Informatica PowerCenter Data Validation Option
(Version 9.1.0)

Installation and User Guide
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/cdd1.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) and the MIT License (http://www.opensource.org/licenses/mit-license.php).

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 Software is protected by U.S. Patent Numbers 5,794,246; 6,014,670; 6,029,178; 6,032,158; 6,035,307; 6,044,374; 6,092,086; 6,208,990; 6,339,775; 6,640,226; 6,789,096; 6,820,077; 6,823,373; 6,850,947; 6,895,471; 7,117,216; 7,162,643; 7,254,590; 7,281,001; 7,421,458; 7,496,588; 7,523,121; 7,584,422; 7,720,842; 7,721,270; and 7,774,791, international Patents and other Patents Pending.

DISCLAIMER: Informatica Corporation provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica Corporation 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.

Part Number: PC-DVO-91000-0001
Chapter 4: New Features and Behavior Changes .............................................. 16
New Features and Enhancements. .......................................................... 16
Behavior Changes. ............................................................................. 17

Chapter 5: Data Validation Option Client Layout ...................................... 19
Data Validation Option Client Layout Overview .................................... 19
Tabs Available in Data Validation Option Client Layout. ......................... 20
Tests Tab .......................................................................................... 20
Table Pairs Tab ................................................................................. 20
Single Tables Tab .............................................................................. 21
SQL Views Tab .................................................................................. 22
Lookup Views Tab .............................................................................. 22

Chapter 6: Repositories .......................................................................... 24
Repositories Overview ......................................................................... 24
Repository Properties .......................................................................... 24
Adding Repositories ........................................................................... 25
Editing Repositories ........................................................................... 26
Deleting Repositories ......................................................................... 26
Refreshing Repositories ...................................................................... 26
Exporting Repository Metadata ............................................................ 27

Chapter 7: Table Pairs ........................................................................... 28
Table Pairs Overview ........................................................................... 28
Table Pair Properties .......................................................................... 28
Selecting Tables .................................................................................. 29
Processing Large Tables ..................................................................... 30
Selecting Connections ........................................................................ 30
Overriding Owner Name ..................................................................... 30
Pushing Test Logic to the Database ..................................................... 30
Source Directory and File Name .......................................................... 31
WHERE Clauses .................................................................................. 31
Description ........................................................................................ 32
External ID ......................................................................................... 32
Joining Tables ..................................................................................... 32
Adding Table Pairs ............................................................................. 33
Editing Table Pairs ............................................................................. 33
Deleting Table Pairs ........................................................................... 33
Viewing Overall Test Results ............................................................... 34
Chapter 15: Command Line Integration ................................. 67
Command Line Integration Overview ................................... 67
CopyFolder Command .................................................... 68
CreateUserConfig Command ......................................... 68
ExportMetadata Command ............................................ 69
ImportMetadata Command ........................................... 70
InstallTests Command ................................................ 70
PurgeRuns Command .................................................. 71
RefreshRepository Command ....................................... 71
RunTests Command .................................................... 72
UpgradeRepository Command ........................................ 73

Chapter 16: Troubleshooting ........................................... 74
Troubleshooting Overview ........................................... 74
Troubleshooting Initial Errors ...................................... 74
Troubleshooting Ongoing Errors ................................... 75
Troubleshooting Command Line Errors ............................ 76

Chapter 17: Other Features and Functionality .................... 77
Folders .................................................................. 77
  Copying Folders .................................................... 77
Menus ................................................................ 78
  Console ............................................................ 79
  Settings Folder ................................................... 79
  Preferences Dialog Box ......................................... 79
Uninstalling Data Validation Option ................................ 79

Appendix A: Datatype Reference ..................................... 80
Test, Operator, and Datatypes Matrix for Table Pair Tests ...... 80
Test, Operator, and Datatypes Matrix for Single-Table Constraints 81

Appendix B: Data Validation Option Report Examples ........... 82
Summary of Testing Activities ....................................... 82
Table Pair Summary .................................................. 83
Detailed Test Results – Test Page ................................... 85
Detailed Test Results – Bad Records Page ....................... 86

Appendix C: Description of Views Used in Reporting ........... 87
Description of Views Used in Reporting Overview .............. 87
results_summary_view ............................................. 87
rs_bad_records_view ............................................... 90
Preface

The *PowerCenter Data Validation Option Installation and User Guide* is written for developers and software engineers responsible for validating moved or transformed data. This guide presents Data Validation Option features and capabilities. This guide assumes you have a solid understanding of PowerCenter applications such as the Designer, Workflow Manager, and Workflow Monitor. This guide also assumes that you understand your operating systems, relational database concepts, and the database engines and flat files in your environment.

Informatica Resources

**Informatica Customer Portal**

As an Informatica customer, you can access the Informatica Customer Portal site at [http://mysupport.informatica.com](http://mysupport.informatica.com). The site contains product information, user group information, newsletters, access to the Informatica customer support case management system (ATLAS), the Informatica How-To Library, the Informatica Knowledge Base, the Informatica Multimedia Knowledge Base, Informatica Product Documentation, and access to the Informatica user community.

**Informatica Documentation**

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

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

**Informatica Web Site**

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

**Informatica How-To Library**

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

Informatica Knowledge Base

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

Informatica Multimedia Knowledge Base

As an Informatica customer, you can access the Informatica Multimedia Knowledge Base at http://msupport.informatica.com. The Multimedia Knowledge Base is a collection of instructional multimedia files that help you learn about common concepts and guide you through performing specific tasks. If you have questions, comments, or ideas about the Multimedia Knowledge Base, contact the Informatica Knowledge Base team through email at KB_Feedback@informatica.com.

Informatica Global Customer Support

You can contact a Customer Support Center by telephone or through the Online Support. Online Support requires a user name and password. You can request a user name and password at http://msupport.informatica.com.

Use the following telephone numbers to contact Informatica Global Customer Support:

<table>
<thead>
<tr>
<th>North America / South America</th>
<th>Europe / Middle East / Africa</th>
<th>Asia / Australia</th>
</tr>
</thead>
</table>
| **Toll Free** Brazil: 0800 891 0202  
Mexico: 001 888 209 8853  
North America: +1 877 463 2435 | **Toll Free** France: 00800 4632 4357  
Germany: 00800 4632 4357  
Israel: 00800 4632 4357  
Italy: 800 915 985  
Netherlands: 00800 4632 4357  
Portugal: 800 208 360  
Spain: 900 813 166  
Switzerland: 00800 4632 4357 or 0800 463 200  
United Kingdom: 00800 4632 4357 or 0800 023 4632 | **Toll Free** Australia: 1 800 151 830  
New Zealand: 1 800 151 830  
Singapore: 001 800 4632 4357 |
| **Standard Rate** North America: +1 650 653 6332 | **Standard Rate** France: 0805 804632  
Germany: 01805 702702  
Netherlands: 030 6022 797 | **Standard Rate** India: +91 80 4112 5738 |
CHAPTER 1

Testing and Methodology

This chapter includes the following topics:

- Testing and Methodology Overview, 1
- Data Testing Approach, 1
- Sample Data Testing Methodology, 2

Testing and Methodology Overview

This guide presents Data Validation Option features and capabilities. Data validation is the process you use to verify that moved or transformed data is complete and accurate and has not been changed because of errors in the movement or transformation process. Use PowerCenter Data Validation Option to verify that your data is complete and accurate.

A sample methodology to help you design a rigorous data validation process is presented below. This approach should act as a helpful starting point.

Most users have some kind of a testing process already in place. Usually, it is a combination of SQL code and Excel spreadsheets. A common temptation is to replicate the current SQL-based process. The first question often asked is, "How do I do this with Data Validation Option?"

It is best to resist this temptation and to take the time to design the proper data testing strategy from the ground up. Often, the easiest tests catch overlooked errors.

Data Testing Approach

Use the following guidelines to set up a data testing approach:

1. Test data, not mappings or workflows. Your test framework should not parallel mappings, sessions, and workflows. Testing mappings is unit testing, which is different from data validation.

2. Do not try to mimic SQL. Step back and think of what you are trying to accomplish. Data Validation Option can make things a lot easier.

3. Assume the worst. If data needs to be moved from last_name to last_name, it may have been moved to city by mistake. If an IF statement was used, assume it was coded wrong. It is always prudent to assume a mistake has been made and be pleasantly surprised when tests return no errors.

4. Do the easy things first. Complicated problems often manifest themselves in simple ways. Simple counts and constraints can point out some obvious errors.
5. Design the initial test framework without taking performance into account. After you are satisfied with your
approach, begin to optimize.

6. Try to split complex SQL into more than one table pair. For example, if you see something like the following
statements:

Select CASE (code='X', TableAFld1, TableBFld1)
Select CASE (code='X', TableAFld2, TableBFld2)

You can create two table pairs:

Table A vs. Target WHERE clause A: code='X'
Table B vs. Target WHERE clause A: code <> 'X'

7. Do not copy formulas from the ETL mapping into Data Validation Option. Sometimes when you need to test a
complex transformation such as complex IF statements with SUBSTR, you might be tempted to just copy it
from the mapping. This approach produces an obvious problem. If there is an error in the ETL mapping
formula, you will replicate it in Data Validation Option, and Data Validation Option will not catch it. Therefore,
you must always maintain a proper separation between ETL and testing.

8. Do not try to do everything in Data Validation Option. If you think that a particular step can be accomplished
more easily with SQL, use SQL. If you run 95% of your validation in Data Validation Option, and can
document it with the audit trail, this is more than enough.

Sample Data Testing Methodology

When validating data, it is better to start with simple tests, and then introduce more complicated scenarios. In this
section, we introduce a methodology to test and validate data.

Basic Counts, Sums, and Aggregate Tests

This approach detects the following problems:

- Lack of referential integrity in the source. (Child with no parent will not be moved.)
- Row rejection by the target system.
- Incorrect ETL logic in the WHERE clauses.
- Other problems that do not move all the required records.

Goal

To make sure that all records were moved.

Approach

- COUNT and COUNT_ROWS
- SUM for numeric fields
- COUNT DISTINCT to compare detail vs. aggregate tables

Check Referential Integrity of Target Tables

Goal

To find lack of referential integrity in the target, either child without a parent or a fact record without
corresponding dimension table.
Approach

- TableA is child table; TableB is parent table.
- FieldA is child FK; FieldB is parent PK.
- Test is SETA_in_B. (Every child FK is in parent PK.)
- In case of a star schema, fact table is child, dimension is parent. (Every fact FK needs to be in parent PK.)
- In case of composite keys, create an expression that concatenates the keys, and run these tests on that expression.

Enforce Constraints on Target Tables

This is one of the most overlooked yet most effective data testing strategies. Often the errors in very complicated transformations manifest themselves in rather simple ways such as NULLs in the target, missing rows, or incorrect formats. Some examples of target table constraints are listed below:

Unique Primary Keys

UNIQUE(PK)

If a composite PK, use UNIQUE(expression).

Valid Individual Values

For example:

VALUE(FldA) Between 10, 50
VALUE(FldB) In {'A','B','C'}
VALUE(FldC) > 0
NOT_NULL(FldD)
FORMAT(Phone) = 'Reg expression' (optional)

You can have more than one test on a specific field and you can create tests on an expression.

Aggregate Constraints

These are often used for sanity checks in terms of rows moved, totals, etc. For example, is this correct?

<table>
<thead>
<tr>
<th>Source (staging file)</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 laptop 1000</td>
<td>1 laptop 1000</td>
</tr>
<tr>
<td>2 desktop 500</td>
<td>2 desktop 1500</td>
</tr>
</tbody>
</table>

This looks correct, but if this is XYZ company’s daily sales, it is not correct, even though it was moved correctly. Somehow you know that XYZ sells more than $2500/day. Therefore, you can say that anything less than 1000 records and anything less than $2m and more than $15m is suspect.

Therefore:

COUNT_ROWS(any f1d) > 1000

SUM(Amount) Between 1000000, 2000000
Compare Individual Records between Sources and Targets

This is another critical step in testing. Read the section that explains the difference between VALUE and OUTER_VALUE tests and the expected results when using each test.

Goal

If a field was moved without transformation, make sure it is the same. If it was transformed, make sure it was done correctly.

Approach

- Simple comparison. Create a table pair, join on a common keys, and then either set up tests automatically (right-click/generate) or manually if field names are different.
- Any row based expression (concatenation, calculation) can be tested similarly, for example:
  
  \[ \text{VALUE(first_name \| \| '_' \|\| last_name \|\| '@dvosoft.com' = email) } \]

Validate Complicated ETL Logic with SQL Views

After the simpler validation is done, you can use Data Validation Option SQL views to construct complicated SQL-based scenarios that involve multiple tables and complicated transformations.

Validate Lookup Integrity with Lookup Views

Testing lookups is an important step in data testing. Data Validation Option lookup views allow you to test the validity of the lookup logic in your transformation layer.
Data Validation Option Overview

This chapter includes the following topics:

- Data Validation Option Overview, 5
- Typical Data Validation Workflow, 6
- Data Validation Options Users, 6
- Architecture, 7
- System Requirements, 7

Data Validation Option Overview

Data Validation Option enables you to test and validate data. It requires the installation of, and does much of its work through, Informatica PowerCenter.

There are two types of data validation generally performed in a data integration setting, source to target comparisons and production to development comparisons. The source to target validation can be done at the end of development of a data integration project on the initial load of a data warehouse, or as reconciliation of the ongoing daily or incremental loads.

Data validation is also performed to compare production and development environments, when data integration software is upgraded, or when RDBMS database software is upgraded.
Finally, data validation can be performed as part of the testing process or as part of the production process, which is also called the reconciliation or Audit/Balance/Control process. Data Validation Option supports all of the use cases described above.

Data Validation Option reads table definitions from PowerCenter metadata repositories, and checks the data at either end of the process. It does not check the correctness of transformations or mappings. Data Validation Option identifies problems or inconsistencies. It does not attempt to identify the source of the problem in the ETL process.

Typical Data Validation Workflow

A typical workflow for data validation consists of multiple tasks.

1. Data Validation Option reads one or more PowerCenter metadata repositories.
2. You define the validation rules in Data Validation Option.
3. You run the rules to ensure the data conforms to the validation rules. When you do this, Data Validation Option performs the following tasks:
   - Creates and executes all tests through PowerCenter.
   - Loads results into the Data Validation Option results database and displays them in the Data Validation Option Client.
4. You examine the results and identify sources of inconsistencies in the ETL process or the source systems.
5. You repeat this process for new records.

Data Validation Options Users

There are many possible users of Data Validation Option:

- Business or Data Analysts
- Data Warehouse Testers
- ETL Developers
- Database Administrators

This guide makes the following assumptions:

- You know and understand data. You have worked with databases in the past and are comfortable with operations such as browsing databases, tables, and flat files.
- You know the data that you are validating. This includes table, field, and file names that may appear cryptic to an outsider. Data Validation Option does not replace these names with business names.
- You have a basic understanding of PowerCenter, which helps you identify the causes of failed tests or errors.
Architecture

Data Validation Option requires installation and setup of PowerCenter. Source and target data table and file definitions are imported from PowerCenter repositories. You set up table pairs and test rules in Data Validation Option. This test metadata is stored in the Data Validation Option repository. When the tests are run, Data Validation Option communicates with PowerCenter through an API to create appropriate mappings, sessions, and workflows, and to execute them. PowerCenter connects to the data being tested instead of Data Validation Option. After the tests are executed, results are stored in the Data Validation Option repository and displayed in the Data Validation Option Client.

System Requirements

The PowerCenter Client must be installed on the machine where Data Validation Option is installed. Any system that supports Informatica PowerCenter will support Data Validation Option. However, Data Validation Option works best on a machine that has at least 1GB of RAM.
CHAPTER 3

Data Validation Option Installation and Setup

This chapter includes the following topics:

- Installation and Setup Overview, 8
- Installation and Upgrade, 8
- User Configuration Directory, 14

Installation and Setup Overview

To install Data Validation Option, complete the following tasks:

1. Review the prerequisites.
2. Review the required system permissions.
3. Collect the required information described.
4. Perform the Data Validation Option setup steps.

After you install Data Validation Option, run a test to verify that the installation was successful.

Installation and Upgrade

You can install the Data Validation Option for the first user and for additional users. You can also upgrade the Data Validation Option.

Prerequisites

To complete the installation, the following prerequisites must be met:

- The machine that hosts Informatica Services must be installed and configured on the same local area network as the Data Validation Option Client machine.
- The Informatica domain must contain at least one PowerCenter Integration Service.
- The PowerCenter Client must be installed and set up on the same machine where Data Validation Option will be installed.
• At least one PowerCenter repository must be set up.

System Permissions Required for Installation

To complete Data Validation Option setup, the following system permissions are required:

• Ability to create a database. This includes schema creation as well as the ability to create tables, indexes, sequences, and views.
• Ability to create a PowerCenter connection object.
• Ability to create a folder in a PowerCenter repository.
• Ability to copy a JAR file onto the machine that hosts Informatica Services.
• Ability to change settings in the Administrator tool.
• Ability to modify the environment variables on the machine where Data Validation Option will be installed.
• Read and write permissions on the Data Validation Option installation directory and subdirectories. These permissions are required to generate reports.

Information Required for Installation

The following information is required to complete Data Validation Option setup:

1. Informatica domain name:

2. PowerCenter Integration Service name:

3. PowerCenter repository name(s):

4. PowerCenter repository username(s) and password(s):

5. Location of the domains.infa file on the client machine:
Installation Instructions – First User

To install Data Validation Option for the first user, perform the following steps:

1. Set up the Data Validation Option repository with a user that has privileges to create and modify tables, indexes, sequences and views. If the Data Validation Option repository database is IBM DB2, the user name and schema name should be the same.

   Enter the database information.
   Database server name: ____________________________________________________________
   Database port number: ____________________________________________________________
   Database name: __________________________________________________________________
   Database username and password: ________________________________________________

2. Open the Workflow Manager and set up a connection to the Data Validation Option repository. Every Data Validation Option user must have the permission to use this connection.

   Enter the connection name: ______________________________________________________

3. Open the Repository Manager and create a folder in the repository for Data Validation Option to store mappings that run tests. Use this folder only for storing Data Validation Option mappings.

   Enter the repository and folder names.
   Repository name: __________________________________________________________________
   Target folder name: __________________________________________________________________

4. On the Data Validation Option Client machine, create a new environment variable called $INFA_HOME$ and set the value to the location of the domains.infa file:
   a. Select Control Panel > System > Advanced > Environment Variables.
   b. Click New System Variable.
   c. Enter $INFA_HOME$ for the variable name.
   d. Enter the domains.infa file path, excluding the domains.infa filename, for the variable value. (See item 5 in “Information Required for Installation ” on page 9.)
   e. Click OK in each of the dialog boxes to close them.

5. Verify that the environment variable is set up correctly:
   a. Open the DOS command window and type set.
   b. The environment variable that was just set up should appear in the list of environment variables. It should read as follows:
      $INFA_HOME = C:\Informatica\<version>\clients\PowerCenterClient\$
   c. If the environment variable is not set, repeat step 4.

6. Install Data Validation Option on the client machine.
7. Create a folder in the root directory on the machine that hosts Informatica Services and copy the dvoct.jar file from the C:\Program Files\ [x86]\Informatica\version\DVO\powercenterlibs directory on the Data Validation Option Client to the new folder.

Enter the file location:

8. Update the Java SDK Classpath for the PowerCenter Integration Service:
   a. Open the Administrator tool on the machine that hosts Informatica Services.
   b. Select the PowerCenter Integration Service from the list on the left side.
   c. Click the Processes tab.
   d. Edit the Service Process Properties > General Properties.
   e. Locate the Java SDK Classpath.
   f. If there is a value already entered for the Java SDK Classpath, add a semi-colon (Windows) or colon (UNIX/Linux) after the existing classpath, and enter the path to the dvoct.jar file on the machine that hosts Informatica Services, including the dvoct.jar file name. If there is no classpath, enter the dvoct.jar path and filename.

If PowerCenter is installed in a grid environment, repeat this step for each node.

