

HP Test Data Management

Software version: 1.10

Designer User Guide

Document release date: October 2010
Software release date: October 2010



Legal notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted rights legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Licensing

The use of HP products is governed by the terms and conditions of the applicable End User License Agreement (EULA).

Copyright notices

© Copyright 2010 Hewlett-Packard Development Company, L.P.

Trademark notices

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft®, Windows®, Windows NT®, Windows XP®, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

Contents

| | |
|---|----|
| About this document | 7 |
| Intended audience | 7 |
| Prerequisites | 7 |
| New and revised information | 7 |
| Related documentation | 8 |
| Document conventions and symbols | 9 |
| Documentation updates | 10 |
| Subscription service | 10 |
| Support | 10 |
| 1 Introduction to test data management | 13 |
| Before you begin | 14 |
| Designer overview | 14 |
| 2 Working with models | 19 |
| Before you begin | 20 |
| About models | 20 |
| Creating a new model | 20 |
| Adding transactional tables | 22 |
| Adding lookup tables | 30 |
| Adding chaining tables | 37 |
| Managing associated tables | 43 |
| Working with rules | 43 |
| Working with virtual unique and foreign keys | 47 |
| Modifying object properties | 52 |
| Locating a table | 53 |
| Drag and drop a table | 53 |
| 3 Working with parameters | 55 |
| About parameters | 55 |
| Creating parameters | 56 |
| Validating parameters | 60 |
| Creating lists of values for parameters | 61 |
| Referencing parameters | 63 |
| Managing model compatibility | 67 |
| Deleting parameters | 67 |

| | | |
|---|---|-----|
| 4 | Previewing models and cartridges | 69 |
| | About preview | 69 |
| | Creating preview rules | 70 |
| | Previewing models and cartridges | 70 |
| 5 | Working with cartridges | 73 |
| | About cartridges | 73 |
| | Creating a new cartridge | 73 |
| | Editing a database to file cartridge | 74 |
| | Editing a schema-based database to file cartridge | 80 |
| | Applying data masks | 85 |
| | Deleting cartridges | 99 |
| 6 | Working with data masking | 101 |
| | About data masking | 101 |
| | Applying pre-built data masks | 102 |
| | Applying string map masks to data | 103 |
| | Applying numeric map masks to data | 106 |
| | Applying string and numeric map masks in non-intrusive environments | 108 |
| | Creating custom data masks | 110 |
| | Unmasking data | 114 |
| | Deploying and running | 115 |
| 7 | Working with projects and connections | 117 |
| | About projects | 117 |
| | Creating a new project | 118 |
| | Creating a connection | 119 |
| | Changing projects | 121 |
| | Annotating projects | 122 |
| | Exporting and importing projects | 124 |
| | Changing the project location | 126 |
| | Mapping schema names | 126 |
| 8 | Working with business flows | 131 |
| | About business flows | 131 |
| | Creating a business flow | 132 |
| | Adding a database to file cartridge | 135 |
| | Adding schema-based cartridges | 136 |
| | Splitting an activity | 136 |
| | Uploading to a test database | 137 |
| | Adding or editing a script | 138 |
| | Adding or editing a condition | 141 |
| | Adding or editing an interrupt | 142 |
| | Testing business flows | 143 |

| | |
|--|------------|
| Deleting a business flow | 143 |
| 9 Generating and deploying | 145 |
| Deploying locally | 145 |
| Deploying remotely | 149 |
| Generating deployment files | 150 |
| 10 Customizing projects | 155 |
| About customization | 155 |
| Guidelines for customizations | 156 |
| Customizing a project | 157 |
| Merging customizations into a new project revision | 158 |
| Using the JDBC test console | 159 |
| Running JDBC driver validation tests | 170 |
| Running the JDBC DDL/SQL configuration utility | 171 |
| | 173 |
| HP_LOOKUP_NUMBER | 176 |
| HP_LOOKUP_STRING | 176 |
| GENERATE_ABA_ROUTING_NUMBER | 176 |
| GENERATE_CREDIT_CARD_NUMBER | 177 |
| GENERATE_SSN | 177 |
| GET_CREDIT_CARD_TYPE | 178 |
| MASK_ABA_ROUTING_NUMBER | 178 |
| MASK_CREDIT_CARD_NUMBER | 179 |
| MASK_SSN | 180 |
| MASK_STRING | 181 |
| RANDOM_NUMBER | 182 |
| RANDOM_STRING | 182 |
| REVERT_HP_LOOKUP_NUMBER | 183 |
| REVERT_HP_LOOKUP_STRING | 183 |
| REVERT_SKEW_DATE | 184 |
| REVERT_SKEW_NUMBER | 184 |
| SKEW_DATE | 185 |
| SKEW_NUMBER | 185 |
| SKEW_PERCENT | 186 |
| VALID_ABA_ROUTING_NUMBER | 186 |
| VALID_CREDIT_CARD_NUMBER | 186 |
| LookupNumber | 189 |
| LookupString | 190 |
| MaskABANumber | 190 |
| MaskCreditCardNumber | 191 |
| MaskSSN | 192 |
| MaskString | 193 |

| | |
|---------------------------|-----|
| RandomInt | 194 |
| RandomString | 195 |
| REVERT_LookupNumber | 195 |
| REVERT_LookupString | 196 |
| REVERT_SkewDate | 196 |
| REVERT_SkewInteger | 197 |
| SkewDate | 197 |
| SkewFloat | 198 |
| SkewInteger | 198 |
| SkewPercent | 199 |
| Glossary | 201 |
| Index | 207 |

About this document

HP Test Data Management provides powerful tools to design a test data management solution that copies data out of your production database for upload into a test database.

This guide provides information about:

- working with projects, models, and rules
- using parameters
- previewing test data files
- working with cartridges and business flows
- generating and deploying cartridges and business flows
- customizing projects

Intended audience

This guide is intended for:

- Test data developers building custom projects

Prerequisites

Prerequisites for using this product include:

- Knowledge of the operating system
- Database knowledge
- Application knowledge

New and revised information

This document includes the following new and revised features for HP Test Data Management:

- DB2 support
- Generic JDBC drivers

- Upload activity for business flows
- Non-intrusive environments
- Groovy-based masking for non-intrusive environments
- Data movement key for selection and data movement for Oracle
- JDBC driver validation and configuration testing
- JDBC connection configuration utility
- SQL and Java APIs for custom masking

See the *HP Test Data Management Installation Guide* for more details about new and revised features for this release.

Related documentation

In addition to this guide, please refer to other documents for this product:

- *HP Test Data Management Installation Guide*
Explains how to use the Installer to install the product.
- *HP Test Data Management Concepts Guide*
Explains the major concepts of test data management in general and HP Test Data Management in particular.
- *HP Test Data Management Tutorial*
Provides step-by-step instructions to build a sample project with business flows, deploy them, run them, and troubleshoot errors.
- *HP Test Data Management Web Console and Query Server User Guide*
Explains how to use the Web Console component to run, monitor, and administer the business flows that copy data to and from the database. This guide also explains how to install, configure, and use the query server to access data that has been extracted from the database.
- *HP Test Data Management Troubleshooting Guide*
Explains how to diagnose and resolve errors, and provides a list of common errors and solutions.
- *HP Test Data Management Release Notes*
Lists any items of importance that were not captured in the regular documentation.

The latest documentation for the most recent HP Test Data Management release can be found on:

<http://support.openview.hp.com/selfsolve/manuals>

Document conventions and symbols

| Convention | Element |
|--|---|
| [] | Indicates that the enclosed element is optional and may be left out. |
| { } | Indicates that you must specify one of the listed options. |
| | Separates alternatives. |
| <parameter_name> | You must supply a value for a variable parameter. |
| ... | <ul style="list-style-type: none"> Indicates a repetition of the preceding parameter. Example continues after omitted lines. |
| Medium blue text: Figure 1 | Cross-reference links and e-mail addresses |
| Medium blue, underlined text (http://www.hp.com) | Web site addresses |
| Bold | <ul style="list-style-type: none"> Key names GUI elements that are clicked or selected, such as menu and list items, buttons, and check boxes |
| <i>Italics</i> | Text emphasis |
| Monospace | <ul style="list-style-type: none"> File and directory names Text displayed on the screen, such as system output and application messages Code syntax |
| <i>Monospace, italic</i> | You must supply a value. <ul style="list-style-type: none"> Code variables Command-line variables |

△ **CAUTION** Indicates that failure to follow directions could result in damage to equipment or loss of data.

NOTE Provides additional information.

TIP Provides helpful hints and shortcuts.

RECOMMENDATION Provides guidance from HP for a best practice or for optimum performance.

Documentation updates

For documentation for all versions of HP Test Data Management, you can go to:

<http://support.openview.hp.com/selfsolve/manuals>

NOTE This documentation is written to the latest patch version. If you have not installed the latest patch, there may be items in this documentation that do not apply to your environment.

Subscription service

HP strongly recommends that customers sign up online using the Subscriber's choice web site:

<http://www.hp.com/go/e-updates>

- Subscribing to this service provides you with e-mail updates on the latest product enhancements, versions of drivers, and firmware documentation updates as well as instant access to numerous other product resources.
- After signing up, you can quickly locate your products under Product Category.

Support

You can visit the HP Software Support web site at:

<http://www.hp.com/go/hpsoftwaresupport>

HP Software Support Online provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services

- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels and register for HP Passport, go to:

http://support.openview.hp.com/new_access_levels.jsp

Introduction to test data management

HP Test Data Management provides powerful tools to design, deploy, and implement solutions that extract data from your production databases for the purposes of populating test databases. Using test data management, you typically copy a subset of data from your production database to a structured data file, such as an XML or comma separated values (CSV) file. Structured files provide open, standards-based formats that can be accessed using standard database mechanisms.

As part of the subsetting process, you perform the following:

- **Apply eligibility requirements to the data.** In most cases, you do not want all of the data from your production database. You need to define criteria that restrict the records copied. For example, you might only copy records that pertain to a certain period of time or customer.
- **Mask sensitive data.** Typically, your test database is not as secure as your production database. Hence, you need to mask any sensitive data, such as names, addresses, phone numbers, social security numbers, and so on. HP Test Data Management provides a variety of masking capabilities to help you protect confidential data.

Copying qualified records out of your active database and masking them is only the first leg in the lifecycle of test data management. Once you have the desired subset of data in a data extract file, you can upload from the file to a compatible test database. HP Test Data Management enables you to upload to heterogeneous databases, if necessary. For example, you may have extracted from a SQL Server database but need to upload to an Oracle database. You can also implement schema mapping in order to change the schema name when you load the data in the test database.

In addition to uploading the data from the extract file, you may also want to create a spreadsheet with some portion of the test data for use with other quality management tools. HP Test Data Management's query server enables you to quickly populate spreadsheets with test data from one or many database tables. Because you are using the extract file, the data you load into the spreadsheet is masked to your specifications and you can join tables, if necessary, without re-querying your source database.

This chapter includes:

- [Before you begin](#) (page 14)
- [Designer overview](#) (page 14)

See also

For a complete conceptual introduction to HP Test Data Management, refer to *HP Test Data Management Concepts Guide*.

Before you begin

Before you begin performing the tasks in this guide, you should:

- 1 Review the *HP Test Data Management Concepts Guide* to become familiar with the software and how you plan to use it.
- 2 Install HP Test Data Management according to the instructions in the *HP Test Data Management Installation Guide*.
- 3 Go through *HP Test Data Management Tutorial*. The tutorial enables you to get hands on with the product quickly and exposes you to many of the most commonly used features.

Designer overview

Most of the work of developing a test data management solution is performed in Designer. Designer is a powerful graphical development environment used to:

- Model data
- Apply rules
- Design cartridges
- Design business flows that employ cartridges and implement additional logic
- Preview models and cartridges for testing purposes
- Deploy cartridges and business flows

Designer drastically improves the productivity of test data developers. Developers no longer need to spend hours writing and debugging complex SQL. They simply drag and drop in Designer's editor to include tables and relate them to other tables. Once the model is defined, developers can point and click to define rules and policies on tables as desired. When the business flows are ready, the developer can deploy them to any supported environment, local or remote, from Designer.

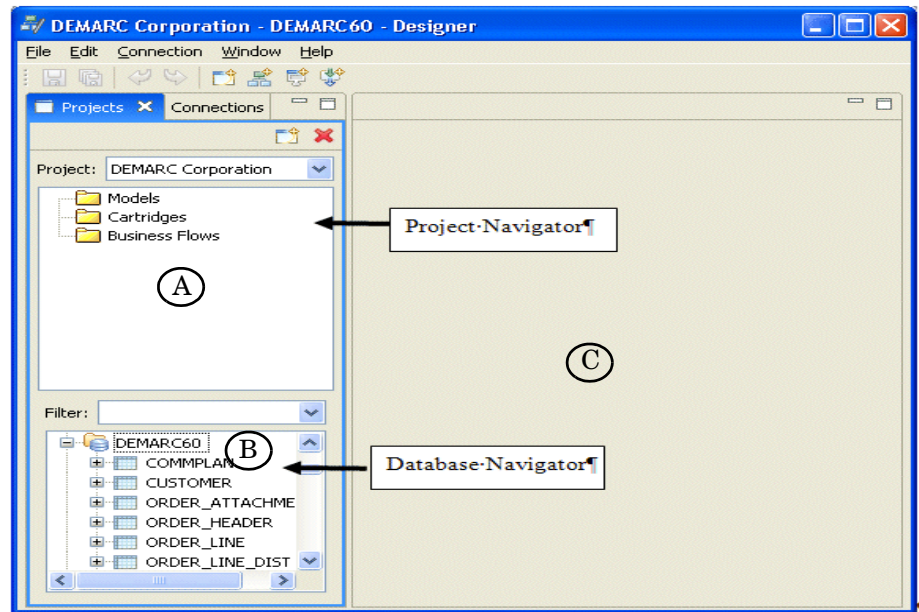
Navigation and interaction with the various components within Designer is consistent, and once familiar with the patterns in one portion of Designer, you should be able to work with any part of Designer.

This section includes:

- [Designer main window](#) (page 14)
- [Toolbar buttons](#) (page 15)
- [Projects and Connections](#) (page 16)
- [Preview](#) (page 17)

Designer main window

The main window is the primary window you see and interact with while using Designer. The following figure shows an example of the main window and descriptions of its editors.














| Legend | Name | Description |
|--------|--------------------|---|
| A | Project Navigator | Shows the files contained within the project |
| B | Database Navigator | Shows the content of the database or local cache that the project is connected to |
| C | Editor | Workspace used to define model, cartridge, and business flow components, to view preview data, and to view database table data. |

Toolbar buttons

This section details the main toolbar buttons used in Designer. While using the tool, you may see different toolbar buttons. The purpose of this section is to introduce you to the most common toolbar buttons.

| Icon | Name | Description |
|------|----------|---|
| | Save | Save the active window. |
| | Save All | Save work in all currently open windows and tabs. |
| | Undo | Undo the last change. |
| | Redo | Redo the last undone action. |

| Icon | Name | Description |
|---|--------------------------|--|
|  | Create New Project | Allows you to start creating a new project |
|  | Create New Model | Allows you to start creating a new model |
|  | Create New Cartridge | Allows you to create a new cartridge, if you have Models created |
|  | Create New Business Flow | Allows you to create a new business flow, if you have Cartridges created |
|  | Custom View | Toggles custom view mode on or off. Custom view provides visual cues for locked projects that alert you to the customizations you are making and whether they are supported. |
|  | Add Chaining Table | Allows you to add a chaining table to a model |
|  | Add Transactional Table | Allows you to add a transactional table to a model |
|  | Add Lookup Table | Allows you to add a lookup table to a model |
|  | Add Rule | Allows you to add a rule to a model table |
|  | Preview | Allows you to access the Designer's preview functionality. |
|  | Refresh | Refreshes the data in preview from the database. |

Projects and Connections

All work to build a cartridge in Designer is performed within the context of a project. A finished project contains all the metadata definitions needed by the test data management software to copy your data.

Projects help you organize your work for a specific goal.

You can define multiple connections, to databases and to local caches. Each project is associated with one connection.

Preview

Within Designer, you can preview the behavior of your model, rules, and cartridges. Preview provides a quick way to test your test data solution as you are building it. You need not wait until you deploy and run your cartridge to see how it will work. The best practice is to preview your solution at each stage of development and preview it repeatedly until you are certain it performs as expected.

For example, in [Figure 1](#) rows excluded by the rule "Product Not Recalled" appear in red, while rows that are eligible for copy and subsequent testing are shown in black. You can preview your model before you create any rules and then preview it again after you add each rule. Preview identifies the rows that are excluded as well as which rule causes their exclusion. When you create your cartridge, you can preview its behavior as well.

Figure 1 Preview window in Designer

| # | Excluded By | ORDERID | CUSTOMERID | ORDERDATE |
|----|----------------------|---------|------------|-----------------------|
| 59 | Product Not Recalled | 2797 | 529 | 1995-11-21 00:00:00.0 |
| 60 | Product Not Recalled | 2818 | 550 | 1995-08-01 00:00:00.0 |
| 61 | Product Not Recalled | 2891 | 56 | 1995-02-20 00:00:00.0 |
| 62 | Product Not Recalled | 216 | 216 | 1994-06-13 00:00:00.0 |
| 63 | | 1 | 1 | 1994-08-18 00:00:00.0 |
| 64 | | 2 | 2 | 1994-08-27 00:00:00.0 |
| 65 | | 3 | 3 | 1994-04-20 00:00:00.0 |
| 66 | | 4 | 4 | 1994-02-13 00:00:00.0 |
| 67 | | 5 | 5 | 1994-08-28 00:00:00.0 |
| 68 | | 6 | 6 | 1994-05-16 00:00:00.0 |
| 69 | | 7 | 7 | 1994-04-21 00:00:00.0 |
| 70 | | 8 | 8 | 1994-01-15 00:00:00.0 |
| 71 | | 9 | 9 | 1994-07-13 00:00:00.0 |

| # | Excluded By | ORDERID | CUSTOMERID | ORDERDATE |
|---|-------------------|---------|------------|-----------|
| 1 | Product Not Re... | 216 | 216 | 1994-0 |
| 2 | | 2 | 2 | 1994-0 |
| 3 | | 1 | 1 | 1994-0 |

Within HP Test Data Management, you can identify the data to be extracted in one of two ways:

- Selecting tables in a schema. This method of identifying data is described in [Chapter 5, Working with cartridges](#).
- Building one or more models that represents the tables and data relationships.

A model represents the tables from which you want to copy data and their relationships. In this context, the term table actually refers to a use of a table, synonym, or view in the model. A table use can represent a table, view, or a synonym to a table or view. The same table can appear multiple times in a model. For example, a table could be appear as a transactional table and a lookup table in the same model.

You apply rules to your tables to:

- Set the scope of extraction, such as only extract orders older than two years.
- Exclude certain records (such as excluding orders that are not open).

In building your model, you use Designer to discover database definitions and add them to your model. However, before you begin creating your model, you must have a good understanding of your business entity and table. You can also use Designer to validate rule definitions added against the active database.

This chapter includes:

- [Before you begin](#) (page 20)
- [About models](#) (page 20)
- [Creating a new model](#) (page 20)
- [Adding transactional tables](#) (page 22)
- [Adding lookup tables](#) (page 30)
- [Adding chaining tables](#) (page 37)
- [Managing associated tables](#) (page 43)
- [Working with rules](#) (page 43)
- [Working with virtual unique and foreign keys](#) (page 47)
- [Modifying object properties](#) (page 52)
- [Locating a table](#) (page 53)
- [Drag and drop a table](#) (page 53)

Before you begin

Before building your model, you need to perform some careful analysis and planning. For some ideas about what you must know and take into account before building your test data solution, refer to *HP Test Data Management Concepts Guide*.

TIP Designing your models is much easier if you already have an entity-relationship diagram and know the cardinality of each role in the relationships.

About models

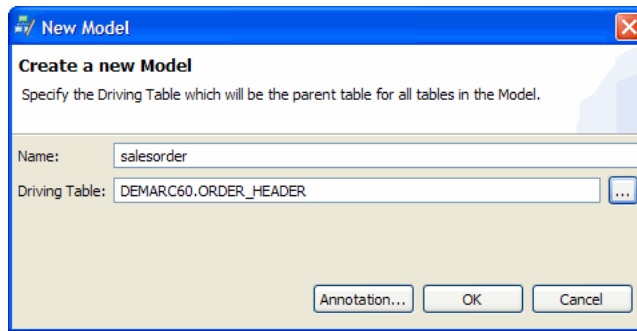
A model represents the tables and relationships that define transactions as test data candidates. A model captures the logic of the application whereas a cartridge based upon the model defines a specific usage of the model. You can define as many rules as needed on any model object (driving, transaction, lookup, or chaining) to further limit the universe of eligible rows for extraction.

Rules articulate business conditions, often parameterized, that determine what data to include in the XML or CSV files. For example, you can create a rule that includes billing transactions that are at least two years old. To further refine the selection of records, you can create rules that define exceptions to the business conditions. For example, you can create a rule to exclude all orders that have not been shipped regardless of their age.

Creating a new model

To create a new model:

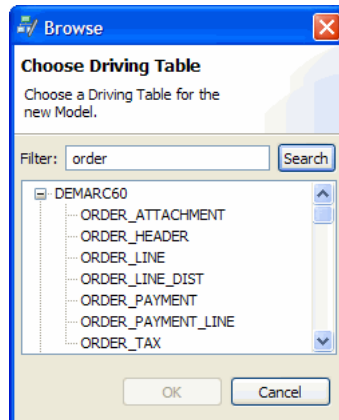
- 1 Select **Models** in the Project Navigator.
- 2 Right-click and select **New Model**. The New Model dialog opens.
- 3 Type a name for the model, for example, `salesorder`.
- 4 If known, type the fully-qualified name of the Driving Table. If not, explore the database to which you are connected and find the table:



- a Click the browse button at the end of the field.
- b Type a string or a partial string in the Filter field, such as `order`.

TIP In the case of Oracle, a lowercase filter criteria returns uppercase, matching names.

- c Click **Search**.



- d Highlight the table you want as the driving table. If Designer can find a primary key or a unique key without nullable columns to act as a data movement key, then you are not prompted further. Otherwise, you may be prompted to add a virtual unique key.

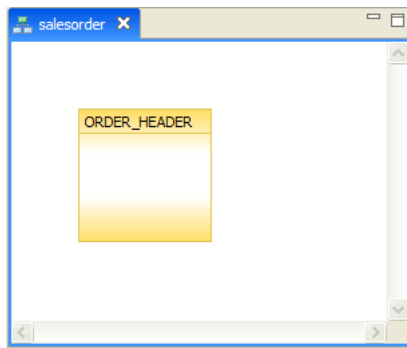
Synonyms and views

In some cases, you may need to copy data from views or synonyms instead of the base tables. For example, if you need to copy data from a table that resides in another database, you can create a synonym in the local database that refers to that remote table. When you use synonyms and views, bear in mind the following:

- Multi-table views are supported by Oracle; however, only modifiable views (also known as *modifiable join views*), which are views that contain more than one table in the top-level FROM clause statement and are not restricted by the WITH READ ONLY clause, support upload and copy of data.
- Synonyms must be recognized as local objects. Remote synonyms or synonyms created as though they were remote are not supported.

- e Click **OK** in the Browse window.

- 5 Once the Driving Table field is populated, click **OK**. The new model opens in the Editor.



TIP To delete a model, right click it in the Project Navigator and choose **Delete** from the menu. If you have created cartridges based on the model, Designer will not allow you to delete it until you have first deleted the cartridges.

Adding transactional tables

If the relationship between two tables is one to many (parent to child), you should add the second (child) table as a transactional table. If the relationship is many to one, you would add it as a lookup or chaining table. Refer to [Adding lookup tables](#) (page 30) or [Adding chaining tables](#) (page 37).

NOTE You cannot add a transactional table or a chaining table to a lookup table.

You can add transactional tables in one of two ways:

- If your database has foreign key relationships defined or if you have already created virtual foreign key relationships for the model, you can use them to relate transactional tables in the model.
- If you cannot use existing foreign key relationships, you can add transactional tables through the All Tables tab.

TIP To ensure optimal performance, you should consider creating an index for any columns that you plan to use in a virtual constraint. Otherwise, performance may be slow and you could encounter deadlocks.

NOTE For DB2 only, you cannot add tables that include the ' (apostrophe) character in the name. Attempting to do so will not immediately yield an error, however any cartridge with DB2 tables containing a ' character in the name will not deploy. If you must use a table with the ' character in the name, then you can create a view or synonym as a workaround.

Both of these methods are described in this section:

- [Adding transactional tables using foreign key relationships](#) (page 23)

- [Adding transactional tables using the All Tables tab](#) (page 25)

Synonyms and views

In some cases, you may need to extract views or synonyms instead of the base tables. For example, if you need to extract a table that resides in another database, you can create a synonym in the local database that refers to that remote table. When you use synonyms and views, bear in mind the following:

- Multi-table views are not supported in the case of Oracle. You should instead extract each of the tables in the multi-table view individually.
- Synonyms must be recognized as local objects. Remote synonyms or synonyms created as though they were remote are not supported.

NOTE Synonyms and views are not supported in non-intrusive environments.

Multiple table uses

You can add the same table multiple times to the same model and each instance of the table can fulfill different functions (transactional table, chaining table, or lookup table). When working with multiple table uses, you should be aware of the following points:

- Whenever you have multiple uses of a table and those uses are disjoint (no overlapping rows), you can use standard selection. When the table uses have overlapping rows, standard selection can yield unexpected results. Hence, when rows overlap among multiple table uses, you must use advanced selection.

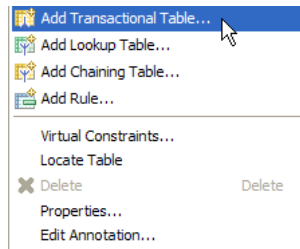
TIP To ensure the set of rows for each table use are unique, you can attach a condition to the relationship that restricts the rows returned. Refer to *HP Test Data Management Tutorial* for an example of multiple table uses where a condition restricts the rows for each usage.

- Multiple table uses can have arbitrarily different operations associated with them. Ultimately, though, each use of the table maps to the same physical table. Hence, a row satisfying two or more table use selections might experience conflicting operations. The data movement process uses the following rule to resolve such operational conflicts:
 - If one table use has no operation and another has any operation, the latter table use's operation is used on a row belonging to both table uses.
- If you plan to use advanced selection, then you are not restricted to using a single driving table (in other words, your model's root table). To designate that a table be used as a driving table, right-click it and select Properties. Click the Use as a Driving Table in Advanced Selection check box in the Property dialog.

Adding transactional tables using foreign key relationships

To add transactional tables using foreign key relationships:

- 1 Open a model in Designer.
- 2 Right click a driving, transactional, or chaining table and select **Add Transactional Table**.

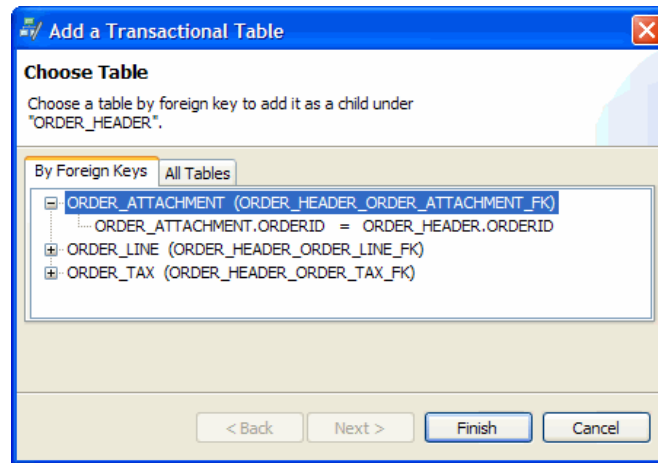


- 3 On the By Foreign Keys tab, select the table you want to add.

For example, **SalesOrderDetail**.

You can expand the details to see the foreign key relationship.

TIP If you select multiple foreign keys, multiple table uses are added to the model when you click **Finish**.



- 4 Click **Next**.

Designer requires a data movement key for all table uses in a model. You can specify an existing unique key or create a new one to act as the data movement key.

Additionally, for Oracle only, you can define the data movement key (via the Data Movement page) as follows:

- Create a new data movement key
- Create a row ID to be used in lieu of a data movement key
- Use an existing unique key as the data movement key.

NOTE You must ensure that, in the context of the data being extracted, the data movement key contains no null column values. Otherwise, the rows associated with the null values will not be included in your extracted data.

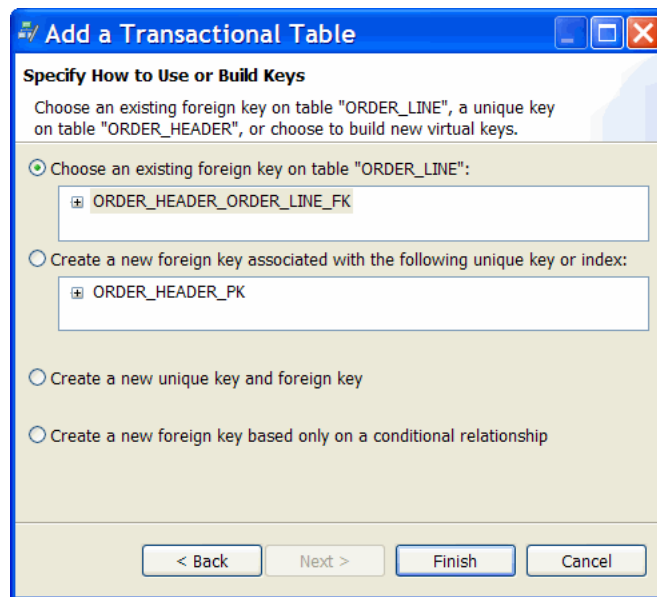
- 5 Click **Finish**.

Adding transactional tables using the All Tables tab

To add transactional tables using the All Tables tab:

- 1 Open a model in Designer.
- 2 Right click the driving table and select **Add Transactional Table**.
- 3 Select the **All Tables** tab.
- 4 Optionally, type a string or a partial string in the Filter field.
- 5 Click **Search** and select the table you want.
- 6 Click **Next**.

The Add a Transactional Table dialog opens and prompts you to Specify How to Use or Build Keys. If the child table has an available foreign key (virtual or database) related to the parent or the parent table has an available unique key (virtual or database), the page displays those keys for your selection. If the child has no such foreign key, the topmost radio button does not appear. If the parent has no unique key, the page appears with only the two bottom radio buttons.



7 Choose one of the following radio buttons:

Table 1 How to use or build keys: key specification

| Selection | Description | Next steps |
|--|---|---|
| Choose an existing foreign key | If you already have a foreign key or a virtual foreign key, you can choose that existing key. | <ul style="list-style-type: none">• Highlight a foreign key from the list.• For Oracle, you can click Finish. For non-Oracle, click Next and, when satisfied with the settings and it becomes enabled, click Finish. The table is added to the model and you can skip the remainder of the steps in this section.• If you select multiple foreign keys, multiple table uses are added to the model when you click Finish. |
| Create a new foreign key associated with the following unique key or index | If the parent table has a unique or primary key but the child table has no foreign key, you can create a new foreign key associated with the existing unique key. | <ul style="list-style-type: none">• Highlight a unique key.• Click Next.• Continue with step 10. |
| Create a new unique key and foreign key | If the parent table has no unique key, you can create a new unique key on the parent table, and then a new foreign key on the child table. | <ul style="list-style-type: none">• Click Next.• Continue with step 8. |
| Create a new foreign key based only on a conditional relationship | If the parent table has no unique key, you can create a foreign key based entirely on a relationship filter. | <ul style="list-style-type: none">• Click Next.• Define a condition for the virtual foreign key.• For non-Oracle, define the data movement key.• Click Finish. |

DB2 restrictions on conditional relationships

For cartridges deployed to a non-intrusive DB2 environment, conditional relationships are subjected to the following limitations:

- They cannot contain any comparison operation whose left-hand expression refers to a parent table column while the right-hand expression refers to a parameter, or vice versa.

Examples include, but are not limited to:

```
ORDER.ORDERID = p_orderid,
p_cutoff_date > ORDER.ORDER_DATE
```

where ORDER represents the parent table in the conditional relationship.

- They cannot contain any comparison operation whose left-hand expression refers to a parent table column while the right-hand expression refers to a child table column, or vice versa.

Examples include, but are not limited to:

```
PARENT.COL_1 = CHILD.COL_2,
CHILD.COL_3 <= PARENT.COL_4
```

where PARENT and CHILD represent the respective parent table and child table in the conditional relationship.

- They cannot contain any expression that applies a function to a parent table column or a parameter.

Examples include, but are not limited to:

```
UPPER(PARENT.COL_1) = CHILD.COL_2,
CHILD.COL_2 < MONTH(PARENT.COL_1),
MONTH(CHILD.COL_2) >= MONTH(PARENT.COL_1),
CHILD.COL_1 = UPPER(p_customer),
LOWER(CHILD.COL_1) = LOWER(p_customer),
p_input_month+3 < MONTH(CHILD.COL_3)
```

where PARENT and CHILD represent the respective parent table and child table in the conditional relationship.

- 8 If you chose Create a new unique key and foreign key, the Unique Key page displays. Type or select the following values:

Table 2 Unique key definition

| Field | Description |
|-----------------------|--|
| Name | Accept or type a new name for the key. |
| Available Columns | These columns in the table could be used to define the key. Highlight values in the Available Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Key Columns | These columns are the ones selected to define the key. Highlight values in the Key Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Shuttle buttons (>,<) | Use shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns field. |
| Sort | Sorts the Available Columns in alphabetical order. |

9 Click **Next**.

10 In the Foreign Key dialog, type or select the following values:

Table 3 Foreign key definition

| Field | Description |
|-----------------------|--|
| Name | Accept or type a new name for the key. |
| Available Columns | These columns in the table could be used to define the key. Highlight values in the Available Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Key Columns | These columns are the ones selected to define the key. Highlight values in the Key Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Shuttle buttons (>,<) | Use shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns field. |
| Sort | Sorts the Available Columns in alphabetical order. |

11 Click **Next**.

Designer requires a data movement key for all table uses in a model. You can specify an existing unique key or create a new one to act as the data movement key.

Additionally, for Oracle only, you can define the data movement key (via the Data Movement page) as follows:

- Create a new data movement key
- Create a row ID to be used in lieu of a data movement key
- Use an existing unique key as the data movement key.

NOTE You must ensure that, in the context of the data being extracted, the data movement key contains no null column values. Otherwise, you will have null values in your extracted data.

12 If you were prompted for a data movement key, click **Next**.

- 13 Optionally, use the Conditional Relationship dialog to define a SQL WHERE clause to filter the relationship.

Table 4 Conditional relationship definition

| Field | Description |
|-------------------|---|
| Database | <p>Choose the database to which the WHERE clause applies. You can have a different WHERE clause for as many of the vendors and versions as necessary. For example, you could associate a different WHERE clause with SQL Server, Sybase, and Oracle. At runtime, HP Test Data Management can determine the database vendor and version, and apply the clause that you assigned to that vendor and version.</p> <ul style="list-style-type: none"> Choose Any for the default WHERE clause. The Any WHERE clause is used for any vendor and version for which does not have an assigned WHERE clause. For example, if you do not assign a WHERE clause to Oracle or any of its versions, the Any WHERE clause will be used when deploying against Oracle. If the clause has vendor or version specific calls or syntax in it, expand the Any node until you see the necessary option for the clause and select it. |
| WHERE | Type or edit the WHERE clause. |
| Columns | Double-click the column name to insert the name into the WHERE clause. |
| <<Insert | Alternatively, highlight a column and use the <<Insert button to move it to the WHERE clause. |
| Associated Tables | Refer to Managing associated tables (page 43). |

- 14 Optionally, use the Conditional Relationship dialog to define a SQL WHERE clause to filter the relationship.

DB2 restrictions on conditional relationships

For cartridges deployed to a non-intrusive DB2 environment, conditional relationships are subjected to the following limitations:

- They cannot contain any comparison operation whose left-hand expression refers to a parent table column while the right-hand expression refers to a parameter, or vice versa.

Examples include, but are not limited to:

```
ORDER.ORDERID = p_orderid,
p_cutoff_date > ORDER.ORDER_DATE
```

where ORDER represents the parent table in the conditional relationship.

- They cannot contain any comparison operation whose left-hand expression refers to a parent table column while the right-hand expression refers to a child table column, or vice versa.

Examples include, but are not limited to:

```
PARENT.COL_1 = CHILD.COL_2 ,  
CHILD.COL_3 <= PARENT.COL_4
```

where PARENT and CHILD represent the respective parent table and child table in the conditional relationship.

- They cannot contain any expression that applies a function to a parent table column or a parameter.

Examples include, but are not limited to:

```
UPPER(PARENT.COL_1) = CHILD.COL_2 ,  
CHILD.COL_2 < MONTH(PARENT.COL_1) ,  
MONTH(CHILD.COL_2) >= MONTH(PARENT.COL_1) ,  
CHILD.COL_1 = UPPER(p_customer) ,  
LOWER(CHILD.COL_1) = LOWER(p_customer) ,  
p_input_month+3 < MONTH(CHILD.COL_3)
```

where PARENT and CHILD represent the respective parent table and child table in the conditional relationship.

- 15 Click **Finish**.

Adding lookup tables

Lookup tables contain non-transactional data, such as status codes or product identifiers. If the relationship between two tables is many to one and the data is non-transactional, you should add the second table as a lookup table. Otherwise, you should add it as a transactional or chaining table. Refer to [Adding transactional tables](#) (page 22) or [Adding chaining tables](#) (page 37).

You can add lookup tables in one of two ways:

- If your database has foreign key relationships defined or if you have virtual foreign key relationships defined in Designer, you can use those to relate tables in the model.
- If you cannot use existing foreign key relationships for some reason, you can add tables through the All Tables tab.

TIP To ensure optimal performance, you should consider creating an index for any columns that you plan to use in a virtual constraint. Otherwise, your performance may be slow and you could encounter deadlocks.

Both of these methods are described in this section:

- [Adding lookup tables using the foreign key relationships](#) (page 31)
- [Adding lookup tables using the All Tables tab](#) (page 32)

Synonyms and views

In some cases, you may need to extract views or synonyms instead of the base tables. For example, if you need to extract a table that resides in another database, you can create a synonym in the local database that refers to that remote table. When you use synonyms and views, bear in mind the following:

- Multi-table views are not supported in the case of Oracle. You should instead extract each of the tables in the multi-table view individually.
- Synonyms must be recognized as local objects. Remote synonyms or synonyms created as though they were remote are not supported.

Multiple table uses

You can add the same table multiple times to the same model and each instance of the table can fulfill different functions (transactional table, chaining table, or lookup table). When working with multiple table uses, you should be aware of the following points:

- Whenever you have multiple uses of a table and those uses are disjoint (no overlapping rows), you can use standard selection. When the table uses have overlapping rows, standard selection can yield unexpected results. Hence, when rows overlap among multiple table uses, you should use advanced selection.

TIP To ensure the set of rows for each table use are unique, you can attach a condition to the relationship that restricts the rows returned. Refer to *HP Test Data Management Tutorial* for an example of multiple table uses where a condition restricts the rows for each usage.

- Multiple table uses can have arbitrarily different operations associated with them. Ultimately, though, each use of the table maps to the same physical table. Hence, a row satisfying two or more table use selections might experience conflicting operations. The data movement process uses the following rule to resolve such operational conflicts:
 - If one table use has no operation and another has any operation, then the latter table use's operation is used on a row belonging to both table uses.
- If you plan to use advanced selection, then you are not restricted to using a single driving table (in other words, your model's root table). To designate that a table be used as a driving table, right-click it and select Properties. Click the Use as a Driving Table in Advanced Selection check box in the Property dialog.

Adding lookup tables using the foreign key relationships

To add lookup tables using foreign key relationships:

- 1 Open a model in Designer.
- 2 Right-click a table and select **Add Lookup Table**.
- 3 On the By Foreign Keys tab, select the table you want to add.

For example, SalesOrderDetails.

You can expand the details to see the foreign key relationship.

TIP If you select multiple foreign keys, multiple table uses are added to the model when you click **Finish**.

- 4 For non-Oracle data sources, when you click **Next**, you are prompted to specify a unique key to use as the data movement key.

Designer requires a data movement key for all table uses in a model. You can specify an existing unique key or create a new one to act as the data movement key.

Additionally, for Oracle only, you can define the data movement key (via the Data Movement page) as follows:

- Create a new data movement key
- Create a row ID to be used in lieu of a data movement key
- Use an existing unique key as the data movement key.

NOTE You must ensure that, in the context of the data being extracted, the data movement key contains no null column values. Otherwise, you will have null values in your extracted data.

