependencies
propagating 118, 119
importing
COBOL sources 56
delimited flat file source definitions 76
fixed-width flat file source definitions 73
flat files 65
mappings 111
mapplets 143
Microsoft Excel source definitions 62
objects 37
relational source definitions 50
relational target definitions 91
user-defined functions 166
indexes
creating for target tables 103
defining on target 103
dropping for target tables 105
importing for target definitions 91
re-creating for target tables 105
Informix
comparing to transformation datatypes 255

initial values
datetime format 153, 158
mapping parameters 151
mapping variables 151
input ports
connecting mapplet 145
defining in mapplets 140
overview 114
Input transformation
configuring 145
in mapplets 140, 145
ports 145
input/output ports
overview 114
Instance Data
window 18
Instance window
Debugger 182
monitoring data 183
integers
constants in expressions 251
converting from strings 251
using in calculations 250
writing to flat files 252
Integration Service
datatype usage 248
invalidating
sessions 112
ISDEFAULT
breakpoint conditions 175
IsExprVar property
mapping parameters 153, 161
mapping variables 158, 161
ISNULL function
breakpoint conditions 175

K

keys
creating for targets 101
key relationships and re-importing source 54
keywords
setting for relational targets 101

L

Layout toolbar
overview 26
levels
adding to dimensions 203
adding to hierarchy 204
definition (metadata) 202
dimension tables 203
link paths
propagating dependencies 118
viewing 112
linking
autolinking by name 117
autolinking by position 116
business component documentation 198
locking
business components 198

M

maintaining
historical data 208
star schemas 208
Mapping Architect for Visio
mapping templates 108
Mapping Composite Report
viewing 46
Mapping Designer
creating mappings 109
usage for 17
mapping parameters
code pages 152
comparing 195
creating 153
datetime format 153
default values 151
defining 155
definition 153
expanding in expressions 161
initial values 151
IsExprVar property 153, 161
naming convention 153
overview 150
tips for using 161
using in expressions 154
mapping variables
aggregation type 156
clearing values 160
code pages 152
comparing 195
creating 158
current value 156
datatype 156
datetime format 158
Debugger 181, 188
default values 151
definition 155
expanding in expressions 161
functions 158
initial values 151
IsExprVar property 158, 161
mapplets 158
naming convention 158
overriding values 160
overview 150
start value 156
tips for using 161
using in expressions 160
values 155
mapping wizards
creating targets 244
Getting Started 210
overview 210
scheduling sessions for 244
Slowly Changing Dimensions Wizard 210
sources 211
mappings
adding comments 111
adding mapplets 144
adding sources 123
adding targets 88
comparing 195
comparing dependencies 195
comparing links 195
connecting flat file FileName port 128
connecting objects 114

mappings *(continued)*
copy as 110
copying 110
creating 109
debugging 112
debugging overview 168
deleting 112
editing 111
editing for relational database changes 49, 53
exporting 111
importing 111
invalidation causes 49
metadata extensions in 41
opening 109
overriding source table name 124
overriding target table name 133
pass-through mapping 210, 212
printing 34
printing differences between 195
process 108
renaming 111
reverting to previous versions 111
slowly growing target 210
slowly growing target mapping 213
star schemas 208
tasks 109
toolbar 26
Type 1 Dimension mapping 216
Type 2 Dimension/Effective Date Range mapping 210, 232
Type 2 Dimension/Flag Current mapping 210, 226
Type 2 Dimension/Version Data mapping 210, 220
Type 3 Dimension mapping 210, 238
updating targets 99
validating 133
validating multiple 135
viewing reports 46
working with mapplets 125
working with targets 125
working with transformations 125
Mapplet Composite Report
viewing 46
Mapplet Designer
usage for 17, 139
mapplet ports
adding to Input/Output transformations 145
configuring 145
creating 145
mapplets
active and passive 142
adding a description 143
comparing 195
comparing dependencies 195
comparing instances 195
comparing links 195
components 140
configuring 142
connecting input ports 145
connecting with SQL override 145
copying 143
creating 142
creating targets from 97
creating transformation logic 142
definition 139
deleting 143
editing 143
exporting 143
illustration of 140
importing 143
mapplets *(continued)*
in mappings 125
input ports 140
input source data 140
mapping variables 158
metadata extensions in 41
monitoring data 183
output groups 140, 145
output overview 140
output ports 140
overview 139
pipeline partitioning 146
port attributes 145
ports 145
printing differences between 195
repository objects supported 140
rules for validation 143
selecting to debug 178
session behavior 139
set tracing level 143
sources within mapplet 140
target data 140
tips for creating 147
toolbar 26
unsupported objects 146
using in mappings 144
validating 142
viewing reports 46
measure
definition (metadata) 202
metadata
aggregate 202
cubes 202
dimension 202
drilling 202
fact table 202
hierarchies 202
levels 202
measure 202
multi-dimensional 201
normalization 202
redundancy 202
snowflake schema 202
star schema 202
viewing cubes and dimensions 206
metadata extensions
comparing across mappings 195
creating 41
deleting 43
domains 41
editing 42
overview 41
Metadata Manager
configuring data lineage 191
displaying data lineage 191
Microsoft Access
datatypes 265
Microsoft Excel
datatypes 265
formatting numeric data 62
importing source definitions 63
Microsoft SQL Server
comparing to transformation datatypes 257
datatypes 257
misalignment
flat file 67
modifying data
Debugger 187