9. Enter the Data Validation Option repository information:
   a. In Data Validation Option, select Tools > Preferences > Data Validation Option.
   b. Enter the following information:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Enter a unique user name.</td>
</tr>
<tr>
<td>Database Type</td>
<td>Select Oracle, SQL Server, or DB2.</td>
</tr>
<tr>
<td>Database Driver</td>
<td>This value is automatically populated by Data Validation Option. It does not need to be changed.</td>
</tr>
<tr>
<td>Database URL</td>
<td>The value automatically populated by Data Validation Option for this field consists of a series of values. There are placeholders for the database host (server) name, database name, and port number, if appropriate. Enter the information from step 1. Remove the angle brackets (&lt;&gt;). Note: If Oracle RAC is used, the URL should look like the following text: jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALENCE=on) (ADDRESS=(PROTOCOL=TCP) (HOST=host1) PORT=1521)) (ADDRESS=(PROTOCOL=TCP) (HOST=host2) (PORT=1521)) (CONNECT_DATA=(SERVICE_NAME=service)) See tsnames.ora for the exact syntax.</td>
</tr>
<tr>
<td>Database User</td>
<td>Enter the database user name from step 1.</td>
</tr>
<tr>
<td>Database Password</td>
<td>Enter the database user password from step 1.</td>
</tr>
</tbody>
</table>

   c. Click Test to make sure the database information is correct.
   d. Click Save, and create the Data Validation Option repository schema when prompted.
10. Optionally, update the mapping properties:
   b. Enter the following information:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Details</td>
<td>Maximum number of bad records written to the Data Validation Option repository for each test.</td>
</tr>
<tr>
<td>DTM Buffer Size</td>
<td>PowerCenter setting that might need to be increased for table pairs involving many tests.</td>
</tr>
<tr>
<td>Max Concurrent Runs</td>
<td>Maximum number of PowerCenter sessions executed at the same time. Each table pair is executed as one session, regardless of how many tests it has.</td>
</tr>
</tbody>
</table>

11. Add the repository that contains the folder where Data Validation Option will create mappings:
   a. Right-click INFA Repositories in the Navigator, and select Add Repository. The Repository Editor dialog box opens with the controls listed below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name for the repository, ideally with the word “target” in it to identify the repository type.</td>
</tr>
<tr>
<td>Client Location</td>
<td>Enter the location of the pmrep.exe file on the client machine using the Browse button. Typically the location is: C:\Informatica&lt;version&gt;\clients \PowerCenterClient\client\bin.</td>
</tr>
<tr>
<td>PowerCenter Version</td>
<td>Select the PowerCenter version.</td>
</tr>
<tr>
<td>PowerCenter Domain</td>
<td>Enter the Informatica domain name defined in item 1 in “Information Required for Installation” on page 9.</td>
</tr>
<tr>
<td>Repository</td>
<td>Name of the PowerCenter repository defined in step 3.</td>
</tr>
<tr>
<td>User name</td>
<td>User name for the PowerCenter repository defined in step 3.</td>
</tr>
<tr>
<td>Password</td>
<td>User password for the PowerCenter repository defined in step 3.</td>
</tr>
<tr>
<td>Security Domain</td>
<td>LDAP security domain. Leave this field blank if you use native authentication.</td>
</tr>
<tr>
<td>Contains Target Folder</td>
<td>Select true.</td>
</tr>
<tr>
<td>Target Folder</td>
<td>Enter the folder name defined in step 3.</td>
</tr>
<tr>
<td>Integration Service</td>
<td>Enter the name of the PowerCenter Integration Service defined in item 2 in “Information Required for Installation” on page 9.</td>
</tr>
<tr>
<td>Data Validation Option Results Warehouse Connection</td>
<td>Enter the PowerCenter connection to the Data Validation Option repository defined in step 2.</td>
</tr>
</tbody>
</table>

   b. Click Save.

   Data Validation Option offers to test the repository settings to make sure they are correct.
c. Test the repository settings. This is highly recommended during initial setup.

After the settings are confirmed, Data Validation Option imports the PowerCenter sources, targets, and connections.

12. Optionally, you can add other repositories. Make sure that **Contains Target Folder** is **false** because only one repository can have a target folder.

13. Create a table pair with one test and run it to make sure the installation was successful.

**RELATED TOPICS:**
* "Troubleshooting" on page 74

### Installation Instructions – Additional Users

For each additional user, perform steps 4, 5, 6, 9, 11, and 13 defined in "Installation Instructions – First User" on page 10 on each client machine.

When you install Data Validation Option for multiple users, step 9 requires that each user enter connection information for the Data Validation Option repository. The connection information includes the database password. Data Validation Option stores connection information in the preferences file for each user. It creates the preferences file in the user configuration directory.

By default, the user configuration directory is one of the following directories:

- **32-bit operating systems:** `C:\Documents and Settings\<user name>\DataValidator\`
- **64-bit operating systems:** `C:\Users\<user name>\DataValidator\`

If you do not want users to have access to the database password, you can use the DVOCmd CreateUserConfig command to create users and preference files. Data Validation Option creates a preference file for each user. Each preferences file contains the same connection information as the preferences file for the user that runs the CreateUserConfig command.

Copy each preferences file from the output directory to the appropriate user configuration directory. When the user opens Data Validation Option, the user does not have to enter Data Validation Option repository information. Data Validation Option reads the connection information from the preferences file.

If you create users through the CreateUserConfig command, each additional user must still perform steps 4, 5, 6, 11, and 13 defined in "Installation Instructions – First User" on page 10.

**RELATED TOPICS:**
* "Command Line Integration" on page 67

### Installation Instructions – Upgrading from Version 3.0

Upgrading from version 3.0 to version 9.1.0 does not affect test metadata or test results. However, it is strongly recommended you make a back-up copy of the Data Validation Option repository database before proceeding with the upgrade.

1. Open Data Validation Option version 3.0, go to **Tools > Properties**, and note the name of the connection to the Data Validation Option repository.

2. Close Data Validation Option.

3. Uninstall Data Validation Option version 3.0.

4. Install Data Validation Option version 9.1.0.

5. Open a command line window, and run the DVOCmd UpgradeRepository command.
6. Optionally, edit C:\Program Files< (x86)\Informatica\version>\DVO\config\JMFProperties.properties and change the number of pmrep processes from 2 to 8. This step is optional, but recommended.

7. Start Data Validation Option. Edit the repository information and the following information:
   - PowerCenter version (new).
   - Name of the connection to the Data Validation Option repository (from step 1).
8. Click Save. You are prompted to verify that the test settings are correct.
9. Refresh the repository by right-clicking the repository name in the Navigator.
10. Close and restart Data Validation Option.

   **Note:** The database URL format changed between versions 3.0 and 9.1.0. If Data Validation Option fails to upgrade the database URL, select Tools > Preferences > Data Validation Option, and update the Database URL manually.

**Installation Instructions – Upgrading from Version 3.1**

Upgrading from version 3.1 to version 9.1.0 does not affect test metadata or test results. However, it is strongly recommended you make a back-up copy of the Data Validation Option repository database before proceeding with the upgrade.

1. Uninstall Data Validation Option version 3.1.
2. Install Data Validation Option version 9.1.0.
3. Open a command line window, and run the DVOCmd UpgradeRepository command.
4. Optionally, edit C:\Program Files< (x86)\Informatica\version>\DVO\config\JMFProperties.properties and change the number of pmrep processes from 2 to 8. This step is optional, but recommended.

   **Note:** The database URL format changed between versions 3.1 and 9.1.0. If Data Validation Option fails to upgrade the database URL, select Tools > Preferences > Data Validation Option, and update the Database URL manually.

**User Configuration Directory**

Data Validation Option creates a configuration directory for each user. The user configuration directory contains the user preferences file, preferences.xml. It also contains directories that store log files, reports, and temporary files that the user generates.

By default, the user configuration directory is one of the following directories:

- **32-bit operating systems:** C:\Documents and Settings\<user name>\DataValidator\n- **64-bit operating systems:** C:\Users\<user name>\DataValidator\\n
You can change the user configuration directory. To do this, create a batch file that starts the Data Validation Option Client and specifies the user configuration directory, or specify the user configuration directory through an environment variable.
Changing the User Configuration Directory through a Batch File

If you work with multiple repositories, you must specify a unique user configuration directory for each repository. For example, you create separate Data Validation Option repositories for your development and production environments. To specify unique user configuration directories for each repository, create a batch file for each repository that starts the Data Validation Option Client and specifies the user configuration directory.

For example, you installed Data Validation Option in the default directory on 32-bit Windows. You want to set the development user configuration directory to C:\DVOCConfig_Dev and the production user configuration directory to C:\DVOCConfig_Prod. Create two batch files that start the Data Validation Option Client.

Enter the following line in the development environment batch file:

"C:\Program Files\Informatica9.1.0\DVO\DVOClient.exe" C:\DVOCConfig_Dev

Enter the following line in the production environment batch file:

"C:\Program Files\Informatica9.1.0\DVO\DVOClient.exe" C:\DVOCConfig_Prod

Use the appropriate batch file to start the Data Validation Option Client. You can run one instance of the Data Validation Option Client at-a-time.

If you run Data Validation Options commands at the command line, use the --confdir option to specify the user configuration directory.

RELATED TOPICS:

- "Command Line Integration" on page 67

Changing the User Configuration Directory through an Environment Variable

You can change the user configuration directory through an environment variable on the Data Validation Option Client machine. If you change the user configuration directory through an environment variable and you run Data Validation Options commands at the command line, you do not need to specify the user configuration directory through the --confdir option.

To change the user configuration directory through an environment variable, complete the following steps:

1. Open the System properties in the Windows Control Panel.
2. Click the Advanced tab.
3. Click Environment Variables.
4. Create the following environment variable for the user:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DV_CONFIG_DIR</td>
<td>Full file path for the user configuration directory.</td>
</tr>
</tbody>
</table>

5. Click OK to save the changes.
New Features and Behavior Changes

This chapter includes the following topics:

- New Features and Enhancements, 16
- Behavior Changes, 17

New Features and Enhancements

Data Validation Option version 9.1.0 contains new features and enhancements.

Client Layout

- Folders. You can organize single tables and table pairs by placing them in folders. When you upgrade from version 3.1, the installation program creates a Default folder for each user and places the table pairs and single tables in the folder. When you create a new user, Data Validation Option creates a Default folder.

- Error reporting. If a test fails to run, Data Validation Option displays the test run error on the Tests tab. Previously, you had to examine the PowerCenter session log file to view test run errors.

- Single Tables tab. The details area contains separate tabs for table pairs and single tables.

Command Line Utility

- New Commands. The Data Validation Option command line utility, DVOCmd.exe, contains new commands that allow you to create users and refresh repositories.

PowerCenter Version

- PowerCenter 9.1.0. Data Validation Option 9.1.0 works with PowerCenter versions 8.5 and later, except for PowerCenter version 9.0.

Reports

- Reports for tests in folders. You can run reports for all tests in a folder.

- Report information. Reports display the folder name, error messages, join expressions, and conditions, if applicable.
Repositories

- Refreshing repositories. When you refresh a repository, you can refresh the entire repository, the connection objects, the folders, or the sources and targets. You can also refresh repository folders individually.

Single Tables and Table Pairs

- Expressions in join conditions. When you join two tables in a table pair, you can enter a PowerCenter expression as a field in the join condition. Enter an expression to join tables with key fields that are not identical.
- Large table processing. When you include a large table in a table pair, you can optimize the way Data Validation Option joins table data. You can specify which table Data Validation Option uses for the master or detail table. You can also use sorted output for the join.
- Pushing sorting logic to the database. To increase the performance of table pair and single table tests, you can push the sorting logic for joins to the source database. Pushing sorting logic to the database causes the database to sort records before it loads them to PowerCenter which minimizes disk input and output.

Tests

- Filter conditions. You can apply a filter condition to table pair and single table tests. If you apply a filter condition to a table pair test, Data Validation Option applies the filter condition after it joins the tables in the table pair.

Behavior Changes

Effective in 9.1.0, Data Validation Option behavior changes in multiple ways.

Client Layout

- Data Sources tab. The Data Sources tab is removed.
- Folders. Table pairs and single tables appear in the Default folder in the Navigator. Previously, single tables and table pairs appeared in the Single Tables and Table Pairs nodes in the Navigator.
- Properties area. The Properties area is moved to the bottom right side of the Data Validation Option Client to make more room for the Navigator. Previously, the Properties area appeared in the bottom left side of the screen.
- Results tab. The tab that lists bad records for tests is renamed to Results. The Results tab displays test summary information for table pairs, single tables, and tests. It also displays bad records for certain types of tests. Previously, Data Validation Option displayed the Details tab only for tests. It displayed bad records only.
- Single Tables tab. In the details area, single tables are listed on the Single Tables tab. Previously, single tables were listed on the Table Pairs tab.

Installation

- Executable name. The Data Validation Option executable file name is DVOClient.exe. Previously, the executable file name was DataValidator.exe.
- Installation directory. The Data Validation Option default installation directory is C:\Program Files \Informatica\version\DVO on 32-bit operating systems and C:\Program Files (x86)\Informatica\version\DVO on 64-bit operating systems. Previously, the default installation directory was C:\Program Files \DVOSoft.
Reports

- Reports for single-table constraints. Reports for single-table constraints display information about the single table only. Previously, reports for single-table constraints were the same as reports for table pairs except they displayed “Table B” values as null values.

Repositories

- Importing metadata. When you add a PowerCenter repository, Data Validation Option imports folder names. If the repository is a target repository, Data Validation Option also imports connection metadata. To import source and target metadata, you must refresh the repository. Previously, when you saved a repository for the first time, Data Validation Option imported folder names and all source and target metadata. If the repository was a target repository, Data Validation Option also imported connection metadata.
- Refreshing repositories. When you refresh a repository, you do not have to close and restart Data Validation Option. Previously, you had to restart Data Validation Option after you refreshed a repository.

Single Tables and Table Pairs

- Single Tables Editor. You create single tables using the Single Table Editor dialog box. Previously, you created single tables using the Table Pairs Editor dialog box.

SQL and Lookup Views

- SQL view definition. When you create an SQL view, you can select the tables and columns to use in the view. Data Validation Option detects the datatype, precision, and scale of the columns. Previously, when you created an SQL view, you had to define the SQL relationships across tables, define the columns, and define the datatype, precision, and scale of each column manually.
Chapter 5

Data Validation Option Client Layout

This chapter includes the following topics:
- Data Validation Option Client Layout Overview, 19
- Data Validation Option Client Tabs, 20

Data Validation Option Client Layout Overview

The Data Validation Option Client contains multiple areas that allow you to perform different tasks.

Statistics Area

The statistics area is in the upper left area of the screen, below the menu bar. This area displays information about the number of repositories, table pairs, single tables, tests, and data sources that exist in the current instance of Data Validation Option. It also displays information about running tests and the user name.

Run Tests Button

The Run Tests button is in the upper right section of the Data Validation Option Client. When you click this button, the tests that correspond to the selected table pairs are run.

Navigator

The Navigator is on the left side of the Data Validation Option Client. It contains the following objects:

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFA Repositories</td>
<td>Lists all PowerCenter repositories that you add to Data Validation Option. Expand a repository to see the repository folders and the sources and targets in each folder.</td>
</tr>
<tr>
<td>SQL views</td>
<td>Lists the SQL views that you create.</td>
</tr>
<tr>
<td>Lookup views</td>
<td>Lists the lookup views that you create.</td>
</tr>
<tr>
<td>Folders</td>
<td>Lists the single tables and table pairs that you create.</td>
</tr>
</tbody>
</table>
Details Area

The details area is in the upper right section of the Data Validation Option Client. It contains tabs that display details about the objects you create in Data Validation Option such as tests, table pairs, single tables, and views.

Properties Area

The Properties area is in the lower right section of the Data Validation Option Client. It displays the properties for the object that you select in the Navigator or details area.

Results Area

The Results area appears in the lower right section of the Data Validation Option Client when you select a test, table pair, or single table in the details area. The Results area displays test summary information, results, and the bad records written to the Data Validation Option repository.

The following figure shows the Data Validation Client areas:

![Data Validation Option Client Tabs](image)

---

Data Validation Option Client Tabs

The Data Validation Option Client contains tabs that display different information.

Tests Tab

The Tests tab in the details area displays all tests set up in this instance of Data Validation Option.

By default, tests are sorted in the order they were created. However, you can sort tests by clicking the column header.
The following table describes the columns on the Tests tab:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test status icon</td>
<td>Indicates whether tests associated with the table pair have been run and the status of the most recent run. If you hold the pointer over the icon, Data Validation Option displays the meaning of the icon.</td>
</tr>
<tr>
<td>Description</td>
<td>The test description. By default, the test description displays the test type, fields, and operators.</td>
</tr>
<tr>
<td>Table Pair</td>
<td>The name of the table pair.</td>
</tr>
<tr>
<td>Test Run Date/Time</td>
<td>The date and time that the tests were last run.</td>
</tr>
<tr>
<td>Test Run Error</td>
<td>If a test failed, this column lists the error.</td>
</tr>
</tbody>
</table>

### Table Pairs Tab

The Table Pairs tab in the details area displays all table pairs set up in this instance of Data Validation Option. By default, table pairs are sorted in the order they were created. However, you can sort table pairs by clicking the column header.

The following table describes the columns on the Table Pairs tab:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test status icon</td>
<td>Indicates whether tests associated with the table pair have been run and the status of the most recent run. If you hold the pointer over the icon, Data Validation Option displays the meaning of the icon.</td>
</tr>
<tr>
<td>Folder</td>
<td>The folder that contains the table pair.</td>
</tr>
<tr>
<td>Description</td>
<td>The table pair description.</td>
</tr>
<tr>
<td>Table A</td>
<td>The name of the table selected as Table A.</td>
</tr>
<tr>
<td>Table B</td>
<td>The name of the table selected as Table B.</td>
</tr>
<tr>
<td>Test Run Date/Time</td>
<td>The date and time that the tests were last run.</td>
</tr>
<tr>
<td>Test Run Error</td>
<td>If a test failed, this column lists the error.</td>
</tr>
</tbody>
</table>

### Single Tables Tab

The Single Tables tab in the details area displays all single tables set up in this instance of Data Validation Option. By default, single tables are sorted in the order they were created. However, you can sort single tables by clicking the column header.
The following table describes the columns on the Single Tables tab:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test status icon</td>
<td>Indicates whether tests associated with the table have been run and the status of the most recent run. If you hold the pointer over the icon, Data Validation Option displays the meaning of the icon.</td>
</tr>
<tr>
<td>Folder</td>
<td>The folder that contains the single table.</td>
</tr>
<tr>
<td>Description</td>
<td>The single table description.</td>
</tr>
<tr>
<td>Table</td>
<td>The name of the table.</td>
</tr>
<tr>
<td>Test Run Date/Time</td>
<td>The date and time that the tests were last run.</td>
</tr>
<tr>
<td>Test Run Error</td>
<td>If a test failed, this column lists the error.</td>
</tr>
</tbody>
</table>

SQL Views Tab

The SQL Views tab in the details area displays all SQL views set up in this instance of Data Validation Option. By default, SQL views are sorted in the order they were created. However, you can sort SQL views by clicking the column header.

The following table describes the columns on the SQL Views tab:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>SQL view description.</td>
</tr>
<tr>
<td>Table Name</td>
<td>Tables you use to create the SQL view.</td>
</tr>
<tr>
<td>SQL Statement</td>
<td>SQL statement that you run against the database to retrieve data for the SQL view.</td>
</tr>
</tbody>
</table>

The right-click menu in the SQL View tab lists the following options: **Add to Table Pair, Add to Single Table, Add SQL View, Edit SQL View, and Delete SQL View**, and **Export Metadata**.

Lookup Views Tab

The Lookup Views tab in the details area displays all lookup views set up in this instance of Data Validation Option. By default, lookup views are sorted in the order they were created. However, you can sort lookup views by clicking the column header.

The following table describes the columns on the SQL Views tab:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Lookup view description.</td>
</tr>
<tr>
<td>Source Table</td>
<td>Source table name.</td>
</tr>
<tr>
<td>Lookup Table</td>
<td>Lookup table name.</td>
</tr>
</tbody>
</table>
The right-click menu in the Lookup Views tab lists the following options: Add to Table Pair, Add to Single Table, Add Lookup View, Edit Lookup View, Delete Lookup View, and Export Metadata.
Chapter 6

Repositories

This chapter includes the following topics:

- Repositories Overview, 24
- Repository Properties, 24
- Adding Repositories, 25
- Editing Repositories, 26
- Deleting Repositories, 26
- Refreshing Repositories, 26

Repositories Overview

Data Validation Option connects to a PowerCenter repository to import metadata for PowerCenter sources, targets, folders, and connection objects. Data Validation Option also connects to a PowerCenter repository to create mappings, sessions, and workflows in the Data Validation Option target folder.