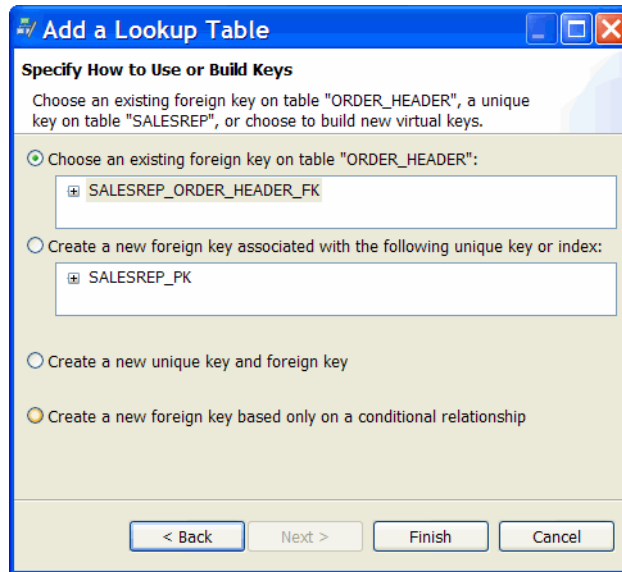
- 5 Click **Finish**.

Adding lookup tables using the All Tables tab

To add lookup tables using the All Tables tab:

- 1 Right-click the driving table and select **Add Lookup Table**.
- 2 Select the **All Tables** tab.
- 3 Optionally, type a string or a partial string in the Filter field.
- 4 Click **Search** and select the table you want.
- 5 Click **Next**.

The Add a Lookup Table dialog opens and you are prompted to specify how to use or build keys. If the parent table has an available foreign key (virtual or database) related to the child or the child table has an available unique key (virtual or database), the page displays those keys for your selection. If the parent has no such foreign key, the topmost radio button does not appear. If the child has no unique key, the page appears with only the two bottom radio buttons.



6 Choose one of the following radio buttons:

Table 5 How to use or build keys: key specification

| Selection | Description | Next steps |
|--|---|---|
| Choose an existing foreign key | If you already have a foreign key or a virtual foreign key, you can choose that existing key. | <ul style="list-style-type: none"> Highlight a foreign key from the list. For Oracle, you can click Finish. For non-Oracle, click Next and, when satisfied with the settings and it becomes enabled, click Finish. The table is added to the model and you can skip the remainder of the steps in this section. If you select multiple foreign keys, multiple table uses are added to the model when you click Finish. |
| Create a new foreign key associated with the following unique key or index | If the child table has a unique or primary key but the parent table has no foreign key, you can create a new foreign key associated with the existing unique key. | <ul style="list-style-type: none"> Highlight a unique key. Click Next. Continue with step 9. |
| Create a new unique key and foreign key | If the child table has no unique key, you can create a new unique key on the child table, and a new foreign key on the parent table. | <ul style="list-style-type: none"> Click Next. Continue with step 7. |
| Create a new foreign key based only on a conditional relationship | If the child table has no unique key, you can create a foreign key based entirely on a relationship filter. | <ul style="list-style-type: none"> Click Next. Define a condition for the virtual foreign key. For non-Oracle, define the data movement key. Click Finish. |

DB2 restrictions on conditional relationships

For cartridges deployed to a non-intrusive DB2 environment, conditional relationships are subjected to the following limitations:

- They cannot contain any comparison operation whose left-hand expression refers to a parent table column while the right-hand expression refers to a parameter, or vice versa.

Examples include, but are not limited to:

```
ORDER.ORDERID = p_orderid,  
p_cutoff_date > ORDER.ORDER_DATE
```

where ORDER represents the parent table in the conditional relationship.

- They cannot contain any comparison operation whose left-hand expression refers to a parent table column while the right-hand expression refers to a child table column, or vice versa.

Examples include, but are not limited to:

```
PARENT.COL_1 = CHILD.COL_2,  
CHILD.COL_3 <= PARENT.COL_4
```

where PARENT and CHILD represent the respective parent table and child table in the conditional relationship.

- They cannot contain any expression that applies a function to a parent table column or a parameter.

Examples include, but are not limited to:

```
UPPER(PARENT.COL_1) = CHILD.COL_2,  
CHILD.COL_2 < MONTH(PARENT.COL_1),  
MONTH(CHILD.COL_2) >= MONTH(PARENT.COL_1),  
CHILD.COL_1 = UPPER(p_customer),  
LOWER(CHILD.COL_1) = LOWER(p_customer),  
p_input_month+3 < MONTH(CHILD.COL_3)
```

where PARENT and CHILD represent the respective parent table and child table in the conditional relationship.

- 7 If you chose Create a new unique key and foreign key, the Unique Key dialog displays. Type or select the following values:

Table 6 Unique key definition

| Field | Description |
|-----------------------|---|
| Name | Accept or type a new name for the key. |
| Available Columns | These columns in the tables could be used to define the key. Highlight values in the Available Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Key Columns | These columns are the ones selected to define the key. Highlight values in the Key Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Shuttle buttons (>,<) | Use shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns field. |
| Sort | Sorts the Available Columns in alphabetical order. |

- 8 Click **Next**.

- 9 Use the Foreign Key page to define a new foreign key:

Table 7 Foreign key definition

| Field | Description |
|-----------------------|---|
| Name | Accept or type a new name for the key. |
| Available Columns | Highlight values in the Available Columns table. Use Shuttle button (>) to move columns from Available Columns to Key Columns. |
| Key Columns | Highlight values in the Key Columns table. Double-click or use Shuttle buttons to move columns from Available Columns to Key Columns or back again. |
| Shuttle buttons (>,<) | Use shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns field. |
| Sort | Sorts the Available Columns in alphabetical order. |

- 10 Click **Next**.

Designer requires a data movement key for all table uses in a model. You can specify an existing unique key or create a new one to act as the data movement key.

Additionally, for Oracle only, you can define the data movement key (via the Data Movement page) as follows:

- Create a new data movement key
- Create a row ID to be used in lieu of a data movement key
- Use an existing unique key as the data movement key.

NOTE You must ensure that, in the context of the data being extracted, the data movement key contains no null column values. Otherwise, you will have null values in your extracted data.

- 11 If you were prompted for a data movement key, click **Next**.
- 12 Optionally, use the Conditional Relationship page to define a SQL WHERE clause to filter the relationship.

Table 8 Conditional relationship definition

| Field | Description |
|-------------------|---|
| Database | <p>Choose the database to which the WHERE clause applies. You can have a different WHERE clause for as many of the vendors and versions as necessary. For example, you could associate a different WHERE clause with SQL Server, Oracle 8i, and Oracle 9i (or Oracle 10g and Oracle 11g). At runtime, HP Test Data Management can determine the database vendor and version, and apply the clause that you assigned to that vendor and version.</p> <ul style="list-style-type: none">• Choose Any for the default WHERE clause. The Any WHERE clause is used for any vendor and version for which does not have an assigned WHERE clause. For example, if you do not assign a WHERE clause to Oracle or any of its versions, the Any WHERE clause will be used when deploying against Oracle.• If the clause has vendor or version specific calls or syntax in it, expand the Any node until you see the necessary option for the clause and select it. |
| WHERE | Type or edit the WHERE clause. |
| Columns | Double-click the column name to insert the name into the WHERE clause. |
| <<Insert | Alternatively, highlight a column and use the <<Insert button to move it to the WHERE clause. |
| Associated Tables | Refer to Managing associated tables (page 43). |

- 13 Click **Finish**.

Adding chaining tables

If the relationship between two tables is many to one and the data is not lookup data, you should add the second table as a chaining table. Otherwise, you would add it as a lookup or transactional table. Refer to [Adding transactional tables](#) (page 22) or [Adding lookup tables](#) (page 30).

You can add chaining tables in one of two ways:

- If your database has foreign key relationships defined or virtual foreign key relationships, you can use those to relate tables in the model.
- If you cannot use existing foreign key relationships for some reason, you can add tables through the All Tables tab.

TIP If you have a chaining table in your model, you must choose the advanced selection option for the associated cartridges. Refer to *HP Test Data Management Concepts Guide* for more information about advanced selection.

TIP To ensure optimal performance, you should consider creating an index for any columns that you plan to use in a virtual constraint. Otherwise, your performance may be slow and you could encounter deadlocks.

Both of these methods are described in this section:

- [Adding chaining tables using the foreign key relationships](#) (page 38)
- [Adding chaining tables using the All Tables tab](#) (page 38)

Synonyms and views

In some cases, you may need to extract views or synonyms instead of the base tables. For example, if you need to extract a table that resides in another database, you can create a synonym in the local database that refers to that remote table. When you use synonyms and views, bear in mind the following:

- Multi-table views are not supported in the case of Oracle. You should instead extract each of the tables in the multi-table view individually.
- Synonyms must be recognized as local objects. Remote synonyms or synonyms created as though they were remote are not supported.

Multiple table uses

You can add the same table multiple times to the same model and each instance of the table can fulfill different functions (transactional table, chaining table, or lookup table). When working with multiple table uses, you should be aware of the following points:

- 1 Whenever you have multiple uses of a table and those uses are not disjoint (overlapping rows), it is advised that you use advanced selection. When you have table uses with overlapping rows, standard selection can yield unexpected results.
- 2 Multiple table uses can have arbitrarily different operations associated with them. Ultimately, though, each use of the table maps to the same physical table. Hence, a row satisfying two or more table use selections might experience conflicting operations. The data movement process uses the following rules to resolve such operational conflicts:

- If one table use has no operation and another has any operation, then the latter table use's operation is used on a row belonging to both table uses.

Adding chaining tables using the foreign key relationships

- 1 Open a model in Designer.
- 2 Right click a driving, transactional, or chaining table and select **Add Chaining Table**.
- 3 On the By Foreign Keys tab, select the table you want to add.
For example, SalesOfferProduct.

You can expand the details to see the foreign key relationship.

TIP If you select multiple foreign keys, multiple table uses are added to the model when you click **Finish**.

- 4 Click **Next**.
Designer requires a data movement key for all table uses in a model. You can specify an existing unique key or create a new one to act as the data movement key.

Additionally, for Oracle only, you can define the data movement key (via the Data Movement page) as follows:

- Create a new data movement key
- Create a row ID to be used in lieu of a data movement key
- Use an existing unique key as the data movement key.

NOTE You must ensure that, in the context of the data being extracted, the data movement key contains no null column values. Otherwise, the rows associated with the null values will not be included in your extracted data.

- 5 Click **Finish**.

Adding chaining tables using the All Tables tab

To add a chaining table using the All Tables tab:

NOTE You cannot add a transactional table or a chaining table to a lookup table.

- 1 Right-click a driving, transactional, or chaining table and select **Add Chaining Table**.
- 2 Select the **All Tables** tab.
- 3 Optionally, type a string or a partial string in the Filter field.
- 4 Click **Search**.
- 5 Highlight the table you want.

6 Click **Next**.

The Specify How to Use or Build Keys page opens. If the parent table has an available foreign key (virtual or database) related to the child or the child table has an available unique key (virtual or database), the page displays those keys for your selection. If the parent has no such foreign key, the top portion of the page does not appear. If the child has no unique key, the page does not appear at all.

7 Choose one of the following radio buttons:

Table 9 Key specification

| Selection | Description | Next steps |
|--|---|---|
| Choose an existing foreign key | If you already have a foreign key or a virtual foreign key, you can choose that existing key. | <ul style="list-style-type: none">Highlight a foreign key from the list.For Oracle, you can click Finish. For non-Oracle, click Next and, when satisfied with the settings and it becomes enabled, click Finish. The table is added to the model and you can skip the remainder of the steps in this section. |
| Create a new foreign key associated with the following unique key or index | If the child table has a unique or primary key but the parent table has no foreign key, you can create a new foreign key pointing to the existing unique key. | <ul style="list-style-type: none">Highlight a unique key.Click Next to create the foreign keyContinue with step 10. |
| Create a new unique key and foreign key | If the child table has no unique key, you can create a new unique key on the child table, and a new foreign key on the parent table. | <ul style="list-style-type: none">Click Next.Continue with step 8. |
| Create a new foreign key based only on a conditional relationship | If the child table has no unique key, you can create a foreign key based entirely on a relationship filter. | <ul style="list-style-type: none">Click Next.Define a condition for the virtual foreign key.Click Finish. |

TIP If you select multiple foreign keys, multiple table uses are added to the model when you click **Finish**.

DB2 restrictions on conditional relationships

For cartridges deployed to a non-intrusive DB2 environment, conditional relationships are subjected to the following limitations:

- They cannot contain any comparison operation whose left-hand expression refers to a parent table column while the right-hand expression refers to a parameter, or vice versa.

Examples include, but are not limited to:

```
ORDER.ORDERID = p_orderid,
p_cutoff_date > ORDER.ORDER_DATE
```

where ORDER represents the parent table in the conditional relationship.

- They cannot contain any comparison operation whose left-hand expression refers to a parent table column while the right-hand expression refers to a child table column, or vice versa.

Examples include, but are not limited to:

```
PARENT.COL_1 = CHILD.COL_2,
CHILD.COL_3 <= PARENT.COL_4
```

where PARENT and CHILD represent the respective parent table and child table in the conditional relationship.

- They cannot contain any expression that applies a function to a parent table column or a parameter.

Examples include, but are not limited to:

```
UPPER(PARENT.COL_1) = CHILD.COL_2,
CHILD.COL_2 < MONTH(PARENT.COL_1),
MONTH(CHILD.COL_2) >= MONTH(PARENT.COL_1),
CHILD.COL_1 = UPPER(p_customer),
LOWER(CHILD.COL_1) = LOWER(p_customer),
p_input_month+3 < MONTH(CHILD.COL_3)
```

where PARENT and CHILD represent the respective parent table and child table in the conditional relationship.

- 8 If you chose Create a new unique key and foreign key, the Unique Key page displays to define a new unique key:

Table 10 Unique key definition

| Field | Description |
|-------------------|---|
| Name | Accept or type a new name for the key. |
| Available Columns | These columns in the tables could be used to define the key. Highlight values in the Available Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Key Columns | These columns are the ones selected to define the key. Highlight values in the Key Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |

Table 10 Unique key definition

| Field | Description |
|--------------------------|---|
| Shuttle buttons (>,<) | Use shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns field. |
| Sort | Sorts the Available Columns in alphabetical order or preserves the database order. |

9 Click **Next**.

10 Use the Foreign Key page to define a new foreign key:

Table 11 Foreign key definition

| Field | Description |
|--------------------------|---|
| Name | Accept or type a new name for the key. |
| Available Columns | These columns in the tables could be used to define the key. Highlight values in the Available Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Key Columns | These columns are the ones selected to define the key. Highlight values in the Key Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Shuttle buttons (>,<) | Use shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns field. |
| Sort | Sorts the Available Columns in alphabetical order. |

11 Click **Next**.

Designer requires a data movement key for all table uses in a model. You can specify an existing unique key or create a new one to act as the data movement key.

Additionally, for Oracle only, you can define the data movement key (via the Data Movement page) as follows:

- Create a new data movement key
- Create a row ID to be used in lieu of a data movement key
- Use an existing unique key as the data movement key.

NOTE You must ensure that, in the context of the data being extracted, the data movement key contains no null column values. Otherwise, the rows associated with the null values will not be included in your extracted data.

- 12 If you were prompted for a data movement key, click **Next**.
- 13 Optionally, use the Conditional Relationship page to define a SQL WHERE clause to filter the relationship.

Table 12 Conditional relationship definition

| Field | Description |
|----------|---|
| Database | <p>Choose the database to which the WHERE clause applies. You can have a different WHERE clause for as many of the vendors and versions as necessary. For example, you could associate a different WHERE clause with SQL Server, Oracle 8i, and Oracle 9i (or Oracle 10g and Oracle 11g). At runtime, HP Test Data Management can determine the database vendor and version, and apply the clause that you assigned to that vendor and version.</p> <ul style="list-style-type: none">• Choose Any for the default WHERE clause. The Any WHERE clause is used for any vendor and version for which does not have an assigned WHERE clause. For example, if you do not assign a WHERE clause to Oracle or any of its versions, the Any WHERE clause will be used when deploying against Oracle.• If the clause has vendor or version specific calls or syntax in it, expand the Any node until you see the necessary option for the clause and select it. |
| WHERE | Type or edit the WHERE clause. |
| Columns | Double-click the column name to insert the name into the WHERE clause. |
| <<Insert | Alternatively, highlight a column and use the <<Insert button to move it to the WHERE clause. |

- 14 Click **Finish**.

Managing associated tables

If the WHERE clause for a conditional relationship or rule refers to tables that are not included in the model, you need to include such tables as associated tables. In Designer, you click Associated Tables and add these tables to the list. For tables in the associated tables list, a GRANT SELECT is automatically issued to ensure that they are accessible to the cartridge when it is deployed.

- 1 From the Rule dialog box or Conditional Relationship page, click **Associated Tables**. The Associated Tables List dialog displays.
- 2 You can populate the list in one of two ways:
 - a Click **Automatic**. Designer discovers the tables referenced in your WHERE clause that are outside the model and adds them to the list.or
 - a Click **Add**. The Add Tables dialog box appears.
 - b Scan or search the tree to find the tables and select them.
 - c After you have selected the tables, click **OK**.

TIP To remove tables from the Associated Tables List dialog, select the table and click **Remove**.

- 3 Click **OK**.

Working with rules

A rule is a condition defined by a SQL WHERE clause that determines whether an object is eligible for copying. Typically, you define rules in the model and then select the model for use in a cartridge. In the case of a schema-based cartridge, where you have no model, a rule may be both defined and used in the cartridge.

When you create the rule, you specify whether it applies to copying data or specify that a rule only be used for Designer preview. For example, you may want to limit the scope of data during development to speed performance for frequent preview operations.

Rules can be defined as *exclusive* or *inclusive*. With exclusive rules, all rows in the model instance must meet the defined criteria. With inclusive rules, only one row must meet the criteria. For example, for the model instance ORDERS_1 in [Figure 2](#), you could define a rule to find orders with items that are equal to abc:

Figure 2 Rule behavior in models: exclusive vs. inclusive

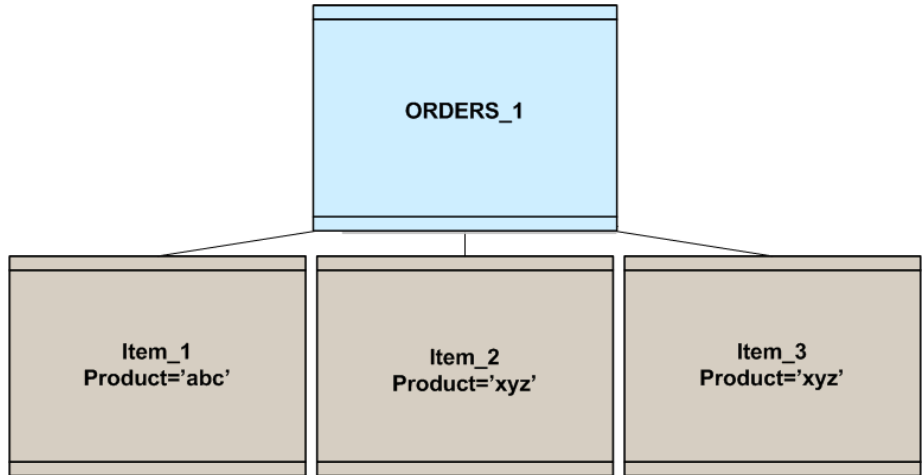


Table 13 Rule types

| If the rule type is | Effect |
|---------------------|--|
| Exclusive | The entire model instance ORDERS_1 is excluded because Item_2 and Item_3 do not meet the criteria to be extracted. All item rows must have a product of 'abc'. |
| Inclusive | The entire model instance ORDERS_1 is included because at least one item (Item_1) meets the criteria to be extracted. Inclusive rules must be associated with a driving table. |

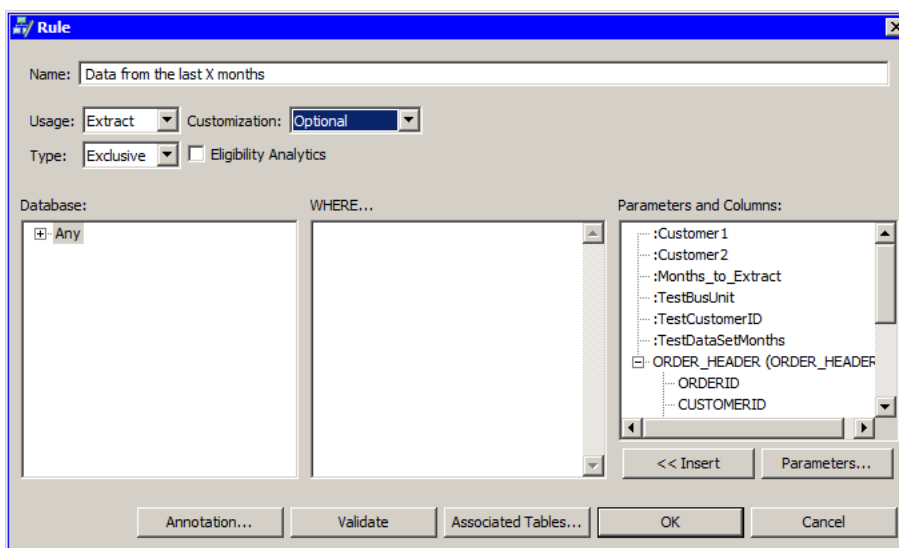
This section includes:

- [Adding a rule](#) (page 44)
- [Editing a rule](#) (page 47)

Adding a rule

To add a rule to a model:

- 1 Open a model.
- 2 Right-click a table and select **Add Rule**. The Rule dialog opens.



3 Type or select values for the following fields:

Table 14 Rule definition

| Field | Description |
|---------------|---|
| Name | Type a name for the rule. The name cannot contain single quotes. |
| Usage | Choose from the list of options: <ul style="list-style-type: none"> • Extract means that the rule applies to extraction operations. • Preview means that the rule applies only when previewing data in Designer. This type of rule is useful for limiting the data returned to improve performance during development. |
| Type | Choose from the list of options: <ul style="list-style-type: none"> • Exclusive means that, for any parent row to be eligible, all of its child rows must meet the defined criteria. • Inclusive means that, for any parent row to be eligible, only one of its child rows must meet the defined criteria. Inclusive rules must be associated with a driving table. |
| Customization | Choose from the list of options: <ul style="list-style-type: none"> • Optional • Mandatory • Recommended Refer to Guidelines for customizations (page 156) for more information about customization. |

Table 14 Rule definition

| Field | Description |
|------------------------|---|
| Eligibility Analytics | <p>Choose whether you want eligibility analytics to be enabled. Because eligibility analytics can impact the performance of a cartridge or business flow, you can choose to enable or disable them on a per rule basis.</p> <p>By default, eligibility analytics are turned off and you must explicitly enable them. Generally, it is recommended that you enable eligibility analytics on rules that will exclude a fairly small percentage of the data (for example, between 1% and 5%).</p> <hr/> <p>NOTE Eligibility analytics are not supported in non-intrusive environments.</p> <hr/> |
| Database | <p>Choose the database to which the WHERE clause applies. You can have a different WHERE clause for as many of the vendors and versions as necessary. For example, you could associate a different WHERE clause with DB2, SQL Server, Oracle 8i, and Oracle 9i (or Oracle 10g and Oracle 11g). At runtime, HP Test Data Management can determine the database vendor and version, and apply the clause that you assigned to that vendor and version.</p> <ul style="list-style-type: none"> Choose Any for the default WHERE clause. The Any WHERE clause is used for any vendor and version which does not have an assigned WHERE clause. For example, if you do not assign a WHERE clause to Oracle or any of its versions, the Any WHERE clause will be used when deploying against Oracle. If the clause has vendor or version specific calls or syntax in it, expand the Any node until you see the necessary option for the clause and select it. <hr/> |
| WHERE | Type or edit the WHERE clause. If you use subqueries, you should fully qualify column names to ensure that they are distinct from the columns in the outer query. |
| Parameters and Columns | Highlight a parameter or column and use the <<Insert button to move it to the WHERE clause. |
| <<Insert | Use to move a highlighted parameter or column to the WHERE clause. |
| Parameters | Click Parameters to manage your parameter definitions (add or edit parameters). Refer to Chapter 3, Working with parameters for more information. |

Table 14 Rule definition

| Field | Description |
|-------------------|---|
| Annotation | Optionally, click this button to enter a description or comment for the rule. Refer to Annotating projects (page 122) for more information. |
| Validate | Click Validate to verify that the syntax of your WHERE clause is correct. Note that validation does not ensure that your rule will behave as you expect. It simply confirms that it meets the syntax requirements of a WHERE clause for the database you are using. |
| Associated Tables | Refer to Managing associated tables (page 43). |

- 4 Optionally, click **Parameters** to create or edit parameters. Refer to [Chapter 3, Working with parameters](#) for more information about parameters.
- 5 Click **OK**.

Editing a rule

To edit a rule:

- 1 Open a model in Designer.
- 2 Double click on a rule, or right-click a rule and select **Edit Rule**. The Rule dialog box opens.
- 3 Edit the rule as appropriate.

Working with virtual unique and foreign keys

A *virtual unique key* is the equivalent of a database unique key. A *virtual foreign key* is similar to a database foreign key, but may also define the condition under which the relation is valid. Virtual unique and foreign keys are created in Designer when you add tables to a model, or from the Virtual Constraints dialog box. Once you create the virtual unique or foreign key, it becomes referenceable and available when you add tables in Designer. Virtual unique and foreign keys are not created in the database, and are not deployed to the database.

Typically, you would only create the virtual unique or foreign key prior to adding the table if you knew that you would need to use it repeatedly.

When working with virtual unique and foreign keys, you should be aware of the following restrictions:

- Because a data movement key is required for every table in non-intrusive environments, you must create a virtual unique key when there is no physical key defined at the source database.

- Because virtual unique and foreign keys are not validated, ensure that the ones you select are correctly defined.
- To ensure optimal performance, you should consider creating an index for all columns that you plan to use as part of a virtual unique or foreign key. Otherwise, your extracting performance may be slow. A non-unique backing index or a unique index that covers some of the columns in the key may be sufficient to ensure good performance.

This section includes:

- [Adding a virtual unique key](#) (page 48)
- [Adding a virtual foreign key](#) (page 49)
- [Editing a virtual unique key or virtual foreign key](#) (page 51)
- [Deleting a virtual key](#) (page 52)

Adding a virtual unique key

To add a virtual unique key:

- 1 Open a model.
- 2 Right click a table and select **Virtual Constraints**. The Virtual Constraints dialog box opens.
- 3 Click **Add Unique Key**. The Virtual Unique Key dialog box opens.
- 4 Type or select the following:

Table 15 Virtual unique key definition

| Field | Description |
|-----------------------|---|
| Name | The default name of the key. You can accept it or type a new name. |
| Available Columns | These columns in the tables could be used to define the key. Highlight values in the Available Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Key Columns | These columns are the ones selected to define the key. Highlight values in the Key Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Shuttle buttons (>,<) | Use Shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns list. |
| Sort | Displays the Available Columns in alphabetical order. |

- 5 Click **OK**.

- 6 Click **Close** to close the Virtual Constraints dialog box.

Adding a virtual foreign key

To add a virtual foreign key:

- 1 Open a model.
- 2 Right click a table and select **Virtual Constraints**. The Virtual Constraints dialog box opens.
- 3 Click **Add Foreign Key**. The Add Virtual Foreign Key dialog box opens.
- 4 Type a string or a partial string in the Filter field to choose a table from which you will select a database or virtual unique key that corresponds to the virtual foreign key you are creating. Note that, if no unique key exists, the wizard walks you through creating one.
- 5 Click **Search** and highlight the table you want.
- 6 Click **Next**.

The Specify How to Use or Build Keys page opens. If the chosen table has no unique keys (database or virtual), the page appears with only the two bottom radio buttons.

- 7 Choose one of the following radio buttons:

Table 16 Key specification

| Selection | Description | Next steps |
|--|--|---|
| Create a new foreign key associated with the following unique key or index | This option creates a new foreign key using existing unique key. | <ul style="list-style-type: none">• Highlight an existing unique key to be used with the new foreign key.• Click Next.• Continue with step 10. |
| Create a new unique key and foreign key | This option creates a new unique key and then a new foreign key that uses the just created unique key. | <ul style="list-style-type: none">• Click Next.• Continue with step 8. |
| Create a new foreign key based only on a conditional relationship | If the chosen table has no unique key, you can create a foreign key based entirely on a relationship filter. | <ul style="list-style-type: none">• Click Next.• Define a condition for the virtual foreign key.• Click Finish. |

8 Use the Unique Key page to define a new unique key:

Table 17 Unique key definition

| Field | Description |
|-----------------------|---|
| Name | The default name of the key. You can accept it or type a new name. |
| Available Columns | These columns in the tables could be used to define the key. Highlight values in the Available Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Key Columns | These columns are the ones selected to define the key. Highlight values in the Key Columns table. Use the shuttle buttons to move columns back and forth from the Available Columns and Key Columns lists as necessary. |
| Shuttle buttons (>,<) | Use Shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns field. |
| Sort | Sorts the Available Columns in alphabetical order. |

9 Click **Next**.

10 Use the Foreign Key page to define a new foreign key:

Table 18 Foreign key definition

| Field | Description |
|-----------------------|---|
| Name | Accept or type a new name for the key. |
| Available Columns | Highlight values in the Available Columns table. Use Shuttle button (>) to move columns from Available Columns to Key Columns. |
| Key Columns | Highlight values in the Key Columns table. Double-click or use Shuttle buttons to move columns from Available Columns to Key Columns or back again. |
| Shuttle buttons (>,<) | Use Shuttle buttons to move columns back and forth between Available Columns and Key Columns. |
| Up and Down buttons | Use to order the columns in the Key Columns field. |
| Sort | Sorts the Available Columns in alphabetical order. |

11 Click **Next**.

- 12 Optionally, use the Conditional Relationship page to define a SQL clause to filter the relationship.

Table 19 Conditional relationship definition

| Field | Description |
|-------------------|--|
| Database | <p>Choose the database to which the WHERE clause applies. You can have a different WHERE clause for as many of the vendors and versions as necessary. For example, you could associate a different WHERE clause with DB2, SQL Server, Oracle 8i, and Oracle 9i (or Oracle 10g and Oracle 11g). At runtime, HP Test Data Management can determine the database vendor and version, and apply the clause that you assigned to that vendor and version.</p> <ul style="list-style-type: none">Choose Any for the default WHERE clause. The Any WHERE clause is used for any vendor and version for which does not have an assigned WHERE clause. For example, if you do not assign a WHERE clause to Oracle or any of its versions, the Any WHERE clause will be used when deploying against Oracle.If the clause has vendor or version specific calls or syntax in it, expand the Any node until you see the necessary option for the clause and select it. |
| WHERE | Type or edit the WHERE clause. |
| Columns | Double-click the column name to insert the name into the WHERE clause. |
| <<Insert | Alternatively, highlight a column and use the <<Insert button to move it to the WHERE clause. |
| Associated Tables | Refer to Managing associated tables (page 43). |

- 13 Click **Finish**.

- 14 Click **Close** to close the Virtual Constraints dialog box.

Editing a virtual unique key or virtual foreign key

To edit a virtual unique or virtual foreign key:

- 1 Open a model.
- 2 Right click a table and select **Virtual Constraints**. The Virtual Constraints dialog box opens.

TIP You can also right-click a line in the model and choose **Edit Virtual Foreign Key**.

- 3 Select a virtual constraint from the list and click **Edit**.
- 4 Edit the virtual unique key or virtual foreign key as required and click **OK**.

- 5 Click **Close** to close the Virtual Constraints dialog box.

Deleting a virtual key

To delete a virtual key:

- 1 Open a model.
- 2 Right click a table and select **Virtual Constraints**. The Virtual Constraints dialog box opens.

TIP You can also right-click a line in the model and choose **Replace Virtual Foreign Key Connection**, if you want to simply change the selected foreign key for the relationship.

- 3 Select a virtual constraint from the list and click **Remove**.

Modifying object properties

Object properties allow you to change the name of an object in the model. For example, you can rename a database table to something that is more meaningful to you for the creation of your cartridge.

This feature also allows you to set aliases and add annotations to the object.

- 1 Right-click a table use in the model and select **Properties**.
- 2 Modify the properties as needed.

Table 20 Object properties

| Property | Description |
|-------------------|---|
| Table Name | The name of the table in the database. If you have mapped schemas through Connection > Map Schemas , you will see Mapped Table Name and Database Table Name instead of just Table Name. |
| Shape Name | The name you want displayed in the diagram. This name has no impact on the cartridge. It is for display purposes in the diagram only. |
| Table Alias | The alias to be used in rule WHERE clauses. (Changing the name will invalidate any existing rules that reference the previous alias.) |
| Data Movement Key | Select the Data Movement Key you want to use for the object. This key is used to join to the selection tables. |

Locating a table

This feature helps you locate model tables in the database navigator. If your database has many tables, you can use this feature to quickly find a table so that you can browse its details.

- 1 Right click a table in the model editor or under the Model node in the Project Navigator and select **Locate Table**.

The table is selected in the database navigator.

Drag and drop a table

This feature allows you to drag a table from the database navigator and attach it to any existing table in the model.

NOTE You cannot add a transactional table or a chaining table to a lookup table.

To drag and drop a table:

- 1 Highlight a table in the database navigator panel.
- 2 Drag the table over to the Model editor.
- 3 Drop the table on any existing table.
- 4 Select one of the following:
 - add `<table_name>` as a transactional table
 - add `<table_name>` as a lookup table
 - add `<table_name>` as a chaining table
- 5 From this point, the wizard behaves much the way it is described in the following sections, except it does not display the Choose Table page because the parent and child are already known to it.
 - [Adding transactional tables](#) (page 22)
 - [Adding lookup tables](#) (page 30)
 - [Adding chaining tables](#) (page 37)

In HP Test Data Management, you can use parameters in your rules, allowing users who run the jobs to change the values at runtime. In addition, you can create parameters that can be given values dynamically at runtime by code or at configuration time by administrators.

This chapter explains how to work with parameters.

This chapter includes:

- [About parameters](#) (page 55)
- [Creating parameters](#) (page 56)
- [Validating parameters](#) (page 60)
- [Creating lists of values for parameters](#) (page 61)
- [Referencing parameters](#) (page 63)
- [Managing model compatibility](#) (page 67)
- [Deleting parameters](#) (page 67)

About parameters

In HP Test Data Management, you have three types of parameters:

- **Runtime parameters** have their values set at runtime by the user running the job. Runtime parameters tend to be best for operational values that tend to change with each execution of a job. For example, if your extraction is based on a specified cutoff date, you most likely need to update that date every time you run the job.
- **Configuration parameters** have their values set by an administrator (someone who has repository privileges from Console) through the administrator interface. Typically, this type of parameter represents values that should be changed very infrequently, perhaps only at deployment time.
- **Dynamic parameters** have their values set once by a Groovy script that runs at deployment time to obtain a value. For example, this type of parameter can supply the type or version of a database or application, which can be obtained programmatically at deployment time.

NOTE After a job starts running in the Web Console, all three types of parameters are treated the same way. In addition, you cannot change the parameter values after the job starts.

Regardless of a parameter's type, it can be referenced from several places within your extraction project:

- WHERE clauses within your rules
- Groovy scripts in runtime and configuration parameter validations
- Groovy scripts in business flows
- SQL for runtime and configuration parameter list of values
- Interrupts in your business flows
- Manage Compatibilities dialog box for the model (dynamic parameters only)

Creating parameters

You can create parameters beforehand and then reference them, or, in SQL, you can create them dynamically as you reference them.

This section includes:

- [Creating or editing parameters from the Project Navigator](#) (page 56)
- [Creating or editing parameters using the Parameters button](#) (page 59)
- [Creating parameters automatically](#) (page 59)
- [Validating parameters](#) (page 60)

Creating or editing parameters from the Project Navigator

To create or edit parameters using Project Navigator:

- 1 In Designer, from the Project Navigator, right-click the Parameters node and choose New Parameter.
- 2 Type or select values for the following Parameter definition properties:

| Field | Description |
|----------------|---|
| Name | Enter a name for the parameter. In choosing parameter names, you must avoid reserved names. Refer to Validating parameters (page 60) for a list of reserved names. |
| Parameter Type | Type of the parameter, choose one from the list of options: <ul style="list-style-type: none">• Runtime parameters get their values from the user at runtime.• Configuration parameters are assigned values by the administrator and cannot be changed by the user at runtime.• Dynamic parameters get their values from Groovy scripts. If you choose this type, the rest of the properties in the dialog are replaced by Data Type and a Code area, where you can enter your Groovy script. |
| Label | Label that appears on the Console Job Launcher page. |
| Data Type | Type of the parameter's value (Date, Number, or String). |

| Field | Description |
|--------------|--|
| Length | Maximum length allowed for the selected data type. |
| Default | Type a default parameter value. |

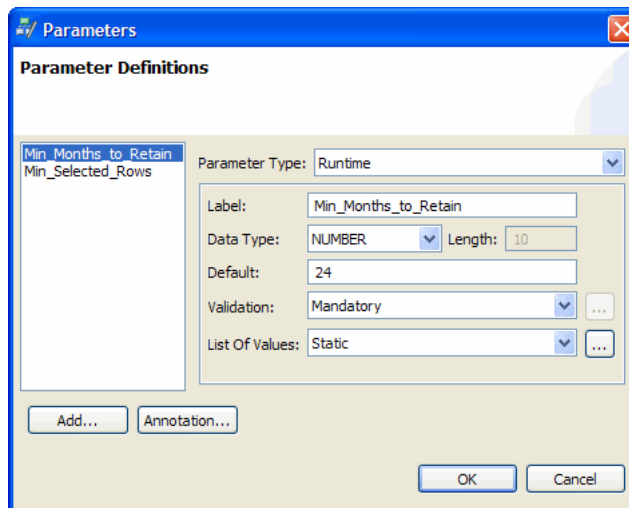
| Field | Description |
|--------------------------------|---|
| Validation | <p>For runtime and configuration parameters only, the type of validation you want to use for the parameter's values. Choose from the list of options:</p> <ul style="list-style-type: none"> • None • Groovy means that you will supply a Groovy script to validate the parameter's value. • Mandatory means that a parameter value must be supplied in order for the job to run. <p>Refer to Validating parameters (page 60) for more information.</p> |
| List of Values | <p>For runtime and configuration parameters only, the kind of list of values you want to use. Choose from the list of options:</p> <ul style="list-style-type: none"> • None • Static means that you can use the adjacent browse button to create a hard-coded list of values for the parameter. • SQL means that a SQL SELECT statement populates the list of values for the parameter. <p>Refer to Creating lists of values for parameters (page 61).</p> |
| Code (dynamic parameters only) | <p>This area only appears when you choose a Parameter Type of Dynamic. Enter a Groovy script that populates the parameter's value. The code must return a value that matches the parameter's data type (Date, Number, or String). For the date data type, you must use <code>java.util.Date</code> to return a value of the appropriate type and format. For the number and string data types, the code must return number or string values, respectively.</p> <p>If you are planning to use the parameter for compatibility checking, it must return a string. Refer to Managing model compatibility (page 67) for more information about compatibility.</p> <p>An example of a Groovy script that populates a dynamic parameter might look like the following:</p> <pre>import java.util.Date import org.codehaus.groovy.runtime.TimeCategory; use([TimeCategory]) { Date now = new Date(); Date cutoff_dt = now - retention.months // Adjust the date to the beginning of the year. GregorianCalendar calendar = new GregorianCalendar(); calendar.setTime(cutoff_dt); calendar.set(GregorianCalendar.MONTH, 0); calendar.set(GregorianCalendar.DAY_OF_MONTH, 1); return calendar.getTime(); }</pre> <p>where <code>retention</code> is a configuration parameter.</p> |

- 3 Click **OK**.

Creating or editing parameters using the Parameters button

To create or edit parameters using the Parameters button:

- 1 In Designer, from either the Rule or Groovy Script dialogs, click **Parameters**.
- 2 To create a new parameter, click **Add**. To edit an existing parameter, click it in the list on the left of the dialog.



- 3 Type or select values for the fields as described in [step 2](#) (page 56).
In choosing parameter names, you must avoid reserved names. Refer to [Validating parameters](#) (page 60) for a list of reserved names.
- 4 Click **OK**.
- 5 To reference the parameter, refer to [Referencing parameters](#) (page 63).

TIP To delete a parameter, right-click it in the Project Navigator under the Parameters node and choose **Delete**. When you delete a parameter, ensure that you also delete all references to it throughout your project.

Creating parameters automatically

To create parameters automatically:

- 1 In Designer, referencing a parameter in SQL that does not already exist causes a new parameter to be automatically created with default properties. Refer to [Referencing parameters](#) (page 63) for information on how to reference parameters.
- 2 After you reference the new parameter from SQL, go to the Project Navigator and expand the Parameters node to find the automatically created parameter.

TIP Some dialog boxes, such as the Rule dialog box, contain a Parameters button, which you can click to see the parameters as well. If you're not in such a dialog, you need to accept the dialogs and go to the Project Navigator.

- 3 Double-click the automatically created parameter. Confirm that its properties are appropriate for your usage. Refer to [step 2](#) (page 56) for more information about parameter definition properties.
- 4 Click **OK**.

Validating parameters

For runtime and configuration parameters, you can choose whether and how to validate the values entered by the user. By validating parameters, you ensure that the user enters a valid value before attempting to run the business flow that references the parameter. If the user enters an invalid value, the job will not run. If you choose not to validate parameter values, then an invalid value is not discovered until the job fails.

- 1 Go to the Parameter Definition dialog box by right clicking the parameter in the Project Navigator and choosing **Edit Parameter**.
- 2 In the Validation field, choose one of these values:
 - **None** means that you want no validation performed on parameter values.
 - **Mandatory** means that a value must be entered for the parameter in order for the business flow to run. If a value is not entered, then the business flow will not run. In general, if you choose Mandatory, it is good practice to also enter a default value for the parameter. That way, if the user does not enter a value, the business flow will run with the default value rather than fail.
 - **Groovy** means that you are providing a Groovy script to check the value. If you choose this option, you must also click the adjacent browse button to create a Groovy script. If the parameter fails validation, the script must return a short error message that describes the reason for the validation failure. If the parameter passes validation, the script must return nothing, or a null or empty string. An example Groovy validation script might look something like the following:

Example

```
import java.util.Date
import org.codehaus.groovy.runtime.TimeCategory;
use([TimeCategory]) {
    Date now = new Date()
    Date maxDate = now - retention.months
    if (cutoff_dt > maxDate){
        return "Date must be older than ${retention} months."
    }
}
```

where `cutoff_dt` is a runtime parameter and `retention` a configuration parameter

Creating lists of values for parameters

Lists of values for runtime and configuration parameters provide another method for reducing the likelihood of user input errors and restricting the possible values the user may enter. You can create a fixed list of values by manually entering values and labels. Alternatively, if the possible values are stored in a database table somewhere, you can create a dynamic list of values by using a SQL SELECT statement to populate the list. The latter option tends to be the most efficient method because it ensures that the list of values will be automatically kept in sync with the database. You do not have to worry about updating the list every time the possible values change.

TIP Lists of values do not apply when you run your business flow from the command line. To restrict the values entered from the command line, you should use validation Groovy scripts. Refer to [Validating parameters](#) (page 60).

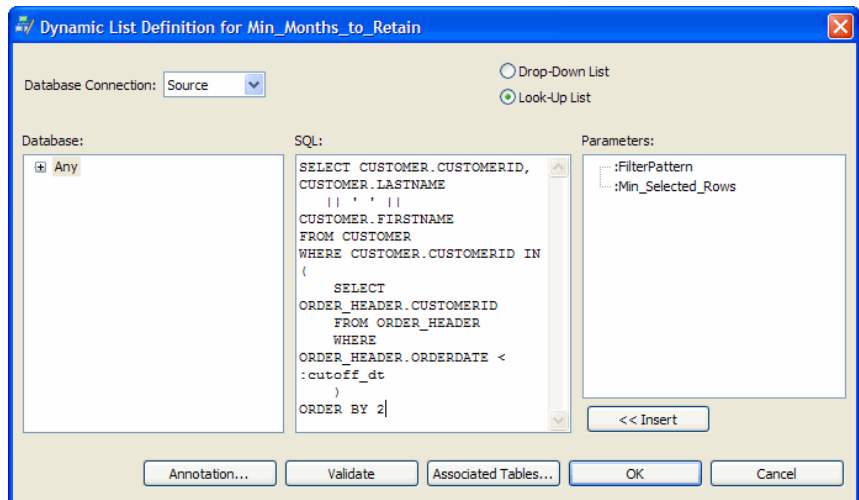
Creating a static list of values

- 1 Go to the Parameter Definition dialog box by right-clicking the parameter in the Project Navigator and choosing **Edit Parameter**.
- 2 In the List of Values field, choose **Static**.
- 3 Click the Browse button to the right of List of Values.
- 4 For Id, enter a valid value for the parameter.
- 5 For Label, enter the label that you want the user to see in the list. For example, suppose the parameter values are the start and close dates of quarters in your fiscal years. For Label, you might enter something like Q1FY08 Start or Q1FY08 Close.
- 6 Click **Add**.
- 7 Repeat [step 4](#) through [step 6](#) until you have added all of the desired values to the list.
- 8 Click **OK**.

Creating dynamic lists of values

To create a dynamic list of values:

- 1 Go to the Parameter Definition dialog box by right-clicking the parameter in the Project Navigator and choosing **Edit Parameter**.
- 2 In the List of Values field, choose **SQL**.
- 3 Click the Browse button to the right of List of Values. The Dynamic List Definition dialog box displays.



- 4 For Database Connection, choose the connection (**Source, Repository**) against which you want to run the SELECT statement that populates the list.
- 5 Choose either the **Drop-Down List** or **Look-Up List** radio button. A drop-down list shows the list of values as a combo box with the field. A look-up list opens a dialog with the values and the user can enter a filter pattern.

TIP If you choose Look-Up List, notice that a parameter named FilterPattern appears in the Parameters area. This parameter contains the search string entered by the user and you can reference this string from your SELECT statement. For example, you might use this string to further qualify your SELECT criteria and thereby reduce the size of the list.

Use a drop-down list if the number of values is reasonably small. Use a look-up list if the number of values is very large.

- 6 For Database, choose the database to which the SELECT statement applies. You can have a different SELECT statement for as many of the vendors and versions as necessary. For example, you could associate a different SELECT statement with DB2, SQL Server, Oracle 8i, and Oracle 9i. At runtime, HP Test Data Management can determine the database vendor and version, and apply the clause that you assigned to that vendor and version.
 - Choose **Any** for the default WHERE clause. The Any WHERE clause is used for any vendor and version which does not have an assigned WHERE clause. For example, if you do not assign a WHERE clause to Oracle or any of its versions, the Any WHERE clause will be used when deploying against Oracle.
 - If the clause has vendor or version specific calls or syntax in it, expand the Any node until you see the necessary option for the vendor and version and select it.
- 7 For SQL, enter a SQL SELECT statement that populates the list of values for the parameter. The SELECT statement must return an *id* and a *label*. The *id* is the first return value, it will appear on the command line. Note that the data

type of the `id` sets the data type for the parameter. The `label` is the second return value. It is displayed to the user in the drop-down list or look-up list in the Job Launcher in the Console.

See [Examples](#) (page 63) for some SQL examples for your list of values.

- 8 Click Validate to confirm the syntax of your SELECT statement.
- 9 If you reference tables not included in your model, click **Associated Tables**.

See also [Managing associated tables](#) (page 43)

- 10 Click **OK**.

Examples Following is an example SELECT statement with numeric values:

```
SELECT CUSTOMER.CUSTOMERID, CUSTOMER.LASTNAME
      || ' ' || CUSTOMER.FIRSTNAME
FROM CUSTOMER
WHERE CUSTOMER.CUSTOMERID IN (
      SELECT ORDER_HEADER.CUSTOMERID
      FROM ORDER_HEADER
      WHERE ORDER_HEADER.ORDERDATE < :cutoff_dt
)
ORDER BY 2
```

where `cutoff_dt` is a runtime parameter.

Following is an example SELECT statement with date values:

```
SELECT TO_CHAR(SHIPDATE, 'YYYY.MM.DD') ,
      TO_CHAR(SHIPDATE, 'DD-MON-YYYY')
FROM ORDER_HEADER
```

NOTE For the `id` (the first value in the SELECT statement), you must format the date value. For `label`, the formatting is optional. Note that Oracle requires `MMM` rather than `MON`.

Referencing parameters

You can reference parameters in your project by:

- In Designer, inside SQL, typing `:parameter_name`. For example, the WHERE clause in a rule might reference a parameter named `Min_Months_to_Retain` as follows:

```
add_months("ORDER_HEADER".SHIPDATE,
:Min_Months_to_Retain) < sysdate
```

NOTE In SQL, if the parameter does not already exist, HP Test Data Management creates a parameter with default properties for you. Refer to [Creating parameters automatically](#) (page 59) for information on how to handle automatically created parameters.

- In Designer, inside a Groovy script, typing `parameter_name`.

System parameters

Table 21 lists the system parameters built into HP Test Data Management, which you can reference within your projects.

NOTE The system parameter names are reserved and you cannot create your own parameters with these names. Hence, when you reference a system parameter name, such as `FilterPattern`, Designer will not create a new parameter. It assumes that you meant to reference the system parameter instead.

Table 21 System parameters

| Parameter Name | Data type | Description |
|-----------------------|-----------|---|
| BUSINESS_FLOW_NAME | STRING | Referenceable from: Groovy activity This parameter is the name of the business flow being run. This parameter is useful when querying the HP Test Data Management repository. It holds values such as <code>Orders_BF</code> or <code>emp_BF</code> . |
| BUSINESS_FLOW_VERSION | STRING | Referenceable from: Groovy activity The version of the business flow being run. |
| CURRENT_GROUP_RUN_ID | NUMBER | Referenceable from: Groovy activity This parameter is the group identifier for the job at runtime. This parameter is useful when querying the HP Test Data Management repository. It holds values such as 1, 22, 34, or 42. |
| CURRENT_RUN_ID | NUMBER | Referenceable from: Groovy activity This parameter is the identifier for the job at runtime. This parameter is useful when querying the HP Test Data Management repository. It holds values such as 1, 22, 34, or 42. |

Table 21 System parameters

| Parameter Name | Data type | Description |
|------------------|----------------|--|
| ENVIRONMENT_NAME | STRING | Referenceable from: any Groovy This parameter defines the name of the environment. |
| FilterPattern | STRING | Referenceable from: SQL popup list of values This parameter is the value entered by the user in the list of values. You can use this parameter to narrow your list of values based upon what the user has entered. Note that this parameter only provides the value entered by the user in the list. It does not append any special characters like % or *. |
| INTF_DB | groovy.sql.Sql | Referenceable from: any Groovy This parameter is the database connection for the interface schema. |
| OBT_INTF_DBTYPE | groovy.sql.Sql | Referenceable from: any Groovy This parameter is the database type, for example, Oracle, SQL Server, or Sybase. |
| OBT_INTF_DBVER | groovy.sql.Sql | Referenceable from: any Groovy This parameter is the database version, for example, 10g, 11g, 2008, 12.5, or 2.5. |

Table 21 System parameters

| Parameter Name | Data type | Description |
|-----------------|----------------|---|
| PRODUCT_HOME | groovy.sql.Sql | Referenceable from: any Groovy The install directory of HP Test Data Management. |
| REPOS_DB | groovy.sql.Sql | Referenceable from: Groovy This parameter is the database connection for the repository. You can use it to connect to the repository database where HP Test Data Management stores its metadata. |
| UPDATE_ROWCOUNT | groovy.sql.Sql | Referenceable from: any Groovy The rowcount for the currently running Groovy job. Useful when you have a Groovy script in business flows that is moving rows and you need to update the row count for a particular job run. |

Symbolic schema names

When specifying the schema name of tables, you can use symbolic notation for the schema names rather than hard coding the actual names. These symbolic schema names are valid in SQL fields (for example, in a rule) or Groovy scripts. Furthermore, any schema mappings you have created in Designer are also applied to these tokens.

For queries against the copy of the Oracle database:

```
${HIST.<source_schema_name>.ORDER_HEADER}
```

For queries against the copy of the SQL Server database:

```
${HIST.<database>.<source_schema_name>.ORDER_HEADER}
```

For queries against the Oracle active database:

```
${SOURCE.<source_schema_name>.ORDER_HEADER}
```

For queries against the SQL Server active database:

```
${SOURCE.<database>.<source_schema_name>.ORDER_HEADER}
```

Managing model compatibility

Each model in your project can have one or more dynamic parameters associated with it to verify the compatibility with the extraction environment. If the parameter returns a string of false, then the cartridge referencing the model will not deploy or run and throw an error. For example, the script for the dynamic parameter could return false for Oracle 10.2 and true for Oracle 11.1 to indicate that a cartridge referencing the model can only deploy and run against Oracle 11.1.

To choose dynamic parameters for the evaluation of model compatibility:

- 1 In the Project Navigator, right click a model node and choose **Manage Compatibilities**.
- 2 In the list, select the dynamic parameters that you want to use for model compatibility. Note that only dynamic parameters appear in the list. If the list is empty, then you do not have any dynamic parameters. The parameter must return a string of true or false.
- 3 Click **OK**.

Deleting parameters

Before you delete a parameter, you must ensure that you understand and remove/replace all references to that parameter throughout your project.

- 1 In Designer, find all references in SQL and Groovy to the parameter you wish to delete and remove or replace those references. If you do not remove all references to the parameter prior to deleting it, it will break any SQL or Groovy that references it.
- 2 In the Project Navigator, expand the Parameters node.
- 3 Right click the parameter you wish to delete and select **Delete**.

Preview is a function of Designer used to view the data eligible for copying in your model and cartridge. Preview enables you to confirm that your model or cartridge is behaving as expected before you deploy and run it.

This chapter includes:

- [About preview](#) (page 69)
- [Creating preview rules](#) (page 70)
- [Previewing models and cartridges](#) (page 70)

About preview

Preview provides a quick way to test your test data solution as you are building it. You need not wait until you deploy and run your cartridge to see if the model and cartridge work properly. It is best practice to preview your test data solution at each stage of development and preview it repeatedly until you are certain it performs as expected.

For example, you can preview your model before you create any rules and then preview it again after you add each rule. Preview identifies the rows that are excluded as well as which rule causes their exclusion. When you create your cartridge, you can preview its behavior as well.

With preview, you can determine:

- What data meets the criteria defined by the rules. You can choose at runtime which rules to enforce for that execution. In this way, you can check each rule by itself as well as in combination with the other rules.
- What data is excluded. Preview displays rows that would be excluded from the extraction in red.
- Which rules prevent certain data from being extracted. In preview, a column called Excluded By shows the name of the rule that excluded a particular row. If a row is included, this column is blank for that row.

You can run preview against models and cartridges with eligibility analytics on or off, and with any combination of rules applied. If you see differences between the preview for a model and a cartridge based on that model, it is most likely because the cartridge is not using the same combination of rules as the model.

If your model contains chaining logic, the data retrieved by preview will differ from the data that will be selected for extraction by the cartridge.

If you have a large amounts of data and are frequently previewing, you can create rules that are Preview Only to improve performance.

NOTE If you're connecting to a non-native data source (for example, a database other than Oracle, DB2, Sybase or SQL Server), then Preview may not be available. If you used the driver certification utility to set up the JDBC driver, and the driver supports Preview, then Preview will be available.

Creating preview rules

When in the development phase for your test data solution, a quick response time is helpful because you are frequently making model enhancements and previewing them. Limiting the data returned for testing purposes is often worthwhile. A preview rule enables you to define a rule that applies only in preview in Designer. The rule is not applied when you generate and deploy your cartridge for actual usage.

To create a preview rule:

- 1 Right click a table in the model and select Add Rule from the pop-up menu.
- 2 For Usage, select **Preview Only**.
- 3 Define the other properties for your rule. For example, in Oracle, you might enter a WHERE clause such as this one to limit the number of rows returned:

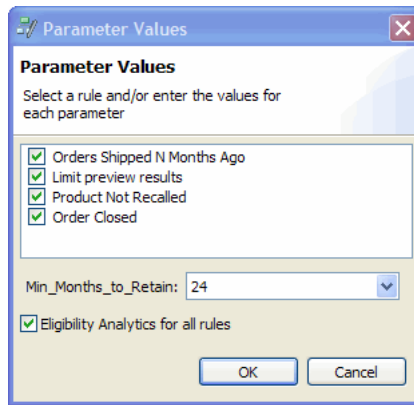
```
ORDER_ID in (100, 132, 5540)
```
- 4 Click **OK**.
- 5 Click Preview in the toolbar to confirm that fewer rows are returned in the Preview tab.

Previewing models and cartridges

To preview a model or cartridge:

- 1 Open a model or cartridge in Designer.
- 2 Click Preview in the tool bar.
- 3 In the Parameters Values window, check the rules to apply and uncheck those to ignore for this run.
- 4 If you have any parameters, type/choose the values in the Parameter Values window.
- 5 Check or uncheck **Eligibility Analytics for all rules**, depending upon whether you want them.

When checked, Designer creates analytics for each rule, regardless of what is set for the individual rules in the Rule dialog box. Enabling analytics causes the ineligible data to appear in red in the preview.



6 Click **OK**.

The Preview pane opens.

The screenshot shows the 'Orders Data' preview pane. The top part is a table with columns: #, Excluded By, ORDERID, CUSTOMERID, and ORDERDATE. The bottom part is a model structure pane showing 'ORDER_HEADER (3000)'.

| # | Excluded By | ORDERID | CUSTOMERID | ORDERDATE |
|----|----------------------|---------|------------|-----------------------|
| 59 | Product Not Recalled | 2797 | 529 | 1995-11-21 00:00:00.0 |
| 60 | Product Not Recalled | 2818 | 550 | 1995-08-01 00:00:00.0 |
| 61 | Product Not Recalled | 2891 | 56 | 1995-02-20 00:00:00.0 |
| 62 | Product Not Recalled | 216 | 216 | 1994-06-13 00:00:00.0 |
| 63 | | 1 | 1 | 1994-08-18 00:00:00.0 |
| 64 | | 2 | 2 | 1994-08-27 00:00:00.0 |
| 65 | | 3 | 3 | 1994-04-20 00:00:00.0 |
| 66 | | 4 | 4 | 1994-02-13 00:00:00.0 |
| 67 | | 5 | 5 | 1994-08-28 00:00:00.0 |
| 68 | | 6 | 6 | 1994-05-16 00:00:00.0 |
| 69 | | 7 | 7 | 1994-04-21 00:00:00.0 |
| 70 | | 8 | 8 | 1994-01-15 00:00:00.0 |
| 71 | | 9 | 9 | 1994-07-13 00:00:00.0 |

| # | Excluded By | ORDERID | CUSTOMERID | ORDERDATE |
|---|-------------------|---------|------------|-----------|
| 1 | Product Not Re... | 216 | 216 | 1994-0 |
| 2 | | 2 | 2 | 1994-0 |
| 3 | | 1 | 1 | 1994-0 |

- The top part of the window shows the rows of the driving table. Select a row or range of rows in the top part of the window to filter the rows displayed in the bottom part. Use Ctrl-click to select more than one row or clear the rows selected.
- The Excluded By column displays the rule that caused a row to be excluded. All rows that are excluded are displayed in red.
- Click on column headers to sort the rows by that value. For example, if you click the Excluded By column header, the rows are sorted according to the values of that column.
- Click and drag the column borders to resize the columns in the display.
- The model structure, including rules, is displayed in the lower left pane. Expand and collapse the node to view the tables and rule you want. Positive numbers in parentheses next to the driving table indicate the

number of driving table rows included by that table. Negative numbers in parentheses next to a rule indicate the number of driving table rows excluded by the rule.

- 7 If you want to export the contents of the top table to a comma delimited file, right click a row and choose **Export Data**. If you want to export the contents of the bottom table to a comma delimited file, right click on a table node in the lower left pane and choose **Export Details Data**.
- 8 Click **Refresh** from the toolbar to refresh the data from the database.
- 9 When you are done with this instance of preview, click the Close icon on the tab.

After you have your model and rules defined, you must consider how to deploy that solution for copying data to your structured data file. A cartridge is an instance of model- or schema-based eligibility criteria used to copy data from one location to another. The cartridge enables you to specify a set of characteristics to apply to a particular data movement activity.

This chapter includes:

- [About cartridges](#) (page 73)
- [Creating a new cartridge](#) (page 73)
- [Editing a database to file cartridge](#) (page 74)
- [Editing a schema-based database to file cartridge](#) (page 80)
- [Applying data masks](#) (page 85)
- [Deleting cartridges](#) (page 99)

About cartridges

HP Test Data Management has the following types of cartridges:

- **Database to file cartridges**
Copy data from a database to a structured file in either XML or comma separated values (CSV) format.
- **Schema-based database to file cartridges**
Extract data from a specified set of tables (without use of a model) to a file (XML or CSV). Schema-based cartridges are useful when you simply want to extract a set of tables without specifying relationships between them (no model).

Creating a new cartridge

To create a new cartridge:

- 1 Select **Cartridges** in the Project Navigator, or select an existing model or cartridge.
- 2 Right click and select **New Cartridge**.
- 3 Type a name for the Cartridge (for example, salesorder).

The name you select appears in the Deployment Assistant and in the Web Console.

- 4 For Type, select Database to File.
- 5 For Source, choose Schema or Model. For information about the difference between schema-based and model-based movement, refer to [About cartridges](#) (page 73).
- 6 If you chose Model in the previous step, select a model from the list.
- 7 Click **OK** to create your new cartridge.

The cartridge opens in the editor.

- 8 Edit the values. For more information, refer to the following sections:
 - [Editing a database to file cartridge](#) (page 74).
 - [Editing a schema-based database to file cartridge](#) (page 80)

TIP You can make annotations from each tab of the cartridge editor according to the behavior described in [Annotating cartridges](#) (page 122).

Editing a database to file cartridge

The database to file cartridge editor consists of the following tabs:

- Overview
- Operations
- Data Masking
- File Indexes
- Column Inclusion
- Name Override
- Custom Properties
- Custom Selection Program

You can edit the information on each tab at any point during the cartridge editing process, as well as edit the model that is associated with the cartridge.

To edit a database to file cartridge:

- 1 Expand Cartridges in the Project Navigator window.
- 2 Select the database to file cartridge that you want to edit.

The Database to File Cartridge page opens in the editor.

Database to File Cartridge

Model: [Orders](#)

Version: [Annotation...](#)

Selection Rules

☒ Order Closed

☒ Orders Shipped N Months Ago

☒ Product Not Recalled

[Select All](#)

[Deselect All](#)

Operations

Extracted:

- BUNIT
- COMPLAN
- CUSTOMER
- ORDER_ATTACHMENT
- ORDER_ATTACHMENT_2
- ORDER_HEADER
- ORDER_LINE
- ORDER_LINE_DIST
- ORDER_TAX
- PRODUCT
- SALESREP
- STATUS
- STORE

Not Extracted: No Table Uses Selected.

Overview | **Operations** | Data Masking | File Indexes | Column Inclusion | Name Override | Custom Properties | Custom Selection Program

3 Type or select values for the following as necessary:

Table 22 Overview properties

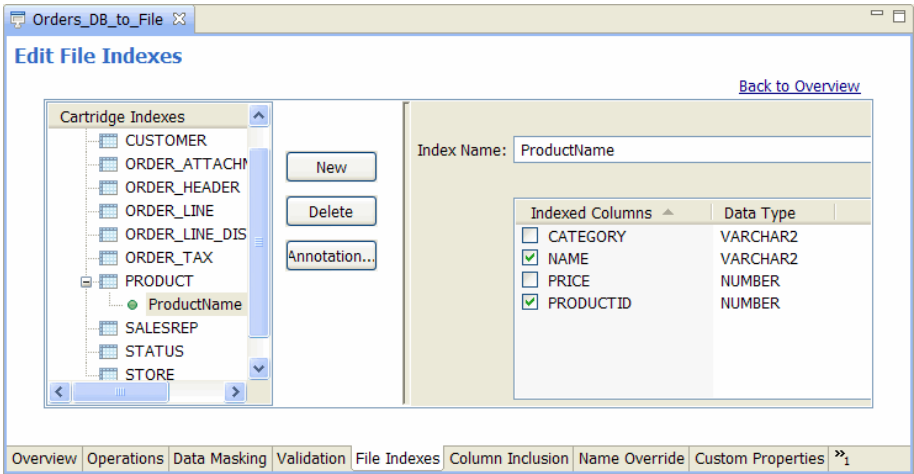
| Field | Description |
|-----------------|---|
| Version | Type a version number for the cartridge or accept the default. |
| Selection Rules | Select the rules to be applied to the cartridge. A model can have any number of rules, and you may or may not want to apply each of them to any particular cartridge. |

- 4 Select the **Operations** tab. The Edit Operations page opens.
- 5 Select the **Data Masking** tab. The Edit Data Masking page opens and you can apply data masks to particular columns as necessary to protect privacy or hide confidential information, such as Social Security Numbers and credit card numbers. Refer to [Applying data masks](#) (page 85).
- 6 Select the **File Indexes** tab. Use this tab to specify indexes for your files for your database to file cartridge. The primary benefit of the file index is that, once the data is extracted, you can easily discover what data is in which file. For example, if you create a file index on customer name, you can find out easily what files contain orders from a particular customer.

You cannot index LOB columns. Furthermore, if a column is indexed, it must also be included on the Column Inclusion tab.

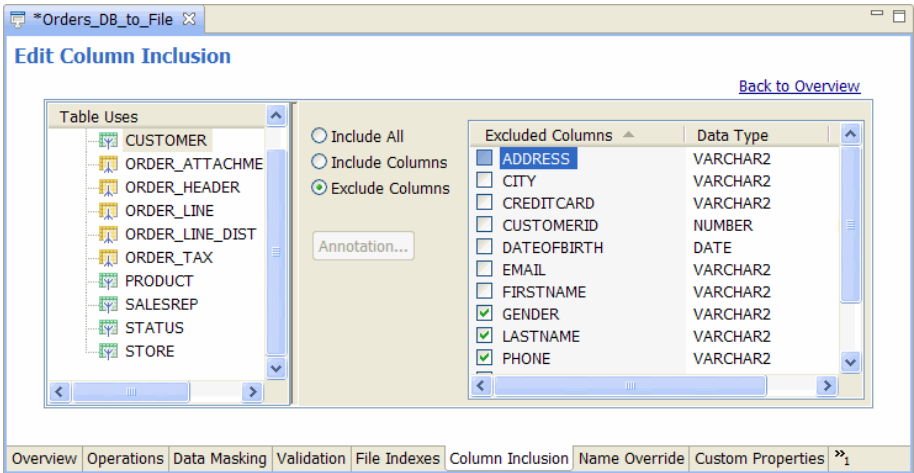
NOTE File indexes are not supported by non-intrusive environments. Rather, in non-intrusive environments, file indexes are ignored during deployment and execution.

See also *HP Test Data Management Web Console User Guide*



7 Select the **Column Inclusion** tab. The Edit Column Inclusion page opens.

TIP If you indexed a column from the File Indexes tab, you cannot exclude it from the Column Inclusion tab.



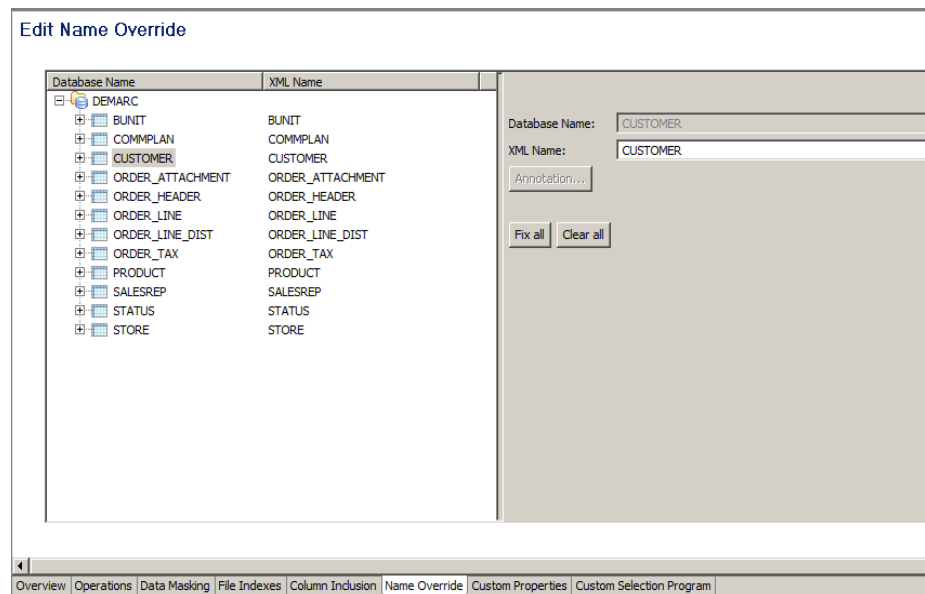
Use this tab to specify how table columns are included or excluded by your cartridge.

Table 23 Column inclusion options

| Select | To |
|-----------------|--|
| Include All | Copies all the columns found at deployment even if the column was not part of the table definition during the design of the cartridge. |
| Include Columns | Ensure that only the columns you specify are copied by the cartridge. |
| Exclude Columns | Ensure that specific columns are excluded from the copy. This option automatically copies all the other columns found at installation and their data, even if they were not part of the table definition during design of the cartridge. |

TIP Include All is not equivalent to selecting Include Columns and explicitly checking all columns. The former includes all columns found at deployment time. The latter only includes the columns you selected at development time. For example, if your development database contains only a subset of the columns in your production database, then selecting Include Columns with all columns checked may not extract all the columns in your production database.

- 8 Select the **Name Override** tab. The Edit Name Override dialog displays.



Use this tab to define the element names to be used in the XML file. By default the element names are identical to the table and column names.

The Name Override alters how a table or column name is represented in the XML document. A Name Override is necessary whenever a table or column name is not a valid XML element name, for example, multi-byte character sets (MBCS).

If you wish to fix all the XML tags for all the tables in a cartridge according to your specifications, then click **Fix all**.

If you wish to restore all default values, then click **Clear all**.

NOTE If you create a name override for a column, the column is included in the XML output, regardless of whether you excluded the column on the Column Inclusion tab.

Related information

See <http://www.w3.org/TR/REC-xml/#NT-Name> for information on valid XML element names.

- a** Expand the database in the Database Name list, and select a column or table name.
 - b** Edit the value in the XML Name field so that the name is compatible with XML element naming conventions.
- 9** Select the **Custom Properties** tab.
- 10** In Custom Properties, you can optionally create, annotate, and delete name-value pairs that provide additional information about your data. Such information could be used by other applications to classify or otherwise operate on the data.
 - a** Click New to create a new custom property.
 - b** Enter a Name for the property.
 - c** Enter a Value for the property.
 - d** Click Annotation to add a comment to the property.
 - e** Use the Delete button to get rid of the currently selected property, if you do not need it.

TIP Each tab of the cartridge editor has its own Annotation button to enable you to enter comments. Refer to [Annotating projects](#) (page 122) for more information.

See also [Generating and deploying](#) (page 145) for information on generating and deploying your cartridge.

Output files

When you extract data to a file, you can extract it as XML or CSV files. The generated XML or CSV files are described in this section.

- [XML output files](#) (page 79)
- [CSV output files](#) (page 80)

XML output files

When you extract to XML, the following files are created:

| File | Description |
|-------------|---|
| summary XML | Contains summary information about the extraction process that was run to create the files. This information includes: <ul style="list-style-type: none">• what files were created• what data was extracted• where the data was extracted from• how much data was extracted |
| summary XSD | XML Schema Definition file that defines the structure and makeup of the summary XML file. |
| group XML | <p>The group XML file contains the following information:</p> <ul style="list-style-type: none">• information about the extraction process used to generate the extracted data within each particular XML file• the extracted database data in XML form <p>The number of group XML files depends on the data movement batch size. For example, if you set the batch size to 100, and there are 1000 driving table rows being extracted, then 10 group XML files are generated. Refer to the <i>HP Test Data Management Web Console and Query Server User Guide</i> for more information.</p> |
| group XSD | XML Schema Definition file that defines the structure and makeup of the group XML file. |

All the information needed to interpret the structure of the test data is contained in these files, making the XML stand-alone for many years to come.

CSV output files

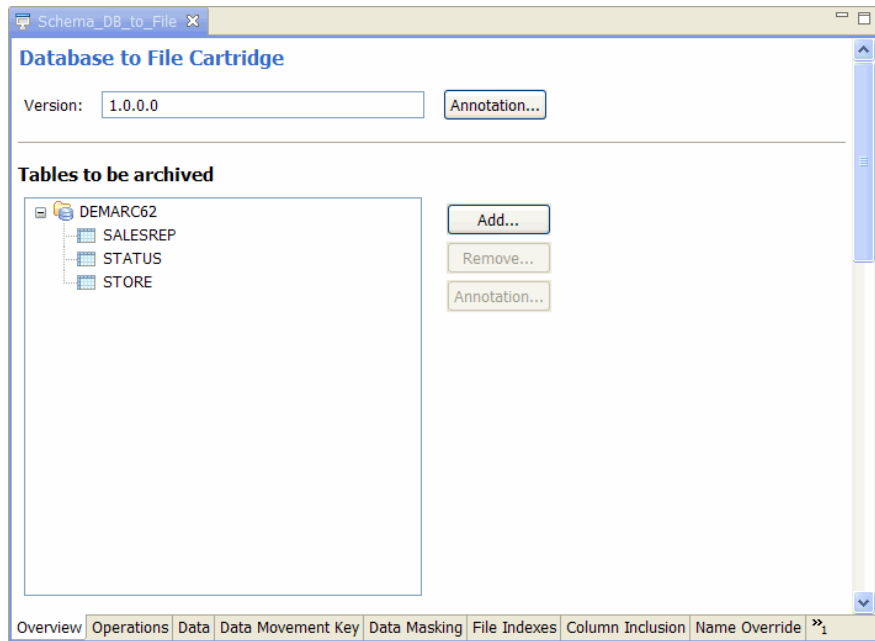
When you extract to CSV, the following files are created:

| File | Description |
|-------------|---|
| summary XML | Contains summary information about the extraction process that was run to create the files. This information includes: <ul style="list-style-type: none">• what files were created• what data was extracted• where the data was copied from• how much data was extracted |
| summary XSD | XML Schema Definition file that defines the structure and makeup of the summary XML file. |
| group XML | The group XML file contains a list of all of the CSV files for the group. |
| group CSV | The group CSV file contains the extracted database data in CSV format. Each group has a set of CSV files for each table use. For example, if you have 10 groups and 3 table uses, you will have 30 group CSV files. Refer to the <i>HP Test Data Management Web Console User Guide</i> for more information. |

Editing a schema-based database to file cartridge

To edit a schema-based database to file cartridge:

- 1 Expand Cartridges in the Project Navigator window.
- 2 Select the schema-based database to file cartridge that you want to edit. The Database to File cartridge page opens in the editor.

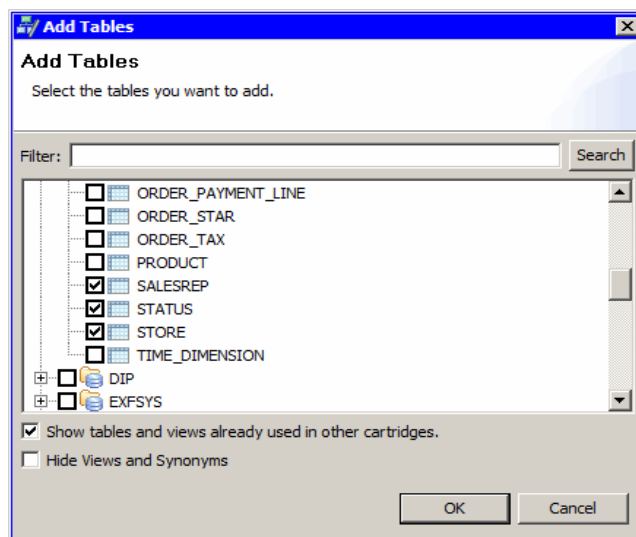


- 3 Type or select values for the following as necessary:

Table 24 Overview properties

| Field | Description |
|---------|--|
| Version | Type a version number for the cartridge or accept the default. |

- 4 Click **Add** to display the Add Tables dialog box and select tables to be added to the extraction process.



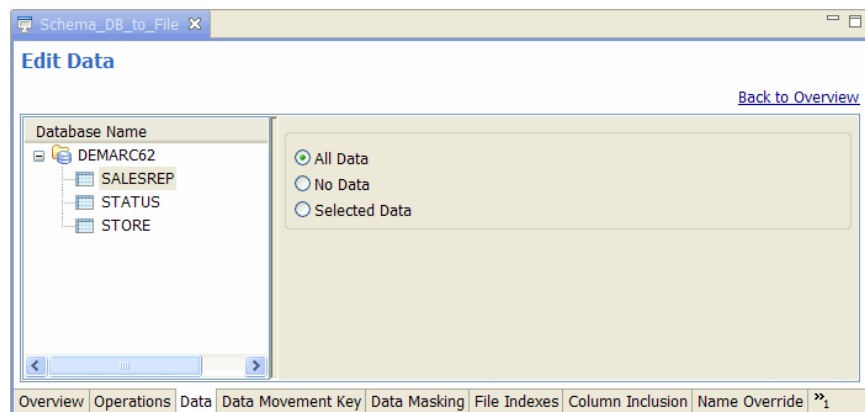
TIP You can extract data from different database schemas (using different access rights) by creating a multi-source data model—this enables you to extract the data from all the schemas at once. Access rights only have an impact if you build the model in Designer.

If you build the model in Designer, then you would need one account with read access to all the schemas you wish to include.

If the schemas are located in different databases, then you can use either views or database links to access the data (although this can impact performance), or, you can create multiple models or cartridges and combine them into a single business flow and run them sequentially.

NOTE You can extract data that is stored in a BLOB and to upload it to the same or different database or schema.

- 5 Click **OK**.
- 6 If necessary, click **Remove** and select tables for removal.
- 7 Select the **Operations** tab. The Edit Operations page opens.
- 8 Select the **Data** tab.

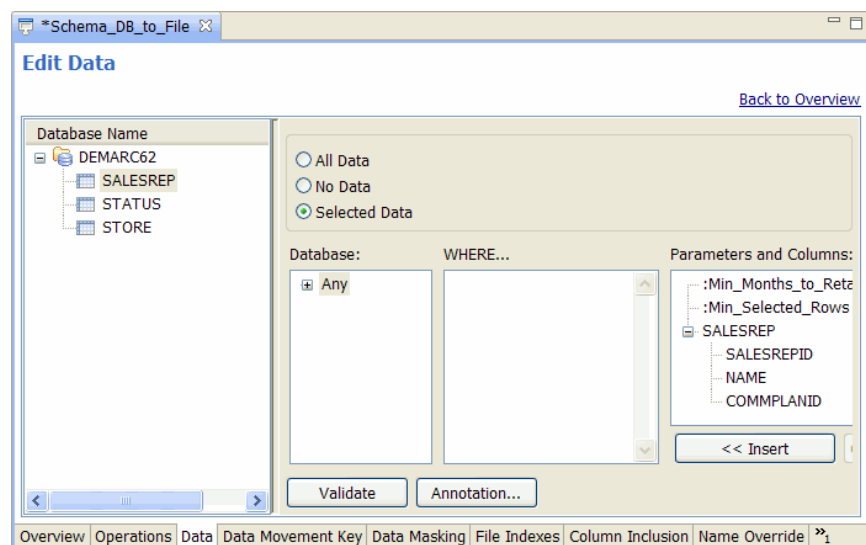


- 9 Choose one of the radio buttons:
 - **All Data** means that the table will be created in the test data store and populated with data.
 - **No Data** means that only XSD files are generated. No data files are generated in this case. This option is useful in cases where you need the tables present in order for the application to run, but you do not necessarily need the data.
 - **Selected Data** means that the table will be created in the test data store and a WHERE clause used to populate it with selected data from the active database.