monitoring
debug indicators 182
debug log 185
Debugger 182, 186
target data 184
multi-dimensional metadata
overview 201
MX (Metadata Exchange)
saving data 19
MX views
performance 19

N

naming conventions
for mappings 109
NaN
described 251
native datatypes
overview 248
Navigator
deleting mappings from 112
usage for 18
Next Instance
Debugger 184
non-normalized
cube attribute 204
dimension tables 201
normalization
definition (metadata) 202
normalized
cube attribute 204
multi-dimensional schema 201
Normalizer transformation
creating a target from 95
debugging 187
description 123
null characters
fixed-width source 73
flat file 75
Number(P,S) values
Oracle 259
numeric operations
converting strings to dates 268
converting strings to numbers 268

O

object validation
overview 134
objects
comparing 35
connecting in mappings 114
copying 36
displaying data lineage 192
viewing older versions 35
OCCURS statement
COBOL source definitions 58
ODBC (Open Database Connectivity)
comparing datatypes to transformation 265
DataDirect driver issues 51, 91
importing source definitions 51
importing target definitions 91
precision in unicode mode 265
OLAP applications
metadata 201
older versions of objects
viewing 35
opening
mappings 109
options
color themes, selecting 23
debug 24
web services 24
options (Designer)
format 22
general 19
tables 21
Oracle
comparing to transformation datatypes 259
Number(P,S) values 259
output groups
connecting to output ports in mapplets 145
definition 140
ports 145
output ports
mapplets 140
overview 114
Output transformation
configuring 145
in mapplets 140, 145
ports 145
Output window
Debugger tab 182
example of mapping validation 135
Session Log tab 182
overflow data
handling 130
overriding
mapping variable values 160

P

parameter files
expression strings, defining 161
specifying location for debug session 178
parameters
mappings 150
pass-through mapping
creating 212
customizing 213
description 210
overview 212
repository objects created 212
session options 213
passive transformations
connecting 115
mapplets 142
persisted values
Debugger 181
pipeline partitioning
mapplets 146
port description
retaining when re-importing a source 54
port-to-port data conversion
overview 267
ports
adding to Input/Output transformations 145
connecting 115
connecting to mapplet input 145
connecting to multiple transformations 114
creating in mapplets 145
deleting 37

ports *(continued)*
errors 115
linking 115
linking by name 117
linking by position 116
propagating attributes 118
renaming 38
source dependencies 113
PowerCenter repository
viewing data lineage analysis 191
PowerCenter Repository Reports
viewing in Designer 46
powrmart.ini
configure for third-party drivers 52, 93
default tab size 57
importing target indexes 91
pre- and post-session SQL
target instances 132
precision
description 73, 79
flat files 87
relational sources 53
relational targets 102
previewing data
target 39
XML data 39
printing
mappings 34
private
configuring for user-defined functions 164
propagate
port attributes 118
public
configuring for user-defined functions 164

Q

QNAN
converting to 1.#QNAN 251

R

Rank transformation
debugging 187
record sets
COBOL source definitions 59
records
updating to include non-key columns 130
REDEFINES statement
COBOL source definitions 58
redundancy
definition (metadata) 202
Reject Truncated/Overflow Rows option
using 130
renaming
source definitions 53
repositories
adding 33
Repository Manager
viewing metadata in 206
repository objects
description 35
editing 38
supported in mapplets 140
Repository toolbar
overview 26
reusable transformations
in mapplets 140
root variable
overview 45
Row type
debug session parameter 178
rules
delimited flat files 78
for target updating 131