When you add a repository to Data Validation Option, you add either a source or target repository. You can add one target repository and multiple source repositories.

The target repository contains metadata for PowerCenter sources, targets, folders, and connection objects. It also contains the Data Validation Option target folder. The target folder stores the mappings, sessions, and workflows that Data Validation Option creates when you run tests. Do not store other PowerCenter mappings, sessions, or workflows in this folder.

A source repository contains metadata for PowerCenter sources, targets, and folders. Add source repositories to Data Validation Option if you want to compare tables from different repositories. When you add a source repository, you must verify that all connection objects in the source repository also exist in the target repository. Data Validation Option uses the connection objects in the target repository when you run tests on table pairs.

The version number for a source repository can differ from the version number for the target repository. The version numbers for two source repositories can also differ.

Repository Properties

When you select a repository in the Navigator, Data Validation Option displays repository properties in the Properties area. Repository properties display the information you enter in the Repository Editor dialog box.
PowerCenter uses these properties to connect to the repository and send metadata to Data Validation Option through the PowerCenter Application Programming Interface (API).

Edit repository properties in the **Repository Editor** dialog box. Data Validation Option displays the **Repository Editor** dialog box when you add or edit a repository.

The following table describes the repository properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the repository. When you create a repository, enter a name that indicates whether the repository is a source or target repository. When you edit a repository, you cannot change the name. The repository name is case sensitive.</td>
</tr>
<tr>
<td>Client Location</td>
<td>Location of the pmrep.exe file on the client machine. Typically the location is: C:\Informatica&lt;version&gt;\clients\PowerCenterClient\client\bin.</td>
</tr>
<tr>
<td>PowerCenter Version</td>
<td>The PowerCenter version.</td>
</tr>
<tr>
<td>PowerCenter Domain</td>
<td>The Informatica domain name.</td>
</tr>
<tr>
<td>Repository</td>
<td>The name of the PowerCenter repository.</td>
</tr>
<tr>
<td>User name</td>
<td>The user name for the PowerCenter repository.</td>
</tr>
<tr>
<td>Password</td>
<td>The user password for the PowerCenter repository.</td>
</tr>
<tr>
<td>Security Domain</td>
<td>The LDAP security domain. Leave this field blank if you use native authentication.</td>
</tr>
<tr>
<td>Contains Target Folder</td>
<td>Indicates whether the repository is a source or target repository. Select false if the repository you create is a source repository. Select true if it is a target repository. When you select true, you must specify the Target Folder, Integration Service, and Data Validation Option Results Warehouse Connection information.</td>
</tr>
<tr>
<td>Target Folder</td>
<td>The name of the folder that has been set up in the PowerCenter repository to store the mappings generated by the tests set up in Data Validation Option.</td>
</tr>
<tr>
<td>Integration Service</td>
<td>The name of the PowerCenter Integration Service.</td>
</tr>
<tr>
<td>Data Validation Option Results Warehouse Connection</td>
<td>The name of the PowerCenter connection to the Data Validation Option repository.</td>
</tr>
</tbody>
</table>

**Adding Repositories**

To add a repository, right-click **INFA Repositories** in the Navigator, and select **Add Repository**. The **Repository Editor** dialog box opens. To add a target repository, enter the repository properties in the **Repository Editor** dialog box, and set **Contains Target Folder** to **true**. To add a source repository, enter repository properties in the **Repository Editor** dialog box, and set **Contains Target Folder** to **false**.

After you enter repository properties, you can test the repository connection. To test the repository connection, click **Test**. Data Validation Option verifies the connection properties. If the repository is a target repository, Data Validation Option also verifies the Integration Service name and that the Data Validation Option target folder and option results warehouse connection exist in the repository.
To save the repository, click **Save**. If you save a new repository or save changes to the connection properties for an existing repository, but you did not test the repository connection, Data Validation Option prompts you to test the repository connection. Testing the repository connection makes it easier to troubleshoot testing errors later.

Saving the repository does not import source or target metadata from the PowerCenter repository. To import source and target metadata, you must refresh the repository.

---

**Editing Repositories**

To edit a repository, right-click the repository, and select **Edit Repository**, or double-click the repository. The **Repository Editor** dialog box opens. You can update any property that is enabled.

After you update repository properties, you can test the repository connection. To test the repository connection, click **Test**. Data Validation Option verifies the connection properties. If the repository is a target repository, Data Validation Option also verifies the Integration Service name and that the Data Validation Option target folder and option results warehouse connection exist in the repository.

---

**Deleting Repositories**

To delete a repository from Data Validation Option, right-click the repository, and select **Delete Repository**. Data Validation Option deletes the repository and all table pairs, single tables, tests, and views based on the repository data.

---

**Refreshing Repositories**

You refresh a source repository when the contents of the PowerCenter repository have changed. You usually refresh a target repository only when there are additions or changes to connection objects.

When you refresh a repository, Data Validation Option reimports source, target, folder, and connection metadata from the PowerCenter repository. Therefore, Data Validation Option objects that use changed or deleted PowerCenter objects might no longer be valid after you refresh a repository. If you created table pairs, single tables, or tests with tables that were deleted from the PowerCenter repository, Data Validation Option deletes them when you refresh the repository.

To refresh all repositories at once, right-click **INFA Repositories** in the Navigator, and select **Refresh All Repositories**. To refresh one repository, right-click the repository, and select **Refresh Repository**.

When you refresh one repository, you can select the objects to refresh. Select one of the following options:

- **Everything**
  - Data Validation Option reimports all source, target, folder, and connection metadata. It updates the folder list, and the Sources and Targets folders in the Navigator.

- **Connections**
  - Data Validation Option reimports connection metadata. Select this option when a PowerCenter user adds, removes, or updates connection objects.
Folder List

Data Validation Option reimports folder metadata. It updates the folder list in the Navigator. Select this option when a PowerCenter user adds or removes folders.

Folders (Sources and Targets)

Data Validation Option reimports source and target metadata. It refreshes the contents of the Sources and Targets folders in each folder in the repository. Select this option when a PowerCenter user adds, removes, or modifies sources or targets in folders.

You can also refresh repository folders individually. You might refresh a folder after you refresh the folder list and Data Validation Option imports a new folder. To refresh a repository folder, right-click the folder in the Navigator, and select Refresh Folder (Sources and Targets). Data Validation Option refreshes the contents of the Sources and Targets folders within the folder you refresh.

Note: Refreshing everything in a repository or refreshing all repositories can take several minutes to several hours, depending on the size of the repositories. If you work with a small number of repository folders, you can shorten refresh time by refreshing the folders individually.

Exporting Repository Metadata

You can export repository metadata to a file. You might want to export repository metadata when migrating from a development to a production environment or if you are asked to do by Informatica Global Customer Support.

To export repository metadata from Data Validation Option, right-click the repository, and select Export Metadata. Data Validation Option prompts you for a file name and file path.
Chapter 7

Table Pairs

This chapter includes the following topics:
- Table Pairs Overview, 28
- Table Pair Properties, 28
- Adding Table Pairs, 33
- Editing Table Pairs, 33
- Deleting Table Pairs, 33
- Viewing Overall Test Results, 34

Table Pairs Overview

A table pair is the basis for all tests that compare one table to another. You can select a relational table, flat file, lookup view, or SQL view as one or both tables in a table pair.

Table Pair Properties

You can view table pair properties by selecting a table pair in either the Navigator or the Table Pairs tab and viewing the properties. Most properties come from the values you enter in the Table Pair Editor dialog box. Other properties come from the tests set up for, and run on, that table pair.

Edit table pair properties in the Table Pair Editor dialog box. Data Validation Option displays the Table Pair Editor dialog box when you add or edit a table pair. The properties vary depending on the types of objects you select for the table pair.

The following table describes the table pair properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Table Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table A/B</td>
<td>All</td>
<td>The first or second table in the table pair.</td>
</tr>
<tr>
<td>Is Large A/B</td>
<td>All</td>
<td>Optimizes join processing for large tables when you run value and set tests.</td>
</tr>
<tr>
<td>Conn A/B</td>
<td>Relational</td>
<td>PowerCenter connection for the table.</td>
</tr>
<tr>
<td>Property</td>
<td>Table Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Override Owner Name A/B</td>
<td>Relational</td>
<td>Overrides the schema or owner name for the table.</td>
</tr>
</tbody>
</table>
| Optimize in Database A/B | Relational | Controls which test logic Data Validation Option converts to a PowerCenter mapping and which test logic it pushes to the database. You can select one of the following options:  
- Disabled. Data Validation Option converts all test logic to a PowerCenter mapping.  
- WHERE clause only. Data Validation Option pushes the WHERE clause to the database. It converts all other test logic to a PowerCenter mapping.  
- WHERE clause, Sorting, Aggregation. Data Validation Option pushes the WHERE clause, sorting logic for joins, and all aggregate tests to the database. It converts all other test logic to a PowerCenter mapping. |
| Source Dir A/B           | Flat file  | Source file directory if the table is a flat file. The path is relative to the machine that hosts Informatica Services.                      |
| Source File A/B          | Flat file  | File name, including file extension, if the table is a flat file.                                                                          |
| Where clause A/B         | All        | Filters the records that the PowerCenter Integration Service reads from the database. Enter the a valid PowerCenter Boolean expression or an SQL WHERE clause without the WHERE keyword. |
| Description              | All        | Table pair description. By default, Data Validation Option uses "Joined <Table A>-<Table B>" for joined table pairs. It uses "<Table A>-<Table B>" for table pairs that are not joined. |
| External ID              | All        | Identifier for the table pair that you can use when you run Data Validation Option tests at the command line.                                |
| Table Join               | All        | Join condition for the tables.                                                                                                             |

**Related Topics:**
- "Single-Table Constraints" on page 45
- "Tests for Single-Table Constraints" on page 48

**Selecting Tables**

To select a table for a table pair, click one of the Browse buttons. The Select Data Source dialog box opens. This dialog box displays all of the tables, flat files, SQL views, and lookup views available in the repositories. You can sort information in this dialog box by clicking the column headers. You can reduce the number of items to select by typing one or more letters of the table, file, or view name in the Search field.

To select the table, click Select or double-click the table in the list. The Select Data Source dialog box closes, and the item is inserted as one of the tables for the table pair.

To select a second table, click the other Browse button and select another table.
**Processing Large Tables**

When you include a large table in a table pair, you can optimize the way Data Validation Option joins table data. Data Validation Option uses joins in value and set tests. To run value tests on a table pair, you must join the tables based on the related keys in each table. When you run set tests, Data Validation Option joins the tables.

By default, Data Validation Option joins the tables with an inner equijoin. The join condition uses the following `WHERE` clause syntax:

Table A.column_name = Table B.column_name

If one of the tables in a table pair contains a large volume of data compared to the other table, you can improve test performance by designating the larger table as the detail source and the smaller table as the master source. The PowerCenter Integration Service compares each row of the master source against the detail source when it determines whether to join two records. To designate the larger table as the detail source, enable the `Is Large` option for the larger table. The table where the `Is Large` option is disabled becomes the master source.

If both tables in a table pair are large, you can improve test performance by using sorted input. When you use sorted data, the PowerCenter Integration Service minimizes disk input and output when it runs a test. To use sorted data when the PowerCenter Integration Service joins the tables in the table pair, enable the `Is Large` option for both tables.

You can further increase performance for large relational tables by pushing the sorting logic to the database. To push sorting logic to the database, enable the `Is Large` option for both tables, and set the `Optimize in Database` property to `WHERE clause, Sorting, Aggregation`.

**Selecting Connections**

You must provide the PowerCenter connection information for the tables used in table pairs. Select a connection object from the `Conn A` or `Conn B` field.

PowerCenter users create connection objects in the Workflow Manager. Data Validation Option imports the connection objects from the PowerCenter repository through the target repository. Connection objects provide the credentials that PowerCenter uses to fetch data from relational tables. If connection objects are changed in the Workflow Manager, you must refresh the target repository to update the connections in Data Validation Option. Selecting an incorrect connection for a table produces an error when you run tests.

**Overriding Owner Name**

If you need to override the schema or owner name for the table to qualify the table name, select the `Override Owner Name` option for Table A or Table B. When you select this option, enter the default owner name in the `Override Owner Name` field.

For example, an SQL Server table is identified by `<database>..<schema>..<table>`. To override the database and the schema, enter `<new database name>..<new schema name>` in the text box. If only the schema should be changed, enter `<new schema name>` in the text box. You cannot change only the database name.

**Pushing Test Logic to the Database**

You can push some test logic for a relational table to the source database. By default, Data Validation Option creates a PowerCenter mapping for each table pair. When you run a test, PowerCenter runs the mapping logic in a session. In some cases, you can significantly increase test performance by processing logic in the database instead of a PowerCenter session.

You can push the `WHERE` clause, logic for aggregate tests, and sorting logic for joins to the source database.
Pushing the WHERE clause to the database can reduce the number of rows the PowerCenter Integration Service reads when it runs a Data Validation Option test. For example, Table A is a table of U.S. customers, and you want to test data only for customers in California. You enter a WHERE clause such as `STATE = 'CA'`. Data Validation Option creates a mapping in PowerCenter that reads all U.S. customers. The mapping might contain a Filter transformation that removes all records for customers outside of California.

If you push the WHERE clause to the database, the database filters customer records. The PowerCenter Integration Service reads records for California customers only. Pushing the WHERE clause to the database increases the performance of the PowerCenter session because the Integration Service reads a small subset of records instead of all records in the table.

Pushing aggregate test logic to the database can also reduce the number of rows the PowerCenter Integration Service reads from the database. For example, you use a COUNT test to compare the number of non-null records in two Customer_ID columns. If you push the test logic to the database, the PowerCenter Integration Service does not have to import all customer records in order to count them.

Pushing the sorting logic for joins to the database causes the database to sort records before it loads them to PowerCenter. This minimizes disk input and output when Data Validation Option runs a test. The reduction in input and output is greatest for large tables, so you might want to push sorting logic to the database when you run tests on tables with large volumes of data.

Pushing test logic to the database slightly increases the load on the database. Before you push test logic to the database, you must decide whether the increased performance of the test outweighs the increased load on the database.

To push test logic to the database, use the **Optimize in Database** property.

The following table describes the values for the **Optimize in Database** property:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>Data Validation Option converts all test logic to a PowerCenter mapping. The PowerCenter Integration Service performs test logic when it runs the associated sessions. This is the default option. If you enter a WHERE clause, you must use valid PowerCenter syntax.</td>
</tr>
<tr>
<td>WHERE clause only</td>
<td>Data Validation Option pushes the WHERE clause to the database. It converts all other test logic for the table pair to a PowerCenter mapping. Because the database processes the WHERE clause, the syntax must be valid database syntax.</td>
</tr>
<tr>
<td>WHERE clause, Sorting, Aggregation</td>
<td>Data Validation Option pushes the WHERE clause and the logic for aggregate tests to the database. If you select the <strong>Is Large</strong> option, Data Validation Option also pushes sorting logic to the database. The PowerCenter Integration Service converts all other test logic for the table pair to a PowerCenter mapping. Because the database processes the WHERE clause, the syntax must be valid database syntax.</td>
</tr>
</tbody>
</table>

The **Optimize in Database** property is available for relational tables only.

**Source Directory and File Name**

If you select a flat file as one of the tables in a table pair, enter the file directory and file name plus file extension in the corresponding text boxes so that the PowerCenter Integration Service can find the file when running the tests. The path you enter must be relative to the machine that hosts Informatica Services.

**WHERE Clauses**

To limit the records returned from a table and made available for a test, enter a WHERE clause in the **Table Pair** dialog box. Because PowerCenter pulls data from each table individually, you can provide separate WHERE
clauses, or filters, for the data pulled from each one. Enter the WHERE clause without the "WHERE" keyword, for example, CITY <> 'London'.

The PowerCenter Integration Service is case-sensitive when it reads WHERE clauses. This functionality corresponds to the use of the Filter transformation in PowerCenter.

Data Validation Option does not check the WHERE clause syntax. If the PowerCenter Integration Service executes the WHERE clause, any valid PowerCenter expression, including expressions that use PowerCenter functions, is allowed. If the database executes the WHERE clause, it must be a valid SQL statement. If the PowerCenter syntax is not valid, a mapping installation error occurs. If the SQL statement is not valid, a runtime error occurs.

When you enter a WHERE clause, consider the following issues:

- Data Validation Option uses two WHERE clauses instead of one. A typical SQL statement has one WHERE clause. Data Validation Option, however, has one WHERE clause for Table A and one for Table B. Therefore, it is possible that more data comes from one table than the other. For example, applying emp_id < 10 to Table A but not Table B results in only nine records coming from Table A and all records from Table B. This affects OUTER_VALUE and aggregate tests, which might or might not be what you intended. However, when you compare production to development where the production environment has three years of data and development only has two weeks, applying a WHERE clause to production equalizes the data sets.

- Certain validation problems can be solved through a nested SQL WHERE clause. For example, if you want to filter for employees with disciplinary issues, use the following WHERE clause (assuming it is executed in the database):

  emp_id IN (SELECT DISTINCT emp_id FROM table_discipline)

- Because the filter condition you enter in the WHERE clause applies to all tests in the table pair, Data Validation Option applies the WHERE clause before it joins the tables. This can improve performance when the WHERE clause filters a large percentage of rows from the source table because the PowerCenter Integration Service processes fewer rows later in the mapping. If you want to enter a condition that filters a small percentage of rows, or you want to apply different filters for different tests, you can enter a filter condition in the Table Pair Test Editor dialog box.

**Description**

By default, the table pair description is "Joined <Table A>-<Table B>" if the table pair is joined, and "<Table A>-<Table B>" if it is not joined. You can edit the table pair description.

**External ID**

To run data validation tests at the command line or embed them into any kind of a workflow, you can create an external ID for the table pair. Enter any unique string that you can reference when you run Data Validation Option tests at the command line.

**Joining Tables**

A table pair can be joined or unjoined. You must join the tables if you want to run a VALUE or OUTER_VALUE test. Data Validation Option ignores joins for all set and aggregate tests.

To create a joined table pair, you define one or more conditions based on equality between the tables. For example, if both tables in a table pair contain employee ID numbers, you might select EMPLOYEE_ID as the join field for one table and EMP_ID as the join field for the other table. Data Validation Option performs an inner equijoin based on the matching ID numbers.
You can join tables only on fields of like datatypes. For example, you can join an INT field to a DECIMAL field, but not to a DATETIME or VARCHAR field. Data Validation Option supports numeric, string, datetime and binary/other datatypes. Joins are not allowed on binary/other datatypes.

You can create a join with one set of fields or with multiple sets of fields from each table if the table requires this to produce unique records. Note that additional sets of fields increase the time necessary to join two sources. The order of the fields in the join condition can also impact the performance of Data Validation Option tests. If you use multiple sets of fields in the join condition, Data Validation Option compares the ports in the order you specify.

When you create a join, you can select a field from each table or enter an expression. Enter an expression to join tables with key fields that are not identical. For example, you have two customer tables that use different cases for the LAST_NAME field. You might enter the following expression for one of the join fields: `lower(LAST_NAME)`

When you enter an expression for one or both join fields, you must specify the datatype, precision, and scale of the result. The datatype, precision, and scale of both join fields must be compatible. The expression must use valid PowerCenter expression syntax. Data Validation Option does not check the expression syntax.

Create a join by selecting a field from each table in the Table Join area of the Table Pair Editor dialog box. To enter an expression, select Expression from the join field, and enter the expression in the Join Expression Editor dialog box.

### Adding Table Pairs

To add a table pair, select two tables in the Navigator, and select Add to Table Pair. To select a second table, hold the Ctrl key and click the second table. If you created folders, Data Validation Option prompts you to select a folder for the table pair. You can also add a table pair by right-clicking a folder in the Navigator and selecting Add Table Pair.

When you add a table pair, the Table Pair Editor dialog box opens.

### Editing Table Pairs

To edit a table pair, right-click it in the Navigator or Table Pairs tab, and select Edit Table Pair. You can also edit a table pair by double-clicking it in the Table Pairs tab.

When you edit a table pair, the Table Pair Editor dialog box opens.

### Deleting Table Pairs

To delete a table pair, right-click a table pair in the Navigator or Table Pairs tab, and select Delete Table Pair. You can also delete a table pair by selecting it and pressing the Delete key.

When you delete a table pair, Data Validation Option deletes all of its tests. Data Validation Option does not delete lookup or SQL views used in the table pair.
Viewing Overall Test Results

After you add a table pair to Data Validation Option, you can view the properties by selecting the table pair in either the Navigator or the Table Pairs tab and viewing the properties in the Properties area. The Overall Result property displays the status of tests run on the table pair.

The following table describes the values for the Overall Result property:

<table>
<thead>
<tr>
<th>Overall Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASS</td>
<td>All tests for the table pair passed.</td>
</tr>
<tr>
<td>FAIL</td>
<td>At least one test for the table pair failed.</td>
</tr>
<tr>
<td>NOT TESTED</td>
<td>The tests for the table pair have not been run.</td>
</tr>
<tr>
<td>INSTALLING</td>
<td>Data Validation Option is creating the PowerCenter mappings, sessions, and workflows for the tests.</td>
</tr>
<tr>
<td>RUNNING</td>
<td>The tests for the table pair are running.</td>
</tr>
<tr>
<td>NO RESULTS</td>
<td>No results were found in the database for this test. This usually means there were no records in the data set.</td>
</tr>
<tr>
<td>INSTALL ERROR</td>
<td>The mapping generated by the test could not be installed into PowerCenter.</td>
</tr>
<tr>
<td>RUN ERROR</td>
<td>The session generated by the test could not be run in PowerCenter. The connection information might not be correct or the source database might not be available.</td>
</tr>
</tbody>
</table>
CHAPTER 8

Tests for Table Pairs

This chapter includes the following topics:
- Tests for Table Pairs Overview, 35
- Test Properties, 36
- Expression Tips, 40
- Adding Tests, 41
- Editing Tests, 42
- Deleting Tests, 42
- Running Tests, 42
- Automatically Generating Value Tests, 42
- Comparing Repository Folders, 43
- Bad Records, 43

Tests for Table Pairs Overview

You can run the following types of tests on table pairs:

Aggregate
- Includes COUNT, COUNT_DISTINCT, COUNT_ROWS, MIN, MAX, AVG, and SUM.

Set
- Includes AinB, BinA, and AeqB.

Value
- Includes VALUE and OUTER_VALUE.

Note: When you run tests, the target folder must be closed in the Designer and Workflow Manager. If the target folder is open, Data Validation Option cannot write to the folder, and the tests return an error.
When you select a test in the Navigator or in the Tests tab, the properties for that test appear in the Properties area. Most properties come from the values you enter in the Table Pair Test Editor dialog box. Other properties apply to the most recent test run.

Edit test properties in the Table Pair Test Editor dialog box when you add or edit a test.

The following table describes the test properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>The test you run such as COUNT, COUNT_DISTINCT, AinB, or VALUE.</td>
</tr>
<tr>
<td>Field A/B</td>
<td>The field that contains the values you want to compare when you run the test. You must select a field from each table in the table pair.</td>
</tr>
<tr>
<td>Condition A/B</td>
<td>Allows you to filter records after Data Validation Option joins the tables in the table pair. Enter a valid PowerCenter Boolean expression.</td>
</tr>
<tr>
<td>Operator</td>
<td>The arithmetic operator that defines how to compare each value in Field A with each value in Field B.</td>
</tr>
<tr>
<td>Threshold</td>
<td>The allowable margin of error for an aggregate or value test that uses the approximate operator.</td>
</tr>
<tr>
<td>Max Bad Records</td>
<td>The maximum number of records that can fail comparison for a test to pass.</td>
</tr>
<tr>
<td>Case Insensitive</td>
<td>Ignores case when you run a test that compares string data.</td>
</tr>
<tr>
<td>Trim Trailing Spaces</td>
<td>Ignores trailing spaces when you run a test that compares string data.</td>
</tr>
<tr>
<td>Null=Null</td>
<td>Allows null values in two tables to be considered equal.</td>
</tr>
<tr>
<td>Comments</td>
<td>Information about a test. Data Validation Option displays the comments when you view test properties in the Properties area.</td>
</tr>
<tr>
<td>Field A/B is Expression</td>
<td>Allows you to enter an expression for Field A or Field B.</td>
</tr>
<tr>
<td>Datatype</td>
<td>The datatype for the expression if Field A or Field B is an expression.</td>
</tr>
<tr>
<td>Precision</td>
<td>The precision for the expression if Field A or Field B is an expression.</td>
</tr>
<tr>
<td>Scale</td>
<td>The scale for the expression if Field A or Field B is an expression B.</td>
</tr>
</tbody>
</table>
The following table describes the table pair tests:

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COUNT</strong></td>
<td>Compares the number of non-null values for each of the selected fields. This test works with any datatype. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime.</td>
</tr>
<tr>
<td><strong>COUNT_DISTINCT</strong></td>
<td>Compares the distinct number of non-null values for each of the selected fields. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime.</td>
</tr>
<tr>
<td><strong>COUNT_ROWS</strong></td>
<td>Compares the total number of values for each of the selected fields. This test counts nulls, unlike the COUNT and COUNT_DISTINCT tests. This test works with any datatype.</td>
</tr>
<tr>
<td><strong>MIN</strong></td>
<td>Compares the minimum value for each of the selected fields. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime.</td>
</tr>
<tr>
<td><strong>MAX</strong></td>
<td>Compares the maximum value for each of the selected fields. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime.</td>
</tr>
<tr>
<td><strong>AVG</strong></td>
<td>Compares the average value for each of the selected fields. This test can only be used with numeric datatypes.</td>
</tr>
<tr>
<td><strong>SUM</strong></td>
<td>Compares the sum of the values for each of the selected fields. This test can only be used with numeric datatypes.</td>
</tr>
<tr>
<td><strong>SET_AinB</strong></td>
<td>Determines whether the entire set of values for Field A exist in the set of values for Field B. This test works with any datatype except binary/other. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime. You can use this test to confirm that all values in a field exist in a lookup table. This test examines all values for a column instead of making a row-by-row comparison.</td>
</tr>
<tr>
<td><strong>SET_BinA</strong></td>
<td>Determines whether the entire set of values for Field B exist in the set of values for Field A. Determines whether the entire set of values for the field selected from Table B exist in the set of values for the field selected from Table A. This test works with any datatype except binary/other. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime. You can use this test to confirm that all values in a field exist in a lookup table. This test examines all values for a column instead of making a row-by-row comparison.</td>
</tr>
<tr>
<td><strong>SET_AeqB</strong></td>
<td>Determines whether the set of values for the selected fields are exactly the same when compared. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime. You can use this test to confirm that all values in a field exist in a lookup table. This test examines all values for a column instead of making a row-by-row comparison.</td>
</tr>
<tr>
<td><strong>SET_ANotInB</strong></td>
<td>Determines whether there are any common values between the selected fields. If there are common values, the test returns an error. If there are no common values, the test succeeds.</td>
</tr>
<tr>
<td><strong>VALUE</strong></td>
<td>For joined table pairs, this test compares the values for the fields in each table, row-by-row, and determines whether they are the same. If there are any rows that exist in one table but not the other, the rows are disregarded which implies an inner join between the tables. If the fields are both null and the Null=Null option is disabled, this pair of records fails the test. This test works with any datatype except...</td>
</tr>
</tbody>
</table>
Test | Description
--- | ---
binary. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime.

OUTER_VALUE | For joined table pairs, this test compares the values for the fields in each table, row-by-row, and determines whether they are the same. If there are any rows that exist in one table but not the other, they are listed as not meeting the test rules which implies an outer join between the tables. For the test to pass, the number of rows for the tables, as well as the values for each set of fields must be equal. If the fields are both null and the Null=Null option is disabled, this set of records fails the test. This test works with any datatype except binary. The fields you compare must be of the same general datatype, for example, numeric-to-numeric or datetime-to-datetime.

**Fields A and B**

To create a test, you must select the fields that contain the values you want to compare from each table in the table pair. The fields available in each table appear in **Field A** and **Field B**. Select a field from each table.

**Conditions A and B**

You can filter the values for each field in a VALUE or OUTER_VALUE test to exclude rows that do not satisfy the test condition. For example, you want to exclude telephone extension numbers that contain fewer than three characters. Use the following VALUE test:

Table A.EXT = Table B.EXT, Condition A = LENGTH(EXT) < 3, Condition B = LEAHGHT(EXT) < 3

The filter condition you enter for a test differs from the WHERE clause you enter for a table pair. Data Validation Option applies the WHERE clause to all tests in the table pair, before it joins the tables. It applies the test filter condition after it joins the tables. You might want to use a test filter condition instead of a filter condition in the WHERE clause when the filter condition does not remove a large percentage of rows. This can improve performance if you run one test on the table pair.

Data Validation Option does not check the condition syntax. Any valid PowerCenter expression, including expressions that use PowerCenter functions, is allowed. If the PowerCenter syntax is not valid, a mapping installation error occurs when you run the test.

To enter a filter condition for either field, enter the filter condition in the **Condition A** or **Condition B** field. Because the PowerCenter Integration Service processes the filter condition, it must use valid PowerCenter syntax. Enter the field name in the filter condition, for example, Emp_ID > 0. Do not include the WHERE keyword.

**Operator**

The operator defines how to compare the test result for Field A with the test result for Field B. Enter an operator for aggregate and value tests.

The following table describes the operators available in the **Operator** field:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equals</td>
<td>Implies that the test result for Field A is the same as the test result for Field B. For example, SUM(Field A) is the same as SUM(Field B).</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Does not equal</td>
<td>Implies that the test result for Field A is not the same as the test result for Field B.</td>
</tr>
</tbody>
</table>
### Operator Table

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Is less than</td>
<td>Implies that the test result for Field A is less than the test result for Field B.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Is less than or equal to</td>
<td>Implies that the test result for Field A is less than or equal to the test result for Field B.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Is greater than</td>
<td>Implies that the test result for Field A is greater than the test result for Field B.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Is greater than or equal to</td>
<td>Implies that the test result for Field A is greater than or equal to the test result for Field B.</td>
</tr>
<tr>
<td>~</td>
<td>Is approximately the same as</td>
<td>Implies that the test result for Field A is approximately the same as the test result for Field B. An approximate test must have a threshold value. You can use this operator with numeric datatypes only.</td>
</tr>
</tbody>
</table>

**Note:** Data Validation Option compares string fields using an ASCII table.

**RELATED TOPICS:**
- “Data Validation Option Report Examples” on page 82

## Threshold

A threshold is a numeric value that defines an acceptable margin of error for a test. You can enter a threshold for aggregate tests and for value tests with numeric datatypes.

In an aggregate test, the threshold defines the number of records that do not have to match for the test to pass. For example, you run a COUNT test that uses the “=” operator and set the threshold to 10. The test passes when the results are within 10 records of each other.

In a value test, the threshold defines the numeric margin of error used when comparing two values. It is especially useful in cases of rounding. For example, you run a VALUE test that uses the “=” operator. The test compares a REAL field with a value of 100.99 to an INTEGER field with a value of 101. The test passes when the threshold value is at least 0.01.

A threshold is required if the test uses the approximate operator. To define a threshold for a test, enter a numeric value in the **Threshold** field.

## Max Bad Records

By default, for a set or value test to pass, all records must compare successfully. Data Validation Option lists records that do not compare successfully as bad records. To disregard bad records, enter a number in the **Max Bad Records** field. The test passes if the number of bad records does not exceed this value.

Value and set tests display bad records on the Results tab. To view bad records for a test, select the test in the Navigator or Tests tab, and click the Results tab.

## Case Insensitive

String comparison in PowerCenter is case-sensitive. If you want the PowerCenter Integration Service to ignore case when you run a test that compares strings, enable the **Case Insensitive** option. This option is disabled by default.
Trim Trailing Spaces

By default, string comparison fails if two strings are identical except one string contains extra spaces. For example, one field value in a test is 'Data Validation' and the other field value is 'Data Validation ' (with three blank spaces after the last character). If you do not trim trailing spaces, the test produces a bad record. If you trim trailing spaces, the comparison passes because the extra spaces are ignored.

You might want to trim trailing spaces when the CHAR datatype, which pads a value entered with spaces to the right out to the length of the field, is used. A field of CHAR(20) compared to CHAR(30) fails, even if both fields have the same value, unless you trim the trailing spaces.

Enable the Trim Trailing Spaces option if there are spaces after the value entered in a field that should be ignored in the comparison. This option is disabled by default.

Null = Null

If null values in Field A and Field B should be considered equal, enable the Null = Null option. For example, a current employee in a table that contains employee information has a null termination date. If two records with null termination dates were compared by a database, they would not be considered equal because SQL does not consider a null value in one field to be equal to a null value in another field.

Because business users often consider null values to be equal, the Null = Null option is enabled by default.

Comments

You can enter information about a test in the Comments field. Data Validation Option displays comments in the Properties window when you select the test in the Navigator or the Tests tab.

Expression Definitions

To substitute an expression for a database value in a test field, enable the Field A/B is Expression option. When you enable this option, Data Validation Option disables the Field A or Field B control in the Table Pair Test Editor dialog box and enables the expression-related fields.

The expression must use valid PowerCenter expression syntax or the test will not run properly. To verify the syntax, you can write the expression in the PowerCenter Expression Editor, validate it there, and then paste it into the expression definition field in Data Validation Option. You must enclose text values in single quotes.

The datatype value represents the datatype of the expression after calculation. If you do not select the correct datatype, the test might produce an error. The datatypes you can select are PowerCenter datatypes.

The precision and scale must match the precision and scale used in PowerCenter for the datatype or the test might produce an error. The scale for any string or text datatype is zero. The precision and scale for a datetime datatype is 23 and 3, respectively.

Expression Tips

Testing often requires the use of different expressions. PowerCenter functions are described at the end of this guide.

The following examples demonstrate how to use expressions for data validation.
Concatenation, RTRIM, and SUBSTR

Often data transformation involves concatenation or the use of substring functions.

The following example tests the result of concatenation transformation:

- Expression A: `UPPER(first_name || ' ' || last_name)`
- Field B: `full_name`

IF Statements

The IF function is arguably the most popular testing function. The syntax for the IF function is as follows:

`IF(condition, if_true_part, if_false_part)`

The following example shows the IF function used in testing:

<table>
<thead>
<tr>
<th>Table A</th>
<th>Table B</th>
</tr>
</thead>
<tbody>
<tr>
<td>sales_usa</td>
<td>region (either 'USA' or 'INTL')</td>
</tr>
<tr>
<td>sales_intl</td>
<td>sales</td>
</tr>
</tbody>
</table>

The aggregate validation can be accomplished by two tests:

- Test 1: SUM
  - Field A: `sales_usa`
  - Expression B: `IF(region='USA', sales, 0)`
- Test 2: SUM
  - Field A: `sales_intl`
  - Expression B: `IF(region='INTL', sales, 0)`

Adding Tests

You can add tests to table pairs one at-a-time or you can auto-generate tests in batches. You can add any test manually.

To add a test to a table pair, right-click the name of the table pair in the Navigator or Table Pairs tab, or right-click in the Tests tab, and select Add Test. The Table Pair Test Editor dialog box opens.

You can generate value tests in a batch for table pairs that have tables with matching field names and datatype families. You can also generate value tests in a batch for tables or files within two target folders that have matching table or file names, field names, and datatype families.

**RELATED TOPICS:**

- "Automatically Generating Value Tests" on page 42
- "Comparing Repository Folders" on page 43
Editing Tests

To edit a test, right-click the test in the Navigator or Tests tab, and select **Edit Test**. You can also double-click the test name. The **Table Pair Test Editor** dialog box opens.

Deleting Tests

To delete a test, right-click the test in the Navigator or the Tests tab, and select **Delete Test**. You can also use the Delete key.

Running Tests

To run tests, select one or more table pairs and click the **Run Tests** button. You can also run tests by right-clicking a folder in the Navigator and selecting **Run Folder Tests** or by right-clicking a test in the Tests tab and selecting **Run Selected Tests**.

Data Validation Option runs all tests for a table pair together. You cannot run tests individually unless only one test is set up for the table pair. If you select an individual test, Data Validation Option runs all tests for the table pair.

After you run a test, you can view the results on the Results tab.

Data Validation Option uses the following logic to determine whether a test passes or fails:

- An aggregate test is calculated as "A <operator> B." If this relationship is true, the test passes. If the operator chosen is approximate, then the test is calculated as ABS(A-B) <= Threshold.

- A value or set test must produce fewer or an equal number of records that do not match compared to the threshold value. If there is no threshold value and there are no records that do not match, the test passes.

When you run tests, the target repository folders must be closed in the Designer or Workflow Manager. If the target folders are open, Data Validation Option cannot write to the folders, and the tests fail.

Automatically Generating Value Tests

To compare two tables with at least some of the same field names, create a joined table pair with the tables. Right-click the table pair in the Navigator, and select **Generate Value Tests**. Data Validation Option automatically generates an OUTER_VALUE test for the set of fields used for the join if the field names and field datatype families are the same. It generates VALUE tests for the remaining fields in which the field names and datatype families (string, numeric, date/time) match.

If there are field names in the tables that do not match or that match but have different datatype families, Data Validation Option does not generate a test for those fields. The OUTER_VALUE test reports any difference in the presence of join field values. The VALUE test reports any differences in actual values for each set of fields. Data Validation Option does not auto-generate tests for binary datatypes.

You can automatically generate VALUE tests to compare numbers of tables during a PowerCenter upgrade or migration from development to production.
Comparing Repository Folders

To compare two versions of a group of target tables or files, use the Ctrl key to select two folders in the same or different source repositories. Right-click and select **Compare Repository Folders.** The **Compare Repository Folders** dialog box opens.

If you compare database tables, you must select a default connection each table. If you compare flat files, enter a default path for each file. If any of the tables or files require connection information that differs from the default information entered (for example, an Oracle and a SQL Server table in folder A, or files located in different directories), update this information with the correct values after the table pairs and tests have been generated.

After you click the **Create** button, Data Validation Option creates table pairs for all tables and files that have matching names in both folders. It creates tests for all fields that have matching names and datatype families in each table pair. Data Validation Option sets up an OUTER.VALUE test for the primary key fields as long as the keys are defined in either the database or PowerCenter. Data Validation Option sets up VALUE tests for the remaining fields if they have a match in the other table or file.

The following table describes the fields in the **Compare Repository Folders** dialog box:

<table>
<thead>
<tr>
<th>Field Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>The name of the source repository in Data Validation Option.</td>
</tr>
<tr>
<td>Folder</td>
<td>The name of the folder selected in the source repository in Data Validation Option</td>
</tr>
<tr>
<td>Subfolder</td>
<td>The name of the selected subfolder, Sources or Targets.</td>
</tr>
<tr>
<td>Connection</td>
<td>The connections available in the Workflow Manager. You can select the connection that matches the largest number of tables or leave this field empty. Update this information, if necessary, after the table pair has been generated.</td>
</tr>
<tr>
<td>Source Dir</td>
<td>The path for the files to be compared from the perspective of the machine that hosts Informatica Services. You can enter the source directory that matches the largest number of files or leave this field empty. Update this information, if necessary, after the table pair has been generated.</td>
</tr>
</tbody>
</table>

Bad Records

When you select a test on the Tests tab, the records that fail either a value test or a set test appear on the Results tab. Different columns appear on the Results tab depending on the test.

**Aggregate Tests**

Aggregate tests do not display bad records. The Results tab displays the test result value from Field A, the test result value from Field B, and the comparison operator.
Value Tests

Value tests display the following columns for each bad record:

♦ The key for Table A
♦ The field or expression from Table A being compared
♦ The key for Table B
♦ The field or expression from Table B being compared

Set Tests

Set tests display the following columns for each bad record:

♦ The result from Table A
♦ The result from Table B
Chapter 9

Single-Table Constraints

This chapter includes the following topics:
- Single-Table Constraints Overview, 45
- Single Table Properties, 45
- Adding Single Tables, 46
- Editing Single Tables, 47
- Deleting Single Tables, 47
- Viewing Overall Test Results, 47

Single-Table Constraints Overview

Use a single-table constraint to run tests on a single table. Single-table constraints define valid data within a table. You can enforce valid values, aggregates, formats, and uniqueness. For example, you might want to verify that no annual salary in an employee table is less than $10,000.

Errors in complex logic often manifest themselves in very simple ways, such as NULL values in the target. Therefore, setting aggregate, value, NOT_NULL, UNIQUE, and FORMAT constraints on a target table is a critical part of any testing plan.