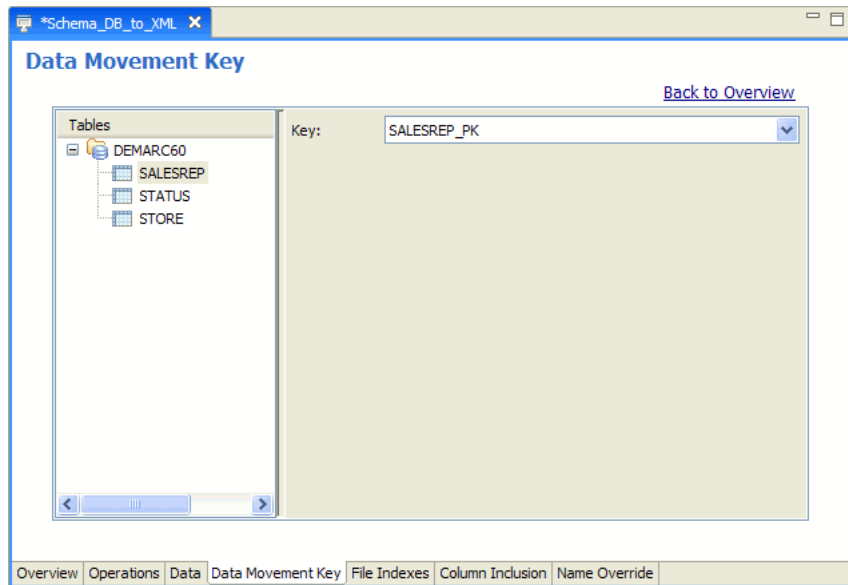
If you choose this radio button, type or select values for the following as necessary:

Table 25 WHERE clause for choosing data

| Field | Description |
|----------|--|
| Database | <p>Choose the database to which the WHERE clause applies. You can have a different WHERE clause for as many of the vendors and versions as necessary. For example, you could associate a different WHERE clause with DB2, SQL Server, Oracle 8i, and Oracle 9i. At runtime, HP Test Data Management can determine the database vendor and version, and apply the clause that you assigned to that vendor and version.</p> <ul style="list-style-type: none"> Choose Any for the default WHERE clause. The Any WHERE clause is used for any vendor and version which does not have an assigned WHERE clause. For example, if you do not assign a WHERE clause to Oracle or any of its versions, the Any WHERE clause will be used when deploying against Oracle. If the clause has vendor or version specific calls or syntax in it, expand the Any node until you see the necessary option for the clause and select it. |
| WHERE | Type or edit the WHERE clause. |
| Columns | Double-click the column name to insert the name into the WHERE clause. |
| Insert | Alternatively, highlight a column and use the <<Insert button to move it to the WHERE clause. |



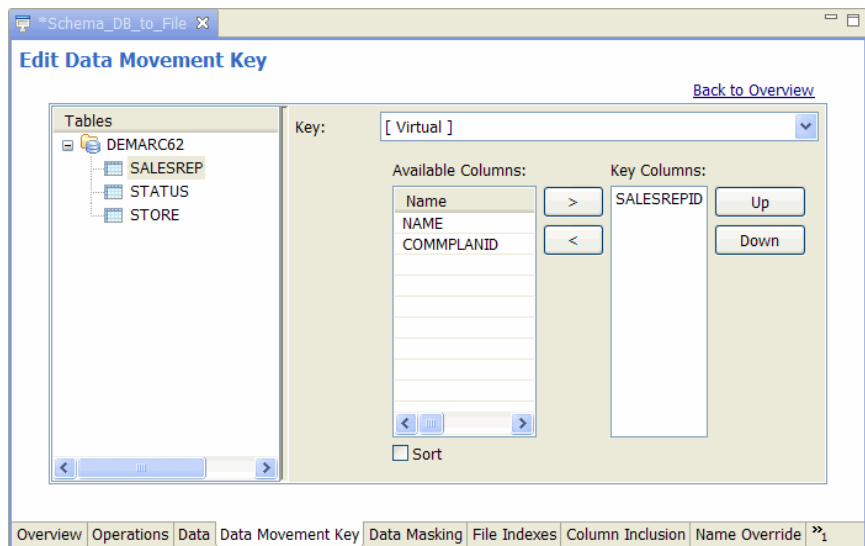
10 Select the **Data Movement Key** tab.



- 11 Choose an option from the Key list of values, the table's primary key, other unique keys in the table, None, or Virtual Key. If you choose Virtual Key, you must specify the columns for the virtual key to use.

TIP For SQL Server, you cannot choose None because it requires a data movement key.

NOTE Designer requires a data movement key for all table uses in a model.



- 12 Select the **Data Masking** tab. The Edit Data Masking page opens and you can apply data masks to particular columns as necessary to protect privacy or hide confidential information, such as Social Security Numbers and credit card numbers. Refer to [Applying data masks](#) (page 85).

- 13 Select the **File Indexes** tab. The Edit File Indexes page behaves similarly to the Edit File Indexes page for a database to file cartridge, [Editing a database to file cartridge](#) (page 74).
- 14 Select the **Column Inclusion** tab. The Edit Column Inclusion page behaves similarly to the Edit File Indexes page for a database to file cartridge, [Editing a database to file cartridge](#) (page 74).
- 15 Select the **Name Override** tab. The Edit Name Override page behaves similarly to the Edit Name Override page for a database to file cartridge, [Editing a database to file cartridge](#) (page 74).
- 16 Select the **Custom Properties** tab. The Custom Properties page opens and behaves similarly to the Custom Properties page for a database to file cartridge, [Editing a database to file cartridge](#) (page 74).

Applying data masks

By applying a data mask to a column, you can obfuscate its values in the extracted data, thus protecting private, confidential information, such as credit card or social security numbers.

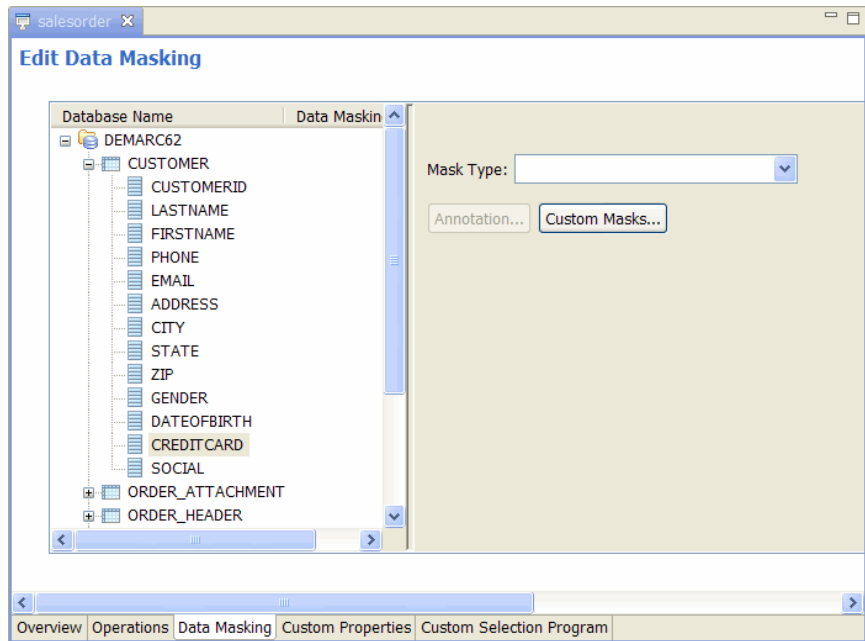
TIP Values are not masked for Designer preview. They are only masked when running the business flow outside of Designer from Web Console or command line.

HP Test Data Management provides pre-built masks for the data which most commonly requires protection, such as, credit card numbers and Social Security numbers. In addition, you can create your own, custom masks for data that you wish to mask.

See also [Chapter 6, Working with data masking.](#)

To apply a data mask:

- 1 In the cartridge editor, select the **Data Masking** tab. The Edit Data Masking page opens.



- 2 If you need to apply a mask to a column, expand the table node that contains the column, for example, CUSTOMER.
- 3 Select the table column, for example, CREDITCARD.
- 4 Choose a value from **Mask Type** or, if you are using a mask you defined yourself, click **Custom Masks**. Refer to [Pre-built masks](#) (page 86) for more information.

Before you choose your data mask, consider all of the following:

- If you choose to mask a primary key, you need to ensure that the masked values are unique (a mapped mask). Otherwise, when you upload the table, it may no longer behave as expected
- You must ensure that the data mask you apply corresponds to the data type of the column. For example, if the column is a numeric value, you must apply a numeric mask to it or use a function to convert the data to a data type that is compatible with the mask. Otherwise, the mask will fail with a SQL error upon deployment.
- If you apply a mask on a primary and/or foreign key, then you should also apply the same mask on a referenced primary key to ensure database integrity.

Pre-built masks

[Table 26](#) lists the string masks provided by HP Test Data Management.

WARNING! If your column contains any invalid data, such as invalid characters, the Social Security, credit card, and ABA/Routing string masks will not mask the value at all. For example, if you have a Social Security Number value that contains an invalid special character like #, none of that value will be masked in the extracted files. Hence, if invalid data is an issue in your columns, you may wish to create a custom mask that includes special logic for handling invalid data as you desire.

TIP The only reversible string mask is String map (DB2 and Oracle only).

NOTE The pre-built data masking functions support generic data types, such as VARCHAR, CHAR, NUMBER, FLOAT, and so on. The pre-built masks will not work on columns that are database specific, such as unique identifier in SQL Server.

Table 26 Pre-built data masks—Strings

| Mask | Description |
|------------------------------|--|
| ABA/Routing number: random | Validates that the given ABA number adheres to a supported format and, if it does, returns a new valid random ABA number. Otherwise, it returns the same input number. ^a This mask supports a nine digit ABA number of the format XXXXXXXXX, where X is any digit [0-9]. |
| Credit card number: random | Validates that the given credit card number adheres to a supported format and, if it does, returns a new, random credit card number of the same type. Otherwise, it returns the same input number. ^a This mask supports Visa, Master Card, American Express, and Discover card numbers. Refer to Credit card masks (page 94) for more information. |
| Credit card number: XXX mask | Validates that the given credit card number adheres to a supported format and, if it does, returns the original number with all digits masked by X, except for the last four digits. Otherwise, the mask returns the same input number. ^a This mask supports Visa, Master Card, American Express, and Discover card numbers. Refer to Credit card masks (page 94) for more information. |

Table 26 Pre-built data masks—Strings

| Mask | Description |
|----------------------------------|---|
| Social Security Number: random | <p>Validates the given US Social Security number adheres to a supported format and, if it does, returns a new, random Social Security Number starting with 897, which is not assigned to any individual and can be safely used. Otherwise, the mask returns the same input number.^a</p> <p>This mask supports both formats, xxx-xx-xxxx and xxxxxxxxx. If the input contains dashes, this function returns a number with dashes inserted in the 4th and 7th place.</p> |
| Social Security Number: XXX mask | <p>Validates that the given US Social Security number adheres to a supported format and, if it does, replaces all digits from the original number with letter X, except the last four digits. Otherwise, the mask returns the same input number.^a</p> <p>This mask supports both formats, xxx-xx-xxxx and xxxxxxxxx. If the input contains dashes, this function returns a number with dashes inserted in the 4th and 7th place.</p> |
| String mask: random length | <p>Returns a string of random length, bounded by a user-specified maximum, using a user-specified character.</p> <ul style="list-style-type: none"> • Character (Data type Character) is the character to use in the string. You can specify any printable character. • Length (Data type Integer) is the maximum allowable length of the string. The generated string will have a random length between 1 and the value of Length. |
| String mask: same length | <p>Returns a string of the same length using a user-specified character.</p> <ul style="list-style-type: none"> • Character (Data type Character) is the character to use in the string. You can specify any printable character. |
| Random string | <p>Returns a string of random length, bounded by a user-specified maximum, using random, printable characters.</p> <ul style="list-style-type: none"> • Length (Data type Integer) is the maximum allowable length of the string. The generated string will have a random length between 1 and the value of Length. |

Table 26 Pre-built data masks—Strings

| Mask | Description |
|----------------------------|---|
| Random string: [a-z] | <p>Returns a string of random length, bounded by a user-specified maximum, using only random lowercase, English alphabetic characters.</p> <ul style="list-style-type: none">• Length (Data type Integer) is the maximum allowable length of the string. The generated string will have a random length between 1 and the value of Length. |
| Random string: [A-Z] | <p>Returns a string of random length, bounded by a user-specified maximum, using only random uppercase, English alphabetic characters.</p> <ul style="list-style-type: none">• Length (Data type Integer) is the maximum allowable length of the string. The generated string will have a random length between 1 and the value of Length. |
| Random string: [a-zA-Z] | <p>Returns a string of random length, bounded by a user-specified maximum, using only random English, alphabetic characters.</p> <ul style="list-style-type: none">• Length (Data type Integer) is the maximum allowable length of the string. The generated string will have a random length between 1 and the value of Length. |
| Random string: [a-zA-Z0-9] | <p>Returns a string of random length, bounded by a user-specified maximum, using only random English, alphanumeric characters.</p> <ul style="list-style-type: none">• Length (Data type Integer) is the maximum allowable length of the string. The generated string will have a random length between 1 and the value of Length. |

Table 26 Pre-built data masks—Strings

| Mask | Description |
|--|--|
| String map (DB2 and Oracle only, standard environment) | <p>Looks up a given string in a map and returns the value to which it is mapped. If a string is not mapped to any value, then a null string is returned.</p> <p>This mask is reversible upon upload and thus the mapping must be complete (encompassing all values). If the mapping is not complete, an exception is thrown during upload.</p> <p>The mapping is stored in a lookup table, which you must create and populate prior to using this function. This mask supports VARCHAR columns up to 256 characters in length. It is best employed on short strings, such as zip code, driver's license number, and telephone number.</p> <p>The data type and lengths of the columns in the lookup table must match those of the columns in the managed tables. For example, if the column in the source table is VARCHAR(100), then the corresponding column in the lookup table must be VARCHAR(100) as well.</p> <ul style="list-style-type: none">• Map table name (Data type VARCHAR) is the name of a lookup table. A lookup table has only two columns, both of type VARCHAR with a maximum size of 256 characters:<ul style="list-style-type: none">— The column named <code>orig</code> holds the key.— The column named <code>masked</code> holds the value. <p>The lookup table must be created and populated before using this mask. In single instance, database to file extractions, the lookup table must be created in the source database.</p> |

Table 26 Pre-built data masks—Strings

| Mask | Description |
|---|--|
| String map (non-intrusive environment only) | <p>In non-intrusive environments, you must enter the String map information in a file where each line follows the format <code>key=value</code>, and key is mapped to the value by the mask. You must also specify the absolute path to this mapping file in the <code>MAP_TABLENAME</code> parameter. Alternatively, you can place it in the default location <code>(obt\extensions\runtime\masking)</code> and the value of the <code>MAP_TABLENAME</code> parameter is then just the filename of the mapping file. The format of the mapping file is:</p> <pre>key1=val1 key2=val2</pre> <p>Enter only one key/value pair per line.</p> |
| Fixed string | <p>Returns a constant, user-specified string.</p> <ul style="list-style-type: none">• Value (Data type String) is the string with which to replace all values. This replacement string should be compatible with the column's data type, for example, it should match the maximum length of the column. <p>a. Because returning the same number could be a security issue, you should confirm that the numbers are in valid format prior to running with this mask.</p> |

Table 27 lists the numeric masks provided by HP Test Data Management.

TIP The only reversible numeric masks are skew number: add and number map (DB2 and Oracle only).

Table 27 Pre-built data masks—Numeric

| Mask | Description |
|-----------------------|--|
| Fixed number | Returns a user-specified constant, numeric value. <ul style="list-style-type: none">• Value (Data type Numeric) is the constant with which to replace the column's values. This number must be compatible with the column data type, for example, the precision and scale must match. |
| Random number | Returns a random number within a user-specified range of values, both inclusive. <ul style="list-style-type: none">• Minimum (Data type Integer) is the lower bound of the range.• Maximum (Data type Integer) is the upper bound of the range. |
| Skew number: add | Returns a skewed value, the original value plus a user-specified number. <ul style="list-style-type: none">• Amount (Data type Integer) is the number by which to increment the original value. It must be a positive or negative whole number. <p>This function is reversible and supports whole numbers on all database platforms. If you attempt to apply this mask to other column types (for example, decimals or strings), then you will encounter an error when deploying the cartridge. For Oracle only, all types of numeric columns—including decimals—are supported.</p> |
| Skew number: multiply | Returns a skewed value, the original number multiplied by a user-specified number. <ul style="list-style-type: none">• Amount (Data type Numeric) is the number by which to multiply the original value. <p>For all databases (except Oracle), the result is rounded off to the nearest whole number.</p> <p>For example, if you input 10.1, then a skew amount of 20.9 will return a result of 200.</p> |

Table 27 Pre-built data masks—Numeric

| Mask | Description |
|--|---|
| Skew number: percent | <p>Returns a skewed value, the original number increased by a user-specified percentage. For example, to increment a number by 10%, enter 10 for the Percent.</p> <ul style="list-style-type: none"> • Percent (Data type Numeric) is the percentage by which the original value will be increased. <p>For all databases (except Oracle), the result is rounded off to the nearest whole number.</p> |
| Number map (DB2 and Oracle standard environments only) | <p>Looks up a given number in a map and returns the value to which it is mapped.</p> <ul style="list-style-type: none"> • Map table name (Data type VARCHAR) is the name of a lookup table. A lookup table has only two columns, both of type VARCHAR: <ul style="list-style-type: none"> — The column named orig holds the key. — The column named masked holds the value. <p>The lookup table must be created and populated before using this mask. In single instance, database to file extractions, the lookup table must be created in the source database.</p> |
| Number map (non-intrusive environments only) | <p>In non-intrusive environments, you must enter the Number map information in a file where each line follows the format <code>key=value</code>, where key is mapped to the value by the mask. You must also specify the absolute path to this mapping file in the MAP_TABLENAME parameter. Alternatively, you can place it in the default location (obt\extensions\runtime\masking) and the value of the MAP_TABLENAME parameter is then just the filename of the mapping file. The format of the mapping file is:</p> <pre>key1=val1 key2=val2</pre> <p>Enter only one key/value pair per line.</p> |

Table 28 lists the date masks provided by HP Test Data Management.

Table 28 Pre-built data masks—Date

| Mask | Description |
|-----------|---|
| Skew date | <p>Returns a skewed value, the original date plus a user-specified number of days.</p> <ul style="list-style-type: none"> Days (Data type Integer) is the number of days by which to increment the original date. <p>This function is reversible.</p> |

Credit card masks

Table 29 lists the credit card types and formats for which HP Test Data Management supports masking.

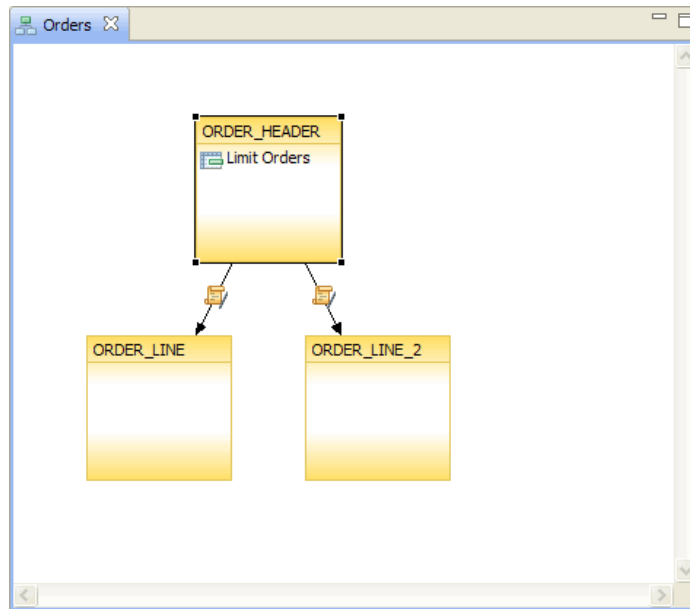
Table 29 Supported credit card types and formats

| Card type | Prefix | Length | Format |
|---------------------------|---------|--------|---------------------|
| American Express | 34,36 | 15 | 123412345612345 |
| | | 17 | 1234-123456-12345 |
| | | 17 | 1234 123456 12345 |
| Discover | 6011 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |
| Mastercard | 51-55 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |
| Visa | 4 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |
| | | 13 | 1234123412345 |
| Diners Card International | 36 | 14 | 12341234561234 |
| | | 16 | 1234-123456-1234 |
| | | 16 | 1234 123456 1234 |
| Diners Card Carte Blanche | 300-305 | 14 | 123412345631234 |
| | | 16 | 1234-123456-1234 |
| | | 16 | 1234 123456 1234 |

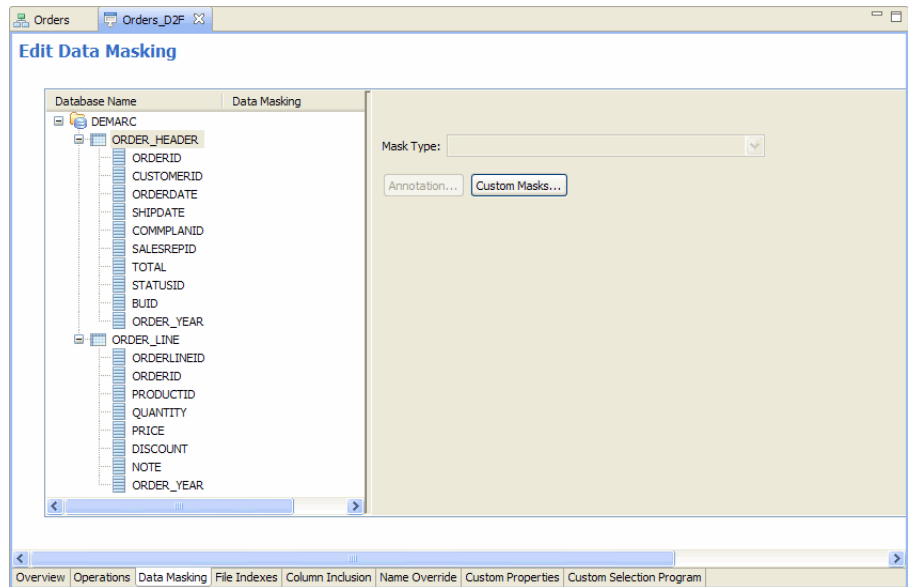
Data masking and referential integrity

To maintain referential integrity in your test database, you must take special care when applying data masks to primary or foreign key columns. Because referential integrity relies on unique keys—both primary and foreign—you must ensure that you mask them consistently. Otherwise, upload will not work as expected when you upload the data into another database. Also note that masks on primary keys must be one-to-one in order to preserve the uniqueness of primary keys.

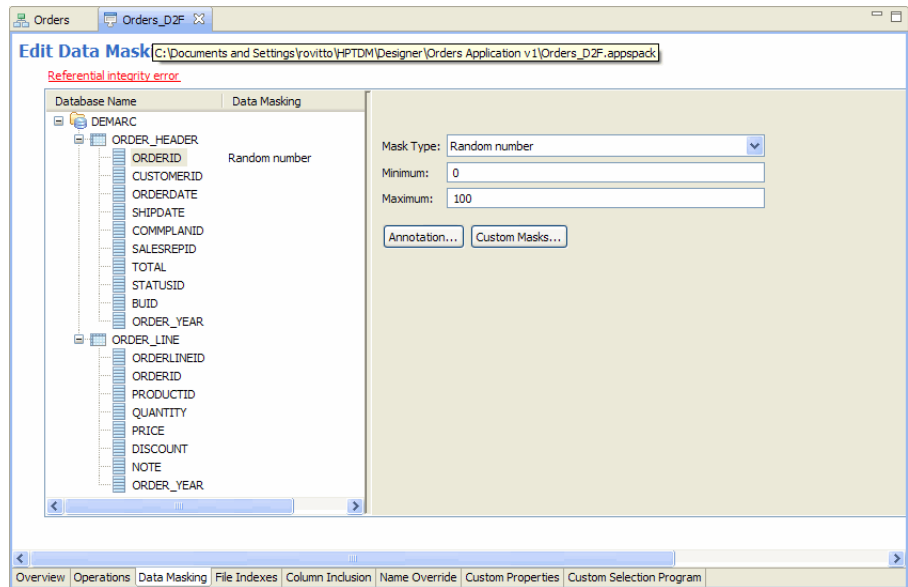
Example Suppose that you have a simple data model such as the following one. Notice that ORDER_LINE is used twice as a child transactional table in this model.



In a cartridge based on this model, you only see two tables in the Data Masking tab. Data masking is applied on a per table basis, not per table use. Hence, in the Data Masking tab, ORDER_LINE appears only once and any masks you set for it are applied to all of its table uses in the cartridge.



If you choose to mask the column containing the primary key (ORDERID) in ORDER_HEADER, you must also mask the column containing the corresponding foreign key in ORDER_LINE with a reversible mask to preserve the primary key constraints (uniqueness). If you try to apply a non-reversible mask or do not mask the column containing the foreign key as well, Designer issues a warning.

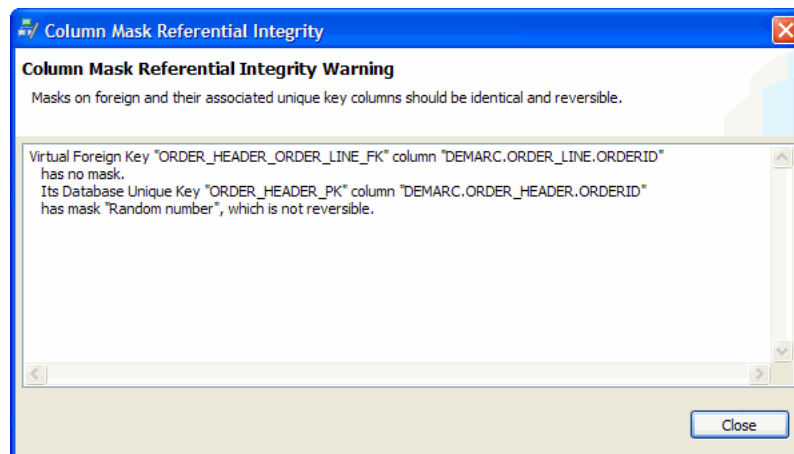


Notice the Referential integrity error link that appears in the upper left corner of the page. Click this link to view a more detailed error message.

TIP If you choose a custom mask on a column containing the primary key, Designer cannot analyze your masking function's logic to determine whether or not it is one-to-one and thus will not issue a warning on a column's primary key masks. It is up to you to verify that the mask's logic is one-to-one. However, a referential integrity warning will be raised when custom masks on columns containing primary and foreign keys are different.

To assure referential integrity when assigning a mask on a foreign key column, ensure that:

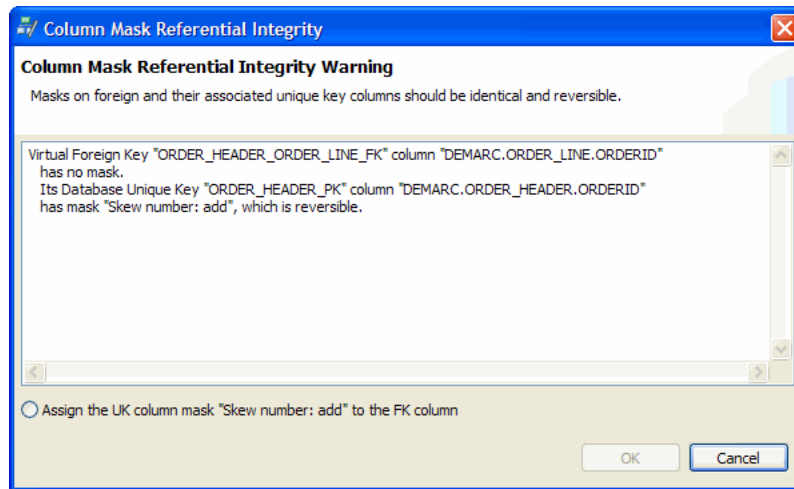
- The mask (whether custom or pre-built) is reversible
- The mask on the foreign key column is identical to the corresponding column on the unique key to which the foreign key is associated.



In this case, the message explains that you have two issues. First, you have masked the column with the primary key without similarly masking the column containing the foreign key. Consequently, the primary and foreign key values are no longer aligned with each other and could therefore no longer be used to find related rows. Second, you have masked the column containing the primary key (ORDERID) in ORDER_HEADER with a Random number mask, which is not a reversible mask.

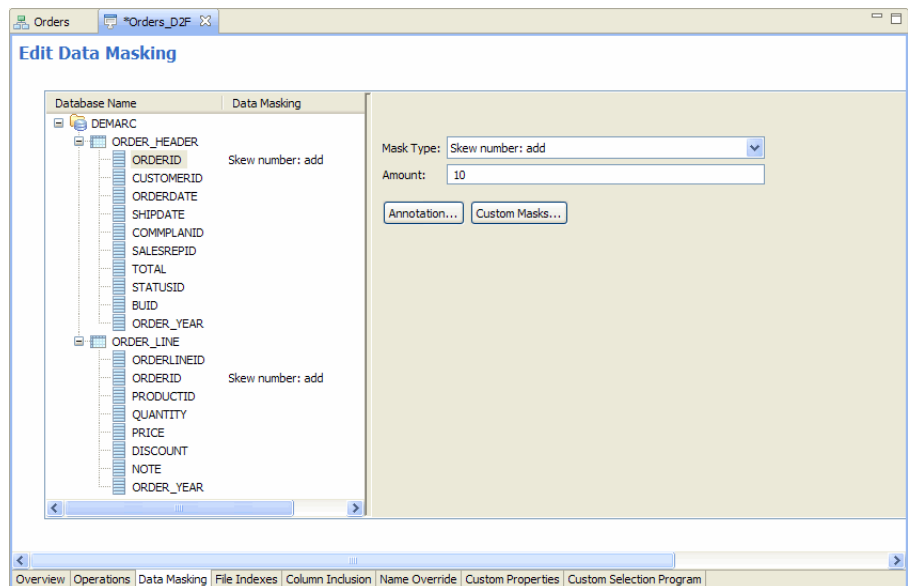
If you change the mask of ORDERID in ORDER_HEADER to Skew number: add, which is reversible, you eliminate the latter error but not the former.

NOTE Notice that the dialog still provides an informational message about the mask on the column containing the primary key (indicating that the mask is reversible). No further action on this item is required on your part.



The first message in the dialog is a warning that indicates that the column containing the foreign key is not masked, but the corresponding column containing the primary key is masked. To maintain referential integrity, columns containing the primary and foreign keys must be masked in a consistent manner. In this case, that would mean applying a Skew number: add mask with the same skew value to the foreign key column.

At the bottom of the dialog is a radio button that, when selected, will automatically apply the same reversible mask to the column containing the foreign key that has already been applied to the column containing the primary key. If you select this option and click OK, the referential integrity warning is removed because both columns (with primary and foreign keys) are now masked by the same reversible mask.



Deleting cartridges

To delete a cartridge:

- 1 Ensure that you have deleted calls to the cartridge from any business flows that reference it.
- 2 Select the cartridge you want to delete in the Project Navigator.
- 3 Right click and select **Delete**, or press the Delete key.

Typically, your test database is not as secure as your production database. Hence, you need to mask any sensitive data, such as names, addresses, phone numbers, social security numbers, and so on. HP Test Data Management provides a variety of masking capabilities to help you protect confidential data.

This chapter includes:

- [About data masking](#)
- [Applying pre-built data masks](#) (page 102)
- [Applying string map masks to data](#) (page 103)
- [Applying numeric map masks to data](#) (page 106)
- [Creating custom data masks](#) (page 110)
- [Unmasking data](#) (page 114)

About data masking

Data masks enable you to obfuscate column values in your copied data as a means of protecting private, confidential information such as credit card or social security numbers. HP Test Data Management provides pre-built masks for data that most commonly requires protection. In addition, you can create your own custom masks for data that you wish to mask.

Before creating and applying a data mask, consider the following:

- You must ensure that the data mask you apply corresponds to the data type of the column. For example, if the column is a numeric value, you must apply a numeric mask to it or use a function to convert the data to a data type that is compatible with the mask. Otherwise, the mask will fail with a SQL error upon deployment.
- If you apply a mask on a primary or foreign key, then you should also apply the same mask on a referenced primary key to ensure database integrity.

Refer to [Chapter 5, Working with cartridges](#) for detailed descriptions of all the available data masks.

Applying pre-built data masks

HP Test Data Management supplies numerous pre-built data masks so that you can quickly and easily mask sensitive data such as the SOCIAL and CREDITCARD columns of the CUSTOMER TABLE. However, with escalating concerns about confidentiality of customer data, that might not be considered sufficient. You might also want to mask columns such as LASTNAME, PHONE, ADDRESS, and EMAIL to further protect the identities of your customers.

Masking all these columns requires masks that can handle strings. For this purpose, HP Test Data Management includes pre-built string masks.

To mask data using a pre-built string mask:

- 1 Go to the cartridge editor for the Orders_D2F cartridge.
- 2 Click the **Data Masking** tab.
- 3 Select the **CUSTOMER** table on the left and expand it. At a minimum, you need to mask last names, addresses, phone numbers, email addresses, credit card numbers, and Social Security Numbers.
- 4 Select the **LASTNAME** column.
- 5 In the Mask Type list of values, choose **String mask: random length**. This option will generate strings of random lengths using the character X (or whatever character you specify).
- 6 For Max Length, enter **30**. This specifies that the random string length will not exceed 30 characters.
- 7 Using the information in [Table 30](#), specify masks for the other columns in the CUSTOMER table.

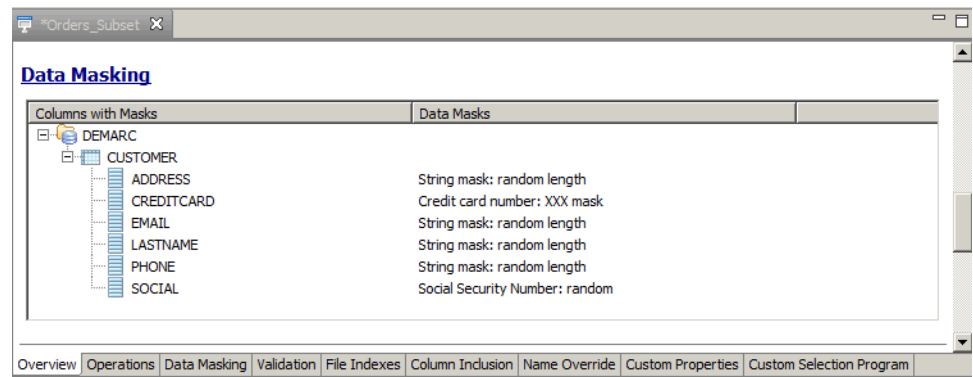
TIP You must ensure that the mask you choose for any column matches the data type of that column. For example, if the column is VARCHAR2(80), use a string mask with a length that is within 80 characters. You can double-click the table in the Database Navigator in the lower left pane of the Designer window to check the characteristics of the columns.

Table 30 Column masking information

| Column name | Mask |
|-------------|---|
| PHONE | String mask: random length Max. Length: 20 |
| EMAIL | String mask: random length Max. Length: 40 |
| ADDRESS | String mask: random length Max. Length: 60 |

- 8 Click the **Overview** tab.

- 9 Scroll down to the Data Masking section to view a summary of the mask.



Applying string map masks to data

String map data masks look up a given string in a map and apply the mapping value.

String map masks are reversible; hence, the mapping must be one-to-one and complete. String map masks are supported on Oracle and DB2 databases for standard database to file environments. For non-intrusive environments, string map masks are supported for all environments and are implemented using a mapping file rather than a mapping table.

NOTE For SQL Server only, you must use the procedures described in [Creating numeric or string map masks for SQL Server only](#) (page 106) to create a custom map mask in lieu of a string map or numeric map mask.

See also [Applying string and numeric map masks in non-intrusive environments](#) (page 108)

To apply a string map mask to data:

- 1 In the source database, create a 2-column mapping table. The data type of these two columns is determined by the data type of the column you wish to mask. In this case, the column being masked is of the data type VARCHAR with a maximum size of 20 characters (if using Oracle the data type is VARCHAR2).

```
create table stringmap (  
    orig varchar(20), masked varchar(20));
```

- 2 After creating the mapping table in your source database, populate it with the necessary original column values and their corresponding masked column values. In this case, we are randomly scrambling all the values in the LASTNAME column of the CUSTOMER table using a PL/SQL script. First, insert the original values:

```
insert into stringmap(orig)  
    (select distinct lastname from customer);
```

After inserting the original values, insert the masked values:

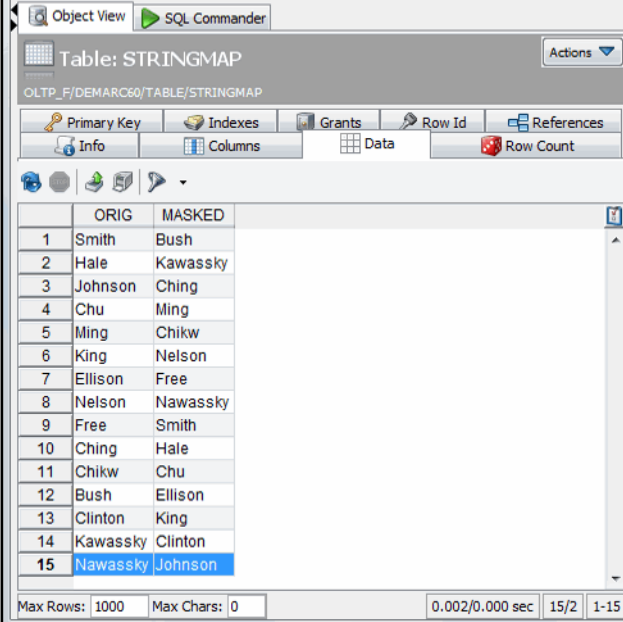
```

declare
    cursor c_emp is select * from stringmap;
    r_emp c_emp%ROWTYPE;
begin
    open c_emp;
    loop
        fetch c_emp into r_emp;
        exit when c_emp%NOTFOUND;
        update stringmap set masked = (SELECT orig FROM
( SELECT
orig FROM stringmap where orig not in (select
NVL(masked, ' ') from stringmap) and orig <> r_emp.orig
ORDER BY dbms_random.value ) WHERE rownum = 1 ) where
orig = r_emp.orig;

        end loop;
    close c_emp;
    commit;
end;
/

```

The size must match the size of the VARCHAR column from the source database that is being masked by the string map. The column MASKED now stores the masked values for the keys in the ORIG column.



The screenshot shows the SQL Developer interface with the 'STRINGMAP' table selected. The table has two columns: 'ORIG' and 'MASKED'. The data is as follows:

| | ORIG | MASKED |
|----|----------|----------|
| 1 | Smith | Bush |
| 2 | Hale | Kawassky |
| 3 | Johnson | Ching |
| 4 | Chu | Ming |
| 5 | Ming | Chikw |
| 6 | King | Nelson |
| 7 | Ellison | Free |
| 8 | Nelson | Nawassky |
| 9 | Free | Smith |
| 10 | Ching | Hale |
| 11 | Chikw | Chu |
| 12 | Bush | Ellison |
| 13 | Clinton | King |
| 14 | Kawassky | Clinton |
| 15 | Nawassky | Johnson |

At the bottom of the window, it shows 'Max Rows: 1000', 'Max Chars: 0', and execution statistics: '0.002/0.000 sec', '15/2', and '1-15'.

Alternatively, you can use the createRandomMapping.bat script to randomly generate new masked values and update the mapping table. Run the script from the obt/bin directory using the following command:

```

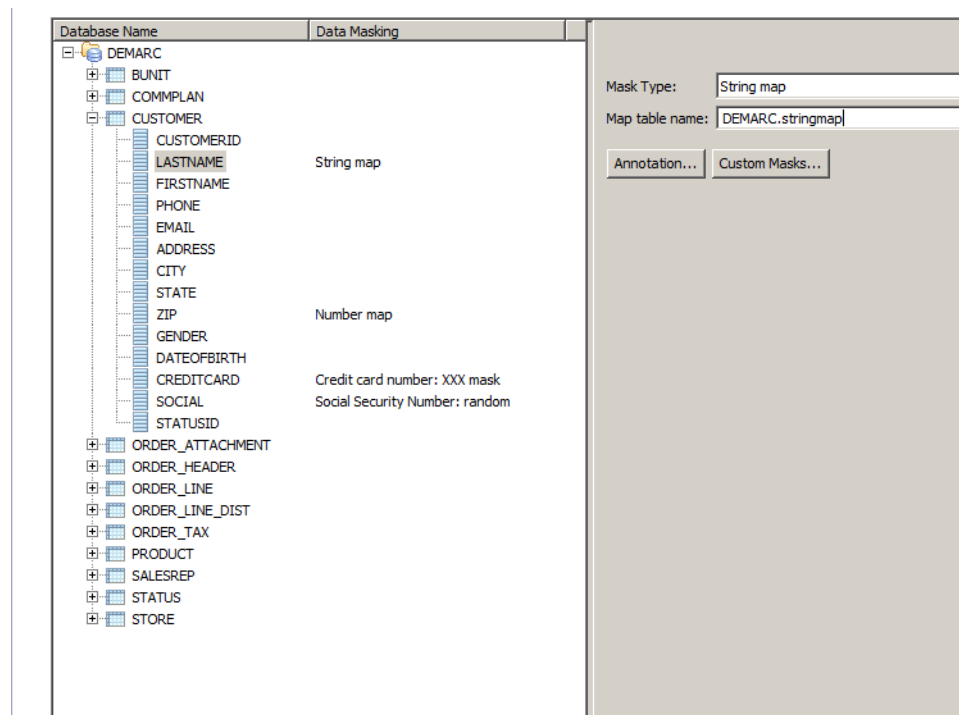
createRandomMapping.bat -e environment-id -k
encryption-key MappingTableName [lower-bound]
[upper-bound] [-h]

```

where MappingTableName is the name of the mapping table. The optional [lower-bound] and [upper-bound] parameters specify the boundaries for the numeric ranges (the defaults are 1 and $2^{31}-1$, respectively).

NOTE You must populate the original values in the mapping table before running the createRandomMapping.bat script. The script then populates the mapping table with the randomly generated masked values.

- 3 After creating and populating the mapping table in your source database, in the cartridge editor click the **Data Masking** tab. The Edit Data Masking dialog opens.
- 4 To apply the string map mask to a column, expand the node that contains the table you want to mask. For example, CUSTOMER.
- 5 Select the table column LASTNAME.
- 6 For **Mask Type**, select String map. In the Map table name field, type the fully-qualified name of the mapping table you created in your source database in [step 1](#) (page 103). In this case, the fully-qualified table name is DEMARC.stringmap. (assuming that the table stringmap is located in the schema DEMARC).



- 7 Optionally, click **Annotation** and provide a description of the string mask you created and are applying to the table column.
- 8 Optionally, create unique key constraints on a mapping table for the columns ORIG and MASKED. Refer to [Working with virtual unique and foreign keys](#) (page 47). Note that the syntax for unique key constraints can vary by platform. Refer to your platform-specific documentation for details about the syntax.

Creating numeric or string map masks for SQL Server only

Because SQL Server does not support dynamic SQL, you cannot create and use numeric map masks or string map masks for this platform in the same way you would for all other platforms.

Instead, you must create a custom mask that will be applied just as the numeric map and string map masks are applied on all non-SQL Server platforms.

See also For details about creating a numeric or string map mask for SQL Server in non-intrusive environments only, refer to [Table 26](#) (page 87) and [Table 27](#) (page 92) in [Chapter 5, Working with cartridges](#).

To create a custom map mask for SQL Server in standard environments:

- 1 Create a mapping table:

```
create table mylookuptable(actualvalue varchar(20),
    valtoget varchar(20));
```

- 2 Create the custom mask:

```
CREATE FUNCTION DRINKLOOKUP (@realvalue varchar(20))
returns varchar(20)
as begin
    declare @returnval varchar(20);
    select @returnval = valtoget from mylookuptable where
actualvalue = @realvalue;
    return (@returnval);
End;
```

- 3 Populate the mapping table with the mapping values. For example:

```
Insert into mylookuptable values ('Water', 'Wine');
```

- 4 Use the new mask (DRINKLOOKUP) as a custom mask.

Applying numeric map masks to data

Just as with string map data masks, numeric map data masks look up a given number in a map and apply the mapping value.

Numeric map masks are reversible; hence, the mapping must be one-to-one and complete. Numeric map masks are supported on Oracle and DB2 databases for standard database to file environments. For non-intrusive environments, numeric map masks are supported for all environments and are implemented using a mapping file rather than a mapping table. For details about applying string or number map masks in non-intrusive environments, see [Applying string and numeric map masks in non-intrusive environments](#) (page 108).

Numeric data masks are applicable to numeric columns (such as NUMERIC, INT, REAL, FLOAT, DECIMALS, etc.).

To apply a numeric map mask to data:

- 1 In the source database, create a 2-column mapping table with a fully-qualified table name. The data type is determined by the data type of the column you wish to mask. In this case, the column being masked is of the data type NUMBER (Oracle).

```
create table zipmap  
  (orig number, masked number);
```

- 2 After creating the mapping table, populate it with the data. In this case, zip codes will be masked by adding 100 to each existing zip code value:

```
insert into zipmap(orig)  
  (select distinct zip from customer);  
update zipmap set masked = orig +100;
```

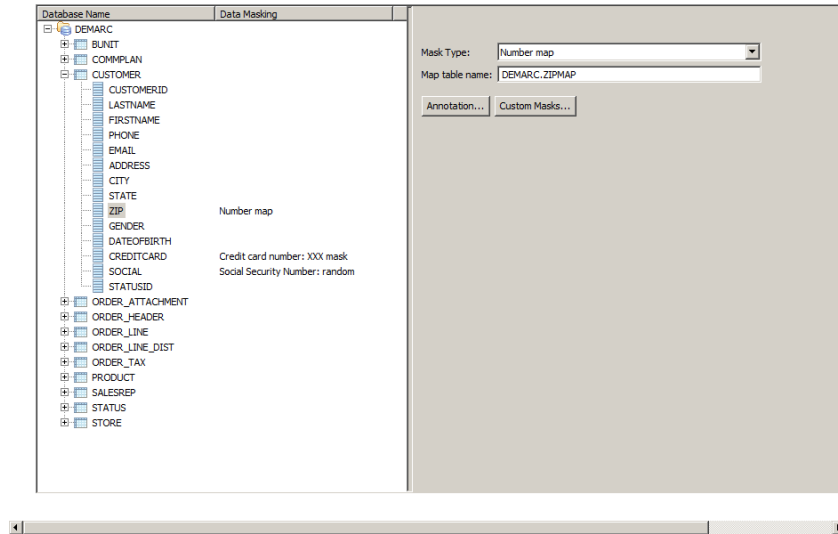
As is the case for string mapping, for numeric mapping the column MASKED stores the masked values for the keys in the ORIG column.



| | ORIG | MASKED |
|----|-------|--------|
| 1 | 19036 | 19136 |
| 2 | 34567 | 34667 |
| 3 | 19013 | 19113 |
| 4 | 10702 | 10802 |
| 5 | 11010 | 11110 |
| 6 | 12301 | 12401 |
| 7 | 10002 | 10102 |
| 8 | 10001 | 10101 |
| 9 | 16301 | 16401 |
| 10 | 11756 | 11856 |
| 11 | 13902 | 14002 |
| 12 | 21235 | 21335 |
| 13 | 10960 | 11060 |
| 14 | 14617 | 14717 |
| 15 | 55467 | 55567 |
| 16 | 15219 | 15319 |

- 3 After creating and populating the mapping table in your source database, in the cartridge editor click the **Data Masking** tab. The Edit Data Masking dialog opens.
- 4 To apply the numeric map mask to a column, expand the node that contains the table you want to mask. For example, CUSTOMER.
- 5 Select the table column, ZIP.
- 6 For **Mask Type**, select Numeric map. In the Map table name field, type the full path name of the mapping table you created in your source database in [step 1](#). In this case, the fully-qualified numeric map table name is DEMARC.zipmap (assuming that the table zipmap is located in the schema DEMARC).

Edit Data Masking



- 7 Optionally, click **Annotation** and provide a description of the numeric data mask you created and are applying to the table column.
- 8 Optionally, create unique key constraints on a mapping table for the columns ORIG and MASKED. Refer to [Working with virtual unique and foreign keys](#) (page 47). Note that the syntax for unique key constraints can vary by platform. Refer to your platform-specific documentation for details about the syntax.

Applying string and numeric map masks in non-intrusive environments

String and number map masks are supported in non-intrusive environments as well. However, HP Test Data Management does not maintain a schema within the source database for non-intrusive environments, so the mapping is defined in a file rather than a table. Also, the absolute path applies to a file instead of a mapping table name and is recorded in the Map table name parameter.

To apply a string or numeric mask in a non-intrusive environment:

- 1 Create a mapping file that defines a map using the syntax: `key=value` per line, where `key` is mapped to `value` by this map mask. If a key or value contains spaces, you must enclose them in them in quotation marks (for example, `Jack="Male Customer 1"`). In this case, we are masking the `FIRSTNAME` column in the `CUSTOMER` table:

```
Jack=MaleCustomer1
Paul=MaleCustomer2
Phillip=MaleCustomer3
Nadine=FemaleCustomer4
Audrey=FemaleCustomer5
.
.
```

- 2 Save the mapping file to the `obt\extensions\runtime\masking` directory.
- 3 In the cartridge editor click the **Data Masking** tab. The Edit Data Masking dialog opens.
- 4 To apply the string map or numeric map mask to a column, expand the node that contains the table you want to mask. For example, CUSTOMER.
- 5 Select the table column, FIRSTNAME.
- 6 For **Mask Type**, select either String map or Numeric map. In the Map table name field, type the full path name of the mapping table you created in [step 1](#). In this case, the fully-qualified numeric map table name is `StringNonintrusive.txt`

NOTE If the mapping file is located in the default folder (`obt\extensions\runtime\masking`), then you can simply specify the file name (in this case, `StringNonintrusive.txt`) in the Map table name field. However, if the mapping file is located in a different folder, for example, `C:\Users\dba user\Documents\DataMasking\mapping`, then you must specify the full path, which in this case would be: `C:\Users\dba user\Documents\DataMasking\mapping\StringNonintrusive.txt`.

- 7 Deploy the cartridge and launch the business flow. See [Deploying and running](#) (page 115).
- 8 Confirm the mask was applied by viewing the results in the `HPTDM\archivedata` directory:

```
<CUSTOMER>
<CUSTOMERID>1</CUSTOMERID>
<LASTNAME>Jones</LASTNAME>
<FIRSTNAME>MaleCustomer1</FIRSTNAME>
<PHONE>111-222-3333</PHONE>
<EMAIL>JonesJ202@msn.com</EMAIL>
<ADDRESS>123 Main</ADDRESS>
<CITY>Aberdeen</CITY>
<STATE>CA</STATE>
<ZIP>10001</ZIP>
<GENDER>Male</GENDER>
<DATEOFBIRTH>1984-02-12T00:00:00.000000000-08:00</
DATEOFBIRTH>
<CREDITCARD>4588347116299261</CREDITCARD>
<SOCIAL>897-01-2221</SOCIAL>
<STATUSID>14</STATUSID>
</CUSTOMER>
.
.
.
```

Creating custom data masks

In situations where a pre-built mask simply cannot satisfy your testing requirements, or you require specific, conditional logic to apply a data mask, custom data masking is available.

Table 31 describes the data masking options available for standard and non-intrusive environments on the various database platforms.

Table 31 Custom data masking availability by cartridge environment setup and database platform

| | Database to File Cartridge, Standard Environment | Database to File Cartridge, Standard Environment | Database to File Cartridge, Non-Intrusive Environment |
|--|--|---|--|
| Database platform | <ul style="list-style-type: none">• DB2• Oracle 9i and later• SQL Server• Sybase 15.0.2 and later | <ul style="list-style-type: none">• Oracle 8i and earlier• Sybase 15.0.1 and earlier | <ul style="list-style-type: none">• All database platforms |
| Custom masking (SQL-based) available? | Yes | No | No |
| Custom masking (Groovy-based) available? | No | Yes | Yes |

Creating custom data masks in standard environments

To create a custom mask for standard deployment environments:

- 1 You must write a masking function using PL/SQL for Oracle, Transact-SQL for SQL Server, or the SQL command line interface for DB2.

NOTE In an Oracle environment, the function name cannot contain the / character.

Custom data masking takes only one parameter and it is the column to be masked. For example, if you are custom masking a column named DEPARTMENT, which is of data type A, then the signature of the custom data mask would be:

```
A MY_MASK(A in);
```

where MY_MASK is the name of the custom data masking function that you are writing and A is the data type of the column being masked.

The following PL/SQL script creates a package, DATAMASKING, with the masking function MY_MASK and the unmasking function (which is optional) REVERT_MY_MASK:

```
create or replace package "DATAMASKING" as
    function MY_MASK
        (p_data in varchar2)
        return varchar2;
    function REVERT_MY_MASK
        (p_data in varchar2)
        return varchar2;
end;
/

create or replace package body "DATAMASKING" as
    function MY_MASK
        (p_data in varchar2)
        return varchar2 is
        v_return_value    varchar2(1024);
    begin
        v_return_value := p_data || '_CustomMasking';
        return v_return_value;
    end MY_MASK;

    function REVERT_MY_MASK
        (p_data in varchar2)
        return varchar2 is
        v_return_value    varchar2(1024);
    begin
        v_return_value := replace(p_data,
        '_CustomMasking', null);
        return v_return_value;
    end REVERT_MY_MASK;

end;
```

The following Transact-SQL script creates the same package for SQL Server:

```
IF OBJECT_ID(N'dbo.my_mask', N'FN') IS NOT NULL
    DROP FUNCTION dbo.my_mask;
GO

create function my_mask(@p_orig varchar(256))
returns varchar(256) as
BEGIN
    return @p_orig + '_CustomMasking';
END;
GO

IF OBJECT_ID(N'dbo.revert_my_mask', N'FN') IS NOT NULL
    DROP FUNCTION dbo.revert_my_mask;
GO
```

```

create function revert_my_mask(@p_orig varchar(256))
returns varchar(256) as
BEGIN
    return replace(@p_orig, '_CustomMasking', '');
END;

```

- 2 Ensure that the function compiles successfully.
- 3 Deploy the function on the source database.
- 4 In Designer, when you are ready to apply a custom data mask, type in the fully qualified name of your custom data mask function. For example, in Oracle, suppose you created a custom data masking function called MY_MASK in the package CUSTOM_MASKING and deployed it in a database named HRDB. The fully qualified name would be:

HRDB.CUSTOM_MASKING.MY_MASK.
- 5 In the environments in which you run the cartridge, ensure that the masking function is available in the source database and that the OBT_IF user has access permissions on it.

Creating custom data masks in non-intrusive environments

You can create custom masks for non-intrusive database to file environments (and for standard environments using Sybase version 15.0.1 and earlier as well as Oracle 8i) using Groovy scripts.

To create a custom mask using a Groovy script:

- 1 Create a Groovy script that includes the following signature:

Object mask(Object param);

Mandatory. Use to apply a data mask data during copy.

Object revert_mask(Object param);

Optional. Use to recover masked data and restore the original values during upload. If you wish to restore data to its original state, then you must provide this function.

Because data types are mapped internally and vary amongst the supported database platforms, it is recommended that you include input parameters for the respective data types. For example, if the column being masked is of data type Decimal, Integer or SmallInt, then the object that is passed to these functions is of the type java.math.BigDecimal.

Refer to [Table 32](#) for data type mappings from databases to non-intrusive environments and vice versa.

Table 32 Column data type mappings

| SQL-based database column type | Java-based database column type |
|--|---------------------------------|
| VARCHAR ^a , NVARCHAR, CHAR, NCHAR | java.lang.String |
| VARBINARY, BINARY | java.lang.Byte[] |
| BIGINT, NUMERIC ^b , DECIMAL | java.math.BigDecimal |
| INTEGER, SMALLINT, TINYINT | java.lang.Integer |
| REAL, DOUBLE | java.lang.Double |
| FLOAT | java.lang.Float |
| BIT | java.lang.Short |
| TIME | java.sql.Time |
| DATE | java.sql.Date |
| TIMESTAMP | java.sql.Timestamp |

a. For Oracle, this data type is VARCHAR2

b. For Oracle, this data type is NUMBER.

NOTE Review the sample Groovy scripts in `obt/extensions/runtime/masking` for additional guidance.

- After verifying that the function compiles successfully, copy the Groovy script into the folder `obt/extensions/runtime/masking`.
So, if your Groovy script name is `MyMask.groovy`, first copy `MyMask.groovy` into the folder `obt/extensions/runtime/masking`, then add the name of custom mask (`MyMask.groovy`) on the Data Masking tab for the cartridge and apply it to columns.
- In Designer, click the Data Masking tab in the cartridge and then click **Custom Masks**.
- Click **Add** and type the filename (case-sensitive) of your Groovy script in the Custom Data Masking dialog, then click **OK**.

NOTE The upload process will automatically pick up the unmasking function based on the masks that were applied during extraction. You need not specify the unmasking function in the cartridge.

Unmasking data

After copying data to a standard environment using masks, you may wish to upload data on a different database and unmask it upon upload. The method for unmasking will vary, depending upon whether you used a numeric, string or custom data mask.

Unmasking numeric or string masks upon upload in a database to file standard environment

When you extract data that has been masked using either a number or string map mask in a database to file standard environment, you can unmask the data upon upload. The masking and mapping table functions used by the number or string map masks are located on the source database, which upload does not use. In these cases, you can provide a mapping file that will unmask the data upon upload.

To generate a mapping file from a mapping table, run the `createMappingFileForMask.bat` script as follows:

```
createMappingFileForMask.[bat|sh] -e environment-id -k  
encryption-key MappingTableName
```

where `MappingTableName` is the name of the mapping table specified in Designer when the number or string map mask was applied. After running the script, the mapping file required to unmask the data upon upload is created.

Be sure to set the Unmask data on upload parameter to true.

Unmasking custom masks upon upload in a database to file standard environment

If the data was extracted using a custom mask rather than a number or string mask, then you can use either of the following options to unmask data upon upload:

- Provide a custom reversible mask and deploy it on the upload database in the same schema as the mask on the source database
- Include a `revert_mask` function in a Groovy script. The name of the Groovy script should be the same as the name of the custom mask used when you copied the data, and should be located in the `obt/extension/runtime/masking` directory.

Regardless of the method you choose, ensure that the Unmask data on upload parameter is set to true, then run your upload.

Deploying and running

To deploy and run a business flow:

- 1 Deploy the Orders_D2F_BF business flow from Designer. Refer to [Chapter 9, Generating and deploying](#).

Working with projects and connections

A project is a container that holds components, such as models, cartridges, and business flows, and connection information used to define your test data solution. The first steps in defining your test data solution are to create a project and a database connection in Designer.

This chapter explains how to work with projects and connections.

This chapter includes:

- [About projects](#) (page 117)
- [Creating a new project](#) (page 118)
- [Creating a connection](#) (page 119)
- [Changing projects](#) (page 121)
- [Annotating projects](#) (page 122)
- [Exporting and importing projects](#) (page 124)
- [Changing the project location](#) (page 126)
- [Mapping schema names](#) (page 126)

About projects

All of the work to build a model, cartridge, or business flow in Designer is performed within the context of a project. A finished project contains all the metadata objects and definitions needed by HP Test Data Management to copy your data from the database to the data extract files (XML or CVS).

Projects can help you organize your work for specific goals. For example, you might create projects according to work phases or business areas. A sales project could copy data for the sales aspect of your business while a development project copies the data about the development efforts of your company.

A project is a logical container for:

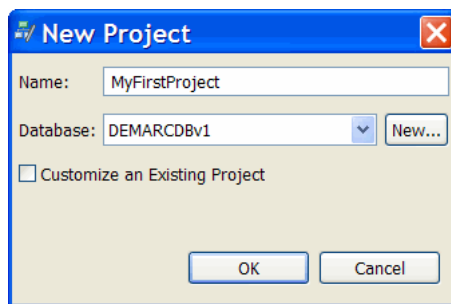
- models
- cartridges
- business flows
- parameters

When logging onto Designer for the first time, it requires you to create a new project or open an existing project. All object definitions made during your work session are stored in a work space owned by the current project. Subsequent to that, Designer starts with your previously created projects available in the Project Navigator.

Creating a new project

To define a new project:

- 1 Start Designer.
- 2 Select **File > New Project**.



- 3 In the Name field, type a name for the project.
For example, Sales Order.
- 4 Select an existing database connection from the list or click **New** to define a new connection.
For more information, see [Creating a connection](#) (page 119).
- 5 If you are creating a project that is based on an existing project file (typically one provided to you by HP or a third party), then check **Customize an Existing Project**. Choosing this option indicates that you want to create a new project from an existing project.
- 6 Click **OK**. If you left Customize an Existing Project unchecked, the new project is created and you are done. If you checked Customize an Existing Project, then the Import Project dialog box displays.
- 7 In the Import Project dialog box, browse for the project file you wish to customize, which has the .hdp extension.

TIP .hdp is an exchange format for HP Test Data Management projects and contains all of the files associated with the project. You create .hdp files by exporting a project from Designer, **File > Export Project**.

- 8 Click **Open**. Designer extracts the project files from the selected .hdp file and the project appears in the Project Navigator. You can now modify the project as you would any other project in the Project Navigator.

TIP If the project you selected is a locked project, then you should only perform limited customizations on it. For more information about customization, refer to [Chapter 10, Customizing projects](#).

Creating a connection

You can work with online connections or offline local caches. An online connection provides you with the data, but, if you are in the process of designing a test data solution on a remote database, you often do not require the data and working from the online connection may slow you down considerably. In such circumstances, it may be more efficient to create a local cache and work from the metadata rather than being continuously connected to the remote database. Furthermore, a local cache allows you to continue designing even when your computer is not connected to your network.

NOTE If using a new, generic JDBC driver in a non-intrusive environment, HP Test Data Management checks the driver's Name types and JDBC type values. If data is missing, then you may encounter an error at runtime. If there is a data mismatch, then you can encounter data loss. Upon making the first connection for a non-intrusive environment, examine your data closely to ensure that there are no "mismatches" between the Name type and JDBC Type.

Refer to the *HP Test Data Management Web Console User Guide* and *HP Test Data Management Troubleshooting Guide* for details about data mapping errors and mismatches when using generic JDBC drivers in non-intrusive environments.

TIP To ensure that your JDBC driver will work properly with HP Test Data Management, it is strongly recommended that you test it using the JDBC driver certification test tool. For more details about the JDBC driver certification tool, refer to [Appendix A, JDBC driver validation and configuration](#).

To create a new connection:

- 1 Click **New** from the New Project dialog box.
Or:
 - Select **Connection > Edit Connections**.
 - Click **New**.

2 Select one of the connection types in [Table 33](#):

Table 33 Database connection types

| Connection Type | Description |
|-----------------------------------|---|
| Online Connection | Create a connection to a database |
| Local Cache | Create a new local cache to capture the metadata information about an existing online connection. This option will enable you to work offline. To preview data when working offline, the underlying online connection must still be available to Designer because only metadata is stored in the local cache. |

[Online Connection](#)

- a Type a name for the connection.
- b Select a database type from the list of values (for example, DB2) and click **Next**.
- c Type the connection information for the active database as described in [Table 34](#):

Table 34 Source database properties

| Field | Description |
|-----------|--|
| User Name | Type the name of the user with permission to access the objects necessary to create your cartridge. For example, <code>sa</code> . |
| Password | Type the password. For example, <code>imSn33ky</code> . You should only check Save Password when the data is not critical. Although the password is saved in an encrypted format, it does present a security risk. For example, if you are working with a development database, it could be considered safe to save the password. On a production database, it would probably not be considered safe. |
| Host | Type the name of the machine where the database is installed. For example, <code>localhost</code> . |
| Port | Type the port number for your database. For example, <code>5001</code> . |
| URL | For JDBC URL, the connection URL for the database. JDBC URLs are often used to connect to an Oracle database with RAC. For example: <code>jdbc:oracle:thin:@(DESCRIPTION=(SDU=32768)(enable=broken)(LOAD_BALANCE=yes)(ADDRESS=(PROTOCOL=TCP)(HOST=gvu2707.austin.hp.com)(PORT=1525))(ADDRESS=(PROTOCOL=TCP)(HOST=gvu2923.austin.hp.com)(PORT=1525))(CONNECT_DATA=(SERVICE_NAME=SAPDEMI)))</code> |

Table 34 Source database properties

| Field | Description |
|------------------------|--|
| SID | For Oracle, the name of the database instance. |
| DB Server | For SQL Server and Sybase, the name of the database server. |
| Database | For DB2 and SQL Server, the name of the database, for example, <i>master</i> . |
| Windows Authentication | For SQL Server, Windows Authentication indicates that the operating system login for the machine is the same as the SQL Server login, and once you are logged into the machine, you need not authenticate again for the SQL Server instance. If you do not select this option, the SQL Server login is distinct from the operating system login for the machine, and logging into the machine does not imply that you are authenticated for the SQL Server instance as well. |

- d Click **Finish**.

Local Cache

- a Type a name for the local cache.
- b Select a database from the list of existing online connections.
- c Perform one of the following:
 - Select All to capture metadata for all your databases, schemas, and tables.
 - Select particular databases, schemas, or tables to work with offline. In most cases, this choice is the most efficient and minimizes the overhead.
- d Click **Finish**.

Changing projects

As mentioned earlier, you may use multiple projects to categorize and organize your model, cartridge, and business flow development. If so, you need to switch among these projects within Designer.

To switch between different projects:

- 1 Go to the Project Navigator.
- 2 Select the project you want to work with from the list of values for the Project field.

You are now working in and saving objects to the project that is selected and active.

Annotating projects

HP Test Data Management enables you to enter comments throughout your project for virtually every component. These comments help other developers and users better understand your project and all of its components. You can also generate a PDF file that contains all of your annotations as well as descriptions of the objects that makeup up the project components.

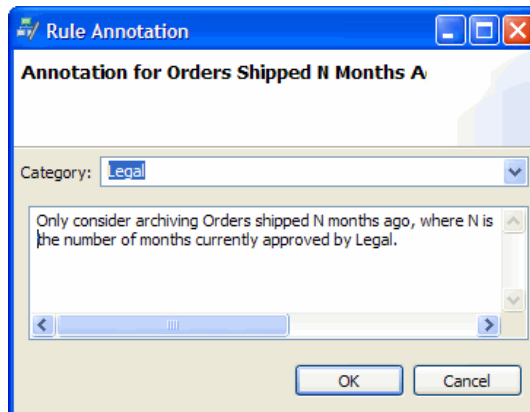
Annotating projects

To create a project annotation:

- 1 From a dialog or page, click **Annotation**. The Annotation dialog displays.

TIP For projects, models, and cartridges, right click them in the Project Navigator and choose **Edit Annotation**.

- 2 For rules, the Annotation dialog box has a Category list from which you can optionally choose a category, such as **Legal** or **Data Integrity**. A category is not required. You can leave it as blank.



- 3 Type your annotation and click **OK**. Typically, an annotation should explain what the element does and its significance. For example, an annotation for a rule should explain the function of the rule and its underlying business condition.

Annotating cartridges

From each tab in the cartridge editor you can add specific annotations as follows:

- From the Operations tab, when you select a table, the Annotation button is enabled and you can add comments for that table.
- For the Data Masking tab, select a table and column, then click Annotation to enter comments about that column's mask.
- For the model-based cartridge editor tabs, annotations behave as follows:

- For the File Indexes tab, the Annotation button is enabled when you select an existing index or click New to create a new one. You can add a separate comment for each named index.
- For the Column Inclusion tab, the Annotation button is enabled when you select a column for inclusion. If you make no changes to the default selections, the assumption is that you do not need to note any changed behavior in the annotations.
- For the Name Override tab, the Annotation button is enabled whenever you change the XML name for a column. If you make no changes, the assumption is that you need not note any changed behavior in the Annotations.
- For the Custom Properties tab, you can enter a comment for each property.
- For the following schema-based tabs, annotations behave as follows:
 - For the Overview tab, the Annotation button is enabled when you select a table to be extracted.
 - For the Data tab, the Annotation button only displays if you choose Selected Data.

Generating and viewing documentation

For any project, you can generate documentation in PDF format that describes the project. The documentation includes a number of items, such as:

- title page and table of contents
- business flow diagrams and activities
- parameter lists
- model diagrams
- table use lists
- cartridge characteristics
- annotations

You can generate this documentation in one of two ways:

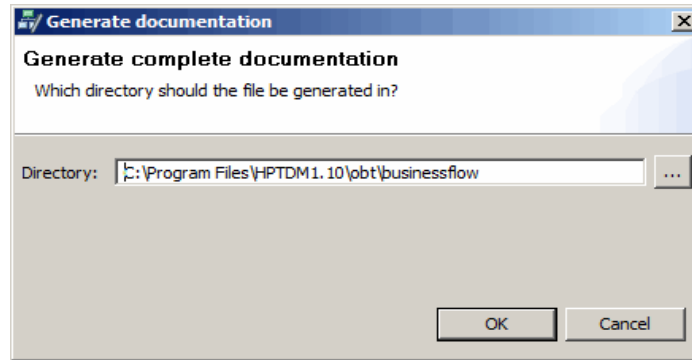
- [Generating documentation from the Project Navigator](#) (page 123)
- [Generating documentation on deployment](#) (page 124)

Generating documentation from the Project Navigator

- 1 In the Project Navigator, right click the project (the top-level node), and choose **Generate Documentation**. The Generate Documentation dialog box displays.

TIP You can also right click on individual cartridges and business flows in the Project Navigator to generate documentation for them. If you right click on the Cartridges or Business Flows nodes, it generates documentation for all cartridges or business flows.

- 2 Type or browse to the location where you want the PDF file generated. The PDF file will be placed in a business flow subdirectory in the chosen location.



- 3 Click **OK**.
- 4 Go to the location you chose and open the PDF file.

Generating documentation on deployment

- 1 As you are deploying your cartridge or business flow in Deployment Assistant, on the Deployment Type page, check **Include Documentation**. Choosing this option creates a PDF file that describes the structure of your business flow along with any annotations you added to the business flow and the model.
- 2 Go to the location of the PDF file. It should be in `install_dir\obt\businessflow`. For example:

```
C:\Program  
Files\HPTDM\obt\businessflow\<environment_name>  
Orders_Bus_Flow.1_0_0_0.pdf
```

- 3 Open the PDF file.

Exporting and importing projects

Projects in Designer can be saved to .hdp files for exchange purposes. Typically, you export projects to .hdp files to share them with other users, who can then reuse and/or modify them.

Exporting projects

To export a project to .hdp:

- 1 In the Project Navigator, select the project from the list at the top.

- 2 Choose **File > Export Project**. The Export Project dialog box displays.
- 3 Navigate to the location where you want to save the .hdp file and enter the File name.
- 4 Click **OK**. The project files are saved to the .hdp file.

Importing projects

To import a project from a file with the .hdp extension:

- 1 Choose **File > Import**. The Import Project dialog box displays.
- 2 In the Import dialog box, choose **Existing Designer project** from the Import list.

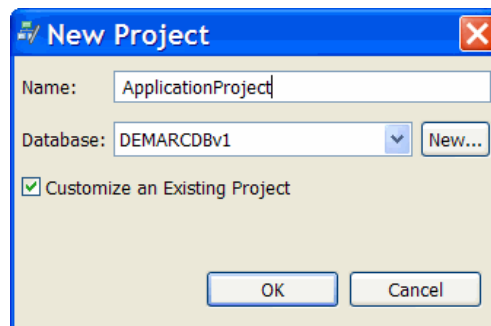
NOTE Do not select the 5.1 cartridge in this context; it is *not* a valid option.

- 3 Click **Next**. The Import Existing Project dialog box displays.
- 4 Navigate to the location where the .hdp file is stored and select it.
- 5 Click **Open**. The project files are extracted and the project appears in the Project Navigator. Please note the following:
 - The content of the imported project is added to the existing content of your current project.
 - Exporting/Importing a project exports/imports the models, cartridges, business flows, and parameters. The connection information is not exported/imported.

Importing a project for customization

To import a project for the purposes of customization:

- 1 In Designer, choose **File > New Project**. The New Project dialog box displays.
- 2 Enter a name for the project and select **Customize an Existing Project**.



- 3 Click **OK**. The Import Project dialog, from which you can select a .hdp file, displays.
- 4 Browse to and select the .hdp file that you received from your vendor.

- 5 Click **Open**. The selected project is imported into Designer under the specified project name. Please note the following:
 - Exporting/Importing a project exports/imports the models, cartridges, business flows, and parameters. The connection information is not exported/imported.
 - If the project you imported is a locked project, then you should only perform limited customizations on it. For more information about customization, refer to [Chapter 10, Customizing projects](#).
 - If you are dealing with a locked project from a third party, you would only import it the first time you receive it. Subsequently, when the third party provides you with project revisions, you would use **Customization > Merge Project** to bring in the updates.

Changing the project location

By default, all projects are saved to the user directory. For example, on MS Windows:

```
C:\Documents and Settings\<username>\HPTDM\Designer
```

where <username> is the name of the Windows user.

If you want to save your projects to a different directory, start Designer from the command line using the -data option. For example:

```
designer.exe -data "c:\<NewDirectory>"
```

where <NewDirectory> is the location where you want to store your project data. Project data will be stored in directories named for the projects under the specified location.

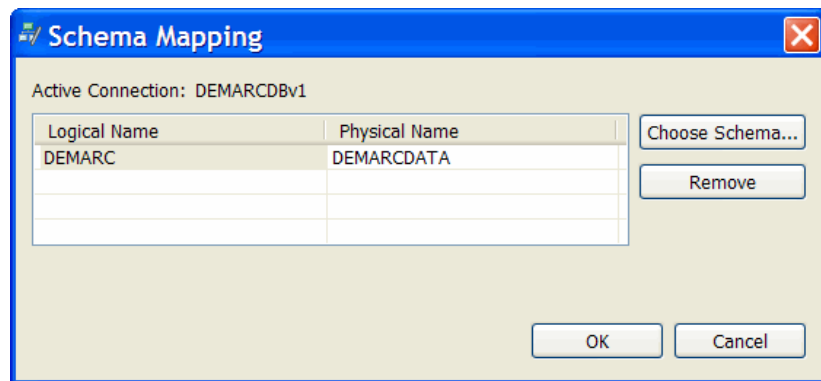
Mapping schema names

In some cases, you may find that you need to run your project against different schema instances with different names. For example, you might be using two database instances. In one instance, the schema might be called DEMARC and, in another, DEMARCDATA. In such situations, it is convenient to be able to map the schema names to avoid problems when switching between connections. Instead of using a hard coded schema name, you can create a logical alias and map it to the physical schema.

TIP You can use schema mapping to develop multi-database cartridges. For example, you could map HR to hr.dbo. In that case, a table called HR.EMPLOYEES on Oracle would be mapped to hr.dbo.EMPLOYEES on SQL Server. Refer to [Example](#) (page 128).

To map schema:

- 1 Open your project in Designer.
- 2 Select **Connection > Map Schemas**.
- 3 In the Schema Mapping dialog box, type a Logical Name or select it by clicking **Choose Schema**. For example, you could type DEMARC as the Logical Name. The logical name is the alias that you want to use within Designer when referring to the physical schema name.
- 4 Click the adjacent cell under Physical Name. Type or choose a Physical Name. The physical name is the name of the schema in the database for which you want to create an alias (logical name).



NOTE Schema mappings are associated with the connection.

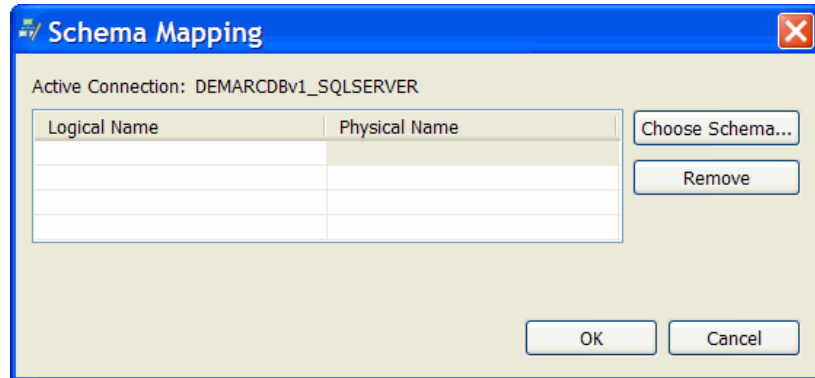
- 5 Click **OK**. Instead of using the physical name (for example, DEMARCDATA), Designer now uses the logical name (for example, DEMARC). If you later must use a different underlying physical name (DEMARCDB), you can simply come back to this dialog and change the Physical Name value accordingly.

TIP To delete a mapping, return to the Schema Mapping dialog box, click on the mapping you wish to remove, and click **Remove**.

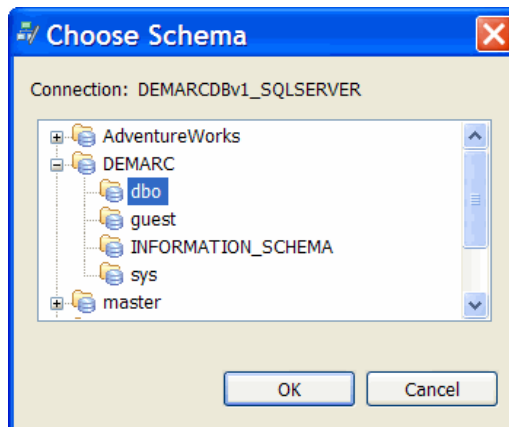
- 6 (Optional) Browse your project and notice some of the changes:
 - Under the Project Navigator, in the Database Navigator, the logical schema name appears in its own top-level node with the physical schema name in parentheses next to it.
 - The Table Use Properties dialog box for a table associated with the logical schema name will appear differently. Instead of showing a read-only Table Name field, the dialog displays two read-only fields, Mapped Table Name and Database Table Name.

Example Suppose that you have imported a project that was built against an Oracle database into a Designer instance running against a SQL Server database with some slight variation in the schema. To avoid reimplementing your model, you can simply map the schemas:

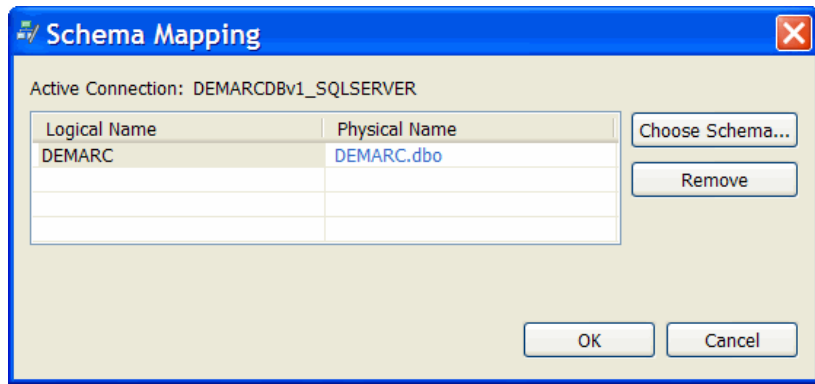
- 1 Select **Connection > Map Schemas**. The Schema Mapping dialog box appears.



- 2 Click in the first cell under Logical Name and type DEMARC.
- 3 Click in the first cell under Physical Name.
- 4 Click **Choose Schema**.
- 5 In the Choose Schema dialog box, expand DEMARC.
- 6 Select **dbo**.



- 7 Click **OK**.



- 8 Click **OK**. The imported model should now work against the SQL Server database.

A business flow is a set of activities including a cartridge and any other activities necessary to accomplish an extraction solution.

This chapter includes:

- [About business flows](#) (page 131)
- [Creating a business flow](#) (page 132)
- [Adding a database to file cartridge](#) (page 135)
- [Adding schema-based cartridges](#) (page 136)
- [Splitting an activity](#) (page 136)
- [Adding or editing a script](#) (page 138)
- [Adding or editing a condition](#) (page 141)
- [Adding or editing an interrupt](#) (page 142)
- [Testing business flows](#) (page 143)
- [Deleting a business flow](#) (page 143)

About business flows

Business flows consist of various activities that you want to perform in sequence. Some activities, such as extraction activities, can be split into steps and you can perform other activities in between those steps. For example, you might run a script that calculates the number of rows to be extracted in between the selection and copy steps of an extraction activity.

A business flow can include any of the following activities:

- Extraction activities that invoke an extract cartridge
- Conditions that enable you to branch your business flow
- Upload a test database immediately after the extraction cartridge
- Interrupts that stop your business flow
- Groovy scripts that perform some additional processing, including but not limited to any of the following:
 - Executing operating system commands, such as copying a file, deleting a file, compressing a file, or transferring a file (ftp)
 - Sending an email or other notification

- Database operations, such as dropping indexes, making a tablespace read-only, or analyzing statistics for a table

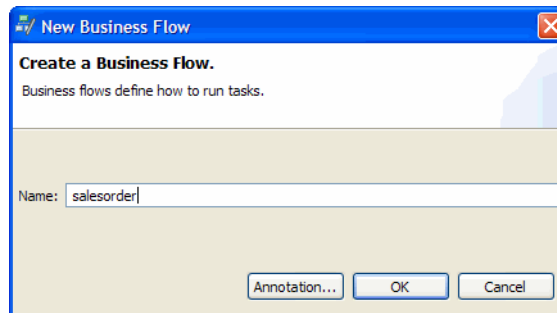
You edit business flows in a graphical environment where you can easily see the relationships among activities.

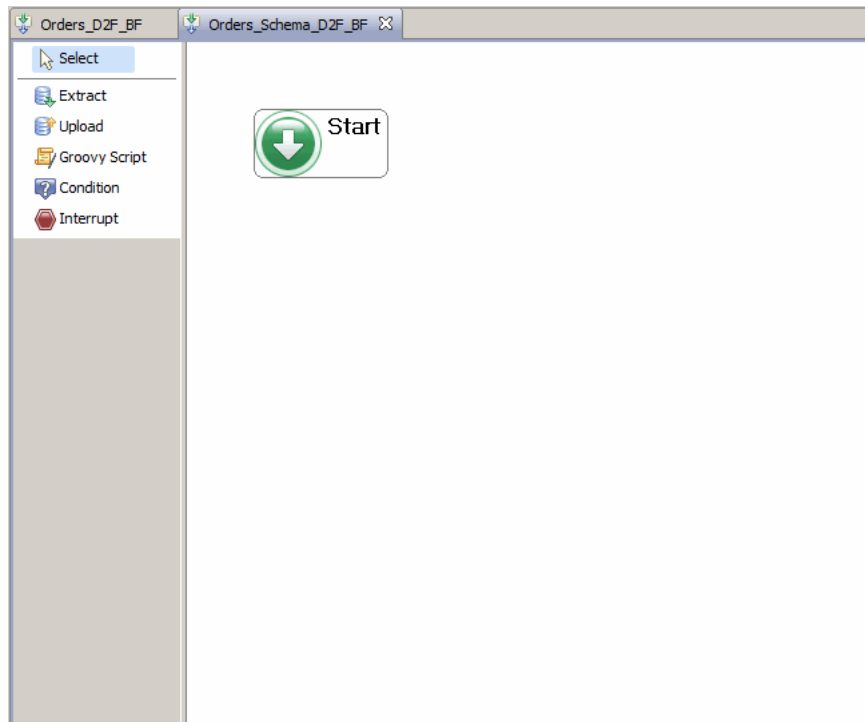
NOTE If you edit a cartridge after deploying a business flow, then you must redeploy the business flow to see the latest changes that you made in the cartridge.

Creating a business flow

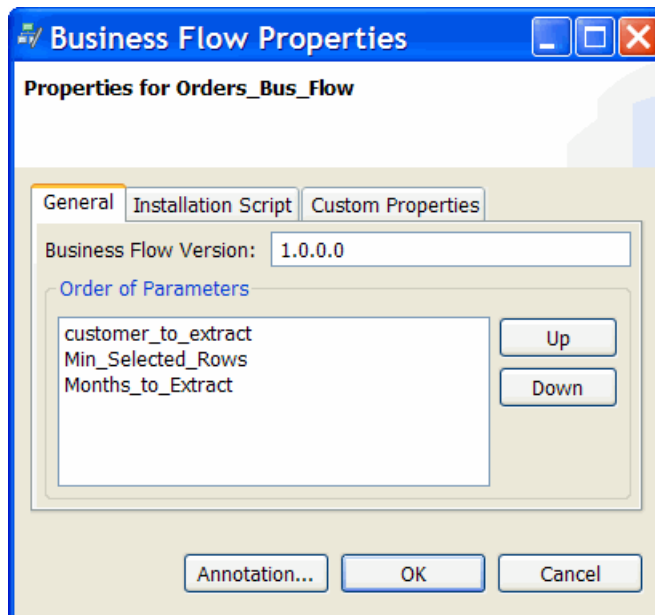
To create a business flow:

- 1 Select the Business Flows node in the Project Navigator.
- 2 Right-click and select **New Business Flow**. The New Business Flow window opens.
- 3 Type a name for the business flow and click **OK**. The Business Flow Editor opens.

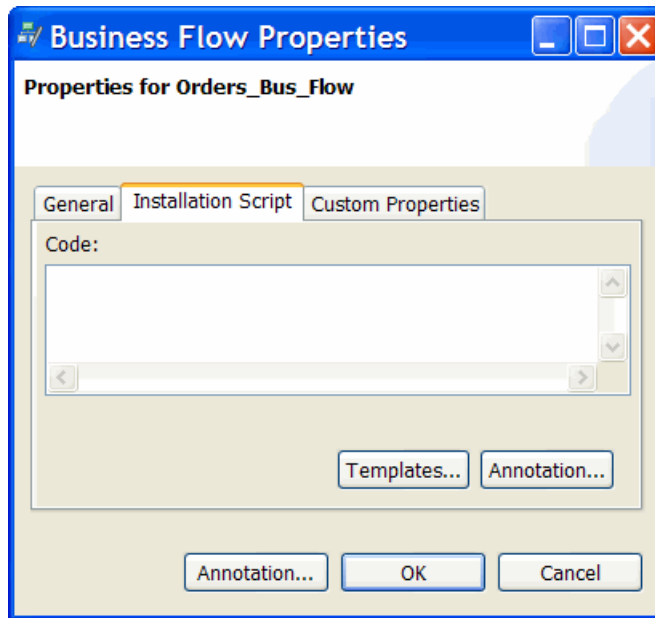




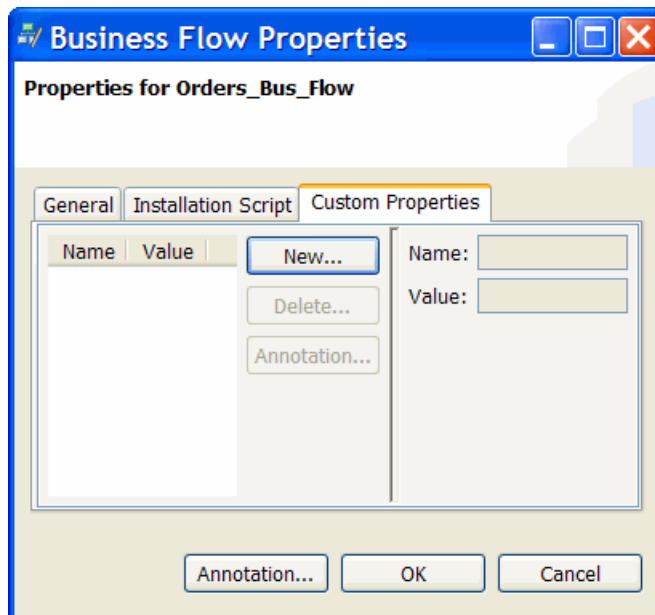
- 4 Click **Properties** to display and optionally change the business flow's properties.
- 5 On the **General tab**, you can see the version of the business flow and order the parameters used by the business flow.



- 6 Select the **Installation tab**. In the Code area, you can enter a Groovy script to be executed at deployment time. **Templates** provide Groovy script templates to help you get started.



- 7 Select the **Custom Properties** tab. The Custom Properties page opens and behaves similarly to the Custom Properties page for a database to file cartridge, [Editing a database to file cartridge](#) (page 74).



- 8 Click **OK**.
- 9 You can now begin to add activities to your business flow. Refer to the following sections for more information:
 - [Adding a database to file cartridge](#) (page 135)
 - [Splitting an activity](#) (page 136)
 - [Uploading to a test database](#) (page 137)
 - [Adding or editing a script](#) (page 138)

- [Adding or editing a condition](#) (page 141)
- [Adding or editing an interrupt](#) (page 142)

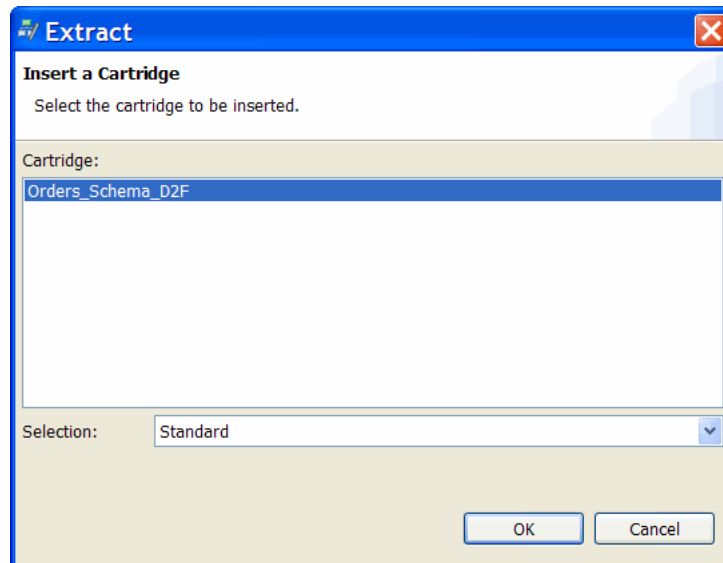
See also [Generating and deploying](#) (page 145) for more information on generating and deploying your business flow.

Adding a database to file cartridge

To add a database to file cartridge to a business flow:

- 1 Expand the Business Flows node in the Project Navigator and double-click a business flow.
- 2 Click the **Extract** tool.
- 3 Click the location in the business flow where you want to add the cartridge. The dialog box displays.
- 4 Select a database to file cartridge from the list.

NOTE You must have already created the cartridge in your project for it to appear in this list. Refer to [Creating a new cartridge](#) (page 73) for more information about creating cartridges.



- 5 For Selection, choose:
 - **Standard selection** is a method of data selection that restricts itself to the rows identified by the model. It does not attempt to traverse related rows across multiple tables.

See also *HP Test Data Management Concepts Guide* for more information about standard data selection methods.

- 6 Click **OK**. The activity is added to the business flow diagram.

Adding schema-based cartridges

Adding a schema-based cartridge works the same way as [Adding a database to file cartridge](#) (page 135).

Splitting an activity

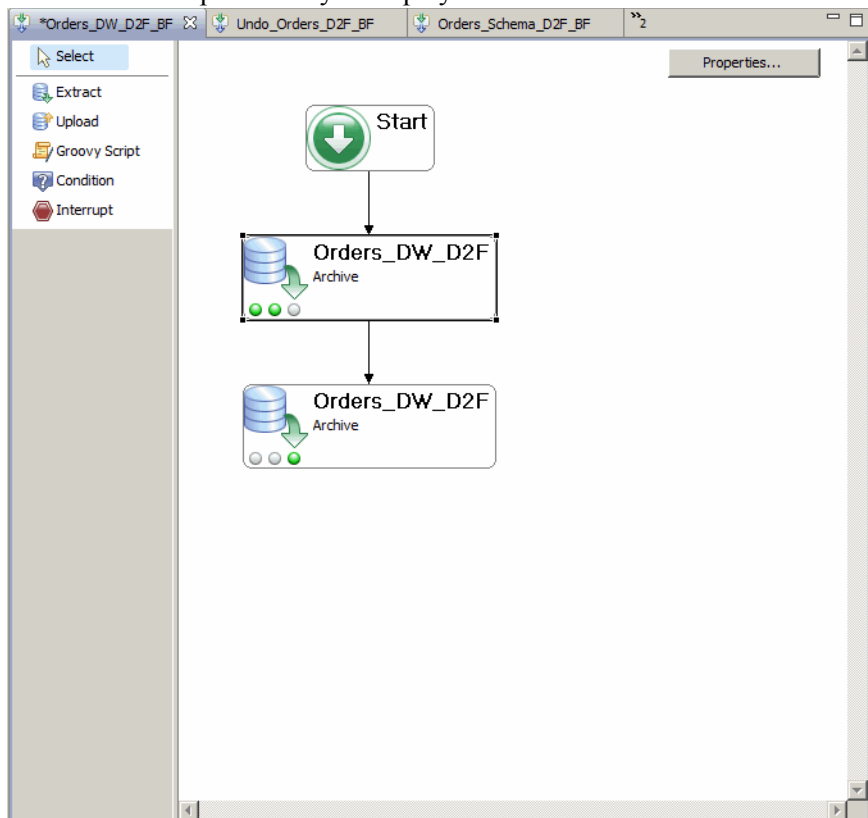
To split an activity:

NOTE In non-intrusive environments only, selection and copy occur at the same time. If you split a cartridge in a non-intrusive environment, selection will result in an empty operation, whereas copy will incorporate both selection and copy. Clean-up also results in an empty operation.

- 1 Expand the Business Flows node in the Project Navigator and double-click a business flow.
- 2 Right-click on an activity in the editor and select **Split**. The Split Activity window opens.

NOTE You cannot split an activity based on a schema-based cartridge.

- 3 Use the shuttle buttons to indicate where the job split should be located.
- 4 Click **OK**. The split activity is displayed.



The bullets represent the steps of the job. Green bullets indicate that the activity contains the corresponding step. Empty bullets indicate that the activity does not contain the step.

The step name is displayed when you move the mouse cursor over a bullet.

- 5 Repeat [step 2](#) through [step 4](#) until you have completed all the splits that you want in your business flow.
- 6 Delete an activity to reverse a split. The steps are merged into the remaining activity.
- 7 Save your work.

See also [Generating and deploying](#) (page 145) for more information on generating and deploying your business flow.

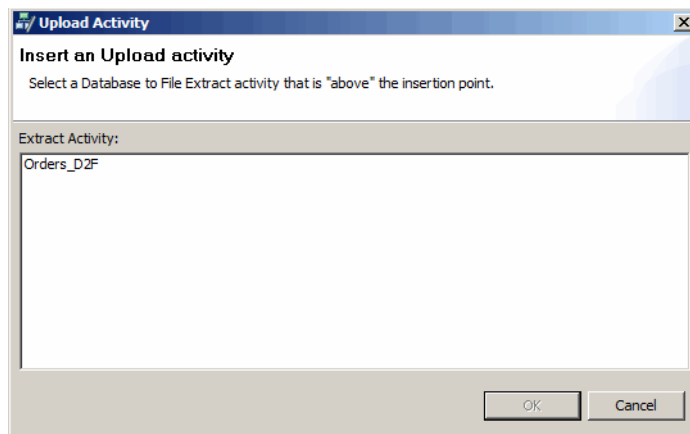
Uploading to a test database

You can use Upload to add an upload to a test database activity immediately after the extraction cartridge activity in the business flow. Uploading to a test database in this manner saves you from having to upload the test database separately from the Web Console.

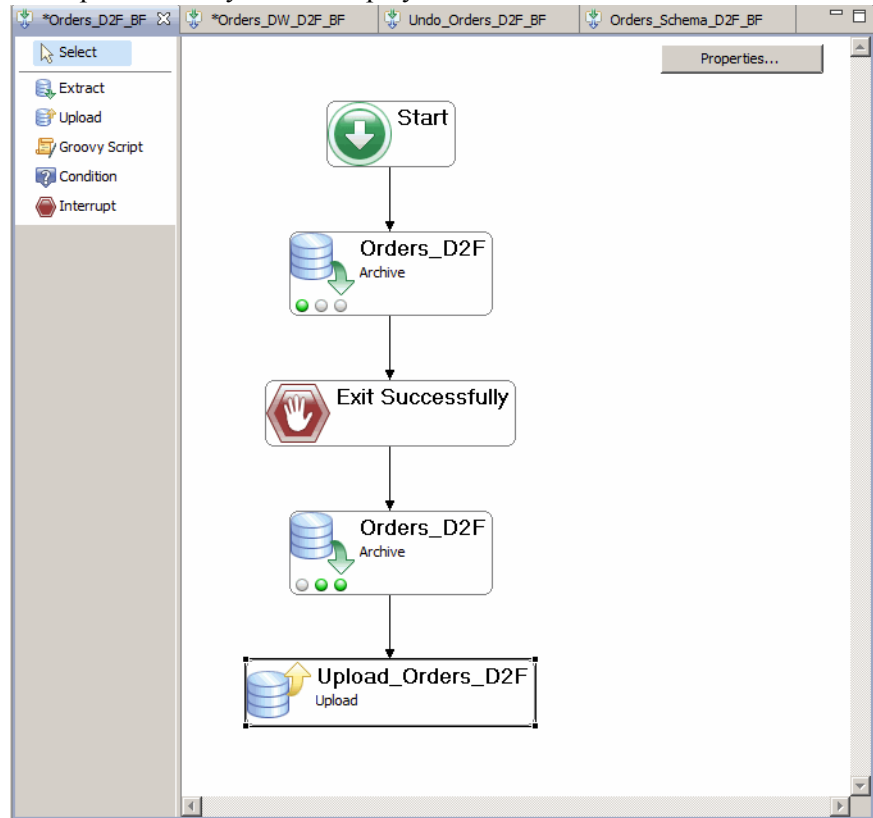
To add an upload activity to a business flow:

- 1 Expand the Business Flows node in the Project Navigator and double-click a business flow.
- 2 Click **Upload** in the tool bar and drop it to the editor below an extraction cartridge. The Upload Activity dialog opens.
- 3 Select an Extract activity that precedes the insertion point and click **OK**.

NOTE If there is no Extract activity before this point, then there is nothing to select. Ensure that the insertion point follows an Extract activity.



The Upload activity is now displayed in the business flow.



4 Save your work.

When you run the business flow from the Web Console, you will be prompted to specify the test database location to which to upload the extracted data. The upload occurs immediately after the extraction successfully completes.

Adding or editing a script

Groovy scripts used in a business flow must be written in the Groovy language version 1.5. You should test your Groovy scripts in a standalone Groovy environment before including them in your business flow.

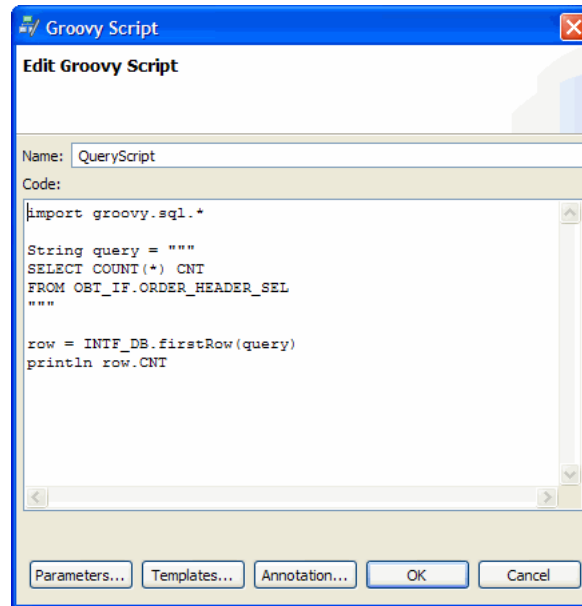
NOTE HP Test Data Management does not validate your code. You must validate and ensure that your code performs as expected at runtime yourself.

Related information <http://groovy.codehaus.org/>

To add or edit a script:

- 1 Expand the Business Flows node in the Project Navigator and double-click a business flow.
- 2 Click **Groovy Script** in the tool bar and drag it to the editor, or double-click an existing script activity. The Groovy Script dialog box opens.
- 3 Type or edit the name for the script.

- 4 Type the Groovy code in the Code field.



- 5 Optionally, click **Parameters** to add or edit a parameter. Refer to [Creating or editing parameters using the Parameters button](#) (page 59).
- 6 Click **Template** to choose a code template for your Groovy script.
 - a Select a template in one of the following categories:
 - SQL
 - Operating System
 - Utilities
 - Utilities - Install BF
 - b Click **OK**.
- 7 Edit the code as necessary.

The code you type is analyzed. If there are any errors, the error message is displayed at the bottom of the window.

Click the error message to move the caret to the error position.
- 8 Click **OK**.
- 9 Save your work.

See also [Generating and deploying](#) (page 145) for information on generating and deploying your Business Flow.

Adding custom Groovy events to a business flow

In addition to including Groovy scripts within business flows, you can insert Groovy events to be executed before and after certain HP Test Data Management events. Groovy events can be executed at the following points:

- **Before deployment.** This Groovy code fires just before HP Test Data Management deploys your business flow to the environment. This event is useful for confirming that anything required for the business flow to deploy successfully is in place prior to deployment. For example, you might use it to check that certain system parameters are properly seeded.
- **After deployment.** This Groovy code fires just after HP Test Data Management completes deployment of your business flow to the environment. This event is useful for any sort of cleanup that you might want to perform after deployment.
- **Before launch.** This Groovy code fires just before HP Test Data Management runs your business flow in the environment. This event is useful for confirming that anything required for the business flow to run successfully is in place prior to execution. For example, you might use it to check that certain locations are accessible and available before trying to execute the business flow.
- **After launch.** This Groovy code fires just after HP Test Data Management completes execution of your business flow in the environment. This event is useful for any sort of cleanup or reporting that you might want to perform after execution. For example, you might use it to generate a report showing you certain results from the business flow's execution.

Your HP Test Data Management installation includes sample Groovy scripts for all of these supported events. You can simply modify these samples for your own purposes. For example, if you need to collect auditing information upon deployment or execution of a business flow, you can customize the sample Groovy files for the after deployment or launch events to include the necessary logic. Or, suppose that you must provide an external locking mechanism for the deployment of a business flow, a custom script can be of help.

The following sample scripts are located in

`<HPDBA_install_directory>\obt\extensions\runtime\events:`

- `beforeDeployBf.sample`
- `afterDeployBf.sample`
- `beforeLaunchBf.sample`
- `afterLaunchBf.sample`

To use the scripts, simply rename the files by replacing the `.sample` file extension with `.groovy`. You can view all output in the `obt\logs` directory.

If multiple events exist, then the order in which the scripts are run is determined by basic alphanumeric sorting rules. The following list shows the order in which a sample group of multiple Groovy events would run:

```
20runme.groovy
S10runme.groovy
S20runme.groovy
Zrunme.groovy
aRunMe.groovy
s10runme.groovy
s11runme.groovy
s1runme.groovy
```

```
s20runme.groovy  
s2runme.groovy  
s3runme.groovy
```

Adding or editing a condition

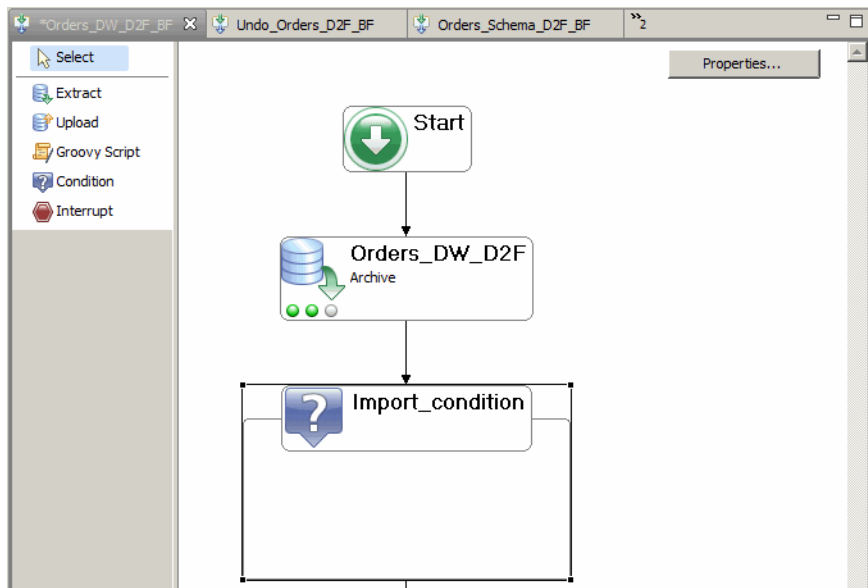
In some cases, you may want to check some condition before proceeding with an activity in your business flow. If the condition is met, then you will proceed with the activity. If not, you skip the activity and go to the next activity, if there is one, or end the business flow.

To add or edit a condition:

- 1 Expand the Business Flows node in the Project Navigator and double-click a business flow.
- 2 Click **Condition** in the tool bar and drag it to the editor, or double-click an existing condition. The dialog box opens.
- 3 Type or edit the name for the script.
- 4 Type or edit the Groovy code in the Code field for your condition. The code must return a Boolean value to indicate whether the condition was met. For example, you might enter something like:

```
import java.io.*  
new File("c:/temp/x.txt").exists()
```

- 5 Click **OK**. Your condition is created or modified. Notice the bounding box around the condition.



- 6 To place an activity within the condition, click a tool in the tool bar (Extract, Groovy Script, Condition, Interrupt).

- 7 Click inside the bounding box of the condition and complete the resulting dialog as necessary. The activity will be governed by the condition. If the condition returns true, the activities inside of its bounding box are executed. If the condition returns false, the activities inside of the bounding box are skipped.

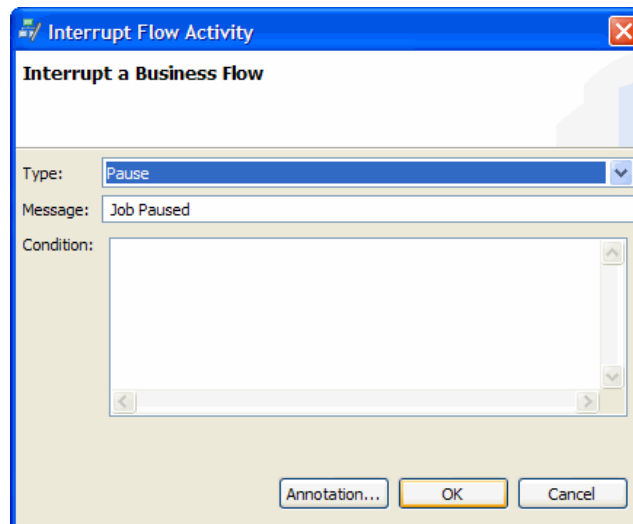
NOTE You cannot split a copy activity and place some of its pre- or post-activities inside of the condition and some outside the condition. All of pre- and post-activities for the copy must be contained either inside or outside of the condition. To stop execution between pre- and post-activities of a copy activity, you can use an interrupt. Refer to [Adding or editing an interrupt](#) (page 142)

- 8 Repeat [step 6](#) and [step 7](#) for any additional activities you want to be under the control of the condition.

Adding or editing an interrupt

In some cases, you may wish to stop or pause your business flow, typically after checking some condition. To stop or pause a business flow at any point, you can introduce an interrupt into the business flow.

- 1 Expand the Business Flows node in the Project Navigator and double-click a business flow.
- 2 Click **Interrupt** in the tool bar and drag it to the editor, or double-click an existing interrupt activity. The Interrupt Flow Activity dialog box opens.



- 3 Choose the Type of interrupt you want to create.
 - **Error** means that the business flow stops with an error condition, meaning that it cannot continue.
 - **Pause** means that the business flow execution is temporarily suspended and can be continued later.

- **Exit Successfully** means that the business flow completed its activities successfully.
 - **Exit with Warning** means that the business flow completed, but some activities caused warnings to be issued.
- 4 Type or edit the Message to display when the interrupt is executed.
 - 5 In Condition, type the Groovy code for your condition. The code must return a Boolean value to indicate whether the condition was met. For example, you might enter something like:

```
import java.io.*
new File("c:/temp/x.txt").exists()
```
-
- TIP** A condition is required if your interrupt type is **Error**. For the other types of interrupt, a condition is optional.
-
- 6 Click **OK**. Your interrupt is created.

Testing business flows

With any business flow, you should carefully test it before running it in your production environment. As best practice, you should:

- Thoroughly test and debug your Groovy scripts in a Groovy IDE.
- For interrupts and conditions, ensure that you have test cases that meet and do not meet the condition. Otherwise, you cannot exercise all branches.
- Preview any models referenced by cartridges in your business flow. Ensure that you vary parameter values within reasonable ranges.
- Preview any extraction cartridges used in your business flow.
- Deploy and run the business flow in a development environment. Carefully check all logs and confirm all results before trying the business flow in production.

See also *HP Test Data Management Troubleshooting Guide*

Deleting a business flow

To delete a business flow:

- 1 In the Project Navigator, select the business flow you want to delete.
- 2 Right click the business flow and choose **Delete**, or simply press the Delete key.

After your cartridge or business flow is complete and tested, you are ready to try running it on the actual active database, outside of Designer. In order to run against your active database, you must first deploy the business flow or cartridge to your deployment environment. The deployment environment is where HP Test Data Management actually perform operations on your specified source (active) database. Until your cartridge or business flow is deployed to the environment, you cannot execute it.

This chapter includes:

- [Deploying locally](#) (page 145)
- [Deploying remotely](#) (page 149)
- [Generating deployment files](#) (page 150)

Deploying locally

Local deployment allows you to deploy a cartridge or business flow to the same system for which Designer is configured. Deployment Assistant displays different pages depending on the type of cartridge you are deploying.

NOTE When you deploy a cartridge by itself, HP Test Data Management wraps it in a business flow for deployment purposes.

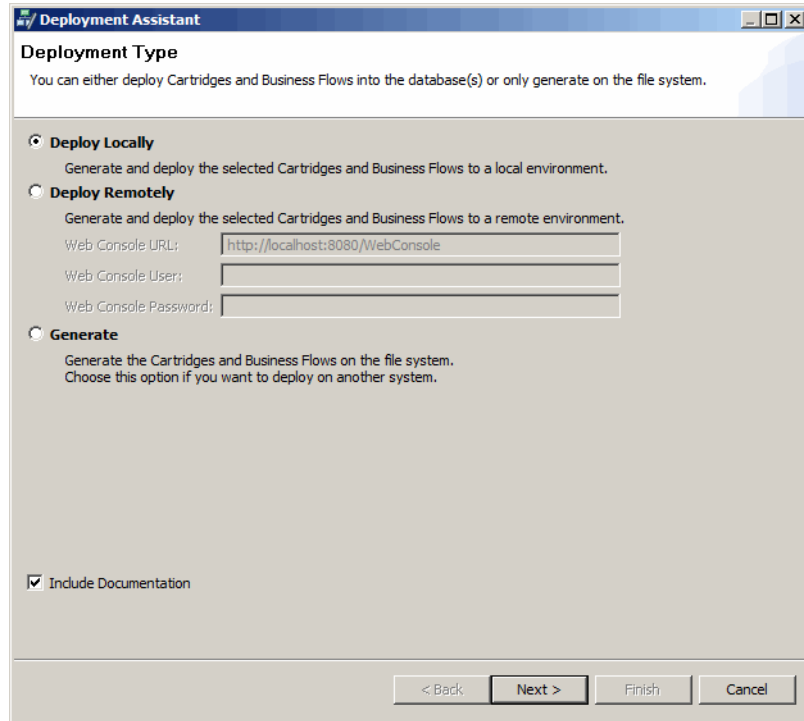
This section includes:

- [Deploying a schema-based database to file cartridge](#) (page 145)
- [Deploying a business flow](#) (page 148)

Deploying a schema-based database to file cartridge

To deploy a schema-based database to file cartridge:

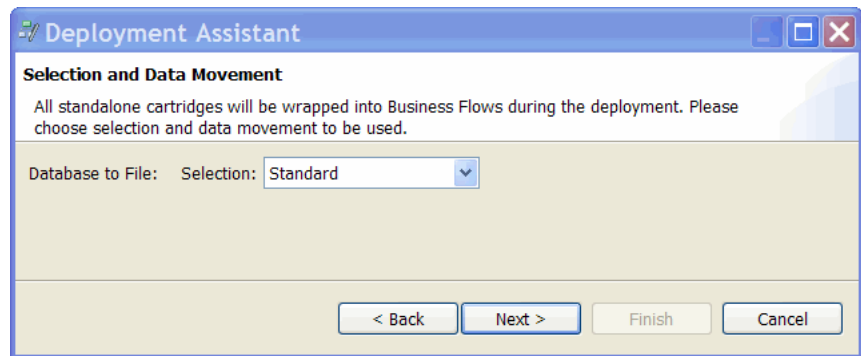
- 1 In Designer, expand Cartridges in the Project Navigator.
- 2 Right click the database to file or schema-based database to file cartridge you want to deploy and select **Deploy**. The Deployment Assistant opens.
- 3 Select the **Deploy Locally** radio button.



- 4 Optionally, check **Include Documentation**.

See also [Annotating projects](#) (page 122)

- 5 Click **Next**.

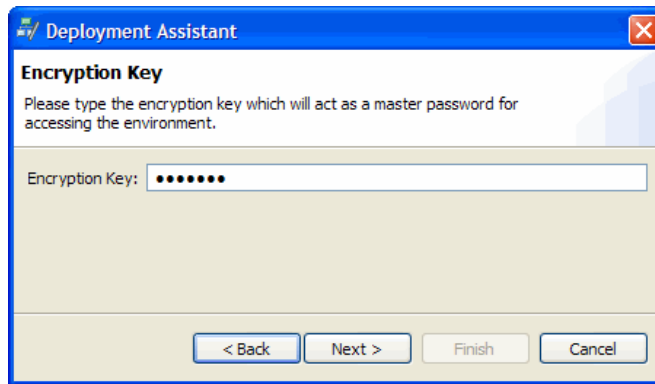


- See also*
 - 6 Choose:

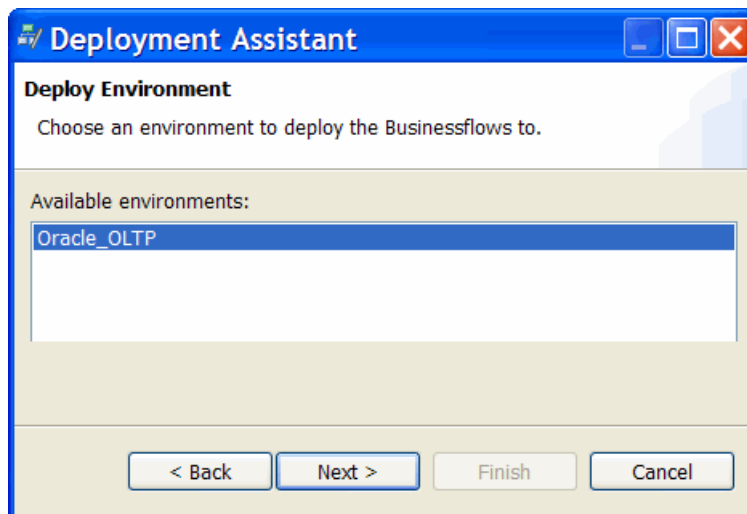
- **Standard.** Standard is a method of data selection that restricts itself to the rows identified by the model. It does not attempt to traverse related rows across multiple tables.

- 7 Click **Next**.

- 8 Type the encryption key value, which was defined during the repository installation. For example, EKEY. The encryption key is case sensitive. The encryption key is remembered for the rest of the session.

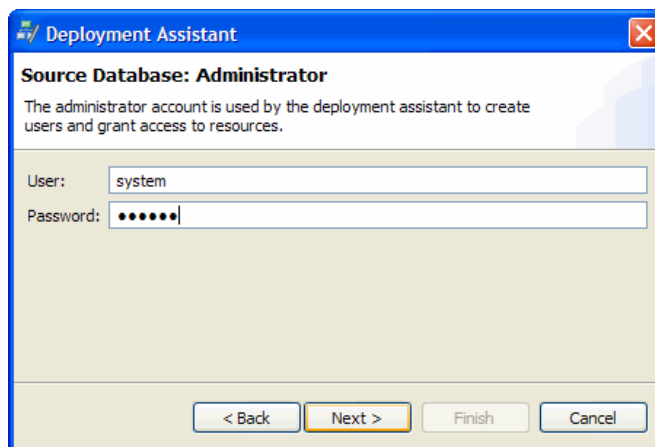


- 9 Click **Next**.
- 10 Choose an environment, then click **Next**.



- 11 Enter the login information for your active database, then click **Next**.

NOTE If you are working in a non-intrusive environment, this step is skipped (because no administrator is required).



- 12 On the Summary page, click **Finish**. Your cartridge may take a few moments to deploy.

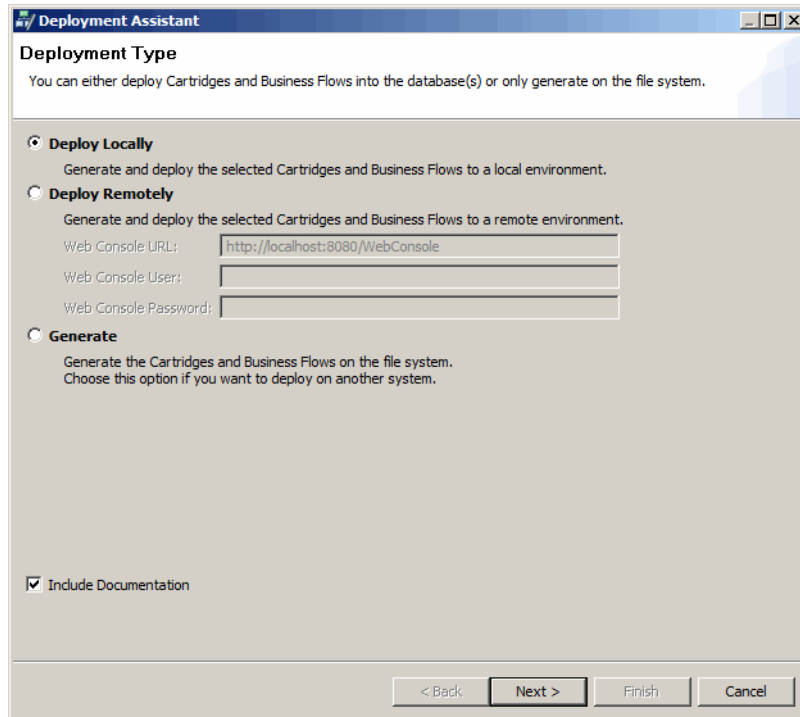
Deploying a business flow

NOTE If using a new, generic JDBC driver in a database to file non-intrusive environment, HP Test Data Management checks the driver's Name Types and JDBC Type values upon your first connection. If data is missing, then you may encounter an error at runtime. If there is a data mismatch, then you can encounter data loss upon business flow deployment. Before deploying a business flow for a database to file, non-intrusive environment, examine your data closely to ensure that there are no "mismatches" between the name Type and JDBC Type.

Refer to the *HP Test Data Management Web Console and Query Server User Guide* and *HP Test Data Management Troubleshooting Guide* for details about data mapping errors and mismatches when using generic JDBC drivers in database to file non-intrusive environments.

To deploy a business flow:

- 1 In Designer, expand Business Flows in the Project Navigator.
- 2 Right click on the business flow you want to deploy and select **Deploy**. The Deployment Assistant opens.



- 3 Select the **Deploy Locally** radio button.
- 4 Optionally, check **Include Documentation**.

See also

[Annotating projects](#) (page 122)

- 5 Click **Next**.
- 6 Type the encryption key value. For example, EKEY. The encryption key is case sensitive. The encryption key is remembered for the rest of the session.
- 7 Enter the login information for your active database as described in [step 11](#) (page 147) through [step 12](#) (page 148).

NOTE If you are working in a non-intrusive environment, this step is skipped (because no administrator is required).

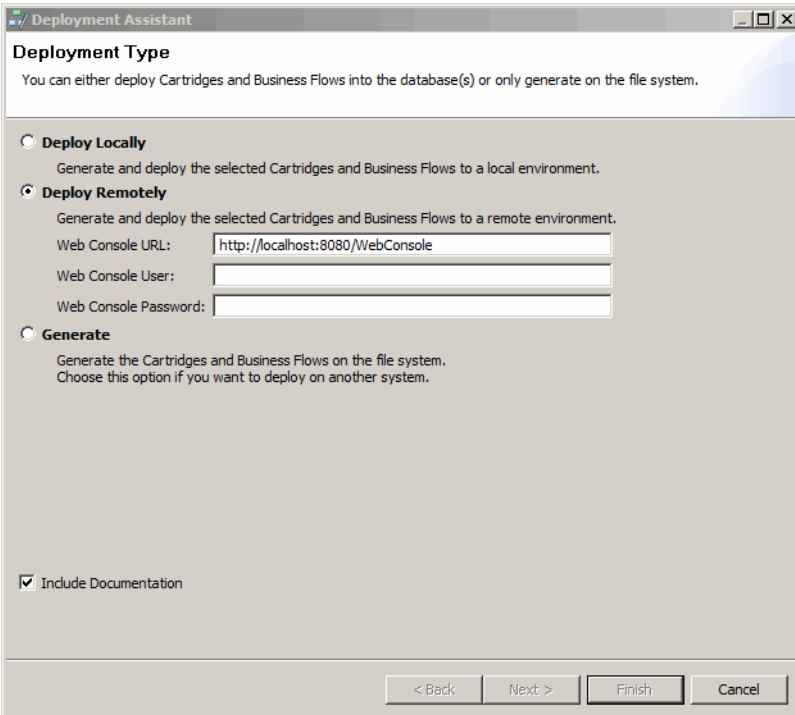
Deploying remotely

Remote deployment allows you to deploy from Designer to a system that is remote from the database for which Designer is installed and configured.

NOTE The Web Console must have already been started on the remote machine for this procedure.

To deploy remotely:

- 1 In Designer, expand Cartridges or Business Flows in the Project Navigator.
- 2 Right click on the cartridge or business flow you want to deploy and select **Deploy**. The Deployment Assistant opens.
- 3 Select the **Deploy Remotely** radio button.



The screenshot shows the 'Deployment Assistant' dialog box. The title bar says 'Deployment Assistant'. The main heading is 'Deployment Type' with a subtext: 'You can either deploy Cartridges and Business Flows into the database(s) or only generate on the file system.' There are three radio button options: 'Deploy Locally' (unselected), 'Deploy Remotely' (selected), and 'Generate' (unselected). Under 'Deploy Locally' is the text: 'Generate and deploy the selected Cartridges and Business Flows to a local environment.' Under 'Deploy Remotely' is the text: 'Generate and deploy the selected Cartridges and Business Flows to a remote environment.' Below this are three text input fields: 'Web Console URL:' with the value 'http://localhost:8080/WebConsole', 'Web Console User:', and 'Web Console Password:'. Under 'Generate' is the text: 'Generate the Cartridges and Business Flows on the file system. Choose this option if you want to deploy on another system.' At the bottom left is a checked checkbox labeled 'Include Documentation'. At the bottom right are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

- 4 For Web Console URL, enter the URL for the Web Console associated with the remote repository database to which you want to deploy.

- 5 Enter the Web Console user name under which you plan to deploy, for example, `admin`.
- 6 Enter the password for the Web Console user you specified in the previous step.
- 7 Optionally, check **Include Documentation**.
- 8 Click **Next**.
- 9 From this point forward, the Deployment Assistant behaves in much the same manner it would for a local deployment:

See also [Annotating projects](#) (page 122)

- See:*
- [Deploying a schema-based database to file cartridge](#) (page 145)
 - [Deploying a business flow](#) (page 148)

Generating deployment files

In some cases, you may not have direct access to the production system where the business flow or cartridge is to be deployed. For example, you might be providing cartridges or business flows to another organization with systems separate from your own. In such situations, you can generate the deployment files for your cartridges and business flows, and then move those files to the other system for subsequent deployment.

