IBM WebSphere Service Registry and Repository Reporting with Business Intelligence and Reporting Tools

This IBM® Redpaper™ publication describes how to create reports for IBM WebSphere® Service Registry and Repository V7.0.0.2 using the Business Intelligence and Reporting Tools (BIRT). The reports created are designed for the WebSphere Service Registry and Repository governance enablement profile (GEP).

Introduction to Business Intelligence and Reporting Tools

WebSphere Service Registry and Repository has a reporting tool that is delivered as both an Eclipse plug-in that can be placed within any Eclipse environment and as an integrated part of the WebSphere Service Registry and Repository Studio tools. You can obtain the full documentation for the reporting tool in the WebSphere Service Registry and Repository Information Center at the following address:


The WebSphere Service Registry and Repository reporting tool is based on a powerful open source reporting system called Business Intelligence and Reporting Tools (BIRT). For more information, refer to the BIRT project website at the following address:

http://www.eclipse.org/birt/phoenix

We develop the reports in this paper by using the WebSphere Service Registry and Repository reporting tool. We have made all the reports available for you to download and use (see “Additional material” on page 89 for more information).
Introduction to the governance enablement profile

The governance enablement profile is a WebSphere Service Registry and Repository profile that provides a starting point to manage services from the initial specification through the deployment in production in a service-oriented architecture (SOA) environment. This profile is installed and activated by default during the deployment of WebSphere Service Registry and Repository.

The governance enablement profile consists of the following components:

- **Models**
  Entities that represent the objects in an organization’s SOA environment

- **Roles**
  A predefined set of roles into which users can be assigned according to their SOA tasks

- **Life cycles**
  A predefined set of states and transitions that are applied to artifacts to indicate their progress in the SOA governance process

For more information about the GEP, see *Service Lifecycle Governance with IBM WebSphere Service Registry and Repository*, SG24-7793.

WebSphere Service Registry and Repository reporting tool

We briefly describe the WebSphere Service Registry and Repository reporting tool and the BIRT capabilities.

There are six concepts that are critical for you to understand when developing a new report:

- **Data source**
- **Data set**
- **Joint data set**
- **Report library**
- **Report table**
- **Report chart**

We briefly explain these concepts. They are a guide to the capabilities that the WebSphere Service Registry and Repository reporting tool offers.

Data source

A report must specify a *data source*, which defines a connection to a specific WebSphere Service Registry and Repository server that will be queried to produce the report data. A report must specify one or more data sources; it is possible to use other data sources to correlate other information in a WebSphere Service Registry and Repository report.

One sort of data source called an XML data source processes XML representing data. In this XML data source, the connection to the WebSphere Service Registry and Repository server is specified in the data source settings as the URL to the WebSphere Service Registry and Repository REST API service. The XML data source defines a specific request for data from the WebSphere Service Registry and Repository server.
Data set

A data set represents a mapping of the result XML to rows and columns that are available to the report. The data set specifies which XML data source is used, so the query runs on the WebSphere Service Registry and Repository server.

The data set specifies which repeating structure in the XML represents a data item in the results, so each occurrence of this structure becomes a row in the data set, containing information about an entity in WebSphere Service Registry and Repository.

Joint data set

You can perform joins on the results of data sets to produce joint data sets. These joins are similar to joins that are used within SQL statements and can take the form of inner joins, left outer joins, right outer joins, or full outer joins.

Briefly, there are four join types:

- **Inner join** Only return rows where the join matches in both data sets.
- **Left outer join** Return all rows from the left data set even when there are not matching rows in the right data set.
- **Right outer join** Return all rows from the right data set even when there are not matching rows in the left data set.
- **Full outer join** Return all rows from both the left and right data sets regardless of whether there are matches.

A joint data set is an extremely powerful feature of reporting, because it allows the report writer to develop complex result sets that simple XPath queries do not allow.

Report library

A report library encapsulates a set of commonly used report data sources, data sets, functions, styles, and look-and-feels. A report library can be reused by many individual reports.

A report library simplifies maintenance of the reports that use it. The report library can be updated to reflect any changes the organization makes to WebSphere Service Registry and Repository servers, the structure of Business Models in WebSphere Service Registry and Repository, or the style that reports should use.

When you develop a new report, you can quickly reuse the components of a report library to speed development of the report. The reports created in this paper contain a report library.

Report table

After the data sets have been defined, you can create a report table to show the results of a specific data set in a tabular format.

The report tables consist of a header row, a data row, and a footer row. The table is associated to a data set, and then you can assign columns from the data set's results to cells within the data row of the table. When the report is generated, the data row of the table is expanded to multiple rows, one row for each of the returned results.
Additionally, you can create result groups. This function allows you to group result column values to improve the readability of the report.