To run single-table constraints, you must create a single table. You can select a relational table, flat file, lookup view, or SQL view as a single table.

Single Table Properties

You can view single table properties by selecting a table in either the Navigator or the Single Tables tab and viewing the properties. Most properties come from the values entered in the Single Table Editor. Other properties come from the tests set up for and run on the table.

Edit single table properties in the Single Table Editor dialog box. Data Validation Option displays the Single Table Editor dialog box when you add or edit a single table. The properties vary depending on the type of object you select for the table.
The following table describes the single table properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Table Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>All</td>
<td>The table name.</td>
</tr>
<tr>
<td>Conn</td>
<td>Relational</td>
<td>PowerCenter connection for the table.</td>
</tr>
<tr>
<td>Override Owner Name</td>
<td>Relational</td>
<td>Overrides the schema or owner name for the table.</td>
</tr>
</tbody>
</table>
| Optimize in Database      | Relational | Controls which test logic Data Validation Option converts to a PowerCenter mapping and which test logic it pushes to the database. You can select one of the following options:  
  - Disabled. Data Validation Option converts all test logic to a PowerCenter mapping.  
  - WHERE clause only. Data Validation Option pushes the WHERE clause to the database. It converts all other test logic to a PowerCenter mapping.  
  - WHERE clause, Aggregation. Data Validation Option pushes the WHERE clause and all aggregate tests to the database. It converts all other test logic to a PowerCenter mapping. |
| Source Dir                | Flat file  | Source file directory if the table is a flat file. The path is relative to the machine that hosts Informatica Services. |
| Source File               | Flat file  | File name, including file extension, if the table is a flat file.           |
| Where clause              | All        | Allows you to limit the number of records that the PowerCenter Integration Service reads from the database. Enter a valid PowerCenter Boolean expression or an SQL WHERE clause without the WHERE keyword. |
| Description               | All        | Single table description. By default, Data Validation Option uses the table name. |
| External ID               | All        | Identifier for the single table that you can use when you run Data Validation Option tests at the command line. |
| Primary Key Column        | All        | Primary key column or columns for the table.                                |

Single table properties function in the same way as table pair properties.

**Related Topics:**
* “Table Pair Properties” on page 28

### Adding Single Tables

To add a single table, select a table in the Navigator, and select **Add to Single Table**. If you created folders, Data Validation Option prompts you to select a folder for the table. You can also add a single table by right-clicking a folder in the Navigator and selecting **Add Single Table**.

When you add a single table, the **Single Table Editor** dialog box opens.
Editing Single Tables

To edit a single table, right-click it in the Navigator or Single Tables tab, and select **Edit Single Table**. You can also edit a single table by double-clicking it in the Single Tables tab.

When you edit a single table, the **Single Table Editor** dialog box opens.

Deleting Single Tables

To delete a single table, right-click it in the Navigator or Single Tables tab, and select **Delete Single Table**. You can also delete a single table by selecting it and pressing the Delete key. When you delete a single table, Data Validation Option deletes all of its tests. Data Validation Option does not delete lookup or SQL views used in the single table.

Viewing Overall Test Results

After you add a single table to Data Validation Option, you can view the properties by selecting the table in either the Navigator or the Single Tables tab and viewing the properties in the Properties area. The Overall Result property displays the status of tests run on the table.

The following table describes the values for the **Overall Result** property:

<table>
<thead>
<tr>
<th>Overall Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASS</td>
<td>All tests for the table passed.</td>
</tr>
<tr>
<td>FAIL</td>
<td>At least one test for the table failed.</td>
</tr>
<tr>
<td>NOT TESTED</td>
<td>The tests for the table have not been run.</td>
</tr>
<tr>
<td>INSTALLING</td>
<td>Data Validation Option is creating the PowerCenter mappings, sessions, and workflows for the tests.</td>
</tr>
<tr>
<td>RUNNING</td>
<td>The tests for the table are running.</td>
</tr>
<tr>
<td>NO RESULTS</td>
<td>No results were found in the database for this test. This usually means there were no records in the data set.</td>
</tr>
<tr>
<td>INSTALL ERROR</td>
<td>The mapping generated by the test could not be installed into PowerCenter.</td>
</tr>
<tr>
<td>RUN ERROR</td>
<td>The session generated by the test could not be run in PowerCenter. The connection information might not be correct or the source database might not be available.</td>
</tr>
</tbody>
</table>
CHAPTER 10

Tests for Single-Table Constraints

This chapter includes the following topics:
- Tests for Single-Table Constraints Overview, 48
- Test Properties, 48
- Adding Tests, 52
- Editing Tests, 52
- Deleting Tests, 53
- Running Tests, 53
- Bad Records, 53

Tests for Single-Table Constraints Overview

Single-table constraints are tests based on a single table. Data Validation Option allows you to run an aggregate test or a VALUE test on single tables. Note that there are no set tests nor an OUTER_VALUE test for single tables. However, there are some additional tests available for single tables that you cannot create for a table pair.

Most single-table constraints allow you to enter a constraint value for the test. The constraint value defines the value or values to which you want to compare the values in a field. For example, you might want to verify that a SALARY field contains values greater than $10,000. Enter the minimum salary as the constraint value.

**Note:** When you run tests, the target folder must be closed in the Designer and Workflow Manager. If the target folder is open, Data Validation Option cannot write to the folder, and the tests return an error.

Test Properties

When you select a test in the Navigator or in the Tests tab, the properties for that test appear in the Properties area. Most properties come from the values you enter in the Single Table Test Editor dialog box. Other properties apply to the most recent test run.

Edit test properties in the Single Table Test Editor dialog box when you add or edit a test.
The following table describes the test properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>The test you run such as COUNT, COUNT_DISTINCT, VALUE, or NOT_NULL.</td>
</tr>
<tr>
<td>Field</td>
<td>The field that contains the values you want to test.</td>
</tr>
<tr>
<td>Condition</td>
<td>Filter condition for the test. Enter a valid PowerCenter Boolean expression.</td>
</tr>
<tr>
<td>Operator</td>
<td>The operator that defines how to compare each value in the field with the constraint value.</td>
</tr>
<tr>
<td>Constraint Value</td>
<td>The value or values you want to compare the field values to.</td>
</tr>
<tr>
<td>Threshold</td>
<td>The allowable margin of error for an aggregate or value test.</td>
</tr>
<tr>
<td>Max Bad Records</td>
<td>The maximum number of records that can fail comparison for a test to pass.</td>
</tr>
<tr>
<td>Case Insensitive</td>
<td>Ignores case when you run a test on string data.</td>
</tr>
<tr>
<td>Trim Trailing Spaces</td>
<td>Ignores trailing spaces when you run a test that on string data.</td>
</tr>
<tr>
<td>Comments</td>
<td>Information about a test. Data Validation Option displays the comments when you view test properties in the Properties area.</td>
</tr>
<tr>
<td>Field is Expression</td>
<td>Allows you to enter an expression for the field.</td>
</tr>
<tr>
<td>Datatype</td>
<td>The datatype for the expression if the field is an expression.</td>
</tr>
<tr>
<td>Precision</td>
<td>The precision for the expression if the field is an expression.</td>
</tr>
<tr>
<td>Scale</td>
<td>The scale for the expression if the field is an expression.</td>
</tr>
</tbody>
</table>

Tests

The following table describes the single table tests:

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>Compares the number of non-null values for the selected field to the constraint value. This test works with any datatype.</td>
</tr>
<tr>
<td>COUNT_DISTINCT</td>
<td>Compares the distinct number of non-null values for the selected field to the constraint value. This test works with any datatype except binary.</td>
</tr>
<tr>
<td>COUNT_ROWS</td>
<td>Compares the total number of values for the selected field to the constraint value. This test counts nulls, unlike the COUNT and COUNT_DISTINCT tests. This test works with any datatype.</td>
</tr>
<tr>
<td>MIN</td>
<td>Compares the minimum value for the selected field to the constraint value. This test works with any datatype except binary.</td>
</tr>
<tr>
<td>MAX</td>
<td>Compares the maximum value for the selected field to the constraint value. This test works with any datatype except binary.</td>
</tr>
<tr>
<td>Test</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AVG</td>
<td>Compares the average value for the selected field to the constraint value. This test can only be used with numeric datatypes.</td>
</tr>
<tr>
<td>SUM</td>
<td>Compares the sum of the values for the selected field to the constraint value. This test can only be used with numeric datatypes.</td>
</tr>
<tr>
<td>VALUE</td>
<td>Examines the values for the field, row by row, and compares them to the constraint value. This test works with any datatype except binary.</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Determines whether the values in the field match the pattern in the constraint value. The PowerCenter Integration Service uses the REG_MATCH function for this test. This test cannot be used with binary datatypes.</td>
</tr>
<tr>
<td>UNIQUE</td>
<td>Confirms that the value in the field is unique. This test does not use a constraint value. This test cannot be used with binary datatypes.</td>
</tr>
<tr>
<td>NOT_NULL</td>
<td>Confirms that the value in the field is not null. This test does not use a constraint value. This test cannot be used with binary datatypes.</td>
</tr>
<tr>
<td>NOT_BLANK</td>
<td>If the value in the field is a string value, this test confirms that the value in the field is not null or an empty string. If the value in the field is a numeric value, this test confirms that the value in the field is not null or zero. This test does not use a constraint value. This test cannot be used with datetime or binary datatypes.</td>
</tr>
</tbody>
</table>

**Field**

To create a single-table constraint, you must select the field that contains the values you want to test. The fields available in the single table appear in the Field drop-down list. Select a field from the list.

**Condition**

You can filter the values for the test field in a VALUE, FORMAT, NOT_NULL, or NOT_BLANK test. Data Validation Option does not test records that do not satisfy the filter condition. For example, you want to test rows in an ORDERS table only if the store ID number is not “1036.” Enter `STORE_ID <> 1036` in the Condition field.

Data Validation Option does not check the condition syntax. Any valid PowerCenter expression, including expressions that use PowerCenter functions, is allowed. If the PowerCenter syntax is not valid, a mapping installation error occurs when you run the test.

Enter the filter condition in the Condition field. Because the PowerCenter Integration Service processes the filter condition, it must use valid PowerCenter syntax. Do not include the WHERE keyword.

**Operator**

The operator defines how to compare the test result for the field with the constraint value. Enter an operator for aggregate, VALUE, and FORMAT tests.
The following table describes the operators available in the **Operator** field:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equals</td>
<td>Implies that the test result for the field is the same as the constraint value. For example, ( \text{SUM(field)} ) is the same as the constraint value.</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Does not equal</td>
<td>Implies that the test result for the field is not the same as the constraint value.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Is less than</td>
<td>Implies that the test result for the field is less than the constraint value.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Is less than or equal to</td>
<td>Implies that the test result for the field is less than or equal to the constraint value.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Is greater than</td>
<td>Implies that the test result for the field is greater than the constraint value.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Is greater than or equal to</td>
<td>Implies that the test result for the field is greater than or equal to the constraint value.</td>
</tr>
<tr>
<td>~</td>
<td>Is approximately the same as</td>
<td>Implies that the test result for the field is approximately the same as the constraint value. The approximate operator requires a threshold value. It only applies to numeric datatypes.</td>
</tr>
<tr>
<td>Between</td>
<td>Is between two values entered</td>
<td>Implies that the test result for the field is between the two constants entered as the constraint value. This operator is generally used for numeric or datetime datatypes.</td>
</tr>
<tr>
<td>Not Between</td>
<td>Is not between two values entered</td>
<td>Implies that the test result for the field is not between the two constants entered as the constraint value. This operator is generally used for numeric or datetime datatypes.</td>
</tr>
<tr>
<td>In</td>
<td>Is included in a list of values entered</td>
<td>Implies that the test result for the field is in the list of constants entered as the constraint value.</td>
</tr>
<tr>
<td>Not In</td>
<td>Is not included in a list of values entered</td>
<td>Implies that the test result for the field is not in the list of constants entered as the constraint value.</td>
</tr>
</tbody>
</table>

**Note:** Data Validation Option compares string fields using an ASCII table.

**RELATED TOPICS:**

- “Data Validation Option Report Examples” on page 82

**Constraint Value**

The constraint value represents a constant value to which you want to compare the field values. For example, you might want to verify that all values in the **ORDER_DATE** field fall between January 1, 2010 and December 31, 2010. Or, you might want to verify that the minimum **ORDER_ID** number is greater than 1000.

The constraint value must be a string, numeric, or datetime constant. The datatype of the constraint value depends on the test.
The following table lists the constraint value datatype allowed for each test:

<table>
<thead>
<tr>
<th>Test</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT, COUNT_DISTINCT, COUNT_ROWS</td>
<td>Integer</td>
</tr>
<tr>
<td>MIN, MAX, SUM, VALUE</td>
<td>Same as the Field datatype.</td>
</tr>
<tr>
<td>FORMAT</td>
<td>String</td>
</tr>
<tr>
<td>AVG</td>
<td>Double</td>
</tr>
<tr>
<td>UNIQUE, NOT_NULL, NOT_BLANK</td>
<td>These tests do not use a constraint value.</td>
</tr>
</tbody>
</table>

Enter a constraint value in the **Constraint Value** field. Enter a constant or list of constants separated by commas.

The number of constants you enter as the constraint value depends on the operator you use:

- **Arithmetic operator such as =, <>, or ~**
  - Enter a single constant.
- **Between or Not Between operator**
  - Enter two constants separated by a comma.
- **In or Not In operator**
  - Enter multiple constants separated by commas.

Enclose each string value, datetime value, or format pattern within single quotes. Datetime values must match the PowerCenter standard datetime format of MM/DD/YYYY HH24:MI:SS.

**Remaining Controls on Test Editor**

The remaining controls on the Single Table Test Editor are used in the same manner that they are used for table pairs.

**Adding Tests**

To add a test to a single table, right-click the name of the table in the Navigator or Single Tables tab, or right-click in the Tests tab, and select **Add Constraint Test**. The **Single Table Test Editor** dialog box opens.

**Editing Tests**

To edit a test, right-click the test in the Navigator or Tests tab, and select **Edit Test**. You can also double-click the test name. The **Single Table Test Editor** dialog box opens.
Deleting Tests

To delete a test, right-click the test in the Navigator or the Tests tab, and select **Delete Test**. You can also use the Delete key.

Running Tests

To run tests, select one or more single tables and click the **Run Tests** button. You can also run tests by right-clicking a folder in the Navigator and selecting **Run Folder Tests** or by right-clicking a test in the Tests tab and selecting **Run Selected Tests**.

Data Validation Option runs all tests for a single table together. You cannot run tests individually unless only one test is set up for the table. If you select an individual test, Data Validation Option runs all tests for the single table.

After you run a test, you can view the results on the Results tab.

Data Validation Option uses the following logic to determine whether a test passes or fails:

- An aggregate test is calculated as “value <operator> constraint.” If this relationship is true, the test passes. If the operator chosen is approximate, then the test is calculated as \( \text{ABS}(\text{value-constraint}) \leq \text{Threshold} \).
- A VALUE test must produce fewer or an equal number of records that do not match compared to the threshold value. If there is no threshold value and there are no records that do not match, the test passes.
- A FORMAT test is calculated as “value <operator> constraint.” If this relationship is true, the test passes.
- A UNIQUE, NOT_NULL, or NOT_BLANK test passes if the field value is unique, is not null, or is not blank, respectively. For string values, not blank means the string is not null or empty. For numeric values, not blank means the number is not null or 0.

When you run tests, the target repository folders must be closed in the Designer or Workflow Manager. If the target folders are open, Data Validation Option cannot write to the folders, and the tests fail.

Bad Records

When you select a test on the Tests tab, the records that fail a test appear on the Results tab. Different columns appear on the Results tab depending on the test.

**Aggregate Tests**

Aggregate tests do not display bad records. The Results tab displays the test result value.

**VALUE, FORMAT, NOT_NULL, and NOT_BLANK Tests**

These tests display the following columns for each bad record:

- The key or expression for the field
- The field value

**UNIQUE Tests**

UNIQUE tests display the non-unique field values.
This chapter includes the following topics:

- SQL Views Overview, 54
- SQL View Properties, 54
- Adding SQL Views, 56
- Editing SQL Views, 56
- Deleting SQL Views, 56

SQL Views Overview

SQL views facilitate the use of more complex functionality for single tables and table pairs. An SQL view allows you to use several tables and several calculations in a query to produce a set of fields that you can use as a table in a single table or table pair. This functionality is similar to the SQL override in PowerCenter or a view in a relational database. You can use any valid SQL statement to create an SQL view.

SQL View Properties

You can view SQL view properties by selecting an SQL view in either the Navigator or the SQL Views tab and viewing the properties. Most properties come from the values entered in the SQL View Editor. Other properties come from the tests set up for and run on the SQL view.

Edit SQL view properties in the SQL View Editor dialog box when you add or edit an SQL view.

The following table describes the SQL view properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>SQL view description.</td>
</tr>
<tr>
<td>Table Definitions</td>
<td>Tables you use to create the SQL view. If you identify the table with an \</td>
</tr>
<tr>
<td></td>
<td>alias, enter the alias name with the table name. All tables you use in the |</td>
</tr>
<tr>
<td></td>
<td>SQL view must exist in the same database.</td>
</tr>
<tr>
<td>Connection</td>
<td>PowerCenter connection for the tables</td>
</tr>
</tbody>
</table>
### Description

Enter a description so you can identify the SQL view. Data Validation Option displays the description in the Navigator and on the SQL Views tab. The description can include spaces and symbols.

### Table Definitions and Connection

To provide Data Validation Option with the information it needs to create an SQL view, you must specify the tables that the SQL statement is based on and the corresponding database connection. When you provide the tables and connection information, Data Validation Option can access the metadata that is necessary for the view to function correctly.

To add a table, click **Add Table**. The Choose Data Source dialog box opens. This dialog box displays all of the relational tables available in the repositories. You can sort information in this dialog box by clicking the column headers. You can reduce the number of items to select by typing one or more letters of the table, file, or view name in the **Search** field. Select a table and click **Select**. All of the tables you use in an SQL view must exist in the same database.

If you identify the table with an alias in the SQL statement you use to create the view, enter the alias name next to the table name.

When you finish adding tables, select the PowerCenter connection for the tables from the **Connection** list.

### Column Definition

After you specify the tables on which the SQL view is based, you must specify the columns that make up the view. The number of columns you define for the view must match the SQL statement.

To import the columns from the tables you select, click **Populate**. Data Validation Option imports all columns in the tables. Delete the columns that you do not want to use. You can rearrange the columns in the view.

You can also create a column. To do this, open a column field in the **Column and Expression Definition** list and select **Column**. Enter the column name in the SQL View Column Editor dialog box. You must also specify the datatype, precision, and scale for the column. The datatype, precision, and scale information must match the transformation datatype that Informatica uses for the column. For datetime, string, and integer datatypes, the scale must be zero.

### SQL Statement

Enter an SQL statement to retrieve data for the SQL view.

The statement that you enter runs as a query against the database, so it must use valid database syntax. Also, the columns that you enter in the SELECT statement must match the columns in the **Column Definition** list in number, position, and datatype.

---

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Definition</td>
<td>The columns that make up the SQL view. When you click Populate, Data Validation Option imports all columns from the tables you select. You can create, delete, and rearrange columns.</td>
</tr>
<tr>
<td>SQL Statement</td>
<td>SQL statement you run against the database to retrieve data for the SQL view.</td>
</tr>
<tr>
<td>Comment</td>
<td>Allows you to enter information about an SQL view. Data Validation Option displays the comment when you view the SQL view in the Properties area.</td>
</tr>
</tbody>
</table>
To avoid errors when you run tests, test the SQL statement in the database before you paste it into the SQL Statement field. Data Validation Option does not check the SQL statement syntax.

**Comment**

You can associate a comment with the view. Data Validation Option displays the comment when you view the SQL view in the Properties area.

**Adding SQL Views**

To create an SQL view, right-click SQL Views in the Navigator or right-click in the SQL Views tab, and select **Add SQL View**. The SQL View Editor dialog box opens.

**Editing SQL Views**

To edit an SQL view, right-click the SQL view in the Navigator or SQL Views tab, and select **Edit SQL View**. The SQL View Editor dialog box opens.

**Deleting SQL Views**

To delete an SQL view, right-click the SQL view in the Navigator or SQL Views tab, and select **Delete SQL View**. You can also select the SQL view and press the Delete key.