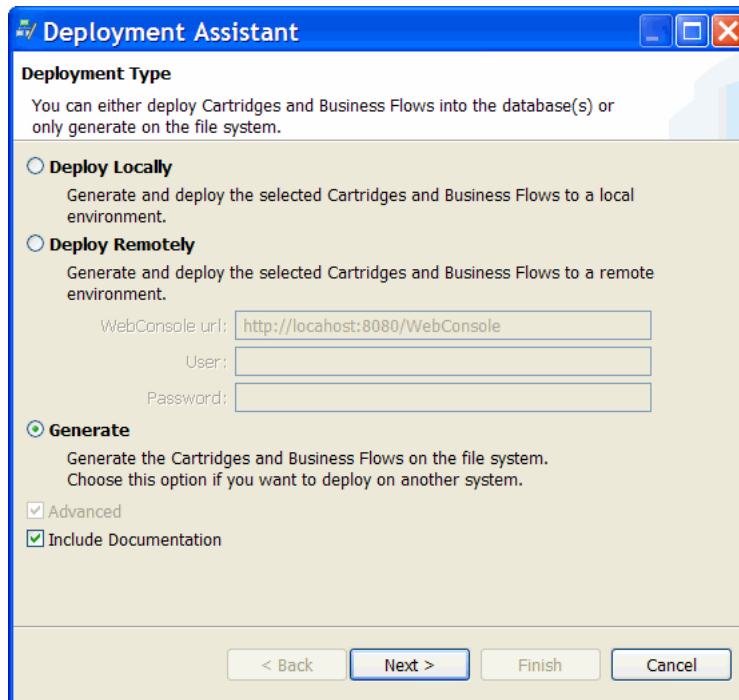
To generate and deploy files:

- 1 Generate a deployment file from the Deployment Assistant configured with your development instance (that is, the database for which Designer is installed and configured).
 - [Generating a cartridge or business flow](#) (page 150)
- 2 Move the deployment file to the remote system and invoke the Web Console associated with the remote system to deploy that file.
 - [Deploying generated files](#) (page 152)

Generating a cartridge or business flow

To generate a cartridge or business flow:

- 1 In Designer, expand Cartridges or Business Flows in the Project Navigator.
- 2 Right click on the cartridge or business flow you want to deploy and select **Deploy**. The Deployment Assistant opens.
- 3 Select the **Generate** radio button.

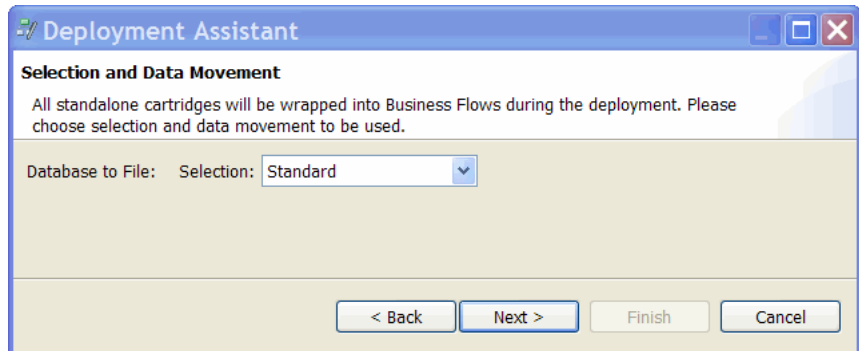


- 4 Optionally, check **Include Documentation**.

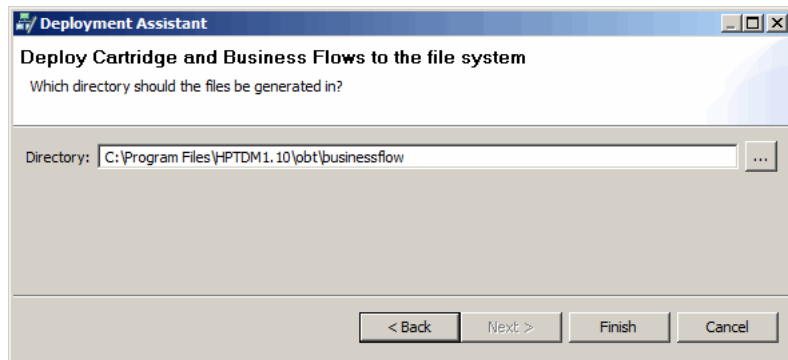
See also

[Annotating projects](#) (page 122)

- 5 Click **Next**.
- 6 If you are deploying a cartridge, you are prompted to choose the data selection and movement methods. If you are deploying a business flow, you will proceed directly to [step 8](#) (page 152).



- 7 Choose the selection and data movement methods from the lists of values and click **Next**.



- 8 Type or browse to the location where you want to store the deployment file. For the sake of convenience, you may wish to choose a path (a network or shared drive) that is accessible from the remote system on which you plan to deploy the cartridge or business flow. Otherwise, you will need to move the deployment files to such a location after you generate them.
- 9 Click **Finish**. The deployment file is generated in the path that you chose in the Deployment Assistant. In that path, you should find a .busflow file with the same name as your business flow or cartridge. If you chose to include documentation, you should also find a PDF file in the same path.
- 10 Perform the steps in [Deploying generated files](#) (page 152) on the remote system where you want to deploy and run the business flow or cartridge.

Deploying generated files

Before performing the steps in this section, you must have performed the steps in [Generating a cartridge or business flow](#) (page 150) and have a login with deployment privileges to the Web Console for the system where you plan to deploy.

NOTE The Web Console must have already been started on the remote machine for this procedure.

- 1 In a browser, invoke the Web Console URL for the system where you want to deploy and run the business flow or cartridge (<http://<hostname>:8080/WebConsole>).
- 2 Login as a user with privileges to deploy business flows, for example, admin. Ensure that the currently active environment is the one in which you want to deploy the business flow.
- 3 From the menu at the top of the page, choose **Business Flow Deployment**. If you stored the generated files in `<install_dir>\obt\businessflow`, then they will appear in the list of business flows available for deployment. If you do not see the business flows you want to choose in the list, do the following:
 - a Click **Import** in the left navigation pane.
 - b Import the business flows in one of two ways:

- Choose the **Path to the Server directory** radio button, enter a valid path on the server where the deployment files (*.busflow) are stored, and click **Copy**.
- Choose the **Business flow file to upload** radio button and click **Browse** to select the deployment files (*.busflow).

The business flows are uploaded from the client to the server.

- c From the menu at the top of the page, choose **Business Flow Deployment**. The business flow(s) that you just imported should now appear in the list.
- 4 Check the business flow(s) in the list that you wish to deploy.
 - 5 Click **Next**.
 - 6 Depending upon the type of cartridge(s) contained in the business flow(s), pages will appear where you can provide the necessary information for deployment. For more information on deployment through the Web Console, refer to *HP Test Data Management Web Console and Query Server User Guide*.
 - 7 When you reach the summary page, click **Finish** to deploy your business flow.

Using Designer, you can safely customize a project you receive from a third party. When the third party releases revisions to the project, you can merge your customizations into the revised project from the third party. Thus, you need not constantly repeat your changes each time you receive a revision from the vendor.

This chapter explains, for consumers, how to customize a project you receive and merge new versions with your customizations.

NOTE If you are a third part vendor developing projects for consumption, you should contact HP about becoming a partner and learning more about developing customizable projects.

This chapter includes:

- [About customization](#) (page 155)
- [Guidelines for customizations](#) (page 156)
- [Customizing a project](#) (page 157)
- [Merging customizations into a new project revision](#) (page 158)

About customization

When you receive a project that was pre-built, you typically need to modify the project for your environment. For example, you might need to add to the model some tables that are required by your environment but that are not part of a default implementation of the packaged application.

As a consumer of a customizable extraction project, you must be aware of several considerations and tasks:

- When you receive a project, it should be locked. When a project is locked by a vendor, it means that the various files that comprise the project are all locked. For example, each model in a project has a file associated with it. When the vendor locks the project, the existing model files are all locked.

TIP Note that if you take an action, such as adding a new model to the project, that results in the creation of a new file, then the new file would not be locked.

- Even though the project is locked, you can perform most modifications that you like on it. You should be aware, though, that some modifications may cause the modified model, cartridge, business flow, or parameter to be

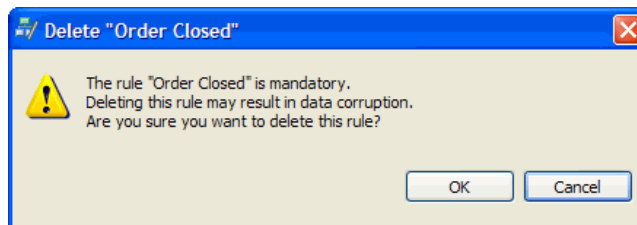
unsupported when upgraded to a newer revision of the project. To ensure that your modifications are completely supported, you should modify the project according to the guidelines in [Guidelines for customizations](#) (page 156).

- Generate a list of unique identifiers (UIDs) to use in validating your customizations.
- When the vendor releases new versions of the project, merge your customizations into the new version.

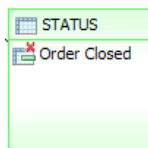
Guidelines for customizations

As you customize a project, you must take the following into account:

- By default, locked projects open in what is known as Customization view. In this mode, Designer gives you various visual cues in the form of icons in the model editor, to call out customizations. You can toggle Customization view on and off from the toolbar using the Custom View tool.
- Ensure that the project files are locked before you begin making modifications. If the Custom View tool in the model editor is disabled, it means that the files are not locked. If the files are not locked, you should lock them before you begin by selecting **Customization > Lock Project**.
- You can make modifications to the model and cartridges in a locked project. As long as you adhere to the guidelines in this section, those customizations are supported, which means you can merge them into a new version of the project when you receive it.
- When you make certain changes to the model or cartridge in a locked project, you receive confirmation dialogs with warning messages to ensure that you really want to make the change.



If you click OK, the operation is carried out and, in the model editor, the object is marked in some fashion to indicate the customization.



- You can add new models, cartridges, business flows, and parameters to the project, but note that they are not locked because they create new unlocked files within the project.

- In a locked project, if you delete an original table use or a mandatory rule, it is marked as deleted rather than actually deleted. If you unlock the project, objects marked as deleted are actually removed. Similarly, if you add new tables or rules to a locked project, these are not added to the base project. If you unlock the project, then the new objects are added to the base project.
- Because original tables are not actually removed from a locked project, you need to take special care when removing a table. If you later recreate that table in the model, you could end up with the same table in the model twice, which could cause problems for you at some point.
- Each rule in the project is categorized according to its Customization property as follows:
 - Optional rules can be changed or deleted without triggering warnings or changing the UID.
 - Recommended rules can also be changed or deleted but only with great caution. Changing a recommended rule will trigger a warning, but the UID is not changed.
 - Mandatory rules cannot be changed or deleted. Such changes trigger a warning and cause the UID to change. If you choose to make the change anyway, the original rule is marked as deleted and a new rule is created to replace it.
- Business flow changes are not merged. If you make changes to a business flow, you cannot merge those customizations into a new version of the original project.

Customizing a project

When you receive a new project to customize, you first import it into Designer for customization. Once you have the project in Designer, you can modify it and generate its UIDs for future reference when you merge a new version from the vendor with your modifications.

- 1 In Designer, choose **File > New Project**. The New Project dialog box displays.
- 2 Enter a name for the project and select **Customize an Existing Project**.
- 3 Click **OK**. The Import Project dialog, from which you can select a .hdp file, displays.
- 4 Browse to and select the .hdp that you received from your vendor.
- 5 Click **Open**. The selected project is imported into Designer under the specified project name.
- 6 When you receive a new project, ensure that the project is locked. Open a model in the project for editing and, if the Custom View tool in the toolbar is disabled, it means that the project is not locked. If the project is not locked, you should lock it before you begin by selecting **Customization > Lock Project**.

- 7 Modify the project as desired but within the guidelines described in [Guidelines for customizations](#) (page 156).
- 8 Save the project.
- 9 Choose **Customization > Generate UIDs**.
- 10 Click **Copy to Clipboard**.
- 11 Paste the UIDs into a document and save it for future reference.
- 12 Click **OK** in the Unique IDs dialog box.

Merging customizations into a new project revision

TIP Before merging customizations, you should backup your project files.

To merge customizations into a new project revision:

- 1 Open your customized version of the project.
- 2 Review the guidelines in [Guidelines for customizations](#) (page 156) to ensure that you understand which of your modifications will be merged and supported.
- 3 Choose **Customization > Merge Project**.
- 4 In the Merge Project dialog box, browse for the new project file from your vendor, which has the .hdp extension.

TIP .hdp is an exchange format for HP Test Data Management projects and contains all of the files associated with the project. You create .hdp files by exporting a project from Designer, File > Export Project.

- 5 Click **OK**.
- 6 Optionally, if you find that some of your customizations were not merged, compare the UIDs from your customized project to those of the developer's original project. Wherever the UIDs do not match, it indicates that you have an unsupported change.

TIP If you want to create a new project based on a project from a vendor, you can do so by choosing File > New Project and checking Customize an Existing Project. Refer to [Creating a new project](#) (page 118) for more details.

A

JDBC driver validation and configuration

This appendix describes options for conducting JDBC driver validation and configuration. The tools available for conducting your JDBC driver validation testing and configuration include:

- **The JDBC test console.** Use the console to perform all JDBC driver validation tests and configuration tasks. Driver validation tests that are run from the console are identical to those performed by the JDBC driver validation tool and the JDBC configuration utility. The console simply provides a consolidated, interactive framework from which to perform all JDBC driver testing and configuration. The console provides you the option to intervene in the case of errors and to perform additional configuration tasks.
- **The JDBC driver validation tool.** Use this tool to validate your JDBC drivers to ensure that they are compatible with HP Test Data Management.
- **The JDBC DDL/SQL configuration utility.** Use this utility to identify and resolve configuration issues with generic JDBC connections before attempting an extraction. Running tests via this utility is essentially the same as running them via the JDBC test console—both are interactive and notify you when there is a failure, and also provide an opportunity to correct any errors with your JDBC driver.

This appendix contains the following sections:

- [Using the JDBC test console](#)
- [Running JDBC driver validation tests](#) (page 170)
- [Running the JDBC DDL/SQL configuration utility](#) (page 171)

Using the JDBC test console

If you prefer to conduct all JDBC testing and configuration within the same console and intervene to make configuration updates and corrections, use the JDBC test console.

The JDBC test console allows you to interactively perform the following testing and configuration:

- JDBC metadata method test

This test examines the metadata associated with the JDBC driver to determine whether or not there is anything present that would lead to a critical error later on when you attempt to extract or copy data. Numerous calls are performed to examine metadata associated with your JDBC driver. For example, metadata

such as the maximum size of a table name, the maximum size of an index name and the database version are just a few of the many pieces of metadata being tested.

Metadata tests produce an output file that enables you to easily identify critical errors, which are preceded by an exclamation mark (!). Errors not preceded by an exclamation mark are unlikely to affect the success of a copy or extraction.

NOTE This is the same test that is performed by the JDBC driver validation tool.

See [Running the JDBC metadata method test](#) (page 162).

- **JDBC DDL/Metadata configuration**

This test examines your database-specific DDL and SQL configuration information (for example, the syntax for creating tables, indexes, and primary keys, as well as SELECT and INSERT statements) for errors in syntax.

NOTE This is the same test that is performed by the JDBC DDL/SQL configuration utility.

See [Testing and configuring JDBC DDL/Metadata](#) (page 162)

- **JDBC table column data type**

This test examines the data types associated with each table (tables that are most likely targets in your copy) and lists the data type information provided. If the data type information is correct and complete, then no further action is required. If the data type information is incorrect, then you need to either add the data type or edit an existing one.

The following steps describe how to change a column type, VARCHAR, to behave as an INTEGER. This test also looks at the JDBC driver to determine whether or not it supports this data type.

See [Running the JDBC table column data type test](#) (page 168).

To launch the JDBC test console for JDBC driver validation and connection testing:

- 1 To launch the JDBC driver validation and connection console from Windows-based machines, navigate to <install_dir>\obt\bin and enter the following command:

```
unified_drivertests.bat
```

To launch the JDBC driver validation and connection console from Linux, Unix or Solaris-based machines, navigate to the <install_dir>/obt/bin directory and enter the following command:

```
./obt-launcher.sh  
com.outerbay.foundation.components.prepare.deploy.Deploy  
mentLauncher 2 -Dobtpa.top="<obt_top_dir>" -buildfile  
"unified_drivertests.xml"
```


- 2 Optionally, you can pass a file with the connection properties to be used with either launcher. To use the properties file on Windows-based machines, enter the following command:

```
unified_drivertests.bat <driver_props>.properties
```

where *<driver_props>.properties* is the actual name of the properties file.

To use the properties file on Linux, Unix or Solaris-based machines, enter the following command where *<properties_file_name>* is the actual name of the properties file:

```
./obt-launcher.sh
com.outerbay.foundation.components.prepare.deploy.Deployme
entLauncher 2 -Dobtpa.top="<obt_top_dir>" -buildfile
"unified_drivertests.xml"
-Dproperty.file.name="<driver_props>.properties"
```

- 3 If you do not provide a properties file, you will be prompted for the connection properties by the console after specifying the JDBC driver you wish to test:

```
test.driver.files:
```

```
test_driver:
```

```
[echo] *****
```

```
[echo] Running Unified Driver Certification
```

```
[echo] *****
```

```
Please select a driver to test:
```

```
[1] ODBC Bridge
```

```
[2] mysql-connector-java-5.1.13-bin.jar
```

```
:2
```

```
Unable to open properties file: home/db2inst1/
```

```
HPDBArchiving630/build_403/obt/config/
```

```
drivertest_connection.properties
```

```
Please enter connection URL for selected JDBC Driver:
```

```
:jdbc:mysql://sardine.cup.hp.com/DEMARC
```

```
Please enter connection User:
```

```
:root
```

```
Please enter connection Password:
```

```
:
```

NOTE The ODBC driver is provided to allow Java to communicate with an ODBC data source. The ODBC Bridge connects to the ODBC client and allows other applications to connect to databases. The ODBC Bridge is present so that you always have a driver choice, even if there is no external driver in the JDBC directory. When connecting using the ODBC Bridge, you must provide a valid JDBC URL containing a data source that is present on your machine, for example: <jdbc:odbc://mydatasource>.

Running the JDBC metadata method test

To run the JDBC metadata method test via the JDBC console:

- 1 After specifying the connection properties, select the JDBC metadata method test:

```
Please select test to run:
[1] JDBC Metadata method test
[2] JDBC DDL/Metadata configuration
[3] JDBC table column data type test
[4] Exit
:1
```

- 2 After you launch the JDBC metadata method test, the console displays the progress along with the results:

```
[unifiedgenericconnectiontesttask] The method
getSchemaTerm returned successful
[unifiedgenericconnectiontesttask] Testing method:
getSearchStringEscape
[unifiedgenericconnectiontesttask] The method
getSearchStringEscape returned successful
[unifiedgenericconnectiontesttask] Testing method:
getTablePrivileges
[unifiedgenericconnectiontesttask] The method
getTablePrivileges returned successful
[unifiedgenericconnectiontesttask] Testing method:
getTableTypes
[unifiedgenericconnectiontesttask] The method
getTableTypes returned successful
[unifiedgenericconnectiontesttask] Testing method:
getTypeInfo
[unifiedgenericconnectiontesttask] The method getTypeInfo
returned successful
[unifiedgenericconnectiontesttask] Testing method:
supportsBatchUpdates
[unifiedgenericconnectiontesttask] The method
supportsBatchUpdates returned successful
[unifiedgenericconnectiontesttask] Please see test
summary in /home/db2inst1/build_403/obt/bin/
metadata_method_test_results.txt
```

- 3 Optionally, review the results displayed in the console in the `/home/db2inst1/build_403/obt/bin/metadata_method_test_results.txt` file.

Testing and configuring JDBC DDL/Metadata

To launch JDBC DDL/Metadata testing and configuration:

- 1 After specifying the connection properties, select JDBC DDL/Metadata configuration:

```
Please select test to run:
[1] JDBC Metadata method test
```

```
[2] JDBC DDL/Metadata configuration
[3] JDBC table column data type test
[4] Exit
:2
```

- 2 Provide the appropriate responses to the configuration inquiries from the JDBC test console. Your responses will be used to create and run the DDL tests. The following is an example—your responses will depend upon your specific configuration:

The JDBC driver indicates that this DB supports catalogs, is this correct?

```
[1] Yes
[2] No
```

:1

The JDBC driver indicates that this DB does not support schemas, is this correct?

```
[1] Yes
[2] No
```

:1

Does this Database support executing DDL using JDBC connections?

```
[1] Yes
[2] No
```

:1

Please enter catalog name where table will be created:

DEMARC

The qualified name 'DEMARC'.hpdba_test_table' will be used for the test, is this correct?

```
[1] Yes
[2] No
```

:1

*****Testing GET_CATALOGS*****

GET_CATALOGS passed

*****Testing GET_SCHEMAS*****

GET_SCHEMAS passed

*****Testing CREATE_TABLE*****

Creating test table using the following DDL:
CREATE TABLE 'DEMARC'.'hpdba_test_table' ('col1'
VARCHAR(100) NOT NULL, 'col2' INTEGER NOT NULL)

CREATE_TABLE passed

*****Testing GET_TABLES*****

GET_TABLES passed

*****Testing INSERT*****

INSERT passed

```
*****Testing SELECT*****
```

```
SELECT passed
```

```
*****Testing VALIDATION*****
```

```
VALIDATE test has failed, Designer rule validation has  
been disabled.
```

```
VALIDATION failed
```

```
*****Testing ADD_PRIMARY_KEY*****
```

```
Creating primary key using the following DDL:
```

```
ALTER TABLE 'DEMARC'.'hpdba_test_table' ADD CONSTRAINT  
'test_hpdba_pk' PRIMARY KEY ('col1')
```

```
ADD_PRIMARY_KEY passed
```

```
*****Testing CREATE_UNIQUE_INDEX*****
```

```
Creating unique index using the following DDL:
```

```
CREATE UNIQUE INDEX 'hpdba_unique_index' ON  
'DEMARC'.'hpdba_test_table' ('col1')
```

```
CREATE UNIQUE INDEX passed
```

```
.  
.   
.
```

- 3 If the JDBC test console encounters any DDL or SQL errors, you will receive a message indicating which test failed. You have the option to ignore the failure and continue with the test or to correct the configuration.

To support generic JDBC sources, HP Test Data Management relies on various templates to create, alter or drop database objects. Because the details of these objects remain unknown until runtime, various *substitution tokens* can be used in place of details such as table names, index names and column descriptions. So, you can build a DDL template and the in place of these details you can provide the tokens.

In the event of an error, you have the option of printing out all the available substitution tokens to aid in building your DDL template. When you view the failure information, you can easily identify the DDL template as well as the information that was replaced by a token, and the token itself. Substitution tokens are pre-defined and are different for each DDL. So, if you have a failure on a CREATE TABLE statement, you will likely not see substitution tokens for index creation.

Also, in the event of a syntax error, closely examine the error as well as the statements that caused the error. It is possible to receive a syntax error only to find out that the syntax is correct. In such cases, you can ignore the error. The following example shows how you can respond to syntax errors and also how to generate a list of the substitution tokens in use:

*****Testing DROP_UNIQUE_INDEX*****

Dropping unique index using the following DDL:

```
DROP INDEX 'DEMARC'. 'hpdba_unique_index'
```

Unable to drop index.

SQL used was:

```
DROP INDEX 'DEMARC'. 'hpdba_unique_index'
```

The error returned was: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'hpdba_unique_index' at line 1

SQL was based on the following template:

```
DROP INDEX ##QUALIFIED_INDEX_NAME##
```

Choose option:

- [1] Update DDL for DROP_INDEX
- [2] Leave DDL unchanged for DROP_INDEX
- [3] Print options

:1

Please provide updated DDL

```
:ALTER TABLE ##QUALIFIED_TABLE_NAME## DROP INDEX  
##QUALIFIED_INDEX_NAME##
```

You have entered

```
ALTER TABLE ##QUALIFIED_TABLE_NAME## DROP INDEX  
##QUALIFIED_INDEX_NAME##
```

Would you like to save Query?

- [1] Yes
- [2] No

:
.
.
.

Dropping index using the following DDL:

```
ALTER TABLE 'DEMARC'. 'hpdba_test_table' DROP INDEX  
'hpdba_test_table'. 'hpdba_non_unique_index'
```

DROP INDEX passed

*****Testing DROP_PRIMARY_KEY*****

Dropping primary key using the following DDL:

```
ALTER TABLE 'DEMARC'. 'hpdba_test_table' DROP CONSTRAINT  
'test_hpdba_pk'
```

Unable to drop primary key

SQL used was:

```
ALTER TABLE 'DEMARC'. 'hpdba_test_table' DROP CONSTRAINT  
'test_hpdba_pk'
```

The error returned was: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'CONSTRAINT 'test_hpdba_pk'' at line 1

SQL was based on the following template:

```
ALTER TABLE ##QUALIFIED_TABLE_NAME## DROP CONSTRAINT
##CONSTRAINT_NAME##
```

Choose option:

- [1] Update DDL for DROP_INDEX
- [2] Leave DDL unchanged for DROP_INDEX
- [3] Print options

:3

| | |
|--------------------------|---------------------------------------|
| ##CONSTRAINT_NAME## | Name of the primary key to be created |
| ##QUALIFIED_TABLE_NAME## | Name of the primary key to be created |
| ##TABLE_CATALOG## | Name of the primary key to be created |
| ##TABLE_SCHEMA## | Name of the primary key to be created |
| ##TABLE_NAME## | Name of the primary key to be created |

Note: To substitute values as JDBC bind variables(?) use !# in place of the ## tags. Using bind variables is generally safer when multi-byte characters are involved.

Choose option:

- [1] Update DDL for DROP_INDEX
- [2] Leave DDL unchanged for DROP_INDEX
- [3] Print options

:1

Please provide updated DDL

```
:ALTER TABLE ##QUALIFIED_TABLE_NAME## DROP PRIMARY KEY
```

.
.
.

Please see test summary in /home/db2inst1/
HPDBArchiving630/build_403/obt/bin/
MySQL_5_4_driver_test_results.txt

Would you like to review test summary?

- [1] Yes
- [2] No

:1

No test failures to report on.

The connection configuration test is complete, would you like to update the configuration for any passed tests?

- [1] Yes
- [2] No

:

Interpreting JDBC DDL/Metadata test results

After running the JDBC DDL/Metadata test you can view a summary of the results in the `obt/log/jdbctestresults.txt` file. The location and name of the log file may vary; however, it is printed at the end of each test for reference. The log file typically follows the format `<working_directory name>/<driver_name>`.

Possible failure codes are:

- **UPLOAD_SEVERE**

Uploads against this connection will likely fail. It is highly recommended that the issue is identified and corrected before attempting to upload using this connection.

- **ARCHIVE_SEVERE**

Extraction attempts using this connection will likely fail. It is highly recommended that the issue is identified and corrected before attempting to extract using this connection.

- **DESIGNER_SEVERE**

Indicates that metadata will not display correctly in Designer. This may make it difficult or impossible to create models in Designer. It is highly recommended that the issue is identified and corrected before attempting to view metadata in Designer.

Following are sample JDBC DDL/Metadata test results for a JDBC driver as they appear in the `obt/log/jdbctestresults.txt` file:

Test Summary

Test Name: DROP_INDEX

Severity: UPLOAD_SEVERE

Message: This error will likely cause any upload against this connection to fail.

It is highly recommended that the cause of this error be fixed before any upload against this connection be attempted.

Test Name: DROP_PRIMARY_KEY

Severity: UPLOAD_SEVERE

Message: This error will likely cause any upload against this connection to fail.

It is highly recommended that the cause of this error be fixed before any upload against this connection be attempted.

Test Name: DROP_UNIQUE_INDEX

Severity: UPLOAD_SEVERE

Message: This error will likely cause any upload against this connection to fail.

It is highly recommended that the cause of this error be fixed before any upload against this connection be attempted.

Running the JDBC table column data type test

To run the JDBC table column data type test:

- 1 Launch the test by selecting the JDBC table column data type test option:

```
Please select test to run:
[1] JDBC Metadata method test
[2] JDBC DDL/Metadata configuration
[3] JDBC table column data type test
[4] Exit
:3
```

- 2 Provide the appropriate responses to the JDBC test console based on your particular configuration. After all data types have been tested, you have the option to update any of the column data type settings. The following is an example—your responses will depend upon your specific configuration:

```
Please enter catalog name where table is located:
DEMARC
Please enter table name:
CUSTOMER
-----
Column name: CUSTOMERID
Data type INT was found with the following properties:
Has length: false
Has precision: false
Has scale: false
This data column will be cloned using: INT
-----
Column name: LASTNAME
Data type VARCHAR was found with the following
properties:
Has length: true
Has precision: false
Has scale: false
This data column will be cloned using: VARCHAR(80)
-----
Column name: FIRSTNAME
Data type VARCHAR was found with the following
properties:
Has length: true
Has precision: false
Has scale: false
This data column will be cloned using: VARCHAR(80)
.
.
.
```

NOTE In the following portion of the session transcript, notice how you can choose to modify column types and datatypes. This functionality enables you to potentially address a problem with the JDBC driver during the configuration.

Update any column type settings?

[1] Yes

[2] No

:1

[unifiedgenericconnectiontesttask] Select option

[unifiedgenericconnectiontesttask] [1] Edit Column types

[unifiedgenericconnectiontesttask] [2] Edit Data types

[unifiedgenericconnectiontesttask] [3] Exit

:1

[unifiedgenericconnectiontesttask] Select Column Type to modify

[unifiedgenericconnectiontesttask] [-1] Back

[unifiedgenericconnectiontesttask] [0] Add Column Type

[unifiedgenericconnectiontesttask] [1] INTEGER

[unifiedgenericconnectiontesttask] [2] SMALLINT UNSIGNED

[unifiedgenericconnectiontesttask] [3] BLOB

[unifiedgenericconnectiontesttask] [4] LONGTEXT

[unifiedgenericconnectiontesttask] [5] VARBINARY

.

.

.

:1

[unifiedgenericconnectiontesttask] Current data type is:

[unifiedgenericconnectiontesttask] Select option

[unifiedgenericconnectiontesttask] [-1] Previous Menu

[unifiedgenericconnectiontesttask] [1] Update dataType

:1

[unifiedgenericconnectiontesttask] Select data type to associate with INTEGER

[unifiedgenericconnectiontesttask] Select JDBC type

[unifiedgenericconnectiontesttask] [-1] Cancel

[unifiedgenericconnectiontesttask] [1] 1 (CHAR)

[unifiedgenericconnectiontesttask] [2] 2 (NUMERIC)

[unifiedgenericconnectiontesttask] [3] 3 (DECIMAL)

[unifiedgenericconnectiontesttask] [4] 4 (INTEGER)

.

.

.

:2

[unifiedgenericconnectiontesttask] Select JDBC type

[unifiedgenericconnectiontesttask] [-1] Cancel

[unifiedgenericconnectiontesttask] [1] 1 (CHAR)

[unifiedgenericconnectiontesttask] [2] 2 (NUMERIC)

[unifiedgenericconnectiontesttask] [3] 3 (DECIMAL)

[unifiedgenericconnectiontesttask] [4] 4 (INTEGER)

.

.

.

Running JDBC driver validation tests

After installing HP Test Data Management, it is strongly recommended that you test your generic JDBC drivers to ensure they are compatible with the application. The JDBC driver validation test tool detects whether or not methods of the `java.sql.DatabaseMetaData` class are supported by the driver.

After running the JDBC driver validation tests you can view a list of all the unsupported methods for specified drivers in the file `obt/log/jdbctestresults.txt`.

The location and name of the log file may vary; however, it is printed at the end of each test for reference. The log file typically follows the format `<working_directory_name>/<driver_name>`. Items preceded by an exclamation mark (!) will prevent a successful extraction.

To run JDBC driver validation tests:

- 1 Move all `driver.jar` files that you want to test into the `obt/lib/jdbc` directory.
- 2 Open the file `obt/config/drivertest_connection.properties` and for each driver that you will be testing, create the following properties:

```
<driver.name>.username=root
<driver.name>.userpassword=all4one
<driver.name>.url=jdbc:mysql://sardine.cup.hp.com:3306/
```

NOTE The `<driver.name>` variable is the name of the driver to which the connection properties belong. For example, if testing `mysql.jar`, then `<driver.name> = mysql`.

It is possible that the `<driver.name>` in the properties file is different from the database vendor name; use the name supplied by the vendor.

- 3 Navigate to the `<install_dir>\obt\bin` directory and run the driver validation test:

```
launch_drivertests.bat "<install_dir>\obt\config\
drivertest_connection.properties"
```

- 4 Carefully review the results of the driver validation test in the `obt/log/jdbctestresults.txt` file. The location and name of the result file may vary; however, it is printed at the end of each test for reference. Note that items preceded by an exclamation mark (!) will prevent a successful extraction.

Running the JDBC DDL/SQL configuration utility

When using generic JDBC drivers, it is recommended that you use the JDBC DDL/SQL configuration utility to determine whether or not there are any configuration issues that must be addressed before running an extraction.

To run the JDBC DDL/SQL configuration utility:

- 1 You can run the utility on any of the drivers in the `obt/lib/jdbc` directory. Before running the utility you can create a properties file that contains the connection information associated with each .jar file.

Following is an example of a typical connection properties file:

```
mysql.username=user
mysql.userpassword=password
mysql.url=jdbc:mysql://localhost:3306/my_database
```

- 2 To run the JDBC DDL/SQL configuration utility from Windows-based machines, navigate to `<install_dir>\obt\bin` and enter the following command:

```
launch_drivertests_stage_2.bat <driver_conn>.properties
```

To run the JDBC DDL/SQL configuration utility from Linux, Unix or Solaris machines, navigate to `<install_dir>/obt/bin` and enter the following command:

```
launch_drivertests_stage_2.sh <driver_conn>.properties
```

- 3 The JDBC DDL/SQL configuration utility runs the tests described in [Table 35](#). After the tests are complete you can view the results, which are located in the `obt/log/jdbctestresults.txt` log file.

The location and name of the log file may vary; however, it is printed at the end of each test for reference. The log file typically follows the format `<working_directory name>/<driver_name>`. Items preceded by an exclamation mark (!) will prevent a successful extraction.

JDBC DDL/SQL configuration utility tests

[Table 35](#) identifies the tests run by the JDBC DDL/SQL configuration utility.

Table 35 JDBC DDL/SQL configuration utility tests

| JDBC Test Name | Description |
|------------------|---|
| DDL Tests | |
| ADD_PRIMARY_KEY | Creates a primary key based on one of the columns in the table that was created by the CREATE_TABLE test. |
| CREATE_TABLE | Creates a basic two-column table with a VARCHAR column and an INTEGER column. |

Table 35 JDBC DDL/SQL configuration utility tests

| JDBC Test Name | Description |
|-----------------------------|--|
| CREATE_INDEX | Creates an index on one of the columns in the table that was created by the CREATE_TABLE test. |
| CREATE_UNIQUE_INDEX | Creates a unique index on one of the columns in the table that was created by the CREATE_TABLE test. |
| DROP_UNIQUE_INDEX | Drops the unique index created by the CREATE_UNIQUE_INDEX test. |
| DROP_INDEX | Drops the index created by the CREATE_INDEX test. |
| DROP_PRIMARY_KEY | Drops the primary key created by the ADD_PRIMARY_KEY test. |
| DROP_TABLE | Drops the table created by the CREATE_TABLE test. |
| Metadata Query Tests | |
| GET_CATALOGS | If catalogs are supported, this test ensures that they can be retrieved. |
| GET_SCHEMA | If schemas are supported, this test ensures that they can be retrieved. |
| GET_TABLES | Ensures that the table created by the CREATE_TABLE test can be retrieved. |
| GET_INDEX | Ensures that the indexes created by the CREATE_INDEX and CREATE_UNIQUE_INDEX tests can be retrieved. |
| GET_UNIQUE_INDEX | Ensures that the index created by the CREATE_UNIQUE_INDEX test can be retrieved. |
| GET_PRIMARY_KEY | Ensures that the primary key created by the ADD_PRIMARY_KEY test can be retrieved. |
| GET_COLUMNS | Ensures that the columns defined in the table created by the CREATE_TABLE test are correct. |
| SQL Query Tests | |

Table 35 JDBC DDL/SQL configuration utility tests

| JDBC Test Name | Description |
|-----------------------|---|
| INSERT | Ensures that an INSERT can be performed on the table created by the CREATE_TABLE test. |
| SELECT | Ensures that a SELECT can be performed on a table created by the CREATE_TABLE test. |
| VALIDATION | Ensures that a prepared JDBC statement validates the SQL. If this test fails, then Designer validation is disabled. |

A SQL API is available that you can use for creating custom masks. For Oracle, the functions are contained in the `obt_masking` package. The API consists of the following functions:

- [GENERATE_ABA_ROUTING_NUMBER](#) (page 176)
- [GENERATE_CREDIT_CARD_NUMBER](#) (page 177)
- [GENERATE_SSN](#) (page 177)
- [GET_CREDIT_CARD_TYPE](#) (page 178)
- [HP_LOOKUP_NUMBER](#) (page 176)
- [HP_LOOKUP_STRING](#) (page 176)
- [MASK_ABA_ROUTING_NUMBER](#) (page 178)
- [MASK_CREDIT_CARD_NUMBER](#) (page 179)
- [MASK_SSN](#) (page 180)
- [MASK_STRING](#) (page 181)
- [RANDOM_NUMBER](#) (page 182)
- [RANDOM_STRING](#) (page 182)
- [REVERT_HP_LOOKUP_NUMBER](#) (page 183)
- [REVERT_HP_LOOKUP_STRING](#) (page 183)
- [REVERT_SKEW_DATE](#) (page 184)
- [REVERT_SKEW_NUMBER](#) (page 184)
- [SKEW_DATE](#) (page 185)
- [SKEW_NUMBER](#) (page 185)
- [SKEW_PERCENT](#) (page 186)
- [VALID_ABA_ROUTING_NUMBER](#) (page 186)
- [VALID_CREDIT_CARD_NUMBER](#) (page 186)

See Also For additional background and details about custom masking, refer to [Chapter 6, Working with data masking](#).

HP_LOOKUP_NUMBER

Masks a number by mapping the number to another number. The function looks up the original number in a table and returns the mapped number. The mapping is based on the mapping that you specify in a mapping table. The table contains two columns 'orig' and 'masked'. Your mapping table must cover all values in a one-to-one fashion. For example, if you are mapping two numbers, the file maps two distinct numbers to two other distinct numbers.

Syntax `HP_LOOKUP_NUMBER(originalNumber, mappingTable)`

Parameters

| Parameter | Data Type | Description |
|----------------|--------------|---|
| originalNumber | NUMBER | Number to mask, using the mapping file. |
| mappingTable | VARCHAR(256) | Name of the mapping table to use. |

Returns Returns a masked value from a mapping table ("masked" column) that corresponds to the input value specified by `originalNumber`. The value returned is of type NUMBER.

HP_LOOKUP_STRING

Masks a string by mapping the string to another string. The method looks up the original string in a table and returns the mapped string. The mapping is based on the mapping that you specify in a mapping table. The table contains two columns 'orig' and 'masked'. Your mapping table must cover all values in a one-to-one fashion. For example, if you are mapping two strings, the file maps two distinct strings to two other distinct strings.

Syntax `HP_LOOKUP_STRING(originalString, mappingTable)`

Parameters

| Parameter | Data Type | Description |
|----------------|--------------|--|
| originalString | VARCHAR(254) | String to mask, using the mapping table. |
| mappingTable | VARCHAR(256) | Name of the mapping table to use. |

Returns Returns a masked value from a mapping table ("masked" column) that corresponds to the input value specified by `originalNumber`. The value returned is of type VARCHAR.

GENERATE_ABA_ROUTING_NUMBER

Generates a random ABA routing number.

Syntax `GENERATE_ABA_ROUTING_NUMBER ()`

Parameters `None`

Returns `Returns an ABA routing number as a VARCHAR(9).`

GENERATE_CREDIT_CARD_NUMBER

Generates a random credit card number.

Syntax `GENERATE_CREDIT_CARD_NUMBER (type, length)`

Parameters

| Parameter | Data Type | Description |
|-----------|-------------|---|
| type | VARCHAR(30) | The type of credit card number to generate. The possible values are: <ul style="list-style-type: none">• AMEX• DISC• MCRD• VISA• DINT• DCBL for American Express• Discover• Master card• Visa• Diners Club• Diners Club Carte Blanche |
| length | NUMBER | The length of the credit card number to return. The following are the valid values: <ul style="list-style-type: none">• 13 for VISA• 14 for DINT and DCBL• 15 for AMEX• 16 for DISC, MCRD and VISA |

Returns `Returns an a credit card number as a VARCHAR(16).`

GENERATE_SSN

Generates a new, random social security number starting with 897. Social security numbers starting with 897 are not assigned to any individual and can be safely used.