We use report tables extensively within the reports that are designed within this paper.

**Report chart**

After the data sets have been defined, you can create a *report chart* to show the results of a specific data set in chart format. Many chart types are available, including bar charts and pie charts.

The Report Chart Properties window allows the assignment of various result columns to separate axes or series. Additionally, functions can be enacted on the result columns. For example, the count function counts how many times a result repeats so that the count can be used in a bar chart or pie chart.

**XML data source and the WebSphere Service Registry and Repository data source**

The WebSphere Service Registry and Repository reporting tool offers multiple data set types; among them are the XML data source and the WebSphere Service Registry and Repository data source.

In this paper, we use the XML data source to create the reports and report library. The WebSphere Service Registry and Repository data source can be used to create WebSphere Service Registry and Repository reports in a similar manner, as discussed in “Using the WebSphere Service Registry and Repository data source” on page 79. An added advantage of the XML data source is it enables us to use the WebSphere Service Registry and Repository ATOM APIs. By using the WebSphere Service Registry and Repository ATOM APIs, we could generate reports on business model systems, classification systems, or configuration items.

Additionally, the XML data source can be used to generate reports that use offline XML files containing captured WebSphere Service Registry and Repository data rather than querying the WebSphere Service Registry and Repository REST API, as discussed in “Reporting against XML files” on page 67.
Designing reports using joint data sets

We use joint data sets extensively within the reports that are designed within this paper. The XML data sets we create are generally of two types:

- A data set that contains all items of a certain type, for example, all business capabilities or all capability versions, as shown in Figure 1.

![Figure 1 Business capability and capability version data sets](image)

- A data set that contains the bsrURI of all the targets of a certain relationship for all items of a certain type in WebSphere Service Registry and Repository, and the bsrURI of the item that has the relationship. For example, we create a data set that, for all business capabilities in WebSphere Service Registry and Repository, contains the bsrURI of all the targets of the gep63_capabilityVersion relationship (its capability versions) and the bsrURI of the business capability that has that relationship, as shown in Figure 2.

![Figure 2 Data set containing one row per target of the gep63_capabilityVersion relationship](image)

We then use a joint data set to join the data set with details of an item, to the data set with information about a relationship. This results in a data set which has details of an item and the bsrURI of another related item, with one row per relationship target. For example, a business capability which has multiple capability versions would result in one row per capability version, with each row containing the details of the business capability and the bsrURI of a capability version.
Figure 3 shows how joining the Business Capability and Business Capability Version Relationship data sets, using the bsrURI of the business capability and an inner join, results in a new joint data set called Business Capability CVRef, where each row contains details about the business capability and the bsrURI of a capability version.

We then make another joint data set to join the data set with the bsrURI of the relationship target to the details of that relationship target. In the business capability example, we join the data set containing the business capability and capability version bsrURI to a data set containing details about the capability version. This results in a data set that has one row per capability version, with each row now containing the details of the business capability and the details of the capability version.
Figure 4 shows how joining the Business Capability CVRef data set and the Capability Version data set using an inner join results in a new joint data set called Business Capability Capability Version, where each row contains details of the business capability and the capability version. This data set could be used to create a report to show business capabilities and their capability versions.

The pattern of making joins can be repeated multiple times to add details of other related entities, for example, adding the service level definitions (SLDs), which are provided by a capability version.

It is not necessary to always use an inner join. If you want to keep items that do not have relationship targets, you can use an outer join. In the example in Figure 3 on page 6, you can use an outer left join with the Business Capability data set on the left, to keep all business capabilities, even if they do not have any capability versions and therefore no entries in the Business Capability Version Relationship data set.

Another technique we use is to create a data set which, for all items in WebSphere Service Registry and Repository, has a row containing a life cycle URI and the bsrURI of the item in that state. A WebSphere Service Registry and Repository item can only be in one life cycle state at one time. We join this item with a data set containing items of a particular type, resulting in a joint data set that has details about the item and the life cycle state that it is in.
Configuring WebSphere Service Registry and Repository connectivity in WebSphere Service Registry and Repository Studio

If your WebSphere Service Registry and Repository is secured using SSL, you need to perform some configuration to enable WebSphere Service Registry and Repository Studio to connect to WebSphere Service Registry and Repository.

Exporting the certificate from WebSphere Service Registry and Repository

To export the SSL certificate, perform the following steps using the WebSphere Application Server administrative console:

1. In the WebSphere Application Server navigation tree, click Security, click SSL certificates and key management, and then click Key stores and certificates.
2. In the Key stores and certificates window, click NodeDefaultKeyStