HP SOA Systinet Workbench

Software Version: 4.00

Report Editor Guide

Document Release Date: September 2010
Software Release Date: September 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.

Copyright Notices

Contents

About this Guide

Welcome to the HP SOA Systinet Report Editor Guide. This guide describes how to use Report Editor to create and modify reports for use with HP SOA Systinet.

This guide contains the following chapters:

- Chapter 1, Report Editor

  Introduces Report Editor, the user interface, and the main use cases.

- Chapter 2, Getting Started

  Explains how to configure your environment and Report Editor.

- Chapter 3, Designing Reports

  Explains how to create and modify report definitions for use with HP SOA Systinet.

- Chapter 4, Using DQL

  Describes the syntax and grammar for creating DQL queries.

- Chapter 5, Exploring the Database

  Explains how to examine, optimize, and access the database.

- Chapter 6, Deploying Reports

  Explains how to deploy your report definitions to HP SOA Systinet.

- Chapter 7, Example: Failure Impact Report for the Reports Tab

  Walks you through the main use case for Report Editor.

- Appendix A, Dialog Boxes

Describes Report Editor dialog boxes in detail.

- Appendix B, Troubleshooting

  Describes some common issues and their solutions.

- Appendix C, Eclipse Plug-in Requirements

  Required plug-ins when installing Workbench as an update.

Document Conventions

This document uses the following typographical conventions:

run.bat make	Script name or other executable command plus mandatory arguments.
[--help]	Command-line option.
either | or	Choice of arguments.
`replace_value`	Command-line argument that should be replaced with an actual value.
{arg1 | arg2}	Choice between two command-line arguments where one or the other is mandatory.

`java -jar`
`hpsystinet.jar`
	User input.

`C:\System.ini`	File names, directory names, paths, and package names.
`a.append(b);`	Program source code.
`server.Version`	Inline Java class name.
`getVersion()`	Inline Java method name.
Shift+N	Combination of keystrokes.
Service View	Label, word, or phrase in a GUI window, often clickable.
OK	Button in a user interface.
New>Service	Menu option.

Documentation Updates

This guide's title page contains the following identifying information:

- Software version number, which indicates the software version
- Document release date, which changes each time the document is updated
- Software release date, which indicates the release date of this version of the software

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport logon page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. For details, contact your HP sales representative.

Support

You can visit the HP Software Support Web site at:

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

HP Software Support Online provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the HP Software Support web 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, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

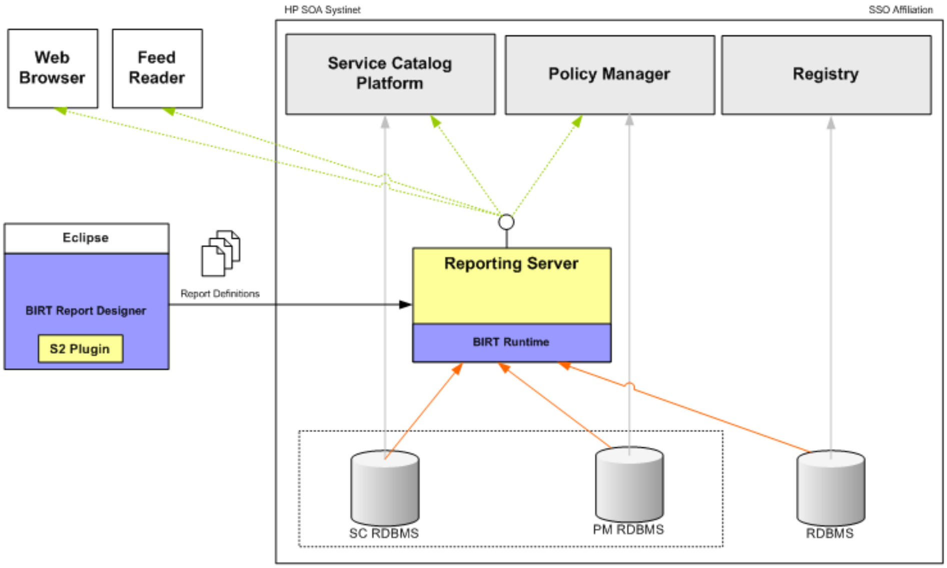
To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

1 Report Editor

Workbench includes Report Editor enabling you to create and deployment new reports easily and simply. The relationship between Report Editor and the HP SOA Systinet Reporting Service is shown in Figure 1, “HP SOA Systinet Reporting Framework”.

Figure 1. HP SOA Systinet Reporting Framework



HP SOA Systinet Report Editor is an implementation of the Built-In Reporting Tool (BIRT), an open source Eclipse-based reporting system. The Help menu includes the BIRT documentation. The *BIRT Report Developer Guide* describes the general use of BIRT to create reports. The Report Editor Guide adds to this by explaining how to use this functionality in conjunction with HP SOA Systinet.

This chapter introduces Report Editor in the following sections:

- Workbench Suite on page 12
- Overview on page 12
- User Interface on page 13
- Use Cases on page 19

Workbench Suite

HP SOA Systinet Workbench is a suite of editor tools enabling you to customize your deployment of HP SOA Systinet 4.00.

Workbench consists of the following editor tools, distributed as a single Eclipse development platform:

- **Customization Editor**

  Customizes the underlying SOA Definition Model (SDM) within HP SOA Systinet.

- **Taxonomy Editor**

  Customizes the taxonomies used to categorize artifacts in HP SOA Systinet.

- **Assertion Editor**

  Customizes the conditions applied by your business policies within HP SOA Systinet.

- **Report Editor**

  Customizes report definitions for use with HP SOA Systinet.

Overview

Report Editor interacts with HP SOA Systinet, enabling you to access existing report definitions, create customized reports, and deploy them to the HP SOA Systinet server.

To access, create, and deploy reports with HP SOA Systinet, follow this process:

1. Create a reporting project.

   For details, see Configuring a Reporting Project on page 27.

2. Create and modify reports.

   For details, see Chapter 3, Designing Reports.

3. Deploy your reports to HP SOA Systinet.

   For details, see Chapter 6, Deploying Reports.

4. Access your reports in HP SOA Systinet.

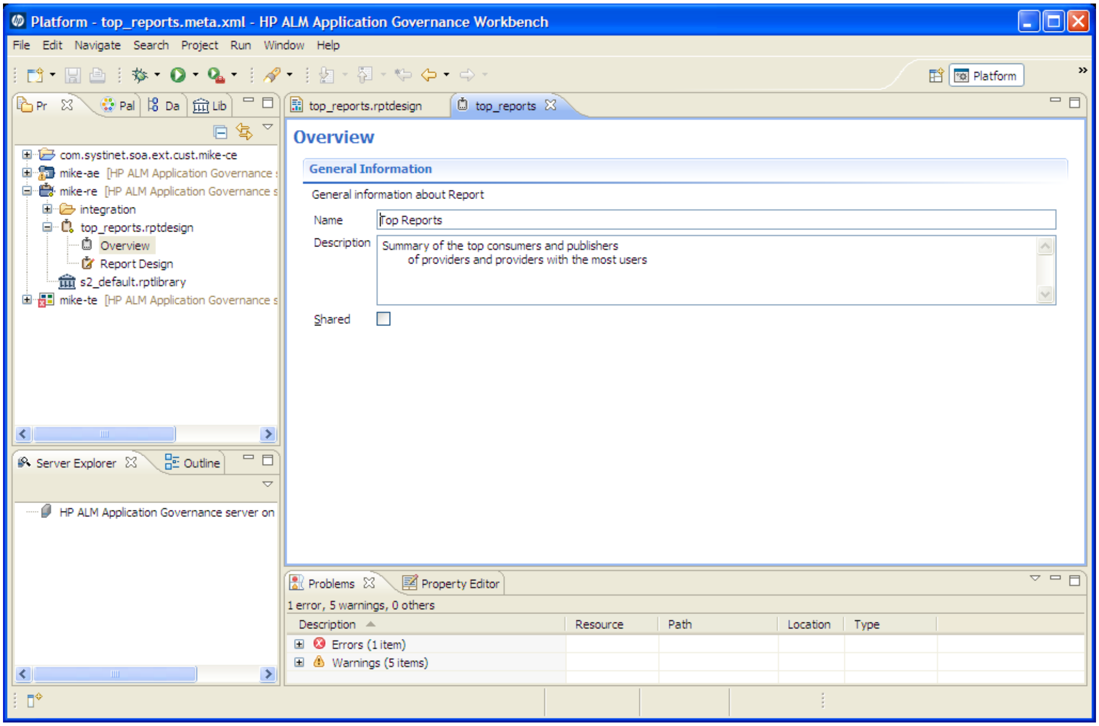
   For details, see:

   - Use Cases on page 19
   - Accessing Reports on page 78
   - Chapter 7, Example: Failure Impact Report for the Reports Tab
   - *HP SOA Systinet User Guide*

User Interface

The default platform perspective is split into a number of views with menu options across the top, as shown in Figure 2, “Platform Perspective”.

Figure 2. Platform Perspective


Platform - top_reports.meta.xml - HP ALM Application Governance Workbench
File Edit Navigate Search Project Run Window Help
Platform
Pr
Pal
Da
Lib
top_reports.rptdesign
top_reports
com.systinet.soa.ext.cust.mike-ce
mike-ae [HP ALM Application Governance
mike-re [HP ALM Application Governance s
integration
top_reports.rptdesign
Overview
Report Design
s2_default.rptlibrary
mike-te [HP ALM Application Governance s
Server Explorer
Outline
HP ALM Application Governance server on
Problems
Property Editor

Overview

General Information

General information about Report

Name: Top Reports

Description: Summary of the top consumers and publishers
of providers and providers with the most users

Shared ☐

1 error, 5 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
Warnings (5 items)				

The perspective consists of the following views:

- **Project Explorer**

  The tree view of your reporting project. For details, see Project Explorer on page 15.

- **Data Explorer**

  The view of the data sources and data sets for a particular report. For details, see Data Explorer on page 17.

- **Server Explorer**

A list of HP SOA Systinet servers connected to Workbench. For details, see Server Explorer on page 18.

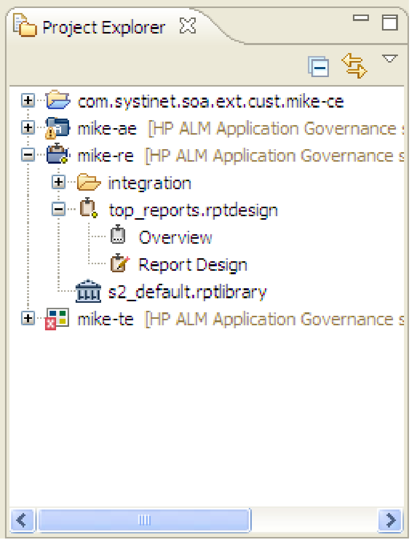
- **Editor Views**

  The main area of the perspective contains report editor views. In this view you can edit the layout and properties of your report definition. For details, see the *BIRT Report Developer Guide*, which you can access from the menu by selecting **Help>Help Contents**.

Project Explorer

The Project Explorer, as shown in Figure 3, “Project Explorer”, is a view of the projects in your workspace and the report definitions they contain. This view is also an interaction point with HP SOA Systinet.

Figure 3. Project Explorer


Project Explorer
com.systinet.soa.ext.cust.mike-ce
mike-ae [HP ALM Application Governance
mike-re [HP ALM Application Governance
integration
top_reports.rptdesign
Overview
Report Design
s2_default.rptlibrary
mike-te [HP ALM Application Governance

Each project contains a set of report definitions and the HP SOA Systinet library.

Each report definition contains the following sections:

- **Overview**

  Double-click to open a properties editor view enabling you to change the name and description, and to mark the report as **Shared** to all users or leave the report as user specific.

- **Report Design**

  Double-click to open a layout editor view enabling you to change the appearance and content of the report definition.

The Project Explorer contains additional context menu options enabling you to interact with a running HP SOA Systinet server. Right-click the project name or a particular report definition, and select **HP SOA Systinet** to view the options listed in Table 1, "Project Context Menu Options" and Table 2, "Report Definition Context Menu Options".

Table 1. Project Context Menu Options

Option	Function
Download Reports	Import report definitions from HP SOA Systinet. For details, see Importing Report Definitions on page 30.
Upload to Server	Export a report to the default HP SOA Systinet server. For details, see Deploying a Report to HP SOA Systinet on page 71.
Update from Server	Update report definitions from HP SOA Systinet.
Remove from Server	Delete reports from HP SOA Systinet.
Upload To Other Server	Export a report to a specified HP SOA Systinet server.
Build Extension	Create a reporting extension for HP SOA Systinet containing all the reports in your project. For details, see Creating a Reporting Extension on page 73.

Table 2. Report Definition Context Menu Options

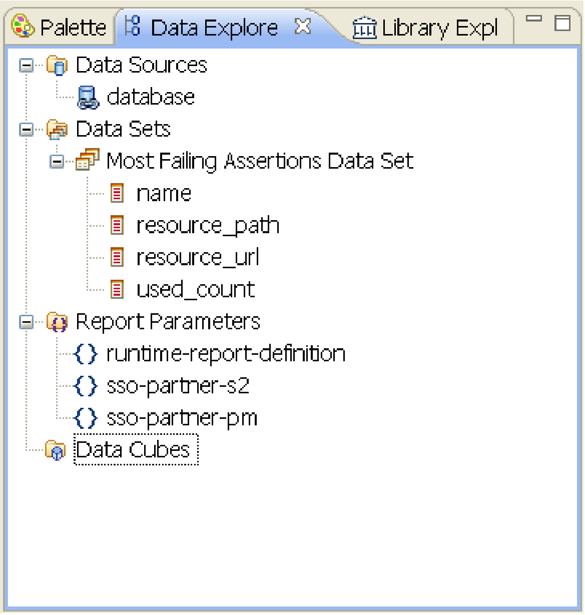
Option	Function
Upload to Server	Export a report to the default HP SOA Systinet server. For details, see Deploying a Report to HP SOA Systinet on page 71.
Update from Server	Update report definitions from HP SOA Systinet. For details, see Importing Report Definitions on page 30.

Option	Function
Remove from Server	Delete the report from HP SOA Systinet.
Upload To Other Server	Export a report to a specified HP SOA Systinet server.
Build Extension	Create a reporting extension for HP SOA Systinet containing all the reports in your project. For details, see Creating a Reporting Extension on page 73.

Data Explorer

The Data Explorer displays the sources and sets of data used by the particular report you are working with, as shown in Figure 4, “Data Explorer View”.

Figure 4. Data Explorer View



Data Explorer contains the following elements:

- **Data Sources**

Open the context menu and select **New Data Source** to access a database.

For details, see Adding a Data Source to a Report on page 34.

- **Data Sets**

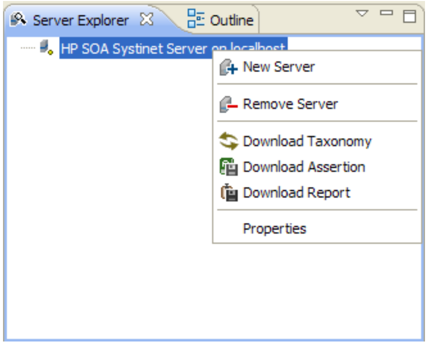
  Open the context menu and select **New Data Set** to define the data used by the report.

  For details, see Creating a Data Set on page 35.

Server Explorer

The Server Explorer displays the HP SOA Systinet servers connected to Workbench, as shown in Figure 5, “Server Explorer View”. The functionality is shared by all the Workbench editors.

Figure 5. Server Explorer View



Right-click a server in the Server Explorer to open the context menu described in Table 3, "Server Explorer Context Menu Options".

Table 3. Server Explorer Context Menu Options

Option	**Function**
New Server	Add a server for downloading assertions and taxonomies (Assertion Editor, Taxonomy Editor, and Customization Editor).
Remove Server	Delete a server from the Server Explorer.

Option	Function
Download Taxonomy	Download a taxonomy from a platform server (Taxonomy Editor and Customization Editor).
Download Assertion	Download assertions from a platform server (Assertion Editor).
Download Report	Download reports from a reporting server (Report Editor).
Properties	View and edit the server name, URL, username, and password.

Use Cases

HP SOA Systinet enables you to add custom BIRT reports created in Report Editor to the Reports tab.

To create and use a custom BIRT report in HP SOA Systinet:

1. Create a new report definition.

   For details, see Creating a Report Definition on page 33.

2. Add a data source for the report.

   For details, see Adding a Data Source to a Report on page 34.

3. Create a data set for the report,

   For details, see Creating a Data Set on page 35.

4. Design your report layout.

   For details, see the *BIRT Report Developer Guide*, which you can access from the main menu by selecting **Help>Help Contents**.

5. Deploy the report to HP SOA Systinet.

   For details, see Deploying a Report to HP SOA Systinet on page 71.

6. Add the report to the HP SOA Systinet Reports Tab.

   For details, see the "Birt Reports" section of the *HP SOA Systinet User Guide*.

For a step-by-step walkthrough demonstrating this use case, see Chapter 7, Example: Failure Impact Report for the Reports Tab.

2 Getting Started

HP SOA Systinet is distributed with a set of predefined reports and various ways to access them.

Report Editor provides a mechanism to customize these existing reports, and also enables you to create new report definitions from scratch.

This chapter describes the setup of Report Editor in the following sections:

- Installing a Report Development Environment on page 21
- Installing Workbench on page 23
- SSL Configuration on page 27
- Configuring a Reporting Project on page 27

Installing a Report Development Environment

This section describes a setup to develop and test reports on a single computer.

For supported platforms, see `readme.txt` in the installation folder.

A typical report development environment consists of the following:

- PC or notebook with at least 2GB RAM and at least 5GB free disk space
- Windows XP or Windows 2000
- Oracle eXpress Edition (XE)

  For deployment details, see Oracle XE Account Setup on page 22.

You do not need to install XE if you have dba-level access to deployed HP SOA Systinet environment using Oracle 10g.

- Java 1.5_9 or higher

  For deployment details, see Java Environment Setup on page 22.

- HP SOA Systinet deployed to JBoss

  For details, see *HP SOA Systinet Installation and Deployment Guide*.

You do not need to install HP SOA Systinet if you have access to a test environment.

- Oracle Developer

  HP Software recommends this tool for tuning SQL queries.

Oracle XE Account Setup

If you install Oracle XE, you must create a user account.

To create an Oracle XE account:

1. Open the XE web console and log on as the system user.
2. Create a new user, `platform`, with the password `changeit`, then grant the new user all available privileges.

Java Environment Setup

After installing Java, you must set an environment variable, JAVA_HOME.

To set the JAVA_HOME environment variable:

1. In Windows, click through **Control Panel>System>Advanced>Environment Variables**.

   The Environment Variables dialog box opens.

2. In the User Variables section, click **New**.
3. Input Variable Name, `JAVA_HOME`, and set Variable Value to the installation folder for Java.
4. Click **OK** to exit each dialog box.

Installing Workbench

HP SOA Systinet Workbench is an Eclipse development platform distributed as a zip file, `hp-soa-systinet-workbench-4.00-win32.zip` or as a plugin for an existing Eclipse environment, `hp-soa-systinet-workbench-4.00-plugin.zip`.

For supported platforms and known issues, see `readme.txt` alongside the archive.

To install HP SOA Systinet Workbench as a new Eclipse platform:

- Extract the archive to your required location, referred to in this document as `WB_HOME`.

The path must not be longer than 97 characters.

To install HP SOA Systinet Workbench to an existing Eclipse platform:

1. Ensure that your Eclipse platform contains the necessary plug-ins. For details, see Appendix C, Eclipse Plug-in Requirements.
2. In your current Eclipse SDK (3.3 or later), use the software updates feature to install HP SOA Systinet Workbench.

   Select **Help>Software Updates>Find and Install...**.

   The Install/Update dialog opens.
3. In the Install/Update dialog, select **Search for new features to install**, and click **Next**.

   The Install – Update Sites to Visit dialog opens.

4. In the Update Sites to Visit dialog, click **New Archived Site**.

   The Select Local Archive Site dialog opens.

5. Locate and select `hp-soa-systinet-workbench-4.00-plugin.zip`, and then click **Open**.

   The Edit Local Site dialog opens.

6. In the Edit Local Site dialog, if required, rename the local archive name, and click **OK**.

7. In the Install – Update Sites to Visit dialog, select the new local archive, and then click **Finish**.

   The Updates – Search Results dialog opens.

8. Select the modules from the archive that you want to install:

   - **Workbench Extra 4.00**

     HP SOA Systinet Workbench splash screen.

   - **Taxonomy Editor 4.00**

     Required for Customization Editor.

   - **Customization Editor 4.00**

   - **Assertion Editor 4.00**

   - **Report Editor 4.00**

   - **Common Plugin 4.00**

     Shared components used by the editors.

   Click **Next**.

   The Install – Feature License dialog opens.

9. In the Feature License dialog, select **I accept the terms in the license agreements**, and click **Next**.

The Install – Installation dialog opens.

10 In the Installation dialog, if required, change the installation location, and then click **Finish**.

11 If you install `Workbench Extra 4.00` make the following configuration changes:

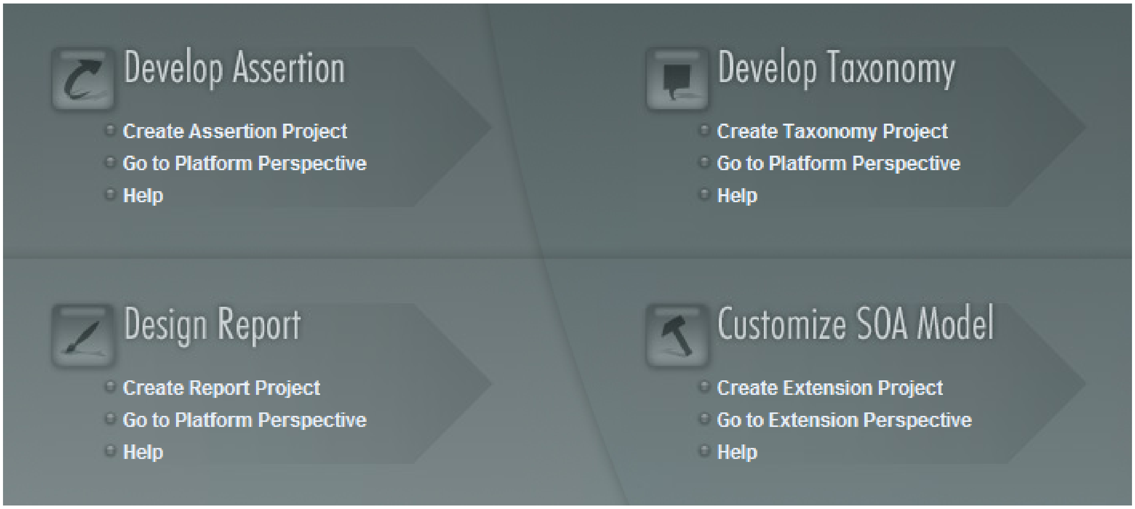
- Remove `-showsplash org.eclipse.platform` from `ECLIPSE_HOME/eclipse.ini`.
- Edit ECLIPSE_HOME/configuration/config.ini and make the following changes:
  - Set `osgi.splashPath=platform:/base/plugins/com.systinet.tools.workbench .`
  - Set `eclipse.product=com.systinet.tools.workbench.ide`

To start HP SOA Systinet Workbench:

- Execute **WB_HOME/workbench/start.exe**.

The first time you start Workbench, the welcome screen opens, as shown in Figure 6, “Workbench Welcome Screen”.

Figure 6. Workbench Welcome Screen



Select one of the options to open one of the editor tools, start a new editing project, or view the documentation set.

You can return to the welcome screen from any of the editor tools by selecting **Help>Welcome** from the menu options.

By default, Workbench runs in 'normal' mode which prevents users from uploading system taxonomies (IDs start with *uddi:systinet.com:soa:model:taxonomies*) and the Report Editor .rptlibrary file to HP SOA Systinet servers. If you need to work with system taxonomies or want to upload the .rptlibrary file you can switch Workbench into 'admin' mode.

⚠ Be extremely careful when working with system taxonomies, HP SOA Systinet uses some hard-coded values from system taxonomies, changing or removing them may cause errors.

To Switch Workbench to Admin Mode

1. Open `WB_HOME/configuration/config.ini` with a text editor.
2. Add *mode=admin* to `config.ini`.
3. Restart Workbench.

⚠ HP SOA Systinet Workbench requires Java SE Development Kit (JDK) 1.5.0 or higher. You must include the path to this version of the JDK in the *`JAVA_HOME`* environment variable.

➤ HP SOA Systinet Workbench is memory-intensive. If you experience performance issues, HP recommends increasing the memory allocation.

To increase the memory allocation for HP SOA Systinet Workbench:

1. Open `WB_HOME/workbench/start.ini` for editing.
2. Set these new values:
   - `-Xms128m`

- `-Xmx1024m`

3. Save your changes.

4. Restart Workbench.

SSL Configuration

By default, Workbench trusts all HP SOA Systinet server certificates. You may want Workbench to verify HP SOA Systinet certificates.

To verify HP SOA Systinet server certificates:

- Add the following options to `WB_HOME/workbench/start.ini`:

```
-Dcom.hp.systinet.security.ssl.verifyCert=true
-Djavax.net.ssl.trustStore=USER_TRUSTSTORE
-Djavax.net.ssl.trustStorePassword=TRUSTSTORE_PASS
-Djavax.net.ssl.trustStoreType=TRUSTSTORE_FORMAT
```

If HP SOA Systinet is configured for 2-way SSL, you must provide Workbench certificates to HP SOA Systinet.

To provide Workbench client certificates to HP SOA Systinet:

- Add the following options to `WB_HOME/workbench/start.ini`:

```
-Djavax.net.ssl.keyStore=USER_KEYSTORE
-Djavax.net.ssl.keyStorePassword=KEYSTORE_PASS
-Djavax.net.ssl.keyStoreType=KEYSTORE_FORMAT
```

Configuring a Reporting Project

To use Report Editor with HP SOA Systinet, you must create or import a project, and import any report definitions you want to modify.

These procedures are described in the following sections:

- Creating a Reporting Project on page 28

- Importing a Reporting Project on page 30
- Importing Report Definitions on page 30

In addition you can configure Report Editor in the following ways:

- Rebranding HP SOA Systinet Reports on page 31
- Configuring the Model on page 31
- Configuring the Reporting Logging Level on page 32

Creating a Reporting Project

The Report Editor organizes your work into project folders in your workspace.

To create a reporting project:

1. Do one of the following:
   - In the Workbench Welcome page, select **Create Report Project**.
   - From the menu, select **File>New>Report Project**.
   - Press **Alt+Shift+N**, and then press **R**. Expand **HP SOA Systinet**, select **Report Project**, and then click **Next**.

   The New Report Project Wizard opens.

2. In the New Report Project dialog box, enter the following parameters:

Parameter	Definition
Project Name	The name of your report project.
Create from Existing Extension	Select this option if you want to create a new project from a previous report extension. If selected, input the path or browse for the location of the report extension.
Use Default Location	If selected, Report Editor stores the project in your default workspace. If unselected, input the path or browse for an alternative workspace.

3. Do one of the following:

   - Select **Create a New Server**, and then click **Next**.

     Continue to Step 4.

   - Select **Use an Existing Server**, select the server from the list and input its credentials, and then click **Next**.

     Continue to Step 5.

4. In the New Server dialog box, add the parameters you require, and then click **Next**.

   For parameters descriptions, see New Report Project Wizard: New Server on page 94.

5. If needed, in the New Data Source dialog box, click **Manage Drivers** to add a JDBC driver.

   For details, see Adding a JDBC Driver on page 64.

6. In the New Data Source dialog box, add the parameters you require.

   For parameters descriptions, see New Report Project Wizard: Default Library Configuration - Create a new data source on page 95.

   If your database server is running, click **Test Connection** to check if your settings are correct.

7. Click **Next** to set project options.

8. Do one of the following:

   - To create a generalized reporting project, click **Finish** and exit the wizard.

   - To create a project associated with a specific report category, click **Next** and continue this procedure.

     The HP SOA Systinet server must be running to access report definitions.

9. Select a report category for your reporting project, and then click **Next**.

10. Select report definitions from the selected category to download to the Report Editor, and then click **Finish**.

Importing a Reporting Project

You can import a reporting project created elsewhere into your Workbench installation.

To import a reporting project:

1. Copy the project folder to your local files ystem.

2. From the menu, select **File>Import**.

   The Import dialog box opens.

3. Expand **General** and select **Existing Projects into Workspace**.

4. Use **Browse**, select the project from your local file system, and then click **OK**.

5. Click **Finish** to import the reporting project.

Importing Report Definitions

The HP SOA Systinet HP SOA Systinet server contains all the report definitions for HP SOA Systinet. To modify existing reports, you must import their definitions from the HP SOA Systinet server.

To import HP SOA Systinet report definitions:

1. In the Project Explorer, open the context menu for the project, and do one of the following:

   - Select **HP SOA Systinet>Download Reports** to import from the current HP SOA Systinet server.

   - Select **Import>Report**, and then select a reporting server.

   The HP SOA Systinet server must be running.

2 Select the report category, and then click **Next**.

3 Select the report definitions you require, and then click **Finish**.

Rebranding HP SOA Systinet Reports

By default, most HP SOA Systinet reports include an HP branded logo. You can replace this image with your own company-specific image.

To rebrand HP SOA Systinet reports:

1 Open `s2_default.rptlibrary` for editing.

2 In the Outline view, expand Embedded Images.

3 Open the context menu for `logoHP.gif` and select **Delete**.

4 Open the context menu for Embedded Images, and select **New Embedded Image**.

5 Upload your new company branded image which must be named `logoHP.gif`.

6 Save `s2.default.rptlibrary`.

7 Redeploy all reports that use this image to HP SOA Systinet. For details, see Chapter 6, Deploying Reports.

Configuring the Model

HP SOA Systinet divides the architecture model into a public and system model. By default, Report Editor accesses the public model which means that you can only create reports that access these public artifact types.

If you need to access system artifact types in your reports then you must modify the library.

To Extend the Reporting Model to access System Artifacts:

1 Open the `s2_default.rptlibrary` file in your workspace in a text editor.

2 Locate the `oda-data-source` element with `name="dqlDatabase` in the file.

3. Edit the `property name="odaURL"` element, adding `|model=sys` to the property value.

4. Save `s2_default.rptlibrary`.

Configuring the Reporting Logging Level

HP SOA Systinet reporting uses the BIRT engine for report execution which uses standard JDK logging with root logger `org.eclipse.birt`. You can configure the logging in that same way as any other component that uses JDK logging, typically using a configuration file passed with the **-Djava.util.logging.config=** switch.

The default logging level is set to SEVERE, but this is overidden if you specify the logging level in a logging configuration file..

For details about JDK logging, see http://java.sun.com/j2se/1.5.0/docs/guide/logging/overview.html.

3 Designing Reports

This chapter describes the functionality of Report Editor in conjunction with HP SOA Systinet. The standard functionality of Report Editor is described in the *BIRT Report Developer Guide*, which you can access from the main menu by selecting **Help>Help Contents**.

This chapter contains the following sections:

Creating a report:

Creating a Report Definition

The first step in defining a report is to create the report definition.

To create a report definition:

1. Select **File>New>Report** from the menu.
2. Input a name and description for the report, and then click **Next**.
3. Select the project folder, and then click **Finish** to create the report definition.
4. Press **Ctrl+S** to save the report definition.

See Step 1 of Chapter 7, Example: Failure Impact Report for the Reports Tab for an example of this procedure.

Adding a Data Source to a Report

By default, new reports contain DQL and SQL data sources setup during project creation. If you need to use a different one, you must add it to the report.

HP SOA Systinet only supports the default datasources. Reports using user-created datasources work within Report Editor but do not display when the report executes on the HP SOA Systinet server.

To add a data source to a report:

1 Open the report definition editor view.

2 In the Data Explorer, open the Data Sources context menu and select **New Data Source**.

The New Data Source dialog box opens.

3 Do one of the following:

- To add a datasource for DQL queries, select **DQL JDBC Data Source**.
- To add a datasource for SQL queries, select **JDBC Data Source**..

Click **Next**.

4 If required, in the Data Source dialog box, click **Manage Drivers** to add a JDBC driver (the DQL driver is pre-installed and there should never be a requirement to add one.

For details, see Adding a JDBC Driver on page 64.

5 In the Data Source dialog box, add the parameters you require.

For parameter descriptions, see New Report Project Wizard: Default Library Configuration - Create a new data source on page 95.

If your database server is running, click **Test Connection** to check if your settings are correct.

6 Click **Finish** to add the data source to your report.

7. Press **Ctrl+S** to save your changes to the report definition.

Creating a Data Set

Each report is associated with one or more queries that each return a data set.

To create a data set:

1. From the Report Design perspective, open the report definition view for the report requiring the new data set.

   If you need to use an alternative data source, see Adding a Data Source to a Report on page 34.

2. In the Data Explorer, open the Data Sets context menu, and select **New Data Set**.

   The **New Data Set** dialog opens.

3. In the New Data Set dialog box, add the parameters you require.

   For parameters descriptions, see New Data Set on page 93.

4. Click **Next** to open the **Query** dialog box.

5. Do one of the following:

   - Copy a query from your development tool to the query editor.

     Skip to Step 8.

   - Create your query in Report Editor as described in Step 6 to Step 7.

6. Use **Schema** (SQL only), **Filter**, and **Type**, and then click **Apply Filter** to focus on the data you require in Available Items.

   For SQL queries, Workbench limits the amount of data collected. To change the settings, see Changing the Data Collection Settings on page 64.

7. Browse Available Items to find the data items you require. Drag and drop items from Available Items to the query editor and construct your query.

    For portability to different HP SOA Systinet installations using different schemas, remove the schema name from the variables in your query if your database schema does not require them.

8. Click **Finish** to add the data set to the report definition.

9. Press **Ctrl+S** to save your changes.

For details about writing DQL queries, see Chapter 4, Using DQL.

For details about the database structure and writing SQL queries, see Chapter 5, Exploring the Database.

For an example of this procedure, see Example: Failure Impact Data Set on page 82.

If possible, use only one dataset per report definition. If you need more datasets in one report (for example, multiple properties in one table row), use JavaScript to build one table.

Designing a Report Layout

The *BIRT Report Developer Guide* contains all the information required to design the layout and content of your reports. Select **Help>Help Contents** to view the guides included with Workbench.

For an example, see Example: Failure Impact Report Layout on page 84.

Shared and User-Specific Reports

The Report Editor enables you to determine the security of report output. A report can be marked as shared to show the same data to all users or not shared. If a report is not shared, then each user only sees the data they have read permission for in the report output.

The default value for a new report is shared.

To mark a report as shared or user-specific:

1. In the **Project Explorer**, expand the report you want to categorize and double-click **Overview** to open a view of the report definition.

2 The General Information section displays a Shared check-box.

Select **Shared** to show the same report output to all users. De-select **Shared** to make the report output restricted according to the user read permissions.

3 Press **Ctrl+S** to save your changes.

Modifying Existing Report Definitions

Report definitions that already exist in HP SOA Systinet can be modified.

To modify existing report definitions:

1 Import the report definition.

For details, see Importing Report Definitions on page 30.

2 To open the report definition, in the Project Explorer, double-click the report definition *`rptdesign`*.

3 Make your modifications, as described in Creating a Report Definition on page 33.

4 Press **Ctrl+S** to save your changes.

5 Redeploy the report definition to HP SOA Systinet.

For details, see Deploying a Report to HP SOA Systinet on page 71.

4 Using DQL

The DQL query language provides a simple query solution for the SOA Definition Model (SDM). It enables you to query all aspects of the model – artifacts, properties, relationships, governance, and compliance.

This chapter describes DQL in the following sections:

Introduction to DQL

DQL is an SQL-like language that enables you to query the repository of artifacts in HP SOA Systinet defined by the SDM model. DQL preserves SQL grammar, but uses artifacts instead of tables, and artifact properties instead of table columns. As DQL is based on SQL you can apply your SQL knowledge to DQL.

A simple example is to return the name and description of all business service artifacts.

```
select name, description from businessServiceArtifact
```

In HP SOA Systinet, you can use DQL queries in the following use cases:

- To create reports in HP SOA Systinet Report Editor. For more details, see the *HP SOA Systinet Report Editor Guide*.
- To customize pages of the HP SOA Systinet user interface. For details, see "Customizing the User Interface" in the *HP SOA Systinet Administrator Guide*.
- You can also use DQL in any SQL designer using the DQL JDBC driver. For more details, see DQL in SQL Designers on page 61.

The following sections contain DQL examples:

Primitive Properties

Primitive properties are simple properties, such as numbers, characters, and dates, that may occur once or multiple times for an artifact depending on the cardinality as defined in the SDM.

For example, in the SDM Model, each person is represented by a person artifact. The person artifact includes a name property with single cardinality and an email property with multiple cardinality.

The following query returns the name and all emails for each person in the repository.

```
select name, email from personArtifact
```

Instances of primitive properties with multiple cardinality are all returned as comma separated values. For example, all the emails for a person return as a concatenated, comma-separated string. If there is no instance of the property for an artifact, a null value is returned.

The following query returns the name, description, and version of all business service artifacts whose version is 2.0.

```
select name, description, version
  from businessServiceArtifact
  where version = '2.0'
```

By default, DQL queries return the latest revisions of artifacts unless you specify revision modifiers. For details, see Modifiers on page 46.

Complex Properties

Complex properties are composed of one or more single or multiple-valued sub-properties (for example, address contains sub-properties addressLines in multiple cardinality, country in single cardinality, etc. The sub-property addressLines is also a complex sub-property, containing a value and useType.). It is only possible to query the sub-property components of primitive types. Components of sub-properties are separated by **.** (in MS Access you can use **$** as a separator).

```
select address.addressLines.value, address.country
  from personArtifact
  where address.city = 'Prague'
```

For a full reference of all complex properties in the default SDM, see "SOA Definition Model" in the *HP SOA Systinet Reference Guide*.

Artifact Inheritance

Artifacts in HP SOA Systinet form a hierarchy defined by the SDM model. Artifacts lower in the hierarchy inherit properties from higher abstract artifact types. `artifactBase` is the root abstract artifact type in the SDM hierarchy. All other artifacts are below it in the hierarchy and inherit its properties. You can query abstract artifacts and return a result set from all the instances of artifact types lower in the hierarchy.

Property groups function in a similar way, querying a property group returns results from all artifact types that inherit properties from the group.

The following query returns results from all implementation artifacts; SOAP Services, XML Services, and Web Applications.

```
select name, serviceName
  from implementationArtifact
```

Notice that in this query, serviceName is a specific property of SOAP Service artifacts. In the result set, name is returned for all implementation artifacts but serviceName is only returned for SOAP service artifacts. For other implementation types, the serviceName is NULL.

⚠ Different artifact types may define the same properties with different cardinalities. In cases where two artifact types define the same property with different cardinality, querying a shared parent abstract artifact for these properties may fail. Examples that fail include **SELECT environment FROM artifactBase** and **SELECT accessPoint FROM artifactBase**.

Categorization Properties

Categorization properties are a special case of complex properties.

Categorization properties have the following sub-properties:

- `val` - machine readable name of the category.
- `name` - human readable name of the category.
- `taxonomyURI` - identifies the taxonomy defining the category set.

taxonomyURI is not defined for named category properties.

HP SOA Systinet uses categorization properties in the following ways:

- **Named category properties** (for example, business service criticality).

  The following query returns the names, descriptions, and versions of all business service artifacts which are categorized using the named criticality categorization property with a high failure impact.

  ```
  select name, description, version
    from businessServiceArtifact
    where criticality.val = 'uddi:systinet.com:soa:model:taxonomies:impactLevel:high'
  ```

  taxonomyURI is not defined for named category properties. The name of the category property implies the taxonomy.

- **categoryBag**

  `categoryBag` is a complex property that includes sub-property `categories` which is a categorization property and `categoryGroups`. `categoryGroups` also contains categorization sub-property `categories` and a `taxonomyURI` defining the meaning of the group. HP Software recommend querying `_category` instead of `categoryBag` to ensure that all categories are queried.

  The following query returns the names, descriptions, and versions of all business service artifacts which are categorized by the Gift certificate category (14111608) of the `uddi:uddi.org:ubr:categorization:unspsc` taxonomy.

```
select name, description, version
  from businessServiceArtifact
  where categoryBag.categories.taxonomyURI =
 'uddi:uddi.org:ubr:categorization:unspsc'
  and categoryBag.categories.val = '14111608'
```

- **identifierBag**

  `identifierBag` is a complex property similar to `categoryBag` that includes sub-property `categories`. `identifierBag` does not contain the `categoryGroups` subproperty. HP Software recommend querying `_category` instead of `identifierBag` to ensure that all categories are queried.

- **_category**

  This generic categorization property holds all categorizations from `categoryBag`, `identifierBag`, and all named categorization properties from the given artifact type.

  The following query returns the names, descriptions, and versions of all business service artifacts which are categorized with a high failure impact.

  ```
  select name, description, version
    from businessServiceArtifact
    where _category.val = 'uddi:systinet.com:soa:model:taxonomies:impactLevel:high'
    and _category.taxonomyURI = 'uddi:systinet.com:soa:model:taxonomies:impactLevel'
  ```

⚠ When you use the generic `_category` property you must specify the taxonomy using the `_category.taxonomyURI` sub-property. When you use a named categorization property the taxonomy is implicitly known and does not need to be specified.

Fixing Multiple Properties

Consider a business service with keywords, 'Finance' and 'Euro'. The intuitive query for finding a 'Euro Finance' service is as follows:

```
select name, description, version
  from businessServiceArtifact b
  where b.keyword.val = 'Finance'
      and b.keyword.val = 'Euro'
```

This query does not work as a single instance of keyword can never be both 'Finance' and 'Euro'

The solution is to fix instances of multiple properties as shown in the following query:

```
select name, description, version
  from businessServiceArtifact b, b.keyword k1, b.keyword k2
  where k1.val = 'Finance'
      and k2.val = 'Euro'
```

Relationships

A relationship is a special kind of complex property pointing to another artifact. HP SOA Systinet uses relationships to join artifacts.

The following queries are semantically identical and return all business services and the contact details of their provider. These queries do not return business services that do not have providers.

- The following query is an example of an *SQL92-like* join which uses the USING clause.

  ```
  select b.name, b.version, b.keyword.name, p.name as contact, p.email
    from businessServiceArtifact b join personArtifact p using provides
  ```

  The relationship property `provides` leads from person artifacts to business service artifacts is specified after the `using` keyword.

- The following query is an example of an *SQL92-like* join which uses the ON clause.

  ```
  select b.name, b.version, b.keyword.name, p.name as contact, p.email
    from businessServiceArtifact b join personArtifact p on bind(provides)
  ```

  The relationship property `provides` leads from person artifacts to business service artifacts is specified with the `bind` predicate in the `WHERE` clause.

- The following query is an example of an *old-style* join which uses the BIND predicate.

  ```
  select b.name, b.version, p.name as contact, p.email
    from businessServiceArtifact b, personArtifact p
    where bind(p.provides, b)
  ```

  The BIND predicate specifies that the provides relationship of the person artifact points to business service artifacts.

The following query also returns all business services and the contact details of their provider. This query is an example of a LEFT JOIN. The LEFT JOIN extends the previous queries by also returning business services that do not have providers.

```
select b.name, b.version, p.name as contact, p.email
```

```
    from businessServiceArtifact b left join personArtifact p using provides
```

Each relationship has the following sub-properties which you can query:

- rType - the SDM QNames of the relationship type.
- useType - the values of the useType relationship property
- target - the UUIDs of the artifact the relationship points to (deprecated).

It is possible to specify a particular provider type using useType. The following queries return all business services and their contact details where the provider is an architect.

```
select b.name, b.version, p.name as contact, p.email
    from businessServiceArtifact b, personArtifact p
    where bind (p.provides, b)
  and p.provides.useType = 'architect'
```

```
select b.name, b.version, p.name as contact, p.email
    from businessServiceArtifact b join personArtifact p
    on bind(p.provides, b)
    and p.provides.useType = 'architect'
```

It is possible to traverse several relationships using several old-style joins or SQL-92-like join clauses in the same query. The following example queries business services in applications, which are also part of a project.

```
select b.name, b.description, a.name as Application, p.name as Project
  from businessServiceArtifact b
    join hpsoaApplicationArtifact a using hpsoaProvidesBusinessService
    join hpsoaProjectArtifact p using contentRelationshipType
```

In cases where artifacts may be joined by multiple properties, you can use a generic `_relation` property together with the additional `rType` condition.

```
select A.name as A_name, B.name as B_name
    from hpsoaApplicationArtifact A left join artifactBase B on bind(A._relation)
    and A._relation.rType in ('{http://systinet.com/2005/05/soa/model/
property}hpsoaProvidesBusinessService',
    '{http://systinet.com/2005/05/soa/model/property}r_providesBusinessProcess');
```

You can use the `target` relationship sub-property to bind the source and target of a relationship.

```
select b.name, b.version, p.name as contact, p.email
    from businessServiceArtifact b, personArtifact p
    where p.provides.target = b._uuid
```

⚠ The `target` property and this style of comparison is deprecated and its use is not recommended. Use the `bind` predicate instead.

Modifiers

Modifiers define primary sets of objects (artifacts and their revisions) to query. If no modifier is specified, the last revisions of undeleted artifacts for which the user has read access are queried.

The following modifiers are available:

- Revision related modifiers (mutually exclusive):
  - **all_rev** - queries all revisions of artifacts.
  - **last_approved_revision** - queries the last approved revisions of artifacts.
- Security related modifiers (mutually exclusive):
  - **my** - queries artifacts belong to the user.
  - **writable** - queries artifact the user has write permission for.
  - **no_acl** - queries all artifacts regardless of security.
- Other modifiers:
  - **include_deleted** - queries all instances, including deleted artifacts.

You can use multiple comma-separated modifiers.

The following query returns all business services that you own that are marked as deleted.

```
select b.name, b.version, b.keyword.name
    from businessServiceArtifact b (my, include_deleted)
    where _deleted = '1'
```

Virtual Properties

DQL defines virtual properties, that are not defined by the SDM. HP SOA Systinet stores or calculates these properties enabling DQL to query meta information about artifacts. These virtual properties provide information about lifecycle, compliance, domains, etc.

The following example returns lifecycle details from the last approved revisions of all business service artifacts, ordered by lifecycle stage.

```
select name, _lastApprovedStage.name Stage, _revision
  from businessServiceArtifact(last_approved_revision)
  order by Stage
```

The following example returns the name and compliance status of last approved revisions of all business services which a compliance status of at least 80%.

```
select b.name, b._complianceStatus
  from businessServiceArtifact b (last_approved_revision)
  where b._complianceStatus >= 80
```

HP SOA Systinet repository content exists within a domain structure where each artifact exists within only one domain. The default functionality of DQL queries all domains but HP SOA Systinet provides virtual properties enabling you to query artifacts within a particular domain. The following example returns business service names and the domain details of all business service artifacts that exist within the EMEA domain.

```
select A.name, A._domainId, A._domainName from businessServiceArtifact A
    where A._domainId="EMEA"
```

DQL provides the following macros for querying within domain hierarchies:

- **#SUBDOMAINS('*`domainId`*')**

  Queries the specified domain and all its sub-domains.

- **#SUPERDOMAINS('*`domainId`*')**

  Queries the specified domain and all its parent domains.

The following query returns all business services in the EMEA domain and any of all of its sub-domains.

```
select A.name from businessServiceArtifact A
    where A._domainId in #SUBDOMAINS('EMEA')
```

The following query returns the name and virtual properties artifactTypeName and owner from the latest revisions of consumer properties (the property group for all consuming artifact types).

```
select name, _artifactTypeName, _owner
  from consumerProperties
```

For details of all virtual properties, see Properties in DQL on page 49.

Embedding SQL Queries

DQL works with SDM entities (artifacts and properties) only and cannot directly access database tables. In some cases it is necessary to obtain values from outside the SDM (for example, system configuration). You can use an SQL subquery in a NATIVE clause of a DQL query. By default, DQL expects SQL to return an unnamed single column of values.

The following example returns business services owned by the administrator using the name defined during installation:

```
select name,description, version
  from businessServiceArtifact
  where _owner in (
    native {select svalue
      from systemConfiguration
      where name='shared.administrator.username'})
```

You can use NATIVE clauses instead of expressions, as a condition in WHERE clauses, as a column in SELECT clauses, and as a artifact reference in FROM clauses. For details, see DQL Grammar on page 53.

If you use a NATIVE clause to formulate part of a FROM clause, you must specify parameters to bind columns defined by SQL to properties used by DQL.

Each parameter consists of the following:

- The property name defines how DQL addresses columns returned from the NATIVE SQL statement.
- The property type which may be returned by the metadata of a column is optional and if not specified is assumed to be a text string.

The parameters are enclosed in brackets in the native clause, delimited by commas, and the type is separated from the name using whitespace.

The following example shows a query with NATIVE SQL in a DQL FROM clause.

```
SELECT B.p_id, B.s_val, A.name, B.state_index
    FROM
        (native(s_val, s_name, state_index integer, p_name, p_id)
            {select S.val as s_val, S.name as s_name, S.state_index as state_index,
 P.name as p_name, P.id as p_id
                  from rylf_state S, rylf_process P WHERE S.fk_rylf_process=P.id and
 P.name='Application Lifecycle'}) B
    LEFT JOIN artifactBase A ON A._currentStage.val = B.s_val
    ORDER BY B.p_id, B.state_index
```

The NATIVE statement returns the following columns; `s_val`, `s_name`, `p_name`, and `p_id` of type String, and `state_index` of type Integer.

Native clauses can not contain variables (? or :<variable>).

DQL Reference

This section provides a reference to properties and DQL grammar in the following sections:

- Properties in DQL on page 49
- DQL and SQL on page 53
- DQL Grammar on page 53

Properties in DQL

Artifact (property group) properties hold values which may be queried in DQL expressions.

DQL recognises the following properties:

- SDM Properties

  Properties defined in the SDM Model. For details, see "SOA Definition Model" in the *HP SOA Systinet Reference Guide*.

- Virtual, System, and Other Properties

  Properties holding metadata about artifact instances.

Properties may be one of the following:

Property Kind	**Description**
Primitive	Holds string, number, or boolean values. For example, name, description, version. A primitive property is defined in DQL statements by the artifact type name or alias, the property delimiter (. or $), followed by the property

Property Kind	Description
	name. For example, personArtifact.name. The artifact name or alias is optional (with the delimiter) when the property is specific to a single artifact type in the query.
Complex	Hold complex structures such as address. Only primitive sub-properties of complex properties may be queried. Properties and sub-properties are separated by **.** or **$**. For example, `personArtifact.address.city`.
Categorization	Hold categorization data and are handled in a similar way to complex properties. Categories consist of `name`, `val`, and `taxonomyURI` components. For example, `businessServiceArtifact.criticality.name`.
Relationships	Properties that specify a directional relationship to other artifacts.

All values that you can query are of a particular data type. The following table describes these data types, and gives examples of how to use them in a query.

Data Type	Description	Example
Number	Numeric values	_revision = 1
String	Text values	_revisionCreator = 'admin'
Date Time	Date and Time (ms since 00:00 1/1/1970)	_revisionTimestamp > 1274447040124 /* revisions made since 15:04 21/5/2010 */
Boolean	True or False flags	_deleted = '1'

Properties may have the following cardinalities:

Property Cardinality	Description
Single	Only one instance of the property exists for an artifact and it may be optional or required.
Multiple	The property is a list of values and so occurs multiple times for an artifact. In WHERE clauses, using multiple properties returns particular artifacts if any instance of the multiple property matches the condition. In SELECT clauses, using multiple properties returns all instances of the multiple property as a concatenated, comma-separated string.

DQL uses the following virtual and system properties:

Property Class	**Property**	**Description**
System Property	_artifactTypeName	The human readable name of the artifact type (the SDM label of the artifact). HP Software recommend using _sdmName in conditions, and _artifactTypeName in SELECT clauses.
	_category	All categorizations for an artifact with `name`, `val`, and `taxonomyURI` components.
	_deleted	The deletion marker flag (boolean).
	_id	The database ID of an artifact instance (number, deprecated - use _uuid).
	_longDescription	HP SOA Systinet supports a long description including HTML tags up to 25000 characters by default. HP Software recommend using the `description` property in DQL queries instead as queries using _longDescription may affect performance and the HTML tags may corrupt report outputs. `description` contains only the first 1024 text characters of `_longDescription` (may vary according to your database type). The `platform.repository.max.description.length` property determines the maximum length of _longDescription. You can modify this property in `SOA_HOME/conf/setup/configuration-propeties.xml`. DB2 has an absolute limit of 32672 characters which is sufficient for descriptions containing ASCII (single-byte encoding) characters but may be exceeded, for example by Japanese descriptions, leading to description truncation.
	_owner	The user, group, or role designated as the artifact owner.
	_ownerName	The human readable name of the user (taken from the use profile), group, or role artifact.

Property Class	Property	Description
	_path	Legacy REST path of the artifact (string, deprecated - use _uuid).
	_relation	A generic virtual property that may be used to specify all outgoing relationships.
	_revision	The revision number of an artifact instance.
	_revisionCreator	The user who created the revision of the artifact.
	_revisionTimestamp	The date and time the revision was created.
	_sdmName	The local name of the artifact type. HP Software recommend using _sdmName in conditions, and _artifactTypeName in SELECT clauses.
	_uuid	The unique artifact indentifier.
UI Property	_isFavorite	Marked by the user as favorite flag (boolean).
	_rating	The average rating of the artifact (double).
Security Property	_writeable	User write permission flag (boolean). ▶ Using this property may have a performance impact. If possible, use the **writable modifier** instead. For details, see Modifiers on page 46.
Contract Property	_enabledConsumer	The artifact is a valid consumer artifact type.
	_enabledProvider	The artifact is a valid provider artifact type. The artifactmust also be marked as 'Ready for Consumption'.
Domain Property	_domainId	Value of the `domainId` property defined for the domain artifact that the artifact belongs to.
	_domainName	The readable name of the domain.
Lifecycle Property	_currentStage	Current working stage of an artifact.
	_governanceProcess	The process applicable to the artifact.
	_isApproved	Lifecycle approval flag (boolean).
	_lastApprovedRevision	Revision number of the last approved revision (number).

Property Class	**Property**	**Description**
	_lastApprovedStage	The name of the last approved stage.
	_lastApprovalTimestamp	Timestamp for the last approval (number, ms since 00:00 1/1/1970).
_lifecycleStatus	The status of the current lifecycle stage.	
Policy Manager Property	_complianceStatus	Compliance status as a percentage (number).

DQL and SQL

DQL supports most features of SQL with the following exceptions:

- SELECT * is not supported.
- RIGHT and FULL OUTER JOIN are not supported.
- It is not possible to use properties with multiple cardinality in GROUP BY, HAVING, or ORDER BY clauses.

DQL Grammar

A DQL query consists of the following elements with their grammar explained in the following sections:

- Select on page 54
- FROM Clause on page 55
- Conditions on page 55
- Expressions on page 57
- Lexical Rules on page 59

Table 4. Typographical Conventions

Convention	Example	Description
KEYWORDS	**SELECT**	A reserved word in DQL (case-insensitive).
parsing rules	*expr*	Name of a parsing rule. A parsing defines a fragment of DQL which consists of keywords, lexical rules, and other parsing rules.
LEXICAL RULES	*ID*	Name of a lexical rule. A lexical rule defines a fragment of DQL which consists of letters, numbers, or special characters.
[ ]	[**AS**]	Optional content.
[...]	[, *select_item*, ...]	Iterations of optional content.
|	**ASC** | **DESC**	Alternatives.
{ }	{ + | - }	Group of alternatives.
..	0..9	A range of allowable characters.

Select

```
select :
  subquery [ ORDER BY order_by_item [, order_by_item ...]]

subquery :
    subquery [ set_operator subquery ...]
  | ( subquery )
  | native_sql
  | subquery_base

subquery_base :
  SELECT [ DISTINCT ] select_item [, select_item ...]
  FROM from_clause_list
  [ WHERE condition ]
  [ GROUP BY expression_list
    [ HAVING condition ]
  ]

select_item :
  expr [  [ AS ] alias ]

alias :
  ID | QUOTED_ID
```

```
order_by_item :
  expr [ ASC | DESC ]

set_operator :
  UNION ALL | UNION | INTERSECT | EXCEPT

native_sql :
  NATIVE [ ( column_name [ column_type ] [ , ... ]  )  ]
  { sql_select }
```

Explanation:

- The { } around the sql_select are required and sql_select is an SQL query.
- The column_name and column_type specify parameters to pass from the SQL query to the DQL query.

FROM Clause

```
from_clause_list :
  { artifact_ref | subquery_ref | fixed_property | native_sql } [ from_clause_item ... ]

from_clause_item :
  , { artifact_ref | subquery_ref | fixed_property | native_sql }  |  [ LEFT [ OUTER
 ] ] JOIN { artifact_ref | subquery_ref } join_condition

artifact_ref :
  artifact_name [ alias ] [ ( artifact_modifiers ) ]

subquery_ref :
  ( subquery ) alias

fixed_property :
  property_ref alias

artifact_modifiers :
  ID [ , ID ... ]

artifact_name :
  ID

join_condition :
  | USING property_ref
```

Conditions

```
condition :
  condition_and [ OR condition_and ... ]
condition_and :
  simple_condition [ AND simple_condition ... ]

simple_condition :
  ( condition )
  | NOT simple_condition
  | exists_condition
  | like_condition
  | null_condition
  | in_condition
  | simple_comparison_condition
  | native_sql
  | bind

simple_comparison_condition :
  expr comparison_op expr

comparison_op :
  = | <> | < | > | <= | >=

like_condition :
  expr [ NOT ] LIKE like_expression [ ESCAPE STRING ]

like_expression :
  STRING
  | variable_ref

null_condition :
  expr IS [ NOT ] NULL

in_condition :
  expr [ NOT ] IN ( { subquery | expression_list } )
  | macro

exists_condition :
  EXISTS ( subquery )

bind :
  BIND ( property_ref [ , alias ] )

macro :
  macro_name [ ( expression_list ) ]

macro_name :
  #ID
```

Explanation:

- Conditions can be evaluated to true, false, or N/A. *condition* consists of one or more *condition_and* that are connected by the **OR** logical operator.
- *condition_and* consists of one or more *simple_condition* connected by the **AND**
- *simple_condition* is one of following:
  - *condition* in parentheses.
  - Negation of *simple_condition*.
  - *exists_condition*
  - *like_condition*
  - *null_condition*
  - *in_condition*
  - *simple_comparison_condition*
  - *native_sql*
- *simple_comparison_condition* is a comparison of two expressions using one of the comparison operators: =, <>, <, >, <=, >=
- *like_condition* compares an expression with a pattern. Patterns can contain wildcards:
  - **_** means any character (including numbers and special characters).
  - **%** means zero or more characters (including numbers and special characters).
  - **ESCAPE** *STRING* is used to prefix _ and % in patterns that should represent those characters and not the wildcard.
- *alias* references the target artifact.

Expressions

```
expr :
  term [  { + | - | CONCAT }  term ... ]
term :
  factor [  { * | / } factor ... ]

factor :
  ( select )
  | ( expr )
  | { + | - } expr
  | case_expression
  | NUMBER
  | STRING
  | NULL
  | function_call
  | variable_ref
  | property_ref
  | native_sql

case_expression :
  CASE
    case_item [ case_item ... ]
    [ ELSE expr ]
  END

case_item :
  WHEN condition THEN expr

function_call :
  ID ( [ DISTINCT ] { [ * ] | [ expression_list ] } )

property_ref :
   { ID | QUOTED_ID } [ { . | $ } { ID | QUOTED_ID } ... ]

expression_list :
  expr [, expr ... ]

variable_ref :
  ? | :ID
```

Explanation:

- Variables are of two kinds:
  - Positional variables - ? in DQL.
  - Named variables - :<name_of_variable>
- When variables are used in DQL, each variable must have a value bound to the variable.

Lexical Rules

```
CONCAT :
  ||

STRING :
  [ N | n ] ' text '

NUMBER :
  [ [INT] . ] INT

INT :
  DIGIT [DIGIT ... ]

DIGIT :
  0..9

ID :
  CHAR [ { CHAR | DIGIT } ... ]

CHAR :
  a..z | A..Z | _
```

Explanation:

- *ID* is sequence of characters, numbers and underscores beginning with a character or underscore.
- *QUOTED_ID* is text in quotes.
- *CONCAT* means a concatenation of strings - syntax ||

DQL with 3rd Party Products

DQL is provided by a JDBC driver which you can use with common SQL designers supporting 3rd-party JDBC drivers (or ODBC with an ODBC-JDBC bridge).

The following sections describe the driver and its use with 3rd party products:

- DQL JDBC Driver on page 60
- DQL in SQL Designers on page 61
- DQL in MS Access on page 61

DQL JDBC Driver

The DQL JDBC driver translates DQL queries into SQL queries and executes them using the underlying JDBC driver for the used database. The translation is provided by a remote invocation of HP SOA Systinet.

All the required JAR files for the DQL driver are available in `SOA_HOME/client/lib/jdbc`:

- `pl-dql-jdbc.jar`
- `hessian-`*`version`*`.jar`
- Database driver JAR files are copied here during installation (for example, `ojdbc6.jar`).

Table 5, "DQL JDBC Driver Configuration" describes the driver configuration required to use the driver with 3rd party products.

Table 5. DQL JDBC Driver Configuration

Property	Description
Connection String	jdbc:systinet:http(s)://<username>@<host:port>/<context>[|schema=schema name][|model=list of allowed models] • <username> is the HP SOA Systinet username who executes the DQL query using HP SOA Systinet permissions security. • <host:port> are the connection details of your HP SOA Systinet installation (for example, localhost:8080 for HTTP or secure:8443 for HTTPS. • <context> is the application server context, the default is `soa`. • |schema=schema name is optional as it is usually identified automatically, but you may require it if you used a non-default schema during DB configuration. • |model=list of allowed models is optional and represents a comma-separated list of models. Only artifacts from allowed models are provided in JDBC metadata as tables. The available models are `sys` and `public`. By default, only artifacts from the public model are provided.

Property	Description
	For example, `jdbc:systinet:http://admin@demoserver.acme.com:8080/soa|schema=SOA320`
DB Credentials	The user requires a DB username and password with read access for all tables for the HP SOA Systinet DB.
DQL JDBC Classname	com.hp.systinet.dql.jdbc.DqlDriver

The DQL JDBC driver must be able to connect to the database from the client. Use the full hostname for your database used during installation or setup. In the event of connection problems, verify the firewall settings between the local server and the database server.

DQL in SQL Designers

SQL Designer software can use the DQL driver if the designer is JDBC-aware.

To configure a JDBC-aware SQL Designer:

1. Add the DQL JDBC JAR files to the classpath.

2. Create a JDBC connection using the properties described in Table 5, "DQL JDBC Driver Configuration".

After you establish the DQL JDBC connection, the following functionality should be available in your SQL Designer:

- Schema introspection, browsing the list of artifact types and property groups as tables, and their properties as columns.

- DQL query execution.

DQL in MS Access

MS Access 2007 can execute DQL queries using an ODBC-JDBC bridge. Before using MS Access, you must configure the ODBC datasource in Windows.

To configure an ODBC-JDBC bridge:

1. Download and install an ODBC-JDBC bridge. For example, *Easysoft ODBC-JDBC Gateway*.

2 Configuration typically consists of:

   - JDBC driver configuration using the properties described in Table 5, "DQL JDBC Driver Configuration".
   - Bridge configuration. For details, see the documentation for the bridge software.

DQL syntax varies from the examples given in Introduction to DQL on page 39 in the following cases:

- Complex properties must use $ notation and be enclosed by [ ].

  ```
  personArtifact.[address$addressLines$value], personArtifact.[address$country]
  ```

- To use modifiers such as (include_deleted) use the Pass-Through option in MS Access.
- Left Joins do not work. Use plain joins instead.
- For fixed properties, use the Pass-Through option in MS Access.
- For timestamps, use the Pass-Through option in MS Access.
- Native queries do not work in MS Access.
- For property aliases, do not use quoted aliases.

5 Exploring the Database

The functionality of Report Editor relies on interaction with the database.

This chapter describes how to examine and setup the database in the following sections:

- SDM to Database Mapping Tool on page 63
- Adding a JDBC Driver on page 64
- Changing the Data Collection Settings on page 64
- Optimizing the Database on page 65
- Writing SQL Queries on page 65

SDM to Database Mapping Tool

Artifacts in HP SOA Systinet are stored in the form of XML documents. Their structure is defined by the SOA Definition Model (SDM). Artifacts are serialized into a database over a standard serialization layer. The serialization of data may differ from the norm, based on customer specific extensions or modifications.

The sdm2dbmap tool is a mapping tool that generates a report containing the mapping between your SDM and database tables.

To generate the report, execute the following command:

SOA_HOME/lib/sdm/bin/sdm2dbmap

The mapping report is output to the following file:

```
SOA_HOME/lib/sdm/build/sdm2dbmap.html
```

The output consists of the following parts:

- A top level 1:1 mapping between SDM artifacts and DB tables. Each artifact listed, maps directly to one table.
- A list of artifacts. Each artifact in the report maps each SDM property to a specific column in the table. There are also associated tables and foreign keys, joined using the primary key of the artifact table.
- A report documenting the DB schema for all database tables coming from the SDM. Tables with names ending in _Rev are used to store older revisions.

Adding a JDBC Driver

To connect to a database you must specify a JDBC driver.

To add a JDBC driver:

1. In the Database Connections dialog box, click **Manage Drivers**.

   The Manage JDBC Drivers dialog box opens.

2. Click **Add** to browse your filesystem for a driver.

   HP Software recommends using the same JDBC driver used during HP SOA Systinet installation.

3. Select the driver, and then click **Open** to add it to the list of drivers in the **Manage JDBC Drivers** dialog box.

Changing the Data Collection Settings

By default, the editor limits the amount of data collected to 20 schemas and 100 tables in the Query dialog box.

To change the data collection settings:

1. From the menu, select **Windows>Preferences**.
2. Expand **Report Design>Data Set Editor**, and then select **JDBC Data Set**.

3. Change the number of schemas and tables as required, and then click **OK**.

Optimizing the Database

If you frequently use particular fields for joins in your queries, create an index for that field to improve performance.

To create an Oracle index:

1. Log on to the sqlplus or isqlplus console as the system user.
2. Enter the following command:

   ```
   create index ix_docrel_optimize2 on ry_docRel(sourceId, rtype);
   ```

As a minimum optimization, HP Software recommends the following:

- **create index ix_res_optimize1 on ry_resource(fk_artifactType, m_deleted, id, m_path);**
- **create index ix_docrel_optimize2 on ry_docRel(sourceId, rtype);**

Writing SQL Queries

Although it is possible to use the query editor in Workbench, it is often more convenient to use a specialized tool to develop queries.

This section describes some typical methods and provides examples to help you design your queries.

These examples were created using Oracle Developer for use with an Oracle Database. Changes may be required to use these queries with other database types.

This section contains the following methods and examples:

- Joining Tables in Queries on page 66
- Example: Business Service SELECT on page 66

- Example: Contract SELECT on page 67
- Example: Combined Business Service and Contract SELECT on page 67
- Example: Accepted Contract Count for a Business Service on page 68

Joining Tables in Queries

To join tables in your queries, add the following to the WHERE clause of your SELECT statement:

- Key or foreign key condition
- Discriminator expression
- Check the deleted flag (0 means false)

Example:

```
WHERE ryga_bsnService.id = rygt_ctgryPty.fk_resource_id
    AND rygt_ctgryPty.discriminator = 'bsnPolicy_iBag'
    AND bsnRes.m_deleted = '0'
```

Example: Business Service SELECT

This example query returns details for business services that are marked as ready for consumption:

```
SELECT
    bsn.id bsn_id,
    bsn.name_val bsn_name,
    bsnRes.m_path bsn_path,
    bsn.serviceVersion_val bsn_version,
    bsn.description_val bsn_description,
    bsn.productionStage_name bsn_productionStage

FROM ryga_bsnService bsn
    INNER JOIN ry_resource bsnRes ON
    (
        bsnRes.id = bsn.id
        AND bsnRes.m_deleted = '0'
    )

WHERE bsn.readyForConsumption_val ='1'
ORDER BY lower(bsn.name_val)
```

Table ry_bsnService contains the main fields for business services.

Table >ry_resource contains general information about every artifact (for all artifact types).

The inner join to ry_resource in the FROM clause excludes deleted artifacts.

Example: Contract SELECT

This example query returns details for contracts that have been accepted:

```
SELECT
    contract.id ctrct_id,
    contract.name_val ctrct_name,
    contract.description_val ctrct_description,
    contract.contractstate_name ctrct_state,
    providerContractRel.targetpath ctrct_targetpath

FROM ry_contract contract
    INNER JOIN ry_docrel providerContractRel ON
    (
        providerContractRel.rtype = '{http://systinet.com/2005/05/soa/model/
property}providerContractor'
        AND providerContractRel.obsolete='0'
        AND providerContractRel.deleted='0'
        AND providerContractRel.targetDeleted='0'
        AND providerContractRel.sourceId = contract.id
    )
    INNER JOIN ry_resource contractRes ON
    (
        contractRes.id =contract.id
        AND contractRes.m_deleted='0'
    )

WHERE contract.contractState_val =
 'uddi:systinet.com:soa:model:taxonomies:contractAgreementStates:accepted'
```

Table ry_contract contains the main fields for contracts.

Table ry_docrel contains artifact relationship details.

The inner join to ry_docrel in the FROM clause excludes contracts where the provider-contract relationship is obsolete, deleted, or where the provider does not exist.

Example: Combined Business Service and Contract SELECT

Example: Business Service SELECT on page 66 and Example: Contract SELECT on page 67 can be combined to list the names of the contracts and the business service to which they apply.

```
SELECT
    contract.id ctrct_id,
    contract.name_val ctrct_name,
    bsn.id bsn_id,
    bsn.name_val bsn_name

FROM ryga_bsnService bsn
    INNER JOIN ry_resource bsnRes ON
    (
        bsnRes.id = bsn.id
        AND bsnRes.m_deleted = '0'
    ),
    ry_contract contract
    INNER JOIN ry_docrel providerContractRel ON
    (
        providerContractRel.rtype = '{http://systinet.com/2005/05/soa/model/
property}providerContractor'
        AND providerContractRel.obsolete='0'
        AND providerContractRel.deleted='0'
        AND providerContractRel.targetDeleted='0'
        AND providerContractRel.sourceId = contract.id
    )
    INNER JOIN ry_resource contractRes ON
    (
        contractRes.id =contract.id
        AND contractRes.m_deleted='0'
    )

WHERE bsn.readyForConsumption_val ='1'
    AND  contract.contractState_val =
 'uddi:systinet.com:soa:model:taxonomies:contractAgreementStates:accepted'
    AND providerContractRel.targetId = bsn.id
ORDER BY lower(bsn.name_val)
```

By merging the queries you create an n:1 relationship between contracts and business services.

Example: Accepted Contract Count for a Business Service

This example returns business service details and the number of accepted contracts for each service:

```
SELECT
    bsn.id bsn_id,
    bsn.name_val bsn_name,
    bsnRes.m_path bsn_path,
    bsn.serviceVersion_val bsn_version,
    bsn.description_val bsn_description,
    bsn.productionStage_name bsn_productionStage,
    ( -- Contracts --
    SELECT count (0)
    FROM ry_contract contract
        INNER JOIN ry_docrel providerContractRel ON
        (
            providerContractRel.rtype = '{http://systinet.com/2005/05/soa/model/
property}providerContractor'
            AND providerContractRel.obsolete='0'
            AND providerContractRel.deleted='0'
            AND providerContractRel.targetDeleted='0'
            AND providerContractRel.sourceId = contract.id
        )
        INNER JOIN ry_resource contractRes ON
        (
            contractRes.id =contract.id
            AND contractRes.m_deleted='0'
        )
    WHERE  contract.contractState_val =
 'uddi:systinet.com:soa:model:taxonomies:contractAgreementStates:accepted'
        AND providerContractRel.targetId = bsn.id
    ) ctrct_count

    FROM ryga_bsnService bsn
        INNER JOIN ry_resource bsnRes ON
        (
            bsnRes.id = bsn.id
            AND bsnRes.m_deleted = '0'
        )

WHERE bsn.readyForConsumption_val ='1'
ORDER BY lower(bsn.name_val)
```

6 Deploying Reports

When your report is ready, you must deploy it to the HP SOA Systinet server.

Report Editor offers two methods:

- Deploy a report directly to the HP SOA Systinet server.

  For details, see Deploying a Report to HP SOA Systinet on page 71.

- Deploy a set of reports as an extension to the HP SOA Systinet server.

  This process consists of the following steps:

  1. Creating a Reporting Extension on page 73
  2. Applying Extensions on page 73
  3. Redeploying the EAR File on page 78

After deployment, you can access your report in various ways.

For details, see Accessing Reports on page 78.

You can remove custom reports from the HP SOA Systinet server.

For details, see Removing Report Definitions from HP SOA Systinet on page 79.

Deploying a Report to HP SOA Systinet

The HP SOA Systinet plug-ins for Report Editor enable you to deploy your report directly to the HP SOA Systinet server.

To deploy a report to HP SOA Systinet:

1. Make sure that the HP SOA Systinet server is running.

2 In the Project Explorer or Navigator view, open the context menu for the report design you want to deploy, and select **HP SOA Systinet>Upload to Server**.

You will be asked if you want to publish the default library. If you do not want to be asked again, you can set this preference in the Preference menu of Report Editor.

Expand **Window>Preferences>HP SOA Systinet>Report Editor** and select whether you want Assertion Editor to prompt you every time you publish, always publish the default library, or never publish the default library.

Custom Policy Manager reports do not contain **Notify** functionality by default. To add this functionality you must add the report name to the notification context file.

To add notify functionality to custom PM reports:

1 Deploy your report to HP SOA Systinet as described above.

2 Open `SOA_HOME\deploy\hp-soa-systinet.ear\lib\pm-persistence.jar\META-INF\pmNotificationContext.xml` with a text editor.

3 Add a `value` element containing the report definition name to the file.

For example:

```
<bean id="pm-persistence.systemReportChecker"
          class="com.hp.systinet.policy.notification.SystemReportChecker">
  <property name="reports">
    <set>
      <!-- Systinet system reports - report definition IDs -->
      <value>acs_basic</value>
      ...
      <value>MyTestReport</value>
    </set>
  </property>
</bean>
```

4 Save `pmNotificationContext.xml`.

5 Restart the HP SOA Systinet server.

Creating a Reporting Extension

You can also deploy a set of reports as an extension.

To create a reporting extension:

1. In the Project Explorer, open the context menu for the project you want to deploy and select **HP SOA Systinet>Build Extension**.

   The Build Extension Wizard opens.

2. In the Build Extension Wizard, input the parameters you require.

   For parameter descriptions, see Build Extension Wizard on page 92.

3. Click **Finish** to create the extension.

Deploy the extension to the HP SOA Systinet server using the Setup Tool.

For details, see Applying Extensions on page 73.

Applying Extensions

You can extend HP SOA Systinet by adding libraries or JSPs to the deployed EAR files, by modifying the data model, by configuring the appearance of the UI, and by importing prepackaged data.

Extensions to HP SOA Systinet come from the following sources:

- **Customization Editor**

  Typical extensions created by Customization Editor contain modifications to the data model and artifact appearance, and possibly data required by the customization (taxonomies). They may also contain new web components, which may include custom JSP and Java code.

  ⚠ If your extension contains new artifact types, HP SOA Systinet does not create default ACLs for them. Set default ACLs for the new artifact types in HP SOA Systinet using the functionality described in "How to Manage Default Access Rights" in the *Administration Guide*.

- **Assertion Editor, Report Editor, and Taxonomy Editor**

  These extensions contain assertion, reporting, and taxonomy data only. They do not involve changes to the data model.

The Setup Tool opens the EAR files, applies the extensions, and then repacks the EAR files.

Apply extensions according to one of the following scenarios:

- Single-Step Scenario on page 75

  The Setup Tool performs all the processes involved in applying extensions, including any database alterations, as a single step.

- Decoupled DB Scenario on page 77

  Database SQL scripts are run manually. The Setup Tool performs the other processes as individual steps that are executable on demand. This scenario is useful in organizations where the user applying extensions does not have the right to alter the database, which is done by a database administrator.

In some specific circumstances (underscores and numbers in property names), extension application may fail because HP SOA Systinet cannot create short enough database table names (31 character maximum for most databases).

The error in `setup.log` resembles the following:

```
[java] --- Nested Exception ---
[java] java.lang.RuntimeException: cannot reduce length of identifier
  'ry_c_es_Artifact02s_c_priEspPty01Group_c_priEspPty01',
  rename identifier elements or improve the squeezing algorithm
[java] at com.systinet.platform.rdbms.design.decomposition.naming.impl.

 BlizzardNameProviderImpl.getUniqueLimitedLengthName(BlizzardNameProviderImpl.java:432
[java] at com.systinet.platform.rdbms.design.decomposition.naming.impl.
  BlizzardNameProviderImpl.filterTableName(BlizzardNameProviderImpl.java:374)
```

This is due to HP SOA Systinet using an older table naming algorithm in order to preserve backward compatibility with HP SOA Systinet 3.00 and older versions.

If you do not require backwards compatibility with these older versions, you can change the table naming algorithm.

To change the table naming algorithm:

1. Open `SOA_HOME/lib/pl-repository-old.jar#META-INF/rdbPlatformContext.xml` with a text editor.

2. In the `rdb-nameProvider` bean element, edit the following property element:

   <property name="platform250Compatible" value="false"/>

3. Save `rdbPlatformContext.xml`

This solution only impacts properties with multiple cardinality. If the problem persists or you need to preserve backwards compatibility, then review the property naming conventions in your extension.

Single-Step Scenario

Follow this scenario if you have permission to alter the database used for HP SOA Systinet.

To apply extensions to HP SOA Systinet in a single step:

1. Make sure that all extensions are in the following directory:

   `SOA_HOME/extensions`

   The Setup Tool automatically applies all extensions in that directory.

   ▶ If you are applying extensions to another server, substitute the relevant home directory for `SOA_HOME`.

2. Stop the server.

3. Start the Setup Tool by executing the following command:

   SOA_HOME/bin/setup.bat(sh)

4. Select the **Apply Extensions** scenario, and click **Next**.

The Setup Tool automatically validates the step by connecting to the server, copying the extensions, and merging the SDM configuration.

5 Click **Next** for each of the validation steps and the setup execution.

> This process takes some time.

6 Click **Finish** to end the process.

7 Deploy the EAR file:

- **JBoss**

  The Setup Tool deploys the EAR file automatically.

  If you need to deploy the EAR file to JBoss manually, see Redeploying the EAR File on page 78.

- **Other Application Servers**

  You must deploy the EAR file manually.

  For application server-specific details, see "Deploying the EAR File" in the *HP SOA Systinet Installation and Deployment Guide*.

8 Restart the server.

> Applying an extension that modifies the SDM model may drop your full text indices.
>
> `SOA_HOME/log/setup.log` contains the following line in these cases:
>
> ```
> Could not apply alteration scripts, application will continue with slower DB
> drop/create/restore scenario. ... .
> ```
>
> In these cases, reapply full text indices as described in the "Enabling Full Text Search" section of the *HP SOA Systinet Installation and Deployment Guide*.

Decoupled DB Scenario

Follow this scenario if the user who applies extensions does not have permission to modify the database.

To apply extensions and modify the database separately:

1. Make sure that all extensions are in the following directory:

   `SOA_HOME/extensions`

   The Setup Tool automatically applies all extensions in that directory.

2. Stop the server.

3. Start the Setup Tool by executing the following command:

   SOA_HOME/bin/setup -a.

4. Select the **Apply Extensions** scenario, and click **Next**.

5. Click **Next**, to execute the extension application, and exit the Setup Tool.

6. Provide the scripts from `SOA_HOME/sql` to the database administrator.

   The database administrator can use `all.sql` to execute the scripts that drop and recreate the database schema.

7. Execute the Setup Tool in command-line mode to finish the extension application:

   SOA_HOME/bin/setup -c

8. Redeploy the EAR file:

   - **JBoss**

     The Setup Tool deploys the EAR file automatically.

     If you need to deploy the EAR file to JBoss manually, see Redeploying the EAR File on page 78.

   - **Other Application Servers**

You must deploy the EAR file manually.

For application server-specific details, see "Deploying the EAR File" in the *HP SOA Systinet Installation and Deployment Guide*.

Redeploying the EAR File

After using the Setup Tool to apply extensions or updates, you must redeploy the EAR file to the application server. For JBoss, you can do this using the Setup Tool.

For other application servers, follow the EAR deployment procedures described in the "Deploying the EAR File" in the *HP SOA Systinet Installation and Deployment Guide*.

To redeploy the EAR file to JBoss:

1. Stop the application server.
2. Start the Setup Tool by executing the following command:

   SOA_HOME/bin/setup.bat(sh).
3. Select the **Advanced** scenario, and click **Next**.
4. Scroll down, select **Deployment**, and then click **Next**.

   When the Setup Tool validates the existence of the JBoss Deployment folder, click **Next**.
5. Click **Finish** to close the Setup Tool.
6. Restart the application server.

Accessing Reports

Use Cases on page 19 describes how HP SOA Systinet uses specific functionality in the UI for each report category.

It is also possible to directly access the reports in the HP SOA Systinet server and obtain alternative outputs.

In your browser, the following URLs use the Service Portfolio report as an example:

- `https://hostname:port/reporting/reports/`
- `https://hostname:port/reporting/reports/service_portfolio?alt=text/plain`
- `https://hostname:port/reporting/reports/service_portfolio/documents/`
- `https://hostname:port/reporting/reports/service_portfolio/documents/1/?alt=text/plain`

The last URL is an instance of the report. The page source contains alternate URLs. Change the URL to return a comma separated output of the report as follows:

- `https://hostname:port/reporting/reports/service_portfolio/documents/1/content?alt=text/csv`

Removing Report Definitions from HP SOA Systinet

You can remove custom report definitions from HP SOA Systinet.

To remove a single report, use the context menu option.

To remove custom reporting extensions:

1. Remove any custom report extensions from `SOA_HOME/extensions`.
2. Execute the Setup Tool using the apply extensions scenario as described, in "Setup Tool" in the *HP SOA Systinet Administrator Guide*.

7 Example: Failure Impact Report for the Reports Tab

To demonstrate the Reports Tab use case described in Use Cases on page 19, follow this example:

To create a Failure Impact Report and add it to the a Reports Tab:

1 Create the failure impact report definition.

Follow the procedure described in Creating a Report Definition on page 33 with the following inputs:

Parameter	Definition
Name	Failure Impact
Description	Summary of Business Service Failure Impact Status

2 Create the Failure Impact data set.

For details, see Example: Failure Impact Data Set on page 82.

3 Create the Failure Impact Report layout.

For details, see Example: Failure Impact Report Layout on page 84.

4 Deploy the Failure Impact Report to HP SOA Systinet.

For details, see Deploying a Report to HP SOA Systinet on page 71.

5 Add the Failure Impact Report to the HP SOA Systinet Reports Tab.

For details, see the "Birt Reports" section of the *HP SOA Systinet User Guide*.

Select **Failure Impact** from the list.

Example: Failure Impact Data Set

To demonstrate the reporting tool use case described in Creating a Data Set on page 35, follow this example.

To create the Failure Impact data set:

1 Follow the procedure described in Creating a Data Set on page 35 with the following inputs.

2 In the New Data Set dialog box, input the following parameters:

Parameter	DQL Option	SQL Option
Data Set Name	Failure Impact	
Data Source	dqlDatabase	Database
Data Set Type	DQL Select Query	SQL Select Query

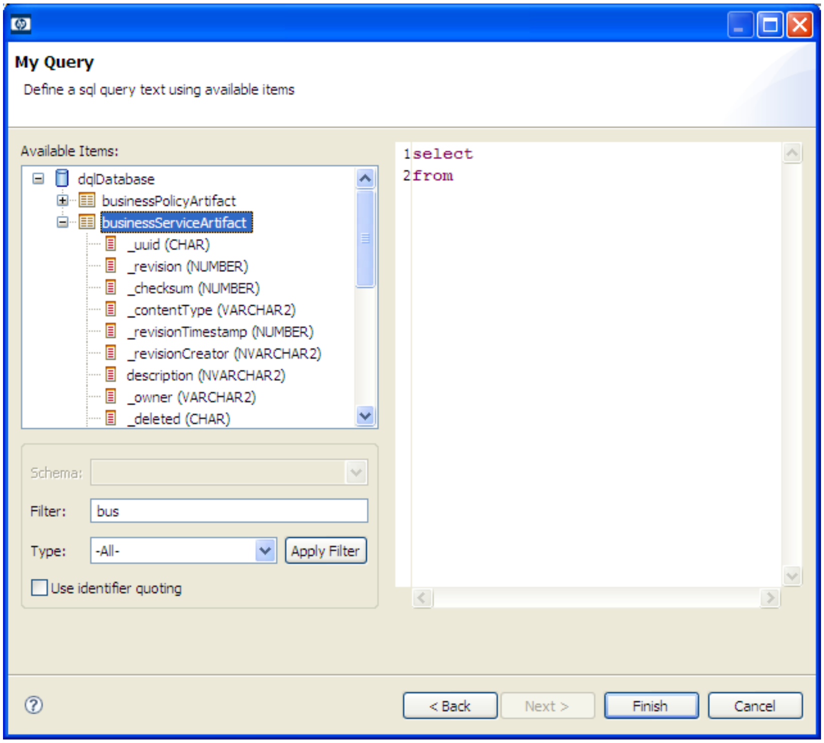
3 Use the filter to only display the data that you want to use.

For SQL queries, all standard tables in HP SOA Systinet begin with RYGA.

4 Use Available Items to explore the data and identify the tables and columns you need for your query.

For this report all the information required is for business services:

- For DQL, artifact type **businessServiceArtifact**.
- For SQL, table **RYGA_BSNSERVICE**.



My Query

Define a sql query text using available items

Available Items:

- dqlDatabase
  - businessPolicyArtifact
  - businessServiceArtifact
    - _uuid (CHAR)
    - _revision (NUMBER)
    - _checksum (NUMBER)
    - _contentType (VARCHAR2)
    - _revisionTimestamp (NUMBER)
    - _revisionCreator (NVARCHAR2)
    - description (NVARCHAR2)
    - _owner (VARCHAR2)
    - _deleted (CHAR)

Schema:

Filter: bus

Type: -All-

□ Use identifier quoting

```
select
from
```

5 Construct your query by typing in the query input area and dragging and dropping items from **Available Items**.

For this report the DQL query is:

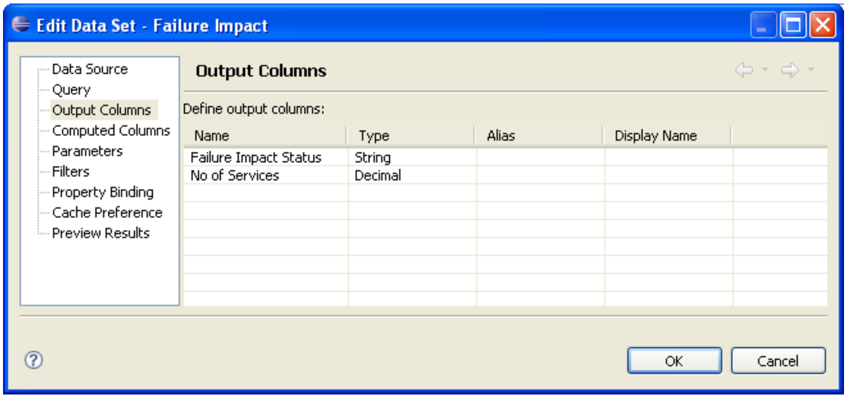
```
select bs.criticality.name as "Failure Impact Status",
            count (bs.criticality.name) as "No. of Services"
 from businessServiceArtifact bs
 group by bs.criticality.name
```

For this report the SQL query is:

```
select RYGA_BSNSERVICE.CRITICALITY_NAME as "Failure Impact Status",
            count(RYGA_BSNSERVICE.CRITICALITY_NAME) as "No of Services"
 from RYGA_BSNSERVICE
```

```
group by RYGA_BSNSERVICE.CRITICALITY_NAME
```

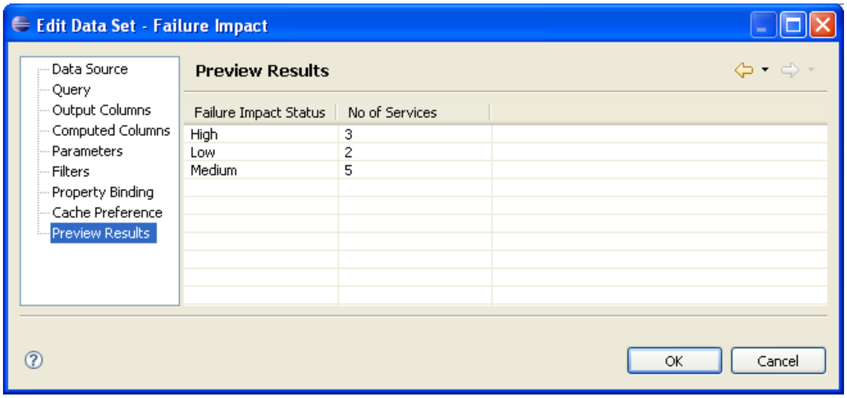
6 Click **Finish** to complete your query and view the Output Columns for the data set.