When you delete an SQL view, Data Validation Option deletes all table pairs, single tables, and tests that use the SQL view.
CHAPTER 12

Lookup Views

This chapter includes the following topics:

- Lookup Views Overview, 57
- Lookup View Properties, 58
- Adding Lookup Views, 59
- Editing Lookup Views, 59
- Deleting Lookup Views, 60
- Lookup Views Example, 60
- Joining Flat Files or Heterogeneous Tables using a Lookup View, 61

Lookup Views Overview

Data Validation Option lookup views allow you to test the validity of the lookup logic in your transformation layer.

Lookup views allow you to validate the process of looking up a primary key value in a lookup, or reference, table using a text value from a source, and then storing the lookup table primary key in the target fact table. For example, a product name in the source system might be in a dimension that serves as the lookup table. The data transformation process involves looking up the product name and placing the primary key from the lookup table in the target fact table as a foreign key. You must validate the product name in the source table against the foreign key in the target table.

The following table lists the keys used in the example:

<table>
<thead>
<tr>
<th>Source Table</th>
<th>Lookup Table</th>
<th>Target Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_id</td>
<td>lookup_id</td>
<td>target_id</td>
</tr>
<tr>
<td>product_name</td>
<td>product_name</td>
<td>source_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lookup_id</td>
</tr>
</tbody>
</table>

The source table product name field is found in the lookup table. After the product name is found, the primary key from the lookup table is stored in the target table as a foreign key.

To test the validity of the lookup table foreign key in the target table, complete the following tasks:

1. Create the lookup view. Add the source table and the lookup table to the lookup view. Then create a relationship between the product name in the source and lookup tables.
2. Create a table pair with the lookup view and the table that is the target of the data transformation process. Join the tables on the source table primary key, which is stored in the target table as a foreign key.
3. Create an OUTER_VALUE test that compares the primary key of the lookup table to the lookup ID that is stored as a foreign key in the target table.

The OUTER_VALUE test checks the validity of the lookup table primary key stored in the target table against the contents of the source table. The test also finds any orphans, which are records in the target table that do not match any records in the lookup table.

## Lookup View Properties

Lookup view properties describe the properties of source and lookup tables.

You can view lookup view properties by selecting a lookup view in either the Navigator or the Lookup Views tab and viewing the properties. Most properties come from the values entered in the Lookup View Editor dialog box. Other properties come from the tests set up for and run on the lookup view.

Edit lookup view properties in the Lookup View Editor dialog box when you add or edit a lookup view.

The following table describes the lookup view properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Table</td>
<td>Source table name.</td>
</tr>
<tr>
<td>Source Conn</td>
<td>PowerCenter connection for the source table.</td>
</tr>
<tr>
<td>Override Owner Name</td>
<td>Overrides the schema or owner name for the source table.</td>
</tr>
<tr>
<td>Source Dir</td>
<td>Source file directory if the source table is a flat file. The path is relative to the machine that hosts Informatica Services.</td>
</tr>
<tr>
<td>Source File</td>
<td>File name, including file extension, if the source table is a flat file.</td>
</tr>
<tr>
<td>Lookup Table</td>
<td>Lookup table name.</td>
</tr>
<tr>
<td>Lookup Conn</td>
<td>PowerCenter connection for the lookup table.</td>
</tr>
<tr>
<td>Lookup Source Dir</td>
<td>Source file directory if the lookup table is a flat file. The path is relative to the machine that hosts Informatica Services.</td>
</tr>
<tr>
<td>Lookup Source File</td>
<td>File name, including file extension, if the lookup table is a flat file.</td>
</tr>
<tr>
<td>Description</td>
<td>Lookup view description.</td>
</tr>
<tr>
<td>Source to Lookup Relation</td>
<td>The fields on which the source table and lookup table are joined.</td>
</tr>
</tbody>
</table>

## Selecting Source and Lookup Tables

A lookup view consists of a source table and a lookup table. Use the Browse button and the Select Data Sources dialog box to select the source and lookup table in the same way that you select tables for table pairs.
Selecting Connections
Select the correct connections for the source and lookup tables in the same way that you select connections for table pairs.

Overriding Owner Name
You can override the owner name for the source table, but not for the lookup table. To specify a different owner or schema name for the lookup table, create a connection in the Workflow Manager, and use that connection for the lookup table.

Source Directory and File
If either the source or lookup tables are flat files, specify the source directory and file name plus file extension, in the same way that you specify source directories and file names for table pairs.

Description
Data Validation Option automatically generates the description for a lookup view based on the tables you select. You can change the description.

Source to Lookup Relationship
In the source and lookup tables, select the values you want to look up in the lookup table.

Adding Lookup Views
To create a lookup view, right-click Lookup Views in the Navigator or right-click in the Lookup Views tab, and select Add Lookup View. The Lookup View Editor dialog box opens.

Select the source table and lookup table, and create the lookup relationship between them. That is, select the field to look up in the lookup table.

The lookup view you create includes fields from both the source and the lookup table, joined on the lookup relationship fields. Data Validation Option precedes the source table field names with "S_."

You can use the lookup view to validate data in the target table, where the lookup table primary key is stored as a foreign key.

Editing Lookup Views
To edit a lookup view, right-click the lookup view in the Navigator or SQL Views tab, and select Edit Lookup View. The Lookup View Editor dialog box opens.
Deleting Lookup Views

To delete a lookup view, right-click the lookup view in the Navigator or Lookup Views tab, and select **Delete Lookup View**. You can also select the lookup view and press the Delete key.

When you delete a lookup view, Data Validation Option deletes all table pairs and tests that use the lookup view.

Lookup Views Example

Use a lookup view to test the validity of the foreign key stored in the target, or fact, table and to confirm that there are no orphans.

The following tables display sample data that is typical of data used to build a target table.

**Source Table**

<table>
<thead>
<tr>
<th>ORDER_ID</th>
<th>PRODUCT_NAME</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>iPod</td>
<td>100</td>
</tr>
<tr>
<td>102</td>
<td>Laptop</td>
<td>500</td>
</tr>
<tr>
<td>103</td>
<td>iPod</td>
<td>120</td>
</tr>
</tbody>
</table>

**Product Lookup Table**

<table>
<thead>
<tr>
<th>LKP_PRODUCT_ID</th>
<th>LKP_PRODUCT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>iPod</td>
</tr>
<tr>
<td>22</td>
<td>Laptop</td>
</tr>
</tbody>
</table>

**Target Table**

<table>
<thead>
<tr>
<th>TARGET_ID</th>
<th>ORDER_ID</th>
<th>LKP_PRODUCT_ID</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
<td>21</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>102</td>
<td>22</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>103</td>
<td>21</td>
<td>120</td>
</tr>
</tbody>
</table>

To test the validity of the lookup table foreign key in the target table, perform the following steps:

**Create the lookup view.**

Create a lookup view with the source and lookup tables. The lookup relationship uses the product name fields in both the source and the lookup tables. The fields that are now included in the lookup view are listed below:

- **S_ORDER_ID**
- **S_PRODUCT_NAME**
Joining Flat Files or Heterogeneous Tables using a Lookup View

One disadvantage of the SQL view is that it does not allow the use of flat files or heterogeneous database tables. You can join two heterogeneous sources with a lookup view. You can think of the source to lookup relationship as an inner join between the two tables or files.
Reports Overview

Data Validation Option stores all test definitions and test results in the Data Validation Option repository. You can run reports to display test definitions and results.

You can generate a report for one or more table pairs or single tables. Right-click the objects for which you want to generate a report, and select Report. The Report Parameters dialog box opens. This dialog box contains options that allow you to generate the report that best fits your needs. Data Validation Option displays reports in a browser window.

Note: Report generation can take several minutes, especially when the report you generate contains hundreds of tests or test runs.

Report Type

Select the report type in the Report Parameters dialog box.

You can select one of the following report types:

Summary of Testing Activities

This report displays the number of table pairs or single tables, the number of tests for each table pair or single table, and the overall test results.

Table Pair Summary

This report lists each table pair or single table with the associated tests. Data Validation Option displays each table pair or single table on a separate page. The report includes a brief description of each test and result.
Detailed Test Results

This is the most detailed report. It displays each test on a separate page with a detailed description of the test definition and results. If one of the test sources is a SQL or a lookup view, the report also displays the view definition.

Related Topics:
- “Data Validation Option Report Examples” on page 82

Other Report Options

After you select the report type, select the other report options in the Report Parameters dialog box. These options allow you to filter test results. For example, you might want to generate a report for all failed tests associated with a table pair so that you can troubleshoot test problems.

The following table describes the other report options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>User that created the tests. By default, Data Validation Option generates a report for the tests that the current user creates and runs. Select All to display tests created and run by all users. Select a user name to display tests created and run by that user.</td>
</tr>
<tr>
<td>Table Pair</td>
<td>Table pairs or single tables for which you want to run a report. You can generate a report on all table pairs and single tables, on the table pairs and single tables in a folder, or on a test. Data Validation Option gives you different options depending on what you select in the Navigator or details area. For example, if you right-click a folder in the Navigator and select Report, you can generate a report on all table pairs and single tables or on the table pairs and single tables in the folder. If you right-click a test and select Report, you can generate a report on all table pairs and single tables or on the selected test.</td>
</tr>
<tr>
<td>Recency</td>
<td>Test runs for which you want to run a report. You can select the latest test runs or all test runs.</td>
</tr>
<tr>
<td>Result Type</td>
<td>Test results for which you want to run a report. You can select all results, tests that pass, or tests that do not pass.</td>
</tr>
<tr>
<td>Run Dates</td>
<td>The test run dates. Enter the from date, to date, or both in the format MM/DD/YYYY. To generate a report for all days, leave the Run Dates fields empty.</td>
</tr>
<tr>
<td>Report Subtitle</td>
<td>Data Validation Option displays the subtitle on each page of the report.</td>
</tr>
</tbody>
</table>

Viewing Reports

Data Validation Option displays reports in a browser window.

Use the arrows in the upper right corner to scroll through the pages of the report. To display a specific page, enter the page number in the Go To Page field.

You can display or hide the table of contents. To do this, click the table of contents icon in the upper left corner of the report. When you display the table of contents, you can click a heading to display that section of the report.

You can also print a report or export it to a PDF file. Click the print icon in the upper left corner of the report.
SQL and Lookup View Definitions

If a test contains either an SQL view or a lookup view as a source, Data Validation Option prints the view definition as part of the report.

You cannot print a report showing the definition of the view by itself because each view is tied to a specific test definition. For example, you create an SQL view, use it in a table pair, and run a test. You then update the view by changing the SQL statement, and re-run the test. Each test result is based on a different view definition, which is why the view definition must be tied to a specific result.

Custom Reports

You can write custom reports against the results database.

All Data Validation Option reports run against database views that are set up as part of the installation process. You can write custom reports against these views. Do not write reports against the underlying database tables because the Data Validation Option repository metadata can change between versions.

RELATED TOPICS:

- "Description of Views Used in Reporting Overview" on page 87
Metadata Export and Import Overview

Data Validation Option allows you to export and import test metadata. Metadata import and export allows users to share tests and allows rapid generation of tests through scripting.

Scripting is particularly useful in the following scenarios:

- You have a very large number of repetitive tests for different table pairs. In this situation, it might be faster to generate the tests programmatically.
- The source-to-target relationships and rules are defined in a spreadsheet. This often happens during data migration. You can script the actual Data Validation Option tests from the spreadsheets.

Related Topics:
- “Metadata Import Syntax” on page 94

Exporting Metadata

Data Validation Option allows you to export selected objects, such as table pairs or SQL views, and all of their dependencies, such as tables, to an XML file. You can also export all objects to an XML file.

To export an object, right-click the object and select Export Metadata. To export all objects, select File > Export All Metadata. Data Validation Option prompts you for the metadata export file name and directory.
Importing Metadata

When you import metadata from an XML file, you can overwrite repository objects such as table pairs and views that have the same type and same name as objects you are importing. To overwrite repository objects when you import, select File > Import Metadata (Overwrite). To import metadata without overwriting repository objects, select File > Import Metadata. When you import metadata without overwriting objects, Data Validation Option stops importing metadata if an object in the import file and an object in the repository are of the same type and have the same name.

When you import metadata, you can automatically generate value tests as you do when you right-click a table pair and select Generate Value Tests. To do this, use the generate-tests command in the import XML file. Place the command at the end of the metadata definition for the table pair.

For example, to generate value tests for a table pair named “CustDetail_CustStage,” add the following lines to the import XML file at the end of the table pair metadata definition:

```xml
<Commands>
    generate-tests("CustDetail_CustStage");
</Commands>
```

Exporting and Importing Metadata at the Command Line

You can use the Data Validation Option command line utility to export and import metadata.

To export and import metadata at the command line, use the following commands:

- `DVOCmd ExportMetadata file_name [--confdir conf_dir]`
- `DVOCmd ImportMetadata file_name [--confdir conf_dir] [--Overwrite]

**RELATED TOPICS:**

- “Command Line Integration” on page 67
Command Line Integration

This chapter includes the following topics:

- Command Line Integration Overview, 67
- CopyFolder Command, 68
- CreateUserConfig Command, 68
- ExportMetadata Command, 69
- ImportMetadata Command, 70
- InstallTests Command, 70
- PurgeRuns Command, 71
- RefreshRepository Command, 71
- RunTests Command, 72
- UpgradeRepository Command, 73

Command Line Integration Overview

You can invoke certain Data Validation Option capabilities at the command line. For example, you can create and run tests without using the Data Validation Option Client.

Running tests at the command line allows you to schedule test execution. It also allows you to embed a specific test as part of the ETL workflow or as part of another process. For example, you can create an ETL process that moves data from source to staging, runs validation, and then moves data into the target or an error table based on the validation results.

The command line utility writes regular messages to the STDOUT output stream. It writes error messages to the STDERR output stream. Normally, command utility messages appear on the screen. To capture messages to a file, use the redirection operator.

The Data Validation Option command line utility is `DVOcmd.exe`. `DVOcmd.exe` exists in one of the following directories:

- 32-bit operating systems: `C:\Program Files\Informatica\<version>\DVO`
- 64-bit operating systems: `C:\Program Files (x86)\Informatica\<version>\DVO`

It uses the following syntax:

`DVOcmd Command [Argument] [--Options] [Arguments]`

**Note:** In the syntax descriptions, options and arguments enclosed in square brackets are optional.
CopyFolder Command

Copies the contents of a folder in a user workspace to a different folder in the same workspace or to another user workspace. The target folder must be a new folder. The target folder cannot exist in the target workspace.

The CopyFolder command copies the table pairs, single tables, and test cases that exist within the source folder. It does not copy test runs or the external IDs associated with table pairs or single tables.

If the table pairs or single tables in the source folder use an SQL or lookup view, the CopyFolder command copies the SQL or lookup view to the target user workspace unless the workspace contains a view with the same name.

Before Data Validation Option copies a folder, it verifies that all data sources associated with the objects being copied exist in the target workspace. If any data source is missing, Data Validation Option does not copy the folder.

The CopyFolder command uses the following syntax:

```
DVCmd CopyFolder [--confdir conf_dir] --fromUser source_user --fromFolder source_folder --toUser target_user [--toFolder target_folder] [--reuseViews]
```

The following table describes CopyFolder options and arguments:

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--confdir</td>
<td>conf_dir</td>
<td>The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.</td>
</tr>
<tr>
<td>--fromUser</td>
<td>source_user</td>
<td>Name of the source user. Data Validation Option copies the folder in this user workspace.</td>
</tr>
<tr>
<td>--fromFolder</td>
<td>source_folder</td>
<td>Name of the source folder. Data Validation Option copies this folder.</td>
</tr>
<tr>
<td>--toUser</td>
<td>target_user</td>
<td>Name of the target user. Data Validation Option copies the folder to this user workspace. The source user and the target user can be the same user.</td>
</tr>
<tr>
<td>--toFolder</td>
<td>target_folder</td>
<td>Name of the target folder. The target folder must be unique in the target workspace. If you do not specify a target folder, Data Validation Option creates a folder in the target workspace with the same name as the source folder.</td>
</tr>
<tr>
<td>--reuseViews</td>
<td>n/a</td>
<td>Reuses an SQL or lookup view in the target workspace when the workspace contains a view with the same name as a source SQL or lookup view. If you specify this option, Data Validation Option does not copy the source view to the target workspace. If you do not specify this option, Data Validation Option copies the source view to the target workspace, but it renames the view in the target workspace.</td>
</tr>
</tbody>
</table>

CreateUserConfig Command

Creates Data Validation Option users with the specified user names.

The CreateUserConfig command uses the following syntax:

```
DVCmd CreateUserConfig user_name [, user_name , ...] [--confdir conf_dir] --outputdir output_dir [--overwrite]
```

Creates a preferences file called <username>-preferences.xml for each user in the output directory. The preferences file contains connection information for the Data Validation Option repository. Copy each preferences
file from the output directory to the user configuration directory and rename it to preferences.xml. This allows each user to access the Data Validation Option repository.

The following table describes CreateUserConfig options and arguments:

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>user_name</td>
<td>The name of the Data Validation Option user. To create multiple users, enter multiple user names separated by commas.</td>
</tr>
<tr>
<td>--confdir</td>
<td>conf_dir</td>
<td>The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.</td>
</tr>
<tr>
<td>--outputdir</td>
<td>output_dir</td>
<td>Directory in which to store user preferences files. Because Windows directories often contain spaces, you must enclose the file path in quotes.</td>
</tr>
<tr>
<td>--overwrite</td>
<td>n/a</td>
<td>Overwrites the configuration files.</td>
</tr>
</tbody>
</table>

**RELATED TOPICS:**
- “Installation Instructions – Additional Users” on page 13

---

**ExportMetadata Command**

Exports all Data Validation Option metadata to an XML file.

The ExportMetadata command uses the following syntax:

```
DVOCmd ExportMetadata file_name [--confdir conf_dir]
```

The following table describes ExportMetadata options and arguments:

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>file_name</td>
<td>The file to which you want to export metadata.</td>
</tr>
<tr>
<td>--confdir</td>
<td>conf_dir</td>
<td>The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.</td>
</tr>
</tbody>
</table>
**Related Topics:**
- "Metadata Export and Import" on page 65

### ImportMetadata Command

Imports metadata into Data Validation Option from an export XML file.

The ImportMetadata command uses the following syntax:

```
DVOcmd ImportMetadata file_name [--confdir conf_dir] [--overwrite]
```

The following table describes ImportMetadata options and arguments:

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>file_name</td>
<td>The file that contains metadata to be imported.</td>
</tr>
<tr>
<td>--confdir</td>
<td>conf_dir</td>
<td>The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.</td>
</tr>
<tr>
<td>--overwrite</td>
<td>n/a</td>
<td>Overwrites existing objects.</td>
</tr>
</tbody>
</table>

**Related Topics:**
- "Metadata Export and Import" on page 65

### InstallTests Command

Prepares all tests for a single table or table pair. For each test, this command generates the PowerCenter mapping in the Data Validation Option target folder in the PowerCenter repository.

The InstallTests command uses the following syntax:

```
DVOcmd InstallTests external_ID [, external_ID, ..] [--confdir conf_dir]
```

The following table describes InstallTests options and arguments:

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>external_ID</td>
<td>The external ID for the single table or table pair.</td>
</tr>
<tr>
<td>--confdir</td>
<td>conf_dir</td>
<td>The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.</td>
</tr>
</tbody>
</table>
PurgeRuns Command

Purges test runs from the Data Validation Option repository. You can purge deleted test runs or purge test runs by date. When you purge test runs by date, you can purge all test runs that occur on or after a specified date, before a specified date, or between two dates.

The PurgeRuns command uses the following syntax:

DVOcmd PurgeRuns [--confdir conf_dir] [--deleted] [--fromdate from_date] [--todate to_date]

The following table describes PurgeRuns options and arguments:

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--confdir</td>
<td>conf_dir</td>
<td>The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.</td>
</tr>
<tr>
<td>--deleted</td>
<td>n/a</td>
<td>Purges deleted test runs.</td>
</tr>
<tr>
<td>--fromdate</td>
<td>from_date</td>
<td>Purges test runs that occur on or after this date.</td>
</tr>
<tr>
<td>--todate</td>
<td>to_date</td>
<td>Purges test runs that occur before this date.</td>
</tr>
</tbody>
</table>

RefreshRepository Command

Refreshes a source or target repository.