Syntax `GENERATE_SSN(include_dashes)`

Parameters

| Parameter | Data Type | Description |
|----------------|------------|--|
| include_dashes | VARCHAR(1) | Specifies whether to include dashes in the SSN. Specify 'Y' to get a number that includes dashes. For example, 897-23-2920. Specify 'N' to get a number with out dashes. For example, 897239320. |

Returns Returns a random SSN, where the first three digits are 897. The value returned is of type VARCHAR(11).

GET_CREDIT_CARD_TYPE

Based on the number of digits and prefix of a credit card number, this function determines if a credit card is American Express, Visa, and so on.

Syntax `GET_CREDIT_CARD_NUMBER(number)`

Parameters

| Parameter | Data Type | Description |
|-----------|-------------|-------------------------------------|
| number | VARCHAR(16) | The credit card number to validate. |

Returns Returns AMEX, DISC, MCRD, VISA, DINT or DCBL, depending on the credit card number input to the function. If no match is found, UKNW is returned. The value returned is of type VARCHAR(4).

MASK_ABA_ROUTING_NUMBER

Validates an ABA routing number. If the number is valid, returns a random routing number.

Syntax `MASK_ABA_ROUTING_NUMBER(abaNumber)`

Parameters

| Parameter | Data Type | Description |
|-----------|------------|---|
| abaNumber | VARCHAR(9) | The ABA routing number to validate then mask. |

Returns If ABA-routing number is valid, `MASK_ABA_ROUTING_NUMBER` returns a randomly generated routing number as type VARCHAR(9). Otherwise, the number input is returned.

MASK_CREDIT_CARD_NUMBER

Validates a credit card number. If the number is valid, returns a randomly generated credit card number of the same type or masks the original number.

Syntax MASK_CREDIT_CARD_NUMBER(creditCardNumber, maskType)

Parameters

| Parameter | Data Type | Description |
|------------------|-------------|---|
| creditCardNumber | VARCHAR(30) | The credit card number to validate then mask. For the supported credit card types and formats, see the following table. |
| maskType | VARCHAR(3) | Specifies the type of mask to return. The maskType parameter can be one of the following: <ul style="list-style-type: none">NEW—generate a new random credit card number.XXX—replace all digits with an X except for the last four digits of the credit card number. |

The following table lists the supported credit cards and formats supported by MASK_CREDIT_CARD_NUMBER.

| Card type | Prefix | Length | Format |
|------------------|--------|--------|---------------------|
| American Express | 34,36 | 15 | 123412345612345 |
| | | 17 | 1234-123456-12345 |
| | | 17 | 1234 123456 12345 |
| Discover | 6011 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |
| Mastercard | 51-55 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |
| Visa | 4 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |
| | | 13 | 1234123412345 |

| Card type | Prefix | Length | Format |
|---------------------------|---------|--------|------------------|
| Diners Card International | 36 | 14 | 12341234561234 |
| | | 16 | 1234-123456-1234 |
| | | 16 | 1234 123456 1234 |
| Diners Card Carte Blanche | 300-305 | 14 | 123412345631234 |
| | | 16 | 1234-123456-1234 |
| | | 16 | 1234 123456 1234 |

Returns If the credit card number is valid, `MASK_CREDIT_CARD_NUMBER` returns one of the following, depending on the mask type specified:

- A random credit card number of the same type.
- The masked credit card number where all digits of the input number are replaced by the letter X, except for the last four numbers.

If the credit card number is invalid, `MASK_CREDIT_CARD_NUMBER` returns the number that was input.

The value returned is of type `VARCHAR(16)`.

MASK_SSN

Validates a social security number. If the number is valid, returns a randomly generated social security number or masks the original number, depending of the type of masking specified.

Syntax `MASK_SSN(ssn, maskType)`

Parameters

| Parameter | Data Type | Description |
|-----------|-------------|---|
| ssn | VARCHAR(11) | The social security number to mask. Supports the formats XXXXXXXXXX and XXX-XX-XXXX. |
| maskType | VARCHAR(3) | Specifies the type of mask to return. The <code>maskType</code> parameter can be one of the following: <ul style="list-style-type: none"> • 'NEW'—generate a new random social security number. • 'XXX'—replace all digits with an X except for the last four digits of the social security number. |

Returns `MASK_SSN` returns one of the following, depending on the mask type specified:

- A random social security number of the same type.
- The masked social security number where all digits of the input number are replaced by the letter X, except for the last four numbers.

If the social security number is invalid, `MASK_SSN` returns the number that was input.

The value returned is of type `VARCHAR(11)`.

MASK_STRING

Returns a masked string based on a specified type and length. The string and string length returned are determined by the substitute character, whether the length of the string should be random or fixed, and the maximum length that should be returned.

Syntax `MaskString(originalString, substituteCharacter, lengthType, maxLength)`

Parameters

| Parameter | Data Type | Description |
|---------------------|---------------|---|
| originalString | VARCHAR(1024) | String to be masked. |
| substituteCharacter | CHAR | Character to use for masking the string. |
| lengthType | CHAR | Length of the string returned. The following are the possible values for <code>lengthType</code> : <ul style="list-style-type: none"> • R specifies that the masked string returned has a random length no longer than <code>maxLength</code>. • O specifies that the masked string is always the same length as the original string. For example, if the original string is 5 characters long, the masked string returned is 5 characters long. • S specifies that only the substitute character is returned as the string mask, and the length of the string is 1. |
| maxLength | NUMBER | Maximum length of the masked string that can be returned when <code>lengthType</code> is 'R'. |

Returns Returns a string created by replacing all characters in `originalString` by the substitute character. Length of the returned string is controlled by `lengthType` and `maxLength`.

RANDOM_NUMBER

Generates a random number between a specified range, inclusive of the upper and lower values for the range.

Syntax `RANDOM_NUMBER(lower, upper)`

Parameters

| Parameter | Data Type | Description |
|-----------|-----------|--|
| lower | NUMBER | The minimum number that the method can return. |
| upper | NUMBER | The maximum number that the method can return. |

Returns Returns a randomly generated whole number in the range specified by the lower to upper values. The value returned is of type NUMBER.

RANDOM_STRING

Generates a random string of the specified length and type.

Syntax `RANDOM_STRING(characterType, length)`

Parameters

| Parameters | Data Type | Description |
|---------------|-------------|---|
| characterType | VARCHAR(20) | The type of string to generate. The types are: <ul style="list-style-type: none">• ANY_ALPHA_CHAR—The string returned contains letters.• ANY_NUMERIC_CHAR—The string returned contains alphanumeric characters.• LOWER_CASE_ALPHA—The string returned contains only lower-case letters.• UPPER_CASE_ALPHA—The string returned contains only uppercase letters. |
| length | NUMBER | The length of string to generate. |

Returns Returns a randomly generated string, containing characters as specified by `characterType` with the length specified by `length`. The value returned is of type `VARCHAR(1024)`.

REVERT_HP_LOOKUP_NUMBER

This function is used to recover a value masked by [HP_LOOKUP_NUMBER](#). It unmask the number by mapping the number to another number. The function looks up the original number in a table and returns the mapped number. The mapping is based on mapping that you specify in a mapping table. The table contains two columns 'orig' and 'masked'. Your mapping table must cover all values in a one-to-one fashion. For example, if you are mapping two numbers, the table maps two distinct numbers to two other distinct numbers.

Syntax `REVERT_HP_LOOKUP_NUMBER(originalNumber, mappingFile)`

Parameters

| Parameter | Data Type | Description |
|----------------|--------------|--|
| originalNumber | NUMBER | Number to recover, using the mapping file. |
| mappingTable | VARCHAR(256) | Name of the mapping table to use. |

Returns Returns a masked value from a mapping table ('masked' column) that corresponds to the input value specified by `originalNumber`. The value returned is of type `NUMBER`.

REVERT_HP_LOOKUP_STRING

This method is used to recover a value masked by [HP_LOOKUP_STRING](#). It unmask the string by mapping the string to another string. The function looks up the original string in a table and returns the mapped string. The mapping is based on mapping that you specify in a mapping table. The table contains the mappings in two columns 'orig' and 'masked'. Your mapping table must cover all values in a one-to-one fashion. For example, if you are mapping two strings, the table maps two distinct strings to two other distinct strings.

Syntax `REVERT_LookupString(originalString, mappingTable)`

Parameters

| Parameter | Data Type | Description |
|----------------|--------------|--|
| originalString | VARCHAR(254) | String to recover, using the mapping file. |
| mappingTable | VARCHAR(256) | Name of mapping table to use. |

Returns Returns a masked value from the mapping table that corresponds to the input value specified by `originalString`. The returned value is of type `VARCHAR`.

REVERT_SKEW_DATE

Performs the opposite function of [SKEW_DATE](#), allowing you to revert the result. `REVERT_SKEW_DATE` subtracts the specified number of days from the date.

Syntax `REVERT_SKEW_DATE(originalDate, skewAmount)`

Parameters

| Parameter | Data Type | Description |
|--------------|-----------|---|
| originalDate | DATA | Original date to unskew. |
| skewAmount | NUMBER | The number of days by which to decrement the original date. |

Returns Returns a date as a DATE type, where the date is skewed by subtracting `skewAmount` days from the original date.

REVERT_SKEW_NUMBER

Performs the opposite function of [SKEW_NUMBER](#), allowing you to revert the result. `REVERT_SKEW_NUMBER` performs the opposite operation from what you specify for `action`. For example, if you specify 'Add', the method performs a subtraction operation.

Syntax `REVERT_SKEW_NUMBER(originalNumber, action, skewAmount)`

Parameters

| Parameter | Data Type | Description |
|----------------|-------------|---|
| originalNumber | NUMBER | Base value to be skewed |
| action | VARCHAR(10) | A string that specifies the method to use to skew the base value specified by target. The type is one of the following: <ul style="list-style-type: none">• Add—subtracts <code>skewAmount</code> from <code>originalNumber</code>• Subtract—adds <code>skewAmount</code> to <code>originalNumber</code>• Multiply—divides <code>originalNumber</code> by <code>skewAmount</code>.• Divide—multiplies <code>originalNumber</code> by <code>skewAmount</code> |
| skewAmount | NUMBER | The amount by which to skew the base value. |

Returns Returns a number skew by the specified amount, using the specified action, where the action performed is the opposite of the action specified. The value returned is of type NUMBER.

SKEW_DATE

Returns a skewed date based on an original date plus a specified number of days.

Syntax `SKEW_DATE(originalDate, skewAmount)`

Parameters

| Parameter | Data Type | Description |
|--------------|-----------|---|
| originalDate | Date | Original date to skew. |
| skewAmount | Number | The number of days by which to increment the original date. |

Returns Returns a date as a Date type, where the date is skewed by adding *skewAmount* days to the original date.

SKEW_NUMBER

Skews an original number skewed by adding, subtracting, multiplying, or dividing the number by a specified amount.

Syntax `SKEW_NUMBER(originalNumber, action, skewAmount)`

Parameters

| Parameter | Data Type | Description |
|----------------|-------------|--|
| originalNumber | NUMBER | Base value to be skewed |
| action | VARCHAR(10) | A string that specifies the action to use to skew <i>originalNumber</i> . The action is one of the following: <ul style="list-style-type: none">• Add—adds <i>skewAmount</i> to <i>originalNumber</i>.• Subtract—subtracts <i>skewAmount</i> from <i>originalNumber</i>.• Multiply—multiplies <i>originalNumber</i> by <i>skewAmount</i>.• Divide—divides <i>originalNumber</i> by <i>skewAmount</i>. |
| skewAmount | NUMBER | The amount by which to skew the original number value. |

Returns Returns the number input to the function by the skew amount, using the specified action. The value returned is of type NUMBER.

SKEW_PERCENT

Returns a skewed value where the original number is increased by a specified percentage.

Syntax `SKEW_PERCENT(originalNumber, skewPercent)`

Parameters

| Parameter | Data Type | Description |
|----------------|-----------|---|
| originalNumber | NUMBER | The value to skew. |
| skewPercent | NUMBER | The percentage of the original number to be added to skew the number. Specify whole number for this parameter instead of decimals. For example, to skew the number by 10%, specify 10 for this parameter, not 0.10. |

Returns Returns $\text{originalNumber} * (1 + \text{skewAmount})/100$ as a value of type NUMBER.

VALID_ABA_ROUTING_NUMBER

Validates an ABA routing number.

Syntax `VALID_ABA_ROUTING_NUMBER(abanumber)`

Parameters

| Parameter | Data Type | Description |
|-----------|------------|-------------------------------------|
| abaNumber | VARCHAR(9) | The ABA routing number to validate. |

Returns Returns VARCHAR(10) value that is the string VALID if the ABANumber is valid or INVALID if it is not valid. For DB2, return type is Boolean, returning true for a valid number and false for an invalid one.

VALID_CREDIT_CARD_NUMBER

Validates a credit card number by using the credit card checksum algorithm (Luhn).

Syntax `VALID_CREDIT_CARD_NUMBER(number, length)`

Parameters

| Parameter | Data Type | Description |
|-----------|-------------|-------------------------------------|
| number | VARCHAR(24) | The credit card number to validate. |

Returns Returns a VARCHAR(10) value that is the string VALID if the credit card number is valid or INVALID if it is not valid. For DB2, the return type is Boolean, returning true for a valid number and false for an invalid one.

A Java API is available that you can use for creating custom masks. The methods for this API are contained in the class `com.outerbay.foundation.components.datamasking.java.DataMasking` as static methods. This class contains the following methods:

- [LookupNumber](#) (page 189)
- [LookupString](#) (page 190)
- [MaskABANumber](#) (page 190)
- [MaskCreditCardNumber](#) (page 191)
- [MaskSSN](#) (page 192)
- [MaskString](#) (page 193)
- [RandomInt](#) (page 194)
- [RandomString](#) (page 195)
- [REVERT_LookupNumber](#) (page 195)
- [REVERT_LookupString](#) (page 196)
- [REVERT_SkewDate](#) (page 196)
- [REVERT_SkewInteger](#) (page 197)
- [SkewDate](#) (page 197)
- [SkewFloat](#) (page 198)
- [SkewInteger](#) (page 198)
- [SkewPercent](#) (page 199)

See Also For additional background and details about custom masking, refer to [Chapter 6, Working with data masking](#).

LookupNumber

Masks a number by mapping the number to another number. The method looks up the original number in a file and returns the mapped number. The mapping is based on the mapping that you specify in a mapping file. The file must contain the mappings in the format `original=masked`. Your mapping file must cover all values in a one-to-one fashion. For example, if you are mapping two numbers, the file maps two distinct numbers to two other distinct numbers.

Syntax `BigDecimal LookupNumber(BigDecimal originalNumber, String mappingFile)`

Parameters

| Parameter | Data Type | Description |
|----------------|------------|--|
| originalNumber | BigDecimal | Number to mask, using the mapping file. |
| mappingFile | String | Absolute path to the mapping file to be used to map the original number. |

Returns Returns a masked value from the mapping file that corresponds to the input value specified by `originalNumber`.

LookupString

Masks a string by mapping the string to another string. The method looks up the original string in a file and returns the mapped string. The mapping is based on the mapping that you specify in a mapping file. The file must contain the mappings in the format `original=masked`. Your mapping file must cover all values in a one-to-one fashion. For example, if you are mapping two strings, the file maps two distinct strings to two other distinct strings.

Syntax `String LookupNumber(String originalString, String mappingFile)`

Parameters

| Parameter | Data Type | Description |
|----------------|-----------|--|
| originalString | String | String to mask, using the mapping file. |
| mappingFile | String | Absolute path to the mapping file to be used to map the original string. |

Returns Returns a masked value from the mapping file that corresponds to the input value specified by `originalString`.

MaskABANumber

Validates an ABA routing number. If the number is valid, returns a random routing number.

Syntax `String MaskABANumber(String abaNumber)`

Parameters

| Parameter | Data Type | Description |
|-----------|-----------|---|
| abaNumber | String | The ABA routing number to validate then mask. |

Returns If ABA-routing number is valid, `MaskABANumber` returns a randomly generated routing number. Otherwise, the same input number is returned.

MaskCreditCardNumber

Validates a credit card number. If the number is valid, returns a randomly generated credit card number of the same type or masks the original number.

Syntax `String MaskCreditCardNumber(String creditCardNumber, String maskType)`

Parameters

| Parameter | Data Type | Description |
|------------------|-----------|--|
| creditCardNumber | String | The credit card number to validate then mask. For the supported credit card types and formats, see the following table. |
| maskType | String | Specifies the type of mask to return. The <code>maskType</code> parameter can be one of the following: <ul style="list-style-type: none">NEW—generate a new random credit card number.XXX—replace all digits with an X except for the last four digits of the credit card number. |

The following table lists the supported credit cards and formats supported by the `MaskCreditCardNumber`.

| Card type | Prefix | Length | Format |
|------------------|--------|--------|---------------------|
| American Express | 34,36 | 15 | 123412345612345 |
| | | 17 | 1234-123456-12345 |
| | | 17 | 1234 123456 12345 |
| Discover | 6011 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |

| Card type | Prefix | Length | Format |
|------------------------------|---------|--------|---------------------|
| Mastercard | 51-55 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |
| Visa | 4 | 16 | 1234123412341234 |
| | | 19 | 1234-1234-1234-1234 |
| | | 19 | 1234 1234 1234 1234 |
| | | 13 | 1234123412345 |
| Diners Card International | 36 | 14 | 12341234561234 |
| | | 16 | 1234-123456-1234 |
| | | 16 | 1234 123456 1234 |
| Diners Card Carte Blanche | 300-305 | 14 | 123412345631234 |
| | | 16 | 1234-123456-1234 |
| | | 16 | 1234 123456 1234 |

Returns If the credit card number is valid, `MaskCreditCardNumber` returns one of the following, depending on the mask type specified:

- A random credit card number of the same type.
- The masked credit card number where all digits of the input number are replaced by the letter X, except for the last four numbers.

If the credit card number is invalid, `MaskCreditCardNumber` returns the number that was input.

MaskSSN

Validates a social security number. If the number is valid, returns a randomly generated social security number or masks the original number, depending of the type of masking specified.

Syntax `String MaskSSN(String ssn, String maskType)`

Parameters

| Parameter | Data Type | Description |
|-----------|-----------|--|
| ssn | String | The social security number to mask. Supports the formats XXXXXXXXXX and XXX-XX-XXXX. |
| maskType | String | Specifies the type of mask to return. The <code>maskType</code> parameter can be one of the following: <ul style="list-style-type: none">• NEW—generate a new random social security number.• XXX—replace all digits with an X except for the last four digits of the social security number. |

Returns If the social security number is valid, `MaskSSN` returns one of the following, depending on the mask type specified:

- A random social security number of the same type.
- The masked social security number where all digits of the input number are replaced by the letter X, except for the last four numbers.

If the social security number is invalid, `MaskSSN` returns the number that was input.

MaskString

Returns a masked string based on a specified type and length. The string and string length returned are determined by the substitute character, whether the length of the string should be random or fixed, and the maximum length that should be returned.

Syntax `String MaskString(String originalString, Character substituteCharacter, Character lengthType, Integer maxLength)`

Parameters

| Parameter | Data Type | Description |
|---------------------|-----------|---|
| originalString | String | String to be masked. |
| substituteCharacter | Character | Character to use for masking the string. |
| lengthType | Character | Length of the string returned. The following are the possible values for <code>lengthType</code> : <ul style="list-style-type: none">• R specifies that the masked string returned has a random length no longer than <code>maxLength</code>.• O specifies that the masked string is always the same length as the original string. For example, if the original string is 5 characters long, the masked string returned is 5 characters long.• S specifies that only the substitute character is returned as the string mask, and the length of the string is 1. |
| maxLength | Integer | Maximum length of the masked string that can be returned when <code>lengthType</code> is 'R'. |

Returns Returns a String created by replacing all characters in `originalString` by the substitute character. Length of the returned String is controlled by `lengthType` and `maxLength`.

RandomInt

Generates a random number between a specified range, inclusive of the upper and lower values for the range.

Syntax `Integer RandomInteger(Integer lower, Integer upper)`

Parameters

| Parameter | Data Type | Description |
|-----------|-----------|--|
| lower | Integer | The minimum number that the method can return. |
| upper | Integer | The maximum number that the method can return. |

Returns Returns a randomly generated whole number in the range specified by the lower to upper values.

RandomString

Generates a random string of the specified length and type.

Syntax `String RandomString(String characterType, Integer length)`

Parameters

| Parameters | Data Type | Description |
|---------------|-----------|---|
| characterType | String | The type of string to generate. The types are: <ul style="list-style-type: none">• ANY_ALPHA_CHAR—The string returned contains letters.• ANY_NUMERIC_CHAR—The string returned contains alphanumeric characters.• LOWER_CASE_ALPHA—The string returned contains only lower-case letters.• UPPER_CASE_ALPHA—The string returned contains only uppercase letters. |
| length | Integer | The length of string to generate. |

Returns Returns a randomly generated String, containing characters as specified by characterType with the length specified by length.

REVERT_LookupNumber

This method is used to recover a value masked by [LookupNumber](#). It unmask the number by mapping the number to another number. The method looks up the original number in a file and returns the mapped number. The mapping is based on mapping that you specify in a mapping file. The file must contain the mappings in the format original=masked. Your mapping file must cover all values in a one-to-one fashion. For example, if you are mapping two numbers, the file maps two distinct numbers to two other distinct numbers.

Syntax `BigDecimal LookupNumber(BigDecimal originalNumber, String mappingFile)`

Parameters

| Parameter | Data Type | Description |
|----------------|------------|--|
| originalNumber | BigDecimal | Number to recover, using the mapping file. |
| mappingFile | String | Absolute path to the mapping file to be used to recover the original number. |

Returns Returns a value from the mapping file that corresponds to the input value specified by originalNumber.

REVERT_LookupString

This method is used to recover a value masked by [LookupString](#). It unmask the string by mapping the string to another string. The method looks up the original string in a file and returns the mapped string. The mapping is based on mapping that you specify in a mapping file. The file must contain the mappings in the format original=masked. Your mapping file must cover all values in a one-to-one fashion. For example, if you are mapping two strings, the file maps two distinct strings to two other distinct strings.

Syntax `String REVERT_LookupString(String originalString, String mappingFile)`

Parameters

| Parameter | Data Type | Description |
|----------------|-----------|--|
| originalString | String | String to recover, using the mapping file. |
| mappingFile | String | Absolute path to the mapping file to be used to recover the original number. |

Returns Returns a value from the mapping file that corresponds to the input value specified by originalString.

REVERT_SkewDate

Performs the opposite function of [SkewDate](#), allowing you to revert the result. REVERT_SkewDate subtracts the specified number of days from the date.

Syntax `Date REVERT_SkewDate(Date originalDate, Integer skewAmount)`

Parameters

| Parameter | Data Type | Description |
|--------------|-----------|---|
| originalDate | Date | Original date to unskew. |
| skewAmount | Integer | The number of days by which to decrement the original date. |

Returns Returns a Date object that contains the original date skewed by subtracting skewAmount days from the date.

REVERT_SkewInteger

Performs the opposite function of [SkewInteger](#), allowing you to revert the result. `REVERT_SkewInteger` performs the opposite operation from what you specify for type. For example, if you specify Add, the method performs a subtraction operation.

Syntax `BigDecimal RevertSkewInteger(BigDecimal originalNumber, String action, Integer skewAmount)`

Parameters

| Parameter | Data Type | Description |
|----------------|------------|---|
| originalNumber | BigDecimal | Base value to be skewed |
| action | String | A string that specifies the method to use to skew the base value specified by target. The type is one of the following: <ul style="list-style-type: none">• Add—subtracts skewAmount from originalNumber• Subtract—adds skewAmount to originalNumber• Multiply—divides originalNumber by skewAmount.• Divide—multiplies originalNumber by skewAmount |
| skewAmount | BigDecimal | The amount by which to skew the base value. |

Returns Returns a BigDecimal value that is skewed from the number input to the method by the skew amount, using the specified action, where the action performed is the opposite of the action specified.

SkewDate

Returns a skewed date based on an original date plus a specified number of days.

Syntax `Date SkewDate(Date originalDate, Integer skewAmount)`

Parameters

| Parameter | Data Type | Description |
|--------------|-----------|---|
| originalDate | Date | Original date to skew. |
| skewAmount | Integer | The number of days by which to increment the original date. |

Returns Returns a Date object that contains the original date skewed by adding skewAmount days to the date.

SkewFloat

Skews a float value. The skewed value is created from an original float value plus a user-specified number.

Syntax `BigDecimal SkewFloat(BigDecimal originalNumber, String action, BigDecimal skewAmount)`

Parameters

| Parameter | Data Type | Description |
|----------------|------------|--|
| originalNumber | BigDecimal | Value to be skewed |
| action | String | A string that specifies the method to use to skew originalNumber. The action is one of the following: <ul style="list-style-type: none">• Subtract—subtracts skew_amount from originalNumber• Add—adds skewAmount to originalNumber• Multiply—multiplies originalNumber by skewAmount.• Divide—divides originalNumber by skewAmount |
| skewAmount | BigDecimal | The amount by which to skew the original number. |

Returns Returns a BigDecimal value that is skewed from the number input to the method by the skew amount, using the specified action.

SkewInteger

Skews an integer value. The skewed value is created from an original number skewed by adding, subtracting, multiplying, or dividing.

Syntax `BigDecimal SkewInteger(BigDecimal originalNumber, String action, Integer skewAmount)`

Parameters

| Parameter | Data Type | Description |
|----------------|------------|--|
| originalNumber | BigDecimal | Base value to be skewed |
| action | String | A string that specifies the action to use to skew originalNumber. The action is one of the following: <ul style="list-style-type: none">• Add—adds skewAmount to originalNumber.• Subtract—subtracts skewAmount from originalNumber.• Multiply—multiplies originalNumber by skewAmount.• Divide—divides originalNumber by skewAmount. |
| skewAmount | Integer | The amount by which to skew the original number value. |

Returns Returns a BigDecimal value that is skewed from the number input to the method by the skew amount, using the specified action.

SkewPercent

Returns a skewed value where the original number is increased by a specified percentage.

Syntax `Float SkewPercent(Float originalNumber, Float skewPercent)`

Parameters

| Parameter | Data Type | Description |
|----------------|-----------|---|
| originalNumber | Float | The value to skew. |
| skewPercent | Float | The percentage of the original number to be added to skew the number. Specify whole number for this parameter instead of decimals. For example, to skew the number by 10%, specify 10 for this parameter, not 0.10. |

Returns Returns $\text{originalNumber} * (1 + \text{skewAmount})/100$ as a Float value.

Glossary

| | |
|---------------------------------|--|
| active database | The database from which you plan to extract data. Typically, this database is your online transaction processing (OLTP) or production database. In a two-tiered configuration, the active database resides on tier one and is the source for data movement operations. |
| active environment | The Web Console views and acts upon only one environment at a time, the active environment. To switch the active environment, you use the Change Active option in the Web Console. |
| activity | In Designer, a component of a business flow, which is added by using the toolbar. Note, activities in a business flow are different from what you see at runtime and therefore do not necessarily map directly to what you see in Console. |
| advanced selection | A method of data selection that discovers all of the interrelated rows from multiple tables and conceptually places them in the same application partition for extraction. |
| annotation | In Designer, a comment associated with the project, or one of its objects or components. These comments are collected and published in a PDF file when you right click a project or business flow and select Generate Documentation. |
| application partitioning | The concept of partitioning related rows together during data selection, regardless of whether they are in one or more tables. Application partitioning is unique to HP Test Data Management and contrasts with the more common table partitioning offered by the database management software, which only groups related rows from one table. |
| business flow | A series of activities, such as extraction operations and scripts, that run in sequence. You build business flows in Designer. |
| business flow status | The Web Console shows the last run of each business flow. The states are Complete/Error/Running. |
| cartridge | An instance of model- or schema-based eligibility criteria used to copy data from one location to another. Cartridges capture the application and business rules to ensure referential integrity of the data. For any one model in your project, you may have many cartridges that use it. |
| chaining table | The lower level table in a many-to-one or a many-to-many relationship between higher level and lower level tables in the model hierarchy. |
| collection | The configuration of a directory location and file pattern to match a set of extracted XML files, thus allowing SQL access to the extracted data. |

| | |
|-------------------------------------|---|
| comma separated values (CSV) | A database to file output format that stores the data as values separated by commas and a metadata file. Each line in the CSV file corresponds to a row in a table. Within a line, fields are separated by commas, each field belonging to one table column. CSV files provide a simple format that many applications can import. |
| command | Command files or JavaScript files launched by the Web Console on your behalf with status displays. |
| condition | In Designer, the way you branch your business flow to run or skip an activity based on some criteria. |
| configuration parameter | A type of parameter that has its values set by an administrator (someone who has repository privileges from Console) through the administrator interface. Typically, this type of parameter represents values that should be changed very infrequently, perhaps only at deployment time. |
| console user | The Web Console identifies individual users, who are distinct from database users. The properties for a Console user are User Name, Full Name, Password, Enabled, Description, Email, Phone, and Privileges. |
| console user name | The login name associated with a Console user. |
| constraint | A column or a list of columns that enables you to identify rows in the database and relate them to one another. |
| customization | A change that an administrator or DBA makes to a project provided by a third party, typically for a packaged application like Oracle PeopleSoft or Oracle E-Business Suite. As long as the customization is allowable by the project, the user can merge the customization into newer revisions of the third party project. |
| customization mode | A Designer mode that provides visual cues to indicate customizations in the model. In a project with locked files, customization mode is on by default, but you can toggle it on and off from the toolbar in the model editor. |
| data masking | The process of replacing private or confidential data during movement with a specified mask. You can choose from pre-defined masks that are part of HP Test Data Management or create your own mask. |
| data movement | The method used by HP Test Data Management to actually copy data. |
| database constraint | A constraint that exists in the database and can be discovered and referenced from Designer. |
| database to file | A movement in which data goes from an active database to a file (XML or CSV format). |
| Deployment Assistant | The user interface component used to deploy or generate business flows. You invoke Deployment Assistant from within Designer. |

| | |
|-------------------------------|--|
| description | A technical description created by the developer for her own reference. These descriptions do not appear in the generated PDF file for the cartridge or business flow. |
| Designer | The user interface component used to develop, test, and deploy your extraction solution. Designer is a powerful graphical development environment for extraction solutions. |
| driving table | A driving object is a root of a model hierarchy. Its relationship to the child tables drives the selection of transactions. |
| dynamic list of values | A list of values for a parameter that obtains its members from a SELECT statement that returns identifiers and labels. |
| dynamic parameter | A type of parameter that has its value set by a Groovy script that runs at deployment time to obtain a value. For example, this type of parameter can supply the type or version of a database or application, which can be obtained programmatically at deployment time. |
| embedded repository | A Java database, installed with HP Test Data Management, that can act as your repository database, where you store your HP Test Data Management metadata. Alternatively, your source database or another database can act as the repository database. |
| environment | The source and (optional) target credentials against which you plan to run commands. You can define multiple environments within your installation to identify various source databases. |
| error | One of the ways in which you can interrupt a business flow. Error indicates that the business flow failed for some reason. |
| exclusive rules | One of the ways in which HP Test Data Management determines whether to include or exclude rows from the extract operation. Exclusive rules require all rows in the constraint table to match for inclusion. Exclusive rules exclude the instance if the condition on any child is false, like STATUS='CLOSED'. |
| exit | One of the ways in which you can interrupt a business flow. You can exit successfully or with a warning. |
| export | The way that you save an HP Test Data Management project to an exchange format (.hdp) from the File menu. See also <i>import</i> . |
| export data | The way that a user can send data to CSV format from Preview using the toolbar item. |
| extract data store | The location where the data is to be copied. Can be an XML or CVS file. |
| generate documentation | The process of collecting and grouping all annotations into a PDF file that also describes the business flow or cartridge structure. |

| | |
|------------------------------|--|
| import | The way that you transfer projects from exchange format (.hdp) into the Project Navigator. |
| inclusive rules | One of the ways in which HP Test Data Management determines whether to include or exclude rows from the extract operation. Inclusive rules require only one row in the constraint table to match the rule and be included. Inclusive rules include the instance if the condition on any child is true, like <code>PRODUCT_RECALLED='Y'</code> . |
| interrupt | The way to stop or pause a business flow (pause, error, exit with warning, exit successfully). |
| local cache | A capture of the metadata for your databases, schemas, and tables used when working offline in Designer. |
| local deployment | The generation and deployment of your cartridge or business flow to an environment on your local, Designer client. Deployment files are generated locally and then deployed to the designated, local environment. |
| lookup table | A table that contains helpful non-transactional information. For example, non-transactional information could be status definitions, or the name of the sales representative. |
| model | A model identifies the tables and table relationships representing a business entity or related business entities. A project can have multiple models. Each model contains a driving table and all of its child and descendent tables. |
| model compatibility | Each model in your project can have one or more dynamic parameters associated with it to verify the compatibility with the target environment. If the compatibility parameter returns false, then the cartridge referencing the model will not deploy or run and throw an error. For example, the script could return false for Oracle 10.2 and true for Oracle 11.1 to indicate that a cartridge referencing the model can only deploy and run against Oracle 11.1. |
| model-based cartridge | A cartridge that moves data based upon a defined data model with relationships. This type of cartridge is typically used for ongoing extract operations. |
| OLTP database | The online transaction processing database that typically is your active or source database. |
| pause | One of the ways in which you can interrupt a business flow. Pausing suspends the business flow while awaiting operator intervention. |
| query server | The component that provides SQL access to XML or CSV files. |
| remote deployment | The generation and deployment of your cartridge or business flow to an environment on a system that is remote from your Designer client. Deployment files are generated locally and then deployed to the designated, remote environment. |

| | |
|------------------------------------|--|
| repository | The location that holds business flow metadata, product configuration data, and data collected during runtime. The repository can be located on your active database, another logical database, or can be embedded database. |
| rule | Qualifications added to the model in order to include or exclude data based on certain criteria. For example, you might add a rule to exclude from extracting any orders that are not yet closed. |
| runtime parameter | A type of parameter that has its values set by the operator executing the job in Console or on the command line. Typically, this type of parameter represents operational values that tend to change frequently and therefore need to be set each time the job is run. |
| schema-based cartridge | A cartridge that moves data based upon the database schema rather than a defined data model with relationships. This type of cartridge is typically used for database retirement or the cleanup of orphan tables. |
| selection | The form of data selection to use (standard or advanced) for choosing data. When creating a cartridge or adding it to a business flow, you must specify the selection method. |
| source | The location (database) from which you are copying or moving data. |
| standard selection | A method of data selection that restricts itself to the rows identified by the model. Unlike advanced selection, it does not attempt to traverse related rows across multiple tables. |
| table use | A database table, view, or synonym that is referenced in Designer, for example, in the model. The same table can be used multiple times in a model. For example, a table could be appear as a transactional table and a lookup table in the same model. |
| target | The location (XML) to which you are copying data. |
| transactional data movement | Transactional movement uses set-based data movement and is the default method of movement. |
| transactional table | A table that contains information about the business transaction. For example, a transactional table might contain detailed tax or payment information related to each business transaction. |
| unique identifiers (UIDs) | A 16 hexadecimal identifier calculated based on the content of a Designer file. This value is used to determine if the user has customized key pieces of a project. |
| virtual constraint | A constraint that you define in Designer that only exists within HP Test Data Management as opposed to a database constraint, which exists within the database. |
| Web Console | A browser-based interface where you can create and manage your deployment environments, and deploy, run, administer, and monitor your business flows. |

Index

A

- ABA
 - masking, 87
- activities
 - splitting, 136
- aliases
 - creating, 52
- annotations
 - about, 122
 - adding, 122
 - cartridges, 122
 - generating documentation, 123, 124
- associated tables
 - managing, 43
- audience
 - intended, 7

B

- before you begin
 - concepts, 14
- business flows
 - about, 131
 - adding conditions, 141
 - adding database to file, 135
 - adding Groovy scripts, 138
 - adding interrupts, 142
 - adding schema-based cartridges, 136
 - creating, 132
 - deploying, 148
 - splitting activities, 136

C

- cartridges
 - creating, 73
 - database to file, 74
 - database to file, schema-based, 80
 - deploying, 145
 - masking, 85

- chaining tables
 - adding, 37
- compatibility
 - models, 67
- concepts
 - before you begin, 14
- conditional relationships
 - for chaining tables, 42
 - for lookup tables, 36
 - for transactional tables, 29
 - for virtual constraints, 51
- conditions
 - adding to business flows, 141
- connections
 - creating, 119
- conventions
 - document, 9
- credit cards
 - masking, 87, 94
- customizing
 - about, 155
 - guidelines, 156
 - merging changes, 158
 - projects, 157
- custom masks using Groovy scripts, 112

D

- database to file
 - adding to business flows, 135
 - deploying, 145
 - editing cartridges, 74
 - editing schema-based cartridges, 80
- data masking
 - applying, 85
 - tutorial, 101
- data movement
 - key for lookup tables, 32

- deploying
 - about, 145
 - business flows, 148
 - cartridges, 145
 - database to file, 145
 - generated files, 152
- Designer
 - overview, 14
- documentation
 - conventions, 9
 - generating, 123, 124
 - HP web site, 8
 - related, 8
 - updates, 10
- E**
- eligibility analytics
 - rules, 46
- examples
 - dynamic lists of values, 63
 - Groovy for dynamic parameter, 58
 - referencing parameters, 63
 - validation script, 60
- exporting
 - projects, 124
- F**
- foreign keys
 - adding transactional tables, 24
 - creating for chaining tables, 39
 - creating for lookup tables, 33
 - creating for transactional tables, 26
- G**
- generating
 - about, 145
 - business flows, 150
 - cartridges, 150
 - then deploying, 152
- GRANT SELECT
 - for associated tables, 43
- Groovy
 - adding scripts to business flows, 138
- Groovy events, 139

- H**
- HP
 - Subscriber's choice web site, 10
- I**
- importing projects, 125
- interrupts
 - adding to business flows, 142
- J**
- JDBC certification testing, 159
- JDBC connection configuration utility, 171
- JDBC DDL/Metadata configuration test, 160
- JDBC metadata method test, 159
- JDBC table column data type test, 160
- JDBC test console, 159
- L**
- licensing, HP
 - end user license agreement, 2
- lists of values
 - dynamic, 61
 - examples of dynamic, 63
 - parameters, 61
 - static, 61
- lookup tables
 - adding, 30
- M**
- mapping
 - schema names, 126
- masking
 - ABA, 87
 - applying, 85
 - cartridges, 85
 - considerations, 86
 - credit cards, 87
 - custom, 110
 - dates, 94
 - numbers, 92
 - pre-built, 86
 - reload, 99
 - social security numbers, 88
 - strings, 87
 - undo, 99

- models
 - about, 20
 - compatibility, 67
 - creating, 20
 - deleting, 22
 - working with, 19

O

- ODBC Bridge, 161

P

- parameters
 - about, 55
 - about runtime, 55
 - configuration, 55
 - creating, 56
 - deleting, 67
 - dynamic, 55
 - lists of values, 61
 - referencing, 63
 - validating, 60
- prerequisites
 - product, 7
- previewing
 - about, 69
 - cartridges, 70
 - data, 69
 - models, 70
 - rules for, 69, 70
- projects
 - about, 117
 - changing location, 126
 - concept of, 16
 - creating, 118
 - customizing, 157
 - exporting, 124
 - importing, 125

R

- referential integrity within data masking, 95

- rules
 - exclusive, 43
 - inclusive, 43

S

- schema
 - mapping names, 126

- schema-based
 - adding to business flows, 136
- scripts
 - adding to business flows, 138
- social security numbers
 - masking, 88

- software
 - version, 1
- Subscriber's choice, HP, 10
- subscription service
 - Subscriber's choice, 10
- substitution tokens, 164
- support
 - web site, 10
- synonyms
 - restriction, 21

T

- tables
 - adding to model, 53
 - adding transactional, 22
 - associated, 43
- test data management
 - overview, 13
- toolbar
 - add chaining table, 16
 - add lookup tables, 16
 - add rule, 16
 - add transactional table, 16
 - create business flow, 16
 - create cartridge, 16
 - create model, 16
 - create project, 16
 - custom view, 16
 - preview, 16
 - refresh, 16
- tutorials
 - data masking, 101

U

- unique keys
 - creating for chaining tables, 39
 - creating for lookup tables, 33
 - creating for transactional tables, 26
 - creating virtual constraints, 48
- Upload tool, 131

V

- views
 - restriction, 21
- virtual constraints
 - working with, 47
- virtual keys
 - deleting, 52
 - editing, 51

W

- web sites
 - HP documentation, 8
 - HP Subscriber's choice, 10
 - support, 10
- WHERE clauses
 - associated tables, 43