Define output columns:

Name	Type	Alias	Display Name
Failure Impact Status	String		
No of Services	Decimal		

7 Click **Preview Results** to view the actual data in your database.



Failure Impact Status	No of Services
High	3
Low	2
Medium	5

8 Click **OK** and press **Ctrl+S** to save the data set as part of the report.

Example: Failure Impact Report Layout

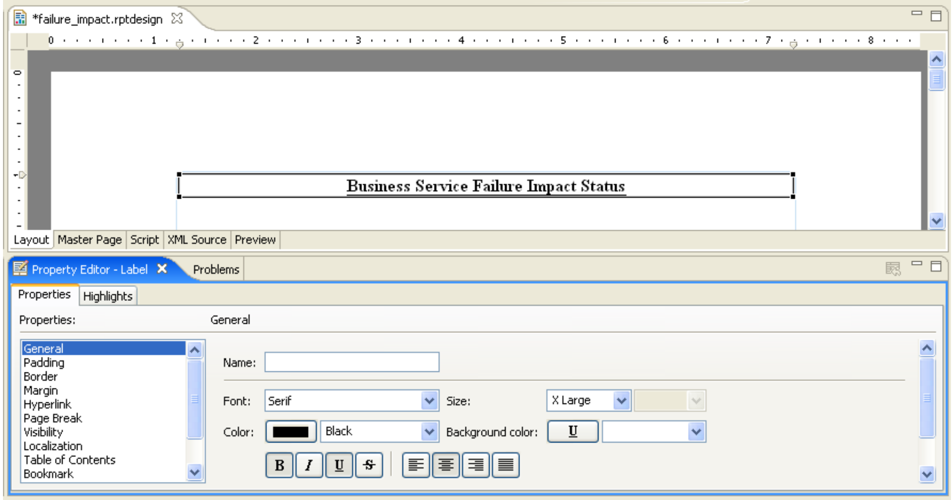
To demonstrate the reporting tool use case referred to in Designing a Report Layout on page 36, follow this example.

To Create the Layout of the Failure Impact Report:

1 In the failure_impact.rptdesign editor view, right-click and select **Insert>Label**.

2 Type the report heading `Business Service Failure Impact Status`.

3 Use the Property Editor view to change the format of the heading.



*failure_impact.rptdesign

Business Service Failure Impact Status

Property Editor - Label

Properties: General

Name:

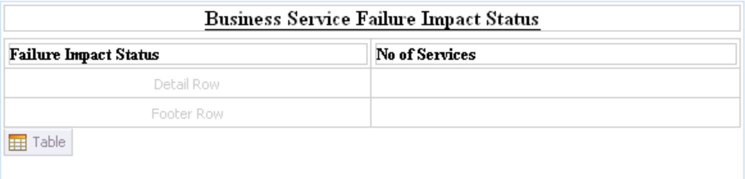
Font: Serif Size: X Large

Color: Black Background color:

4 In a blank area of the failure_impact.rptdesign editor view, right-click and select **Insert>Table**.

5 Input the table parameters and select the data set. In this case, 2 columns and the **Failure Impact** data set.

6 Select the left column header row, right-click and select **Insert>Text**. Input column header `Failure Impact Status`, and then click **OK**. Repeat for the right column header, `No of Services`.

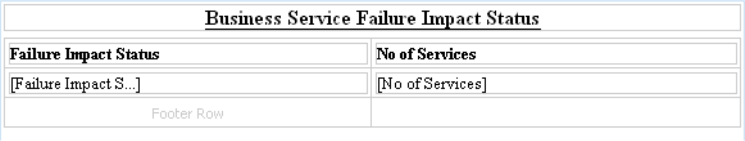


Business Service Failure Impact Status

Failure Impact Status	No of Services
Detail Row	
Footer Row	

7 In the Data Explorer, expand **Data Sets>Failure Impact**.

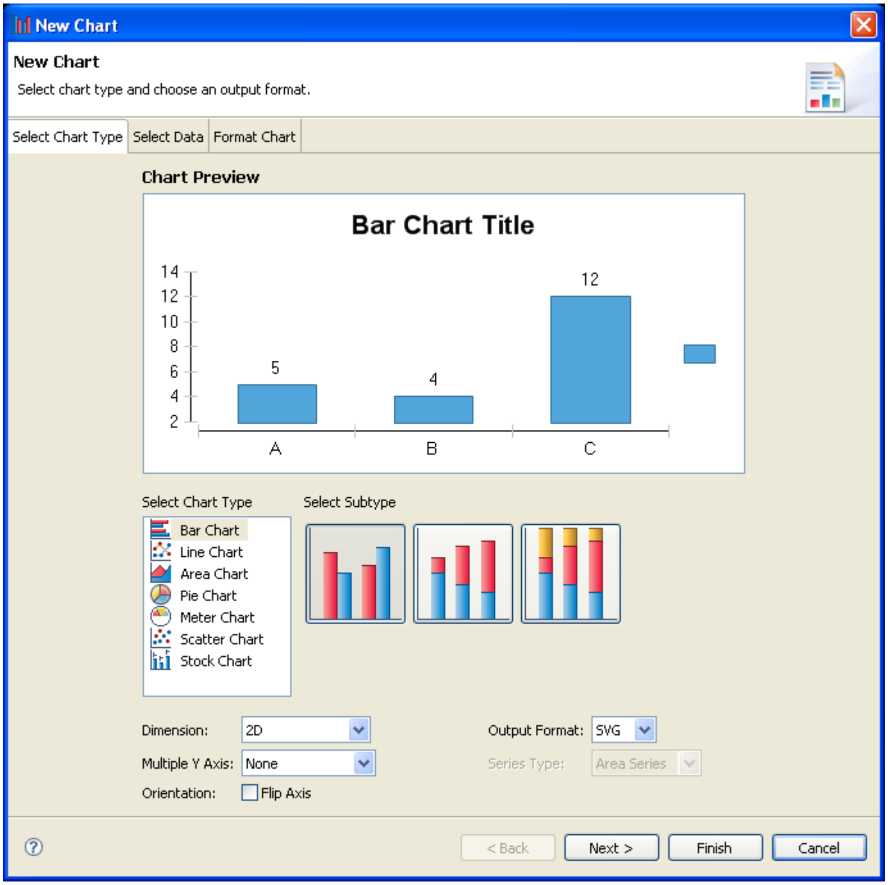
8 Drag **Failure Impact Status** from the Data Explorer to the left detail cell. Repeat for **No of Services** and the right detail cell.



Business Service Failure Impact Status

Failure Impact Status	No of Services
[Failure Impact S...]	[No of Services]
Footer Row	

9 In a blank area of the failure_impact.rptdesign view, right-click and select **Insert>Chart** to open the New Chart dialog box.



New Chart

Select chart type and choose an output format.

Select Chart Type | Select Data | Format Chart

Chart Preview

Select Chart Type

- Bar Chart
- Line Chart
- Area Chart
- Pie Chart
- Meter Chart
- Scatter Chart
- Stock Chart

Select Subtype

Dimension: 2D

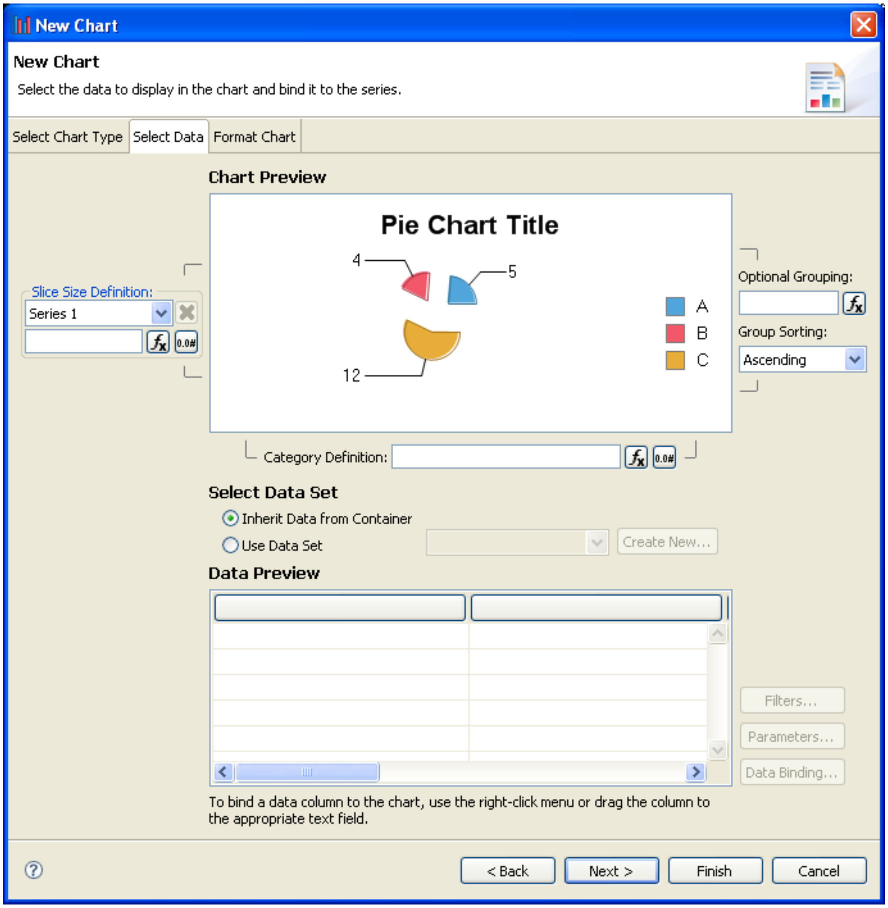
Multiple Y Axis: None

Orientation: □ Flip Axis

Output Format: SVG

Series Type: Area Series

10 This report requires a pie chart, select **Pie Chart**, and then click **Next**.



New Chart

Select the data to display in the chart and bind it to the series.

Slice Size Definition: Series 1

Group Sorting: Ascending

● Inherit Data from Container

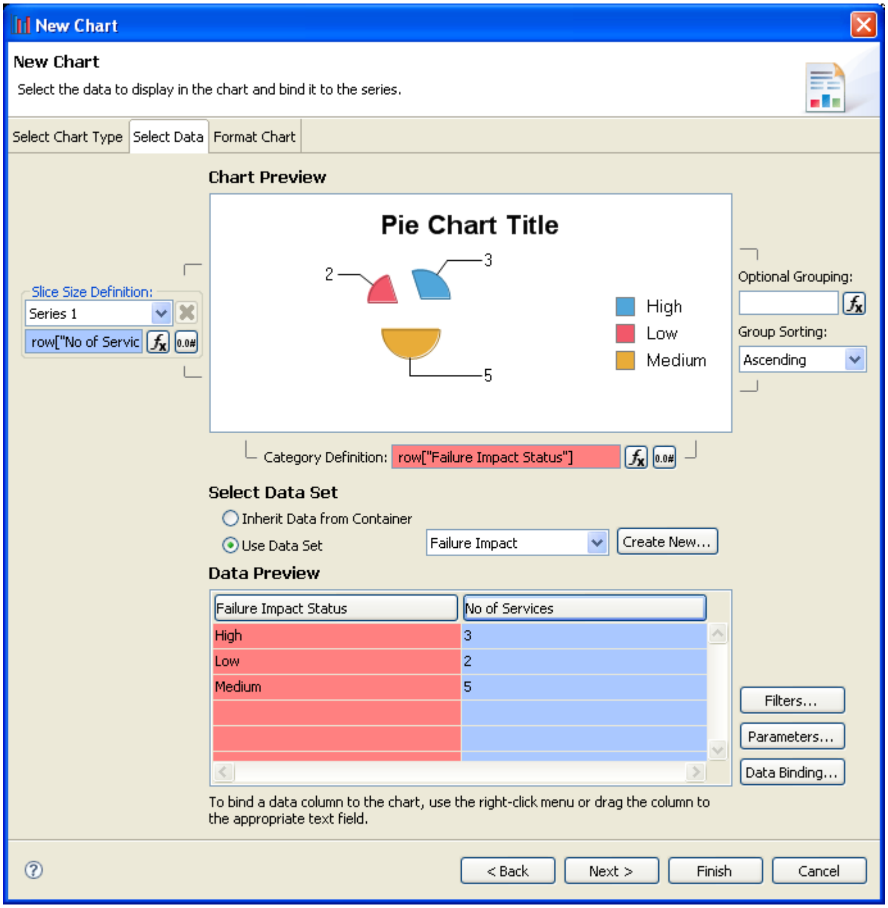
○ Use Data Set

To bind a data column to the chart, use the right-click menu or drag the column to the appropriate text field.

11 Under Select Data Set, select **Use Data Set** and **Failure Impact** from the drop-down list.

12 Drag **Failure Impact Status** from **Data Preview** to the **Category Definition** input.

13 Drag **No of Services** from **Data Preview** to the **Slice Size Definition** input.



New Chart

Select the data to display in the chart and bind it to the series.

Slice Size Definition: Series 1

row["No of Servic

Category Definition: row["Failure Impact Status"]

Optional Grouping:

Group Sorting: Ascending

Select Data Set

○ Inherit Data from Container

● Use Data Set Failure Impact Create New…

Data Preview

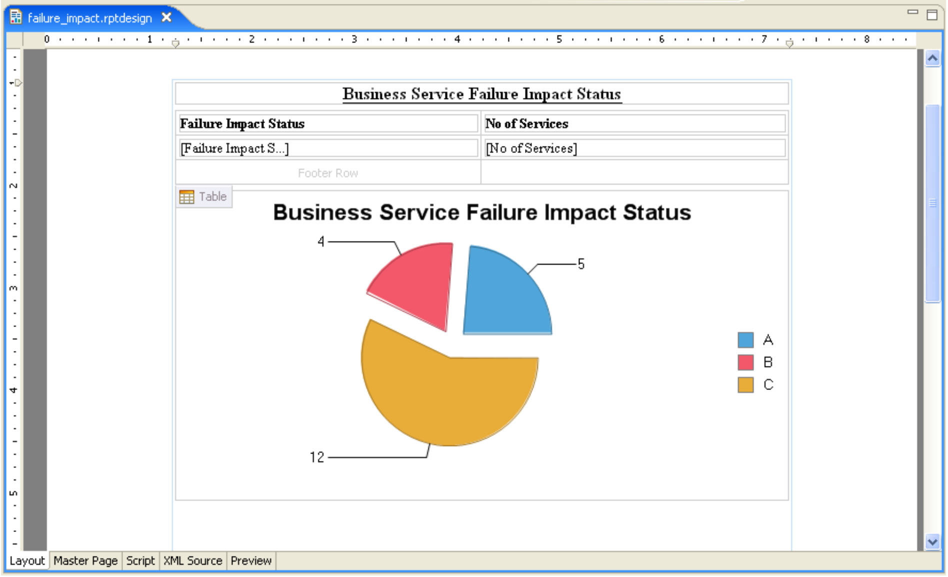
Failure Impact Status	No of Services
High	3
Low	2
Medium	5

To bind a data column to the chart, use the right-click menu or drag the column to the appropriate text field.