The RefreshRepository command uses the following syntax:


Tip: Use the --folderlist and --folder options to get the sources and targets in a new PowerCenter repository folder. For example, if the repository name is "DVTgtRepo" and the new folder name is “NewOrders,” enter the following command:

DVOcmd RefreshRepository DVTgtRepo --folderlist --folder NewOrders

The following table describes RefreshRepository options and arguments:

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>repo_name</td>
<td>Name of the repository you want to refresh. Note: This can take several minutes to several hours depending on the size of the repository.</td>
</tr>
<tr>
<td>--confdir</td>
<td>conf_dir</td>
<td>The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option.</td>
</tr>
</tbody>
</table>
Because Windows directories often contain spaces, you must enclose the file path in quotes.

---

### Related Topics:
- “Repositories Overview” on page 24

---

## RunTests Command

Runs all tests for a single table or table pair.

For example, in a typical client installation of Data Validation Option, to run tests for a table pair with the external ID "abc123," you might enter the following command:

```
DVOCmd RunTests abc123
```

If one test fails, the overall result also fails. The exit status code for a successful test is 0. A non-zero status code designates a failed test or an error.

The RunTests command uses the following syntax:

```
DVOCmd RunTests external_ID [, external_ID, ...] [--confdir conf_dir]
```

The following table describes RunTests options and arguments:

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>external_ID</td>
<td>The external ID for the single table or table pair.</td>
</tr>
<tr>
<td>--confdir</td>
<td>conf_dir</td>
<td>The user configuration directory. Specify the configuration directory if you have multiple Data Validation Option repositories on a client machine. If you have one Data Validation Option repository on a client machine and have not changed the configuration directory, you do not need to specify this option. Because Windows directories often contain spaces, you must enclose the file path in quotes.</td>
</tr>
</tbody>
</table>
UpgradeRepository Command

Upgrades the Data Validation Option repository. Use this command when you upgrade from a previous version of Data Validation Option.

The UpgradeRepository command uses the following syntax:

```
DVOcmd UpgradeRepository
```

The UpgradeRepository command takes no options or arguments.

**RELATED TOPICS:**

- “Data Validation Option Installation and Setup” on page 8
Troubleshooting

This chapter includes the following topics:

- Troubleshooting Overview, 74
- Troubleshooting Initial Errors, 74
- Troubleshooting Ongoing Errors, 75
- Troubleshooting Command Line Errors, 76

Troubleshooting Overview

When you run a test, Data Validation Option performs the following tasks:

1. Creates a mapping in the specified PowerCenter folder.
2. Creates the PowerCenter workflow.
3. Runs the PowerCenter session.

Besides initial installation problems, Data Validation Option errors can occur in one of the following steps:

**Installation Error**
Data Validation Option cannot create a PowerCenter mapping.

**Run Error**
Data Validation Option cannot create or run a PowerCenter workflow.

**No Results**
The PowerCenter session runs but fails, or there are no results in the results database.

Troubleshooting Initial Errors

This section assumes that Data Validation Option has just been installed and no successful tests have been executed. It also assumes that the first test is a simple test that does not contain expressions or SQL views.
The following table describes common initial errors:

<table>
<thead>
<tr>
<th>Error</th>
<th>Possible Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot connect to the Data Validation Option repository</td>
<td>Database credentials are incorrect. Check the server, port, and database name specified in the URL line. If the problem persists, contact your database administrator. If the problem is database username and password, the error message explicitly states this.</td>
</tr>
<tr>
<td>Cannot read the PowerCenter repository</td>
<td>Check the repository settings up to Security Domain. (Informatica Domain names are not used until later.) If you cannot resolve the error, contact the Informatica administrator. Another way to troubleshoot this error is by trying to log into the repository through the pmrep command line utility.</td>
</tr>
<tr>
<td>Installation Error</td>
<td>Verify that the Data Validation Option folder is closed in the Designer, Workflow Manager and the Repository Manager. The Workflow Monitor can be open. Verify that the INFA_HOME environment variable is set. Verify that the Data Validation Option folder exists.</td>
</tr>
<tr>
<td>Run Error</td>
<td>Verify that the Data Validation Option folder is closed. Check the Informatica Domain name and Integration Service names. Verify that they are running. Verify that the PowerCenter connection name (connection to the Data Validation Option repository) is correct, and that the user has the privilege to use it. Open the session log and look for session errors. Most session failures are caused by an incorrect connection. If the error is “Cannot get object class for dvo/infact/PivotPluginImpl,” the dvoct.jar file cannot be read either because it is not on the server, because of its privileges, or because the information entered in the Administrator tool is incorrect. Verify that the user has the privilege to use the connection to the Data Validation Option repository specified in the Tools &gt; Preferences &gt; Data Validation Option database. This will also be apparent in the session log.</td>
</tr>
<tr>
<td>No Results</td>
<td>Verify that there is data in the data set you are analyzing. Tables should have records, and filters and joins should not result in an empty set. Verify that the connection to the Data Validation Option repository specified in the Workflow Manager points to the Data Validation Option repository.</td>
</tr>
</tbody>
</table>

**Troubleshooting Ongoing Errors**

This section assumes that successful tests have been created and run before the error occurred.

In general, you should always check for the following sources of errors:

- Incorrect connection
- The Data Validation Option folder is open in the Designer, Workflow Manager, or Repository Manager

The following table describes common ongoing errors:

<table>
<thead>
<tr>
<th>Error</th>
<th>Possible Cause and Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation or run errors</td>
<td>Verify that the Data Validation Option folder is closed in the Designer, Workflow Manager and Repository Manager. Verify that the PowerCenter environment is functioning correctly, for example, services are running and the repository is up. If the error occurred right after you created an expression either in a test editor dialog box or as a WHERE clause, check the syntax.</td>
</tr>
<tr>
<td>Error</td>
<td>Possible Cause and Solution</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>No results</td>
<td>Verify that there is data in the data set you are analyzing. Tables should have records. Filters or joins should not result in an empty set.</td>
</tr>
<tr>
<td>Inability to generate reports</td>
<td>Verify that you have read and write permissions on the Data Validation Option installation directory and subdirectories.</td>
</tr>
<tr>
<td>Inability to copy folders</td>
<td>Verify that the repository, data sources, and folders that contain the data sources have identical names in the source and the target workspaces. Object names in Data Validation Option are case sensitive. Verify that all data sources associated with the objects to copy exist in the target workspace in the same location and that the names match.</td>
</tr>
</tbody>
</table>

**Troubleshooting Command Line Errors**

I ran a DVOCmd command that got an error and used the redirection operator to write the messages to a file. The redirection operator does not redirect all messages to the output file.

When you run a DVOCmd command, the command line utility writes regular messages to the STDOUT output stream and writes error messages to the STDERR output stream. You use the redirection operator to write messages to an output file. To merge messages from both output streams, enter "2>&1" after the output file name.

For example, you encounter an error while refreshing folder "MyFolder" in Data Validation Option repository "DVORepo." To write all messages, including the error messages, to a text file called "Log.txt," use the following command:

```
DVOCmd RefreshRepository DVORepo --folder MyFolder > C:\Log.txt 2>&1
```

To append messages to an existing log file called "DVOLog.txt," use the following command:

```
DVOCmd RefreshRepository DVORepo --folder MyFolder >> C:\DVOLog.txt 2>&1
```
This chapter includes the following topics:

- Folders, 77
- Menus, 78
- Uninstalling Data Validation Option, 79

**Folders**

Folders store the single tables and table pairs that you create.

By default, Data Validation Option places the single tables and table pairs that you create in the default folder. If you create a folder, you can create single tables or table pairs within the folder. You can move single tables or table pairs between folders. Within a folder, you can expand a single table or table pair to view the tests associated with it. Folder names are case sensitive.

You can also copy folders. You can copy the contents of a folder in your workspace to a different folder in your workspace or to a folder in another user workspace. You must copy folder contents to a new folder. You cannot copy folder contents to a folder that exists in the target workspace.

When you copy a folder, Data Validation Option copies all table pairs, single tables, and test cases in the source folder to the target folder. Data Validation Option does not copy test runs or the external IDs associated with table pairs or single tables.

If a table pair or single table in the source folder uses an SQL view or a lookup view, Data Validation Option copies the view to the target user workspace unless the workspace contains a view with the same name. If the target workspace contains a view with the same name, Data Validation Option gives you the following options:

- You can use the view in the target workspace.
- You can copy the view to the target workspace with another name. Data Validation Option names the view in the target workspace "Copy <number> <source view name>".

Before Data Validation Option copies a folder, it verifies that the repository and all data sources associated with the objects to copy exist in the target workspace. Object names in Data Validation Option are case sensitive. Therefore, the repository, data sources, and folders that contain the data sources must have identical names in the source and the target workspaces. If the repository or any required data source does not exist in the target workspace, Data Validation Option does not copy the folder.

**Copying Folders**

You can copy the contents of a folder in your workspace to a different folder in your workspace or to a folder in another user workspace.
1. Select **Tools > Copy Folder**.
   The **Copy Folder Contents** dialog box opens.
2. Enter the following information:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy from User</td>
<td>Name of the source user. Data Validation Option copies the folder in this user workspace.</td>
</tr>
<tr>
<td>Copy from Folder</td>
<td>Name of the source folder. Data Validation Option copies this folder.</td>
</tr>
<tr>
<td>Copy to User</td>
<td>Name of the target user. Data Validation Option copies the folder to this user workspace. The source user and the target user can be the same user.</td>
</tr>
<tr>
<td>Copy to Folder</td>
<td>Name of the target folder. The target folder must be unique in the target workspace. If you do not specify a target folder, Data Validation Option creates a folder in the target workspace with the same name as the source folder.</td>
</tr>
</tbody>
</table>
3. Click **OK**.

## Menus

The following table describes the Data Validation Option menu items:

<table>
<thead>
<tr>
<th>Menu</th>
<th>Menu Item</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Reload</td>
<td>Reloads repositories, data sources, folders, SQL views, lookup views, and table pairs.</td>
</tr>
<tr>
<td></td>
<td>Import Metadata</td>
<td>Imports scripted test metadata from a text file.</td>
</tr>
<tr>
<td></td>
<td>Import Metadata Overwrite</td>
<td>Imports scripted test metadata from a text file and overwrites existing metadata.</td>
</tr>
<tr>
<td></td>
<td>Export All Metadata</td>
<td>Exports all metadata to a file.</td>
</tr>
<tr>
<td>Exit</td>
<td></td>
<td>Closes the Informatica Data Validation Option Client.</td>
</tr>
<tr>
<td>Tools</td>
<td>Copy Folder</td>
<td>Copies the contents of a folder in one user's workspace to a folder in a target user's workspace.</td>
</tr>
<tr>
<td></td>
<td>Show Console</td>
<td>Opens the Console window to help you diagnose problems with tests.</td>
</tr>
<tr>
<td></td>
<td>Open Settings Folder</td>
<td>Opens the Data Validation Option folder in the &quot;Documents and Settings&quot; directory in Windows Explorer.</td>
</tr>
<tr>
<td></td>
<td>Preferences</td>
<td>Opens the Preferences dialog box, which contains setup information for the results database and settings for the mappings generated by Data Validation Option.</td>
</tr>
<tr>
<td></td>
<td>Report</td>
<td>Allows you to view reports. Data Validation Option displays reports in a browser window.</td>
</tr>
</tbody>
</table>
Menu | Menu Item | Definition
--- | --- | ---
Help | Help | Opens Data Validation Option online help.
About | Displays information about PowerCenter Data Validation Option. Click the dialog box to close it.

**Console**

The Console window displays Data Validation Option status messages to help you diagnose problems that occur. To display the Console window, select **Tools > Show Console**.

**Settings Folder**

When you select **Tools > Open Settings Folder**, Windows Explorer displays the contents of the Data Validation Option folder in the Documents and Settings directory for that installation of the application. The data folder also contains an XML file that contains the information entered in the **Preferences** dialog box.

**Preferences Dialog Box**

The Preferences dialog box appears when you select **Tools > Preferences**.

The Data Validation Option properties are discussed in detail in step 9 in “Installation Instructions – First User” on page 10.

The following table describes the mapping properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Details</td>
<td>Maximum number of detail, or bad, records to be displayed in the detail area of the Tests tab, and written to the results database. Default is 100. <strong>Note:</strong> Performance decreases somewhat when you increase the maximum number of detail records generated and stored in the results database from the default value.</td>
</tr>
<tr>
<td>DTM Buffer Size</td>
<td>Adjusts the amount of memory available to PowerCenter to run tests/mappings. If the session log indicates that a table pair requires additional memory to run the set of tests created, adjust the DTM buffer size here. Do not adjust it in the session in the Workflow Manager.</td>
</tr>
<tr>
<td>Max Concurrent Runs</td>
<td>Maximum number of PowerCenter sessions executed at the same time. Each table pair or single table runs as one session, regardless of how many tests are associated with it. Default is 10.</td>
</tr>
</tbody>
</table>

**Uninstalling Data Validation Option**

You can uninstall Data Validation Option through the uninstaller in the Windows **Start** menu.

The uninstallation process deletes the **DVO** directory within **C:\Program Files\ (x86)\Informatica\version**. It does not delete the Data Validation Option user configuration directory. Therefore, all repositories, views, table pairs, single tables, and tests that you set up in previous versions of Data Validation Option are present when you reinstall.
Datatype Reference

This appendix includes the following topics:
- Test, Operator, and Datatypes Matrix for Table Pair Tests, 80
- Test, Operator, and Datatypes Matrix for Single-Table Constraints, 81

Test, Operator, and Datatypes Matrix for Table Pair Tests

Table pair tests use the following datatypes:
- s = string datatypes
- n = numeric datatypes
- d = date/time datatypes
- b = binary/other datatypes

The following table describes the operators and datatypes for table pair tests:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operators Allowed</th>
<th>Datatypes Allowed: Approx. Operator</th>
<th>Datatypes Allowed: All Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>All</td>
<td>s,n,d,b</td>
<td>s,n,d,b</td>
</tr>
<tr>
<td>COUNT_DISTINCT</td>
<td>All</td>
<td>s,n,d</td>
<td>s,n,d</td>
</tr>
<tr>
<td>COUNT_ROWS</td>
<td>All</td>
<td>s,n,d,b</td>
<td>s,n,d,b</td>
</tr>
<tr>
<td>MIN</td>
<td>All</td>
<td>n</td>
<td>s,n,d</td>
</tr>
<tr>
<td>MAX</td>
<td>All</td>
<td>n</td>
<td>s,n,d</td>
</tr>
<tr>
<td>AVG</td>
<td>All</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>SUM</td>
<td>All</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>SET_AinB</td>
<td>--</td>
<td>s,n,d</td>
<td>s,n,d</td>
</tr>
<tr>
<td>SET_BinA</td>
<td>--</td>
<td>s,n,d</td>
<td>s,n,d</td>
</tr>
<tr>
<td>Operators Allowed</td>
<td>Datatypes Allowed: Approx. Operator</td>
<td>Datatypes Allowed: All Other</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>SET_AeqB</td>
<td>--</td>
<td>s,n,d</td>
<td></td>
</tr>
<tr>
<td>VALUE</td>
<td>All</td>
<td>n</td>
<td>s,n,d</td>
</tr>
<tr>
<td>OUTER_VALUE</td>
<td>All</td>
<td>n</td>
<td>s,n,d</td>
</tr>
</tbody>
</table>

**Note:** SET tests do not use operators and allow string, numeric and date/time datatypes.

Test, Operator, and Datatypes Matrix for Single-Table Constraints

Single-table constraints use the following datatypes:

- **s** = string datatypes
- **n** = numeric datatypes
- **d** = date/time datatypes
- **b** = binary/other datatypes

The following table describes the operators and datatypes for single-table constraints:

<table>
<thead>
<tr>
<th>Operators Allowed</th>
<th>Datatypes Allowed</th>
<th>Result Expression Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>All</td>
<td>s,n,d,b</td>
</tr>
<tr>
<td>COUNT_DISTINCT</td>
<td>All</td>
<td>s,n,d</td>
</tr>
<tr>
<td>COUNT_ROWS</td>
<td>All</td>
<td>s,n,d,b</td>
</tr>
<tr>
<td>MIN</td>
<td>All</td>
<td>s,n,d</td>
</tr>
<tr>
<td>MAX</td>
<td>All</td>
<td>s,n,d</td>
</tr>
<tr>
<td>AVG</td>
<td>All</td>
<td>n</td>
</tr>
<tr>
<td>SUM</td>
<td>All</td>
<td>n</td>
</tr>
<tr>
<td>VALUE</td>
<td>All</td>
<td>s,n,d</td>
</tr>
<tr>
<td>FORMAT</td>
<td>=, &lt;&gt;</td>
<td>s,n,d</td>
</tr>
<tr>
<td>UNIQUE</td>
<td>--</td>
<td>s,n,d</td>
</tr>
<tr>
<td>NOT_NULL</td>
<td>--</td>
<td>s,n,d</td>
</tr>
<tr>
<td>NOT_BLANK</td>
<td>--</td>
<td>s,n,d</td>
</tr>
</tbody>
</table>
Appendix B

Data Validation Option Report Examples

This appendix includes the following topics:

- Summary of Testing Activities, 82
- Table Pair Summary, 83
- Detailed Test Results – Test Page, 85
- Detailed Test Results – Bad Records Page, 86

Summary of Testing Activities

The following figure shows an example of a Summary of Testing Activities report:
### Table Pair Summary

The following figure shows an example of a Table Pair Summary report:
### Table Pair Summary

#### Table Pair Definition

**User:** lori  
**Folder Name:** Lori_TablePairsTests  
**Table Pair:** Joined EMPLOYEES-T_EMPLOYEES

**Table A:** PC_Target_Repo/Orders_LT/Sources/Sources/EMPLOYEES  
**Connection:** lori_target  
**Where Clause:** Owner Name:  
**Optimize in DB:** Disabled  
**Is Large Table:** No

**Table B:** PC_Target_Repo/Orders_LT/Targets/T_EMPLOYEES  
**Connection:** lori_target  
**Where Clause:** Owner Name:  
**Optimize in DB:** Disabled  
**Is Large Table:** No

**Join:** EMPLOYEE_ID = EMPLOYEE_ID  
**External ID:**  
**Run Date:** Apr 26, 2011 2:11 PM PDT  
**Latest Result:** Yes  
**Result:** Pass

#### Test Description

**OUTER VALUE EMPLOYEE_ID = EMPLOYEE_ID**  
**VALUE FIRST_NAME = FIRST_NAME**  
**VALUE LAST_NAME = LAST_NAME**  
**VALUE OFFICE_PHONE = OFFICE_PHONE**

#### Test Result

<table>
<thead>
<tr>
<th>Test Description</th>
<th>Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass</td>
<td>4</td>
</tr>
<tr>
<td>Fail</td>
<td>0</td>
</tr>
<tr>
<td>Error</td>
<td>0</td>
</tr>
<tr>
<td>Total Run</td>
<td>4</td>
</tr>
</tbody>
</table>

#### Pass %

<table>
<thead>
<tr>
<th>Users:Current: TablePair=Selected, Recency=Latest Tests Only; Results=All Results; Run Dates: All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report run on: Apr 26, 2011 2:12 PM Run By: lori</td>
</tr>
<tr>
<td>Pass %: 100%</td>
</tr>
</tbody>
</table>

Page 1 of 1
### Detailed Test Results – Test Page

The following figure shows an example of a test page in the Detailed Test Results report:

**Detailed Test Results**

![Detailed Test Results](image)

<table>
<thead>
<tr>
<th>Test</th>
<th>Folder Name:</th>
<th>Lori_SingleTableTests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Table</td>
<td>CUSTOMERS</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>VALUE STATE = 'CA'</td>
<td></td>
</tr>
<tr>
<td>Run Date:</td>
<td>Apr 26, 2011 2:27 PM</td>
<td></td>
</tr>
<tr>
<td>Test Result:</td>
<td>FAIL</td>
<td></td>
</tr>
</tbody>
</table>

**Test Result Detail**

- **Value**: 8
- **Max Bad Records**: 0

**Test Definition**

- **Test**: VALUE STATE = 'CA'
- **Condition**:  
- **Case Inensitive**: No
- **Trim Trailing**: No
- **Null = Null**: No