S

saving
historical data 208
scale
description 73, 79
relational sources 53
relational targets 102
scheduling
sessions for mapping wizards 244
searching
Designer 29
versioned objects in the Designer 35
Sequence Generator transformation
debugging 187
Session Log tab
Output window 182
session reject file
using for overflow data 130
using for truncated data 130
sessions
configuring Type 1 Dimension 220
configuring Type 2 Dimension/Effective Date Range 238
configuring Type 2 Dimension/Flag Current 232
configuring Type 2 Dimension/Version 225
configuring Type 3 Dimension 243
invalidating 112
pass through, configuring 213
running in debug mode 177
scheduling for star schemas 244
slowly growing target, configuring 216
updates from source definitions 53
shift-sensitive
consecutive shift characters 86
flat files 85
requirements 86
shortcut keys
keyboard 38
shortcuts
as business components 199
creating 34
in mapplets 140
Simple Pass Through mapping
description 210
Slowly Changing Dimensions Wizard
creating targets for 244
description 210
scheduling sessions 244
sources 211
Type 1 Dimension mapping 216
Type 2 Dimension/Effective Date Range mapping 210, 232
Type 2 Dimension/Flag Current mapping 210, 226
Type 2 Dimension/Version Data mapping 210, 220
Type 3 Dimension mapping 210, 238
slowly growing target mapping
creating 213
description 210

slowly growing target mapping *(continued)*
keys, handling 214
overview 213
repository objects created 214
session options 216
small integers
constants in expressions 251
using in calculations 250
writing to flat files 252
snowflake schema
definition 201
Source Analyzer
editing source definitions 53
importing relational source definitions 52
usage for 17
source column dependencies
viewing 113
$Source connection value
debug session parameter 178
source definitions
adding to mappings 123
changing database type 53
COBOL 59
connecting ports 114
creating from target definitions 55
editing COBOL 59
editing fixed-width 71
editing relational 53
for mapping wizards 211
importing delimited flat file 76
importing Microsoft Excel 62
importing relational 50
in mapplets 140
overview 47, 65
re-importing 54
renaming 53
special character handling 48
updating 53
updating ASCII 71
updating relational 53
using as base for target definitions 93
source description
retaining when re-importing a source 54
Source Qualifier transformation
business names 44
description 123
option to create automatically 21
option to create manually 21
using string parameters 152
using variables 152
source tables
overriding table name 124
sources
adding to dimensions 203
adding to mappings 123
comparing 194
connecting targets 114
flat file code page 73, 76
for mapping wizards 211
globalization features 47
metadata extensions in 41
null character 75
overriding source table name 124
targets connected 113
troubleshooting 64
within mapplets 140
special character handling
flat file definitions 66
special character handling *(continued)*
source table and field names 48
target table and field names 89
SQL
generating and executing to create targets 104
SQL Editor
using to modify UPDATE statement 130
Standard toolbar
overview 26
star schema
definition 201
maintaining 208
start value
mapping variables 156
status bar
definition 18
Step to Instance
Debugger 184
string datatypes
overview 254
strings
converting to dates 268
converting to numbers 251, 268
Sybase ASE
comparing to transformation datatypes 261
Sybase IQ
comparing to transformation datatypes 261
synonym datatypes
comparisons 258
Teradata 264
syntax
configuring for user-defined functions 163

T

table name prefix
target owner 132
table names
overriding source table name 124
overriding target table name 133
table owner name
targets 132
tables
Designer options, configuring 21
$Target connection value
debug session parameter 178
Target Data window
description 18
target definitions
adding columns 73, 102
adding to mappings 123
comparing datatypes with transformations 97
connecting ports 114
creating from a mapplet 97
creating from a Normalizer transformation 95
creating from a transformation with multiple groups 95
creating from source definitions 93
creating from transformations 95, 98, 99
creating indexes 103
creating keys 101
creating, overview 88
editing relational definitions 99
generating target tables from 104
importing indexes 91
importing relational targets 91
overview, relational 88
previewing data 39

target definitions *(continued)*
renaming 101
setting options 99
special character handling 89
steps to create from transformations 98
troubleshooting 105
updating, relational 101
using as base for source definitions 55
Target Designer
creating cubes and dimensions in 201
usage for 17
using 88
target instance
pre- and post-session SQL 132
target load order
comparing across mappings 195
setting 126
target load order groups
description 126
target owner
table name prefix 132
target SQL override
overview 130
target tables
creating 104
overriding table name 133
previewing data 39
re-creating 105
target update
adding statement 131
overview 130
Target window
Debugger 182
monitoring 184
targets
adding to mappings 88
code pages 89
comparing 194
connecting multiple transformations to 114
database types 88
editing cubes 205
editing dimensions 205
generating flat files by transaction 127
in mappings 125
mapping wizards 244
mapplet data 140
metadata extensions in 41
overriding target table name 133
overriding update to include non-key columns 130
overview, relational 88
previewing data 39
rules for updating 131
table name prefix, owner name 132
target SQL override 130
target update override 130
viewing source dependencies 113
Teradata
comparing datatypes to transformation 264
datatypes 264
time
display format 34
tips
creating mapplets 147
cubes and dimensions 206
toolbars
creating 28
customizing 28
description 26
toolbars *(continued)*
Designer 26
displaying 27
tools
configuring custom 28
debug options 24
toolbar 26
web service options 24
tracing levels
in mapplets 143
Transaction Control transformation
example with an Expression transformation 128
transformation data
monitoring 183
transformation data types
list of 249
transformation datatypes
comparing to COBOL 266
comparing to flat file 266
comparing to IBM DB2 255
comparing to Informix 255
comparing to ODBC 265
comparing to Oracle 259
comparing to Sybase ASE 261
comparing to Sybase IQ 261
comparing to Teradata 264
Microsoft SQL Server 257
overview 248
transformations
comparing 194
connecting 115
creating 115
creating targets from 95
in mappings 125
in mapplets 140
Input transformation 145
metadata extensions in 41
Output transformation 145
pass-through mapping 212
reusing in mapplets 139
slowly growing target mapping 214
toolbar 26
Type 1 Dimension mapping 218
Type 2 Dimension/Effective Date Range mapping 235
Type 2 Dimension/Flag Current mapping 229
Type 2 Dimension/Version Data mapping 223
Type 3 Dimension mapping 241
troubleshooting
flat files 87
sources 64
target definitions 105
truncated data
handling 130
Type 1 Dimension mapping
creating 216, 219
description 210
handling keys 216
overview 216
repository objects created 216
session options 220
transformations 218
Type 2 Dimension/Effective Date Range mapping
creating 232, 237
description 210
handling keys 233
overview 232
repository objects created 233
session options 238

Type 2 Dimension/Effective Date Range mapping *(continued)*
transformations 235
Type 2 Dimension/Flag Current mapping
creating 226, 231
description 210
handling keys 226
overview 226
repository objects created 227
session options 232
transformations 229
Type 2 Dimension/Version Data mapping
creating 220, 224
customizing 225
description 210
handling keys 221
numbering versions 221
repository objects created 222
session options 225
transformations 223
Type 3 Dimension mapping
creating 238, 242
description 210
handling keys 239
overview 238
repository objects created 239
session options 243
transformations 241

U

Unicode mode
ODBC precision 265
updating
flat file source definitions 71
modifying SQL statement 130
relational source definitions 53
relational targets 101
rules for target updates 131
user-defined functions
configuring arguments 163
configuring function syntax 163
copying 167
creating 165
deleting 166
editing 166
exporting 166
importing 166
in the Expression Editor 167
nesting 164
overview 163
validating 167

V

validating
mappings 133
mapplets 142
multiple mappings 135
user-defined functions 167
values
mapping variables 155
variables
expanding in expressions 161
mappings 150
versioned objects
Allow Delete without Checkout option 25
versioned objects *(continued)*
checking in 35
checking out 35
comparing older versions 35
searching for in the Designer 35
viewing in workspace 35
viewing multiple versions 35
versioning
toolbar 26
versions
reverting to previous mappings 111
View toolbar
overview 26
viewing
business documentation 45
full screen 33
link paths 112
metadata for cubes and dimensions 206
older versions of objects 35
source column dependencies 113

W

web services
backward compatibility 24
options 24
wizards
Cube Wizard 204
Flat File Wizard 65
Getting Started Wizard 210
overview of mapping types 210
Slowly Changing Dimensions Wizard 210
word storage
COBOL 59
Workflow Generation Wizard
overview 136
rules and guidelines 136
Workflow Manager
adding repositories 33
Workflow Monitor
monitoring Debugger 186
workspace
full screen view 33
iconizing 31
navigating 29
printing 34
restoring 31
usage for 18
zooming 32
workspace objects
iconizing 31
restoring 31
WSDL
creating for generated mappings 24

X

XML
datatypes 267
previewing data 39
XML Editor
previewing data 39
XML Source Qualifier transformation
description 123

Z

zoom
display format 32