14 Click **Next** to format the chart appearance.

15 Select **Chart Area**, amend **Chart Title** to `Business Service Failure Impact Status`, and then click **Finish** to add the chart to the report definition.

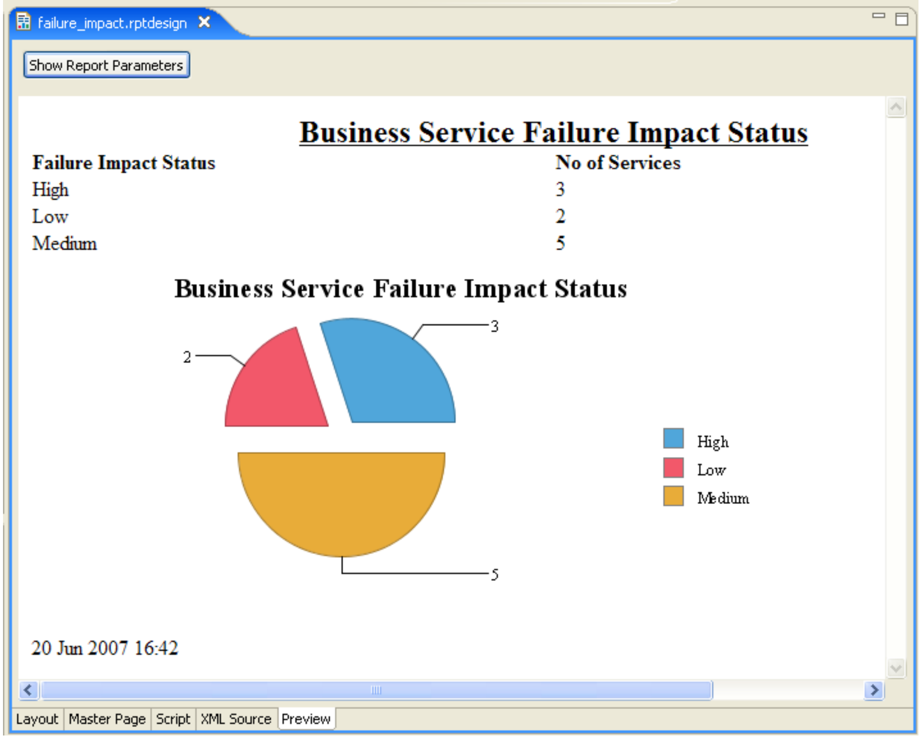
16 Drag the chart area outline to the size you require, and then press **Ctrl+S** to save the report definition.



Business Service Failure Impact Status

Failure Impact Status	No of Services
[Failure Impact S...]	[No of Services]

17 Select the **Preview** tab to view the report with data from your data source.



Business Service Failure Impact Status

Failure Impact Status	No of Services
High	3
Low	2
Medium	5

20 Jun 2007 16:42

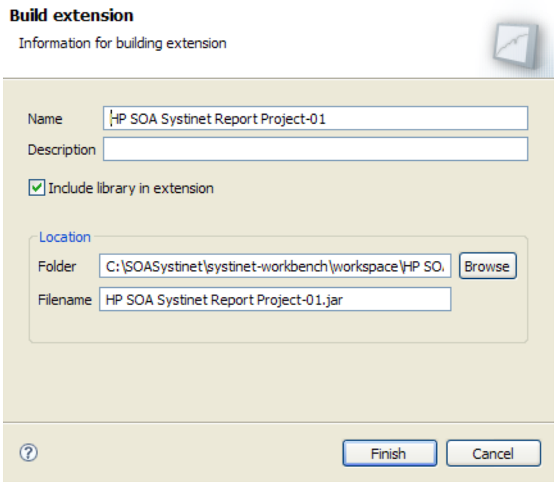
A Dialog Boxes

Each Report Editor input dialog is described in the following sections:

- Build Extension Wizard on page 92. Creating a report extension.
- New Data Set on page 93 Creating a new data set.
- New Report Project Wizard on page 94 Creating a new report project.

Build Extension Wizard

Enter general parameters for the report extension.



Build extension

Information for building extension

Name: HP SOA Systinet Report Project-01

Description:

☑ Include library in extension

Location

Folder: C:\SOASystinet\systinet-workbench\workspace\HP SO

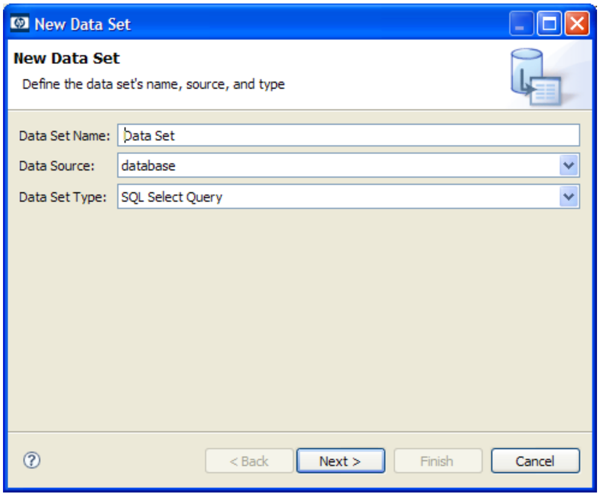
Filename: HP SOA Systinet Report Project-01.jar

Option	Definition
Name	The name of the reporting extension.
Description	A description for the reporting extension.

Option	Definition
Include library in extension	Select this check-box if you want to include the default library in the extension.
Folder	The location of the reporting extensions folder.
Filename	The filename of the reporting extension.

New Data Set

Enter a name, then enter a source and type for the new data set.



New Data Set

New Data Set

Define the data set's name, source, and type

Data Set Name: Data Set

Data Source: database

Data Set Type: SQL Select Query

Parameter	Definition
Data Set Name	Input a name for the data set.
Data Source	Select the data source from the drop-down list. The selection of a DQL or SQL datasource determines the exact query language required for your query.
Data Set Type	Choose one of the options from the drop-down list: • Select Query • Stored Procedure Query

New Report Project Wizard

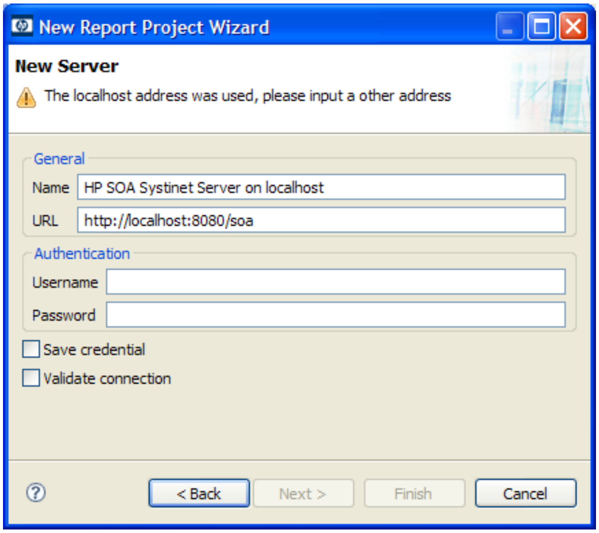
The New Report Project Wizard consists of the following stages depending on the options you select:

1 New Report Project Wizard: New Server on page 94

2 New Report Project Wizard: Default Library Configuration - Create a new data source on page 95

New Report Project Wizard: New Server

Create a new server for the project and optionally specify your authentication credentials.



New Report Project Wizard

New Server

⚠ The localhost address was used, please input a other address

General

Name: HP SOA Systinet Server on localhost

URL: http://localhost:8080/soa

Authentication

Username:

Password:

☐ Save credential

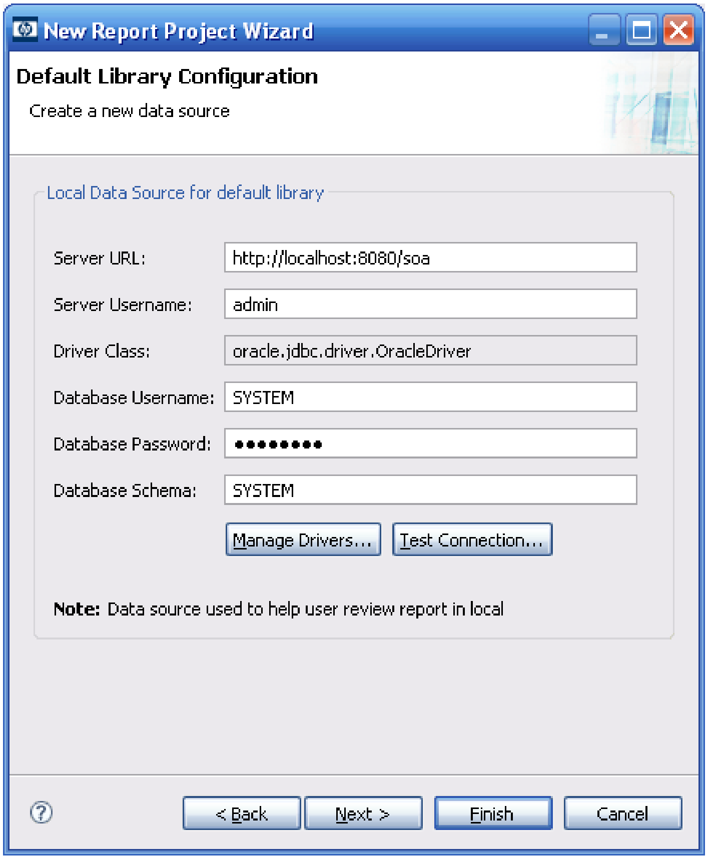
☐ Validate connection

Parameter	Definition
Name	The name of the server to display in the report editor.
URL	The location of the HP SOA Systinet server.
Username and Password	Credentials for access to the HP SOA Systinet server.
Save credentials	Select to store the input credentials.
Validate connection	Select to validate the server connection.

Parameter	Definition
	➤ The HP SOA Systinet server must be running to validate.

New Report Project Wizard: Default Library Configuration - Create a new data source

Configure the new data source and specify your login credentials.



New Report Project Wizard

Default Library Configuration

Create a new data source

Local Data Source for default library

Server URL: http://localhost:8080/soa

Server Username: admin

Driver Class: oracle.jdbc.driver.OracleDriver

Database Username: SYSTEM

Database Password: ●●●●●●●●●

Database Schema: SYSTEM

Note: Data source used to help user review report in local

Parameter	**Definition**
Server URL	The location of the HP SOA Systinet server.
Server Username	Your HP SOA Systinet login.
Driver Class	Select the driver class for your database from the drop-down list.
Database Username and Password	Your database credentials.
Database Schema	The database schema name.
Manage Drivers	Add, delete, or restore JAR files, and edit drivers.
Test Connection	Test the server connection using the specified parameters.

B Troubleshooting

This appendix describes some common problems and their resolutions:

Workbench Looks Different from the Pictures in the Documentation

Problem

Workbench does not look exactly as depicted in the user documentation. The most likely reason for this disparity is that you are in the wrong perspective.

Solution

Switch to the Platform perspective:

- From the menu, select **Window>Open Perspective>Platform**.

The Platform Perspective is Missing Views

Problem

There are views missing from the Platform perspective. The most likely reason for this is that some of the views are closed.

Solution

Reset the perspective:

- From the menu, select **Window>Reset Perspective**.

Incomplete Table List

Problem

The list of tables is incomplete. Workbench limits the number of schemas and tables accessed for performance reasons.

Solution

Change the settings, as described in Changing the Data Collection Settings on page 64.

Oracle Developer Queries Cause Errors in Workbench

Problem

Workbench returns errors for imported queries. The most likely reason is that your table names do not include the schema.

Solution

Include the schema name in the **FROM** clause of your query. For example, `soadb.ry_docrel`.

Workbench Displays Scrambled Table Names

Problem

The tables names are unreadable. These are most likely to be deleted tables stored in Oracle 10gR2.

Solution

Either change the maximum number of tables to 3000, as described in Changing the Data Collection Settings on page 64, or purge the deleted tables.

To purge deleted tables:

1. Log on to the sqlplus ot isqlplus console as the system user.
2. Enter the following commands:
   - **PURGE RECYCLEBIN**
   - **PURGE DBA_RECYCLEBIN**

The Report Result does not Appear

Problem

The report result is not deployed to the HP SOA Systinet server.

Solution

Check the report result:

1. In your browser, navigate to the latest report instance page. For example, `https://hostname:port/reporting/reports/service_portfolio/documents/1/?alt=text/plain`.
2. Check the following status elements:
   - `<category label="finished" scheme="urn:com:systinet:reporting:execution:status" term="finished"/>`
   - `<category label="report document" scheme="urn:com:systinet:reporting:kind" term="urn:com:systinet:reporting:kind:document"/>`

   A correctly finished report contains the *`finished`* value.

If the report execution does not finish, check the JMS settings, JMS queue, or the reporting service logs to analyze the bottleneck.

SQL Error on Report Previews

Problem

When viewing the Preview page of a report, the following SQL error is shown:

```
ReportDesign (id = 1):
+ Cannot get the result set metadata.
SQL statement does not return a ResultSet object.
SQL error #1: Invalid column name ROWNUM
```

This error occurs because the Preview is not aware of the database type that the query is applicable to.

Solution

Check the database type parameter:

1. In the Preview page, click **Show Report Parameters**.
2. Verify that the database-type parameter matches the database you use.

C Eclipse Plug-in Requirements

Ensure that the following prerequisite plug-ins are added to your Eclipse development platform based on the required Workbench components:

- DTP SDK 1.5

  Required for Customization Editor and Report Editor.

  http://www.eclipse.org/datatools/downloads.php

- EMF SDO Runtime 2.3.0

  Required for Customization Editor and Report Editor.

  http://www.eclipse.org/modeling/emf/downloads/

- GEF Runtime 3.3

  Required for Customization Editor and Report Editor.

  http://archive.eclipse.org/tools/gef/downloads/drops/R-3.3-200706281000/index.php

- WTP 2.0

  Required for Customization Editor and Report Editor.

  http://www.eclipse.org/webtools/releases/2.0/

- XSD Runtime 2.3

  Required for Customization Editor and Report Editor

  http://www.eclipse.org/modeling/mdt/downloads/?project=xsd

- BIRT 2.2.0

Required for Report Editor.

http://download.eclipse.org/birt/downloads/build_list.php