**Single Table Definition**

- **Table**: PC_Target_Repo/Orders_LT/Sources/Sources/CUSTOMERS
- **Owner Name**:  
- **Connection**: DQMsources
- **Where Clause**: Disabled
- **Is Large Table**: No
- **Primary Key**: CUSTOMER_ID
- **External ID**:  
- **Last Modified**: Apr 26, 2011 2:27 PM PDT

**Runtime Information**

- **Session Name**: DV_S_CUSTOMERS_8
- **Mapping Name**: DV_M_CUSTOMERS_8
- **Workflow Name**: DV_MF_CUSTOMERS_8

---

_See next page for bad records..._
Detailed Test Results – Bad Records Page

The following figure shows an example of a bad records page in the Detailed Test Results report:

![Detailed Test Results - Bad Records Page](image)

This page displays the results of data validation checks, with a table listing records identified as bad. Each record includes a key and a value, indicating the state of the data. The table is used to highlight records that need further review or correction to ensure data integrity.
Description of Views Used in Reporting Overview

All Data Validation Option reports run against database views that are set up as part of the installation process. You can write custom reports against these views. Do not write reports against the underlying database tables because the Data Validation Option repository metadata can change between versions.

results_summary_view

This view combines all table pair and test metadata and all test results. This view consists of the following general sections:

- tp_. Table pair metadata
- tc_. Test metadata
- tc_rs_. Test results
- tr_, ti_. Table pair runtime information
The following table describes table pair information:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tp_user_id</td>
<td>User ID of the person who ran this test</td>
</tr>
<tr>
<td>tp_username</td>
<td>User name of the person who ran this test</td>
</tr>
<tr>
<td>tp_obj_id</td>
<td>Unique table pair ID</td>
</tr>
<tr>
<td>tp_version</td>
<td>Table pair version</td>
</tr>
<tr>
<td>tp_name</td>
<td>Table pair description</td>
</tr>
<tr>
<td>tp_time_stamp</td>
<td>Time table pair was last edited</td>
</tr>
<tr>
<td>tp_comments</td>
<td>Table pair comments</td>
</tr>
<tr>
<td>tp_description</td>
<td>Table pair description. Same as tp_name.</td>
</tr>
<tr>
<td>tp_table_a,</td>
<td>Either full table name, including the PowerCenter repository directory or view name</td>
</tr>
<tr>
<td>tp_table_b</td>
<td></td>
</tr>
<tr>
<td>tp_table_version_a</td>
<td>Version name for SQL and lookup views, otherwise empty</td>
</tr>
<tr>
<td>tp_table_version_b</td>
<td></td>
</tr>
<tr>
<td>tp_type_a,</td>
<td>A and B source type: 1 = relational, 2 = flat file, 3 = SQL view, 4 = lookup view</td>
</tr>
<tr>
<td>tp_type_b</td>
<td></td>
</tr>
<tr>
<td>tp_conn_name_a,</td>
<td>A and B connection names</td>
</tr>
<tr>
<td>tp_conn_name_b</td>
<td></td>
</tr>
<tr>
<td>tp_owner_name_a,</td>
<td>A and B owner names</td>
</tr>
<tr>
<td>tp_owner_name_b</td>
<td></td>
</tr>
<tr>
<td>tp_src_dir_a,</td>
<td>A and B directory and file names if flat files</td>
</tr>
<tr>
<td>tp_src_file_a,</td>
<td></td>
</tr>
<tr>
<td>tp_src_dir_b,</td>
<td></td>
</tr>
<tr>
<td>tp_src_file_b</td>
<td></td>
</tr>
<tr>
<td>tp_in_db_a,</td>
<td>Run aggregations in database for A or B</td>
</tr>
<tr>
<td>tp_in_db_b</td>
<td></td>
</tr>
<tr>
<td>tp_where_clause_a,</td>
<td>A and B WHERE clauses</td>
</tr>
<tr>
<td>tp_where_clause_b</td>
<td></td>
</tr>
<tr>
<td>tp_is_where_clause_dsq_a,</td>
<td>Run WHERE clause in database for A or B</td>
</tr>
<tr>
<td>tp_is_where_clause_dsq_b</td>
<td></td>
</tr>
<tr>
<td>tp_join_list_str</td>
<td>Joined fields as one string</td>
</tr>
<tr>
<td>tp_external_id</td>
<td>Table pair external ID</td>
</tr>
</tbody>
</table>
The following table describes test information:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tc_index</td>
<td>Internal ID</td>
</tr>
<tr>
<td>tc_description</td>
<td>Test description</td>
</tr>
<tr>
<td>tc_comment</td>
<td>Test comment</td>
</tr>
<tr>
<td>tc_type</td>
<td>AGG if aggregate, otherwise equal to test (VALUE, OUTER_VALUE, etc.)</td>
</tr>
<tr>
<td>tc_agg_func</td>
<td>Aggregate test if AGG (COUNT, SUM, etc.), otherwise blank</td>
</tr>
<tr>
<td>tc_column_a</td>
<td>Field A</td>
</tr>
<tr>
<td>tc_operator</td>
<td>Operator</td>
</tr>
<tr>
<td>tc_column_b</td>
<td>Field B</td>
</tr>
<tr>
<td>tc_tables_type</td>
<td>0 = two-table pair, 1 = one-table constraint</td>
</tr>
<tr>
<td>tc_threshold</td>
<td>Threshold</td>
</tr>
<tr>
<td>tc_max_bad_records</td>
<td>Maximum bad records</td>
</tr>
<tr>
<td>tc_is_case_insensitive</td>
<td>Case insensitive checkbox: 0 = false, 1 = true</td>
</tr>
<tr>
<td>tc_is_treat_nulls_equal</td>
<td>Null = Null, 0 = false, 1 = true</td>
</tr>
<tr>
<td>tc_is_trim_right_ws</td>
<td>Trim trailing spaces: 0 = false, 1 = true</td>
</tr>
<tr>
<td>tc_expression_a</td>
<td>Expression A</td>
</tr>
<tr>
<td>tc_expression_b</td>
<td>Expression B</td>
</tr>
</tbody>
</table>

The following table describes test results information:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tc_rs_result</td>
<td>Test Result: 1 = pass, 0 = fail; -1 = no_results, -2 = error</td>
</tr>
<tr>
<td>tc_rs_failure_count</td>
<td>Number of bad records</td>
</tr>
<tr>
<td>tc_rs_agg_value_a, tc_rs_agg_value_b</td>
<td>Aggregate results A and B</td>
</tr>
</tbody>
</table>

The following table describes table pair runtime information:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr_id</td>
<td>Unique internal run ID incremented for each run</td>
</tr>
<tr>
<td>tr_state</td>
<td>Result state. 2 = install_error, 4 = run_success, 5 = run_error</td>
</tr>
</tbody>
</table>
### Metadata

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr_start_time</td>
<td>Table pair run time start, in milliseconds, since 1970 UTC</td>
</tr>
<tr>
<td>tr_finish_time</td>
<td>Run time finish</td>
</tr>
<tr>
<td>tr_is_latest</td>
<td>Whether this is the latest run of a given table pair: 1 = latest; 0 = not latest</td>
</tr>
<tr>
<td>ti_id</td>
<td>Internal ID. Do not use.</td>
</tr>
<tr>
<td>ti_folder_name</td>
<td>PowerCenter folder name where mapping was created</td>
</tr>
<tr>
<td>ti_mapping_name</td>
<td>PowerCenter mapping name</td>
</tr>
<tr>
<td>ti_session_name</td>
<td>PowerCenter session name</td>
</tr>
<tr>
<td>ti_workflow_name</td>
<td>PowerCenter workflow name</td>
</tr>
</tbody>
</table>

#### rs_bad_records_view

This view is a detailed view of all bad records. It can be joined with `results_summary_view` on `tr_id` and `tc_index`.

The following table describes bad records information:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr_id</td>
<td>Run ID, joined to tr_id in results_summary_view</td>
</tr>
<tr>
<td>tc_index</td>
<td>Joined to tc_index in results_summary_view</td>
</tr>
<tr>
<td>tc_rs_br_key_a</td>
<td>Key A</td>
</tr>
<tr>
<td>tc_rs_br_value_a</td>
<td>Value A</td>
</tr>
<tr>
<td>tc_rs_br_key_b</td>
<td>Key B</td>
</tr>
<tr>
<td>tc_rs_br_value_b</td>
<td>Value B</td>
</tr>
</tbody>
</table>

#### results_id_view

This view uses `results_summary_view` as the source and aggregates it on the table pair level. This view contains the table pair result.
The following table describes results ID information:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr_id</td>
<td>Internal run ID incremented for each run</td>
</tr>
<tr>
<td>tr_is_latest</td>
<td>Whether this is the latest run of a given table pair: 1 = latest; 0 = not latest</td>
</tr>
<tr>
<td>tr_start_time</td>
<td>Table pair run time start, in milliseconds, since 1970 UTC</td>
</tr>
<tr>
<td>ti_id</td>
<td>Internal ID</td>
</tr>
<tr>
<td>tp_obj_id</td>
<td>Table pair ID</td>
</tr>
<tr>
<td>tp_version</td>
<td>Table pair version</td>
</tr>
<tr>
<td>tp_user_id</td>
<td>User ID of the person who ran this test</td>
</tr>
<tr>
<td>tp_rs_result</td>
<td>Table pair result: 1 = pass, 0 = fail, -1 = no results, -2 = error</td>
</tr>
</tbody>
</table>

**meta_sv_view**

This view returns all SQL view information.

The following table describes SQL view information:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sv_id</td>
<td>Unique ID</td>
</tr>
<tr>
<td>sv_name</td>
<td>Internal name</td>
</tr>
<tr>
<td>sv_obj_id</td>
<td>View object ID</td>
</tr>
<tr>
<td>sv_version</td>
<td>View version</td>
</tr>
<tr>
<td>sv_description</td>
<td>Description</td>
</tr>
<tr>
<td>sv_comment</td>
<td>Comment</td>
</tr>
<tr>
<td>sv_owner_name</td>
<td>Owner name</td>
</tr>
<tr>
<td>sv_conn_name</td>
<td>Connection</td>
</tr>
<tr>
<td>sv_dsnname</td>
<td>Table name</td>
</tr>
<tr>
<td>sv_sql_query</td>
<td>SQL statement</td>
</tr>
<tr>
<td>svf_name</td>
<td>Column name</td>
</tr>
<tr>
<td>svf_business_name</td>
<td>Not used</td>
</tr>
</tbody>
</table>
meta_lv_view

This view returns all lookup view information.

The following table describes lookup view information:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lv_id</td>
<td>Unique ID</td>
</tr>
<tr>
<td>lv_name</td>
<td>Internal name</td>
</tr>
<tr>
<td>lv_obj_id</td>
<td>View object ID</td>
</tr>
<tr>
<td>lv_version</td>
<td>View version</td>
</tr>
<tr>
<td>lv_tp_name</td>
<td>Table pair description</td>
</tr>
<tr>
<td>lv_time_stamp</td>
<td>Time lookup view was last edited</td>
</tr>
<tr>
<td>lv_comments</td>
<td>Comments</td>
</tr>
<tr>
<td>lv_description</td>
<td>Lookup view description</td>
</tr>
<tr>
<td>lv_table_a</td>
<td>Source table name</td>
</tr>
<tr>
<td>lv_table_b</td>
<td>Lookup table name</td>
</tr>
<tr>
<td>lv_type_a,</td>
<td>Source and lookup types: 1 = relational, 2 = flat file</td>
</tr>
<tr>
<td>lv_type_b</td>
<td></td>
</tr>
<tr>
<td>lv_conn_name_a,</td>
<td>Source and lookup connections</td>
</tr>
<tr>
<td>lv_conn_name_b</td>
<td></td>
</tr>
<tr>
<td>lv_owner_name_a,</td>
<td>Source and lookup owner names</td>
</tr>
<tr>
<td>lv_owner_name_b</td>
<td></td>
</tr>
<tr>
<td>lv_src_dir_a,</td>
<td>Source and lookup directory and file names if flat files</td>
</tr>
<tr>
<td>lv_src_file_a,</td>
<td></td>
</tr>
<tr>
<td>lv_src_dir_b,</td>
<td></td>
</tr>
<tr>
<td>lv_src_file_b</td>
<td></td>
</tr>
<tr>
<td>Metadata</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>lv_where_clause_a, lv_where_clause_b</td>
<td>Not used</td>
</tr>
<tr>
<td>lv_is_where_clause_dsq_a, lv_is_where_clause_dsq_b</td>
<td>Not used</td>
</tr>
<tr>
<td>lv_join_list_str</td>
<td>Lookup relationship (join) as a string description</td>
</tr>
</tbody>
</table>
APPENDIX D

Metadata Import Syntax

This appendix includes the following topics:
- Metadata Import Syntax Overview, 94
- Table Pair with One Test, 94
- Table Pair with an SQL View as a Source, 95
- Table Pair with Two Flat Files, 95
- Single-Table Constraint, 96
- SQL View, 96
- Lookup View, 97

Metadata Import Syntax Overview

The following sections display examples of metadata syntax definition.

Table Pair with One Test

```xml
<TablePair>
  Name = "CLIDETAIL_CLISTAGE"
  Description = "CLIDETAIL-CLISTAGE"
  ExternalID = "cliTest"
  <TableA>
    Name = "pc_repository/cli_demo/Sources/demo_connection/cliDetail"
    Connection = "dvo_demo_connection"
    WhereClause = "" -
    WhereClauseDSQ = false
    InDS = false
  </TableA>
  <TableB>
    Name = "pc_repository/cli_demo/Targets/cliStage"
    Connection = "dvo_demo_connection"
    WhereClause = "" -
    WhereClauseDSQ = false
    InDS = false
  </TableB>
  <TestCase>
    TestType = "AGG"
    Aggregate = "SUM"
    ColumnA = "ProductAmount"
    ColumnB = "CustomerAmount"
    Operator = "="
    Comments = ""
    CaseInsensitive = false
  </TestCase>
</TablePair>
```
Table Pair with an SQL View as a Source

```xml
<TablePair>
  Name = "Joined_MVIEW_FACTORDERS"
  Description = "Joined_MVIEW_FACTORDERS"
  ExternalID = ""
  <TableA>
    Name = "SQLView_470"
    WhereClause = ""
    WhereClauseDSQ = false
    InDB = false
  </TableA>
  <TableB>
    Name = "pc_repository/dvo_demo/Targets/factOrders"
    Connection = "dvo_demo_connection"
    WhereClause = ""
    WhereClauseDSQ = false
    InDB = false
  </Join>
  <TestCase>
    ColumnA = "MyID"
    ColumnB = "LineID"
    ColumnA = "MyCurrency"
    ColumnB = "CurrencyName"
  </TestCase>
  TestType = "VALUE"
  ColumnA = "MyCurrency"
  ColumnB = "CurrencyName"
  Operator = "="
  Comments = ""
  CaseInsensitive = false
  TrimRightWhitespace = true
  TreatNullsEqual = true
  <TestCase>
    TestType = "AGG"
    Aggregate = "SUM"
    <ExpressionA>
      Expression = "if(MyID>10,10,MyID)"
      Datatype = "integer"
      Precision = 10
      Scale = 0
      ColumnA = "LineID"
      Operator = "="
      Comments = ""
      CaseInsensitive = false
      TrimRightWhitespace = false
      TreatNullsEqual = true
    </ExpressionA>
    <ExpressionB>
    </ExpressionB>
  </TestCase>
</TablePair>
```

Table Pair with Two Flat Files

```xml
<TablePair>
  Name = "FLATFILE_FLATFILE"
  Description = "FLATFILE-FLATFILE"
  ExternalID = ""
  <TableA>
    Name = "pc_repository/dvo_demo/Sources/FlatFile/FlatFile"
    SourceDirectory = "C:\FlatFile\Sources"
    SourceFilename = "flatfile.txt"
    WhereClause = ""
    WhereClauseDSQ = false
    InDB = false
  </TableA>
  <TableB>
  </TableB>
</TablePair>
```
Single-Table Constraint

```xml
<vector>
    <Table>
        <Key>
            <ColumnA>EmployeeID</ColumnA>
        </Key>
        <TestCases>
            <TestCase>
                <Type>AGG</Type>
                <Aggregate>COUNT</Aggregate>
                <FunctionB/>
                <Expression>100,200</Expression>
                <Parameter>10</Parameter>
                <Operator>Between</Operator>
                <CaseInsensitive>false</CaseInsensitive>
                <CaseInsensitiveSpace>false</CaseInsensitiveSpace>
                <TreatNullsEqual>true</TreatNullsEqual>
            </TestCase>
            <TestCase>
                <Type>NOT NULL</Type>
                <ColumnA>LastName</ColumnA>
                <Operator>"="</Operator>
                <CaseInsensitive>false</CaseInsensitive>
                <CaseInsensitiveSpace>false</CaseInsensitiveSpace>
                <TreatNullsEqual>true</TreatNullsEqual>
            </TestCase>
        </TestCases>
    </Table>
</vector>
```

SQL View

```xml
<vector>
    <Table>
        <SQLView>
            <Name>SQL View 991</Name>
            <Description>MyView991</Description>
            <Columns>
                <ColumnA>* from srcOrders</ColumnA>
            </Columns>
            <Connection>dvo_demo_connection</Connection>
        </SQLView>
    </Table>
</vector>
```
Comments = "This is a comment"
<Columns>
  <Column>
    Name = "MyID"
    Datatype = "int"
    Precision = 10
    Scale = 0
  </Column>
  <Column>
    Name = "MyCurrency"
    Datatype = "varchar"
    Precision = 30
    Scale = 0
  </Column>
  <Column>
    Name = "MyAmount"
    Datatype = "decimal"
    Precision = 10
    Scale = 2
</Columns>

Lookup View

<LookupView>
  Name = "LookupView"
  Description = "Lookup srcOrders --> dimProducts"
  <SourceTable>
    Name = "pc_repository/dvo_demo/Sources/demo_connection/srcOrders"
    Connection = "dvo_demo_connection"
  </SourceTable>
  <LookupTable>
    Name = "pc_repository/dvo_demo/Targets/dimProducts"
    Connection = "dvo_demo_connection"
  </LookupTable>
  <Join>
    ColumnA = "ProductID"
    ColumnB = "ProductID"
    ColumnA = "ProductName"
    ColumnB = "ProductName"
  </Join>
</LookupView>
INDEX

A
architecture
  Data Validation Option 7

B
bad records
  single-table constraints 53
  table pair tests 43
  behavior changes
    Data Validation Option 17

c
client layout
  Data Validation Option 19
  Lookup Views tab 22
  Single Tables tab 21
  SQL Views tab 22
  Table Pairs tab 21
  Tests tab 20
  Console window
    displaying 79
  CopyFolder command
    syntax 68
  CreateUserConfig command
    syntax 68

D
data validation
  definition 1
  purpose 5
  sample testing methodology 2
  testing approach 1
  typical workflow 6
  Data Validation Option
    architecture 7
    assumptions 6
    behavior changes 17
    client layout 19
    Console window 79
    installation for additional users 13
    installation for first user 10
    installation overview 8
    installation prerequisites 8
    installation required information 9
    mapping properties 79
    menus 78
    new features 16
    overview 5
    preferences 79
  required system permissions 9
  sample testing methodology 2
  Settings folder 79
  system requirements 7
  uninstalling 79
  upgrading from version 3.0 13
  upgrading from version 3.1 14
  users 6
datatypes
  single table tests 81
  table pair tests 80

DVOCmd
  CopyFolder command 68
  CreateUserConfig command 68
  ExportMetadata command 69
  ImportMetadata command 70
  InstallTests command 70
  location 67
  overview 67
  PurgeRuns command 71
  RefreshRepository command 71
  RunTests command 72
  syntax 67
  troubleshooting 76
  UpgradeRepository command 73

E
ExportMetadata command
  syntax 69
  external ID
    table pairs 32

F
folders
  comparing 43
  copying 78
  copying at the command line 68
  overview 77
  refreshing 26
  restrictions on copying 77
  Settings folder 79
  troubleshooting copying folders 75

I
ImportMetadata command
  syntax 70
  installation
    instructions for additional users 13
    instructions for first user 10
    overview 8
repository connections 74
run errors 74, 75

U
UpgradeRepository command
syntax 73
upgrading
  version 3.0 to current version 13
  version 3.1 to current version 14
user configuration directory
  changing with a batch file 15
  changing with an environment variable 15
  creating at the command line 68
  location 14

W
WHERE clause
  table pairs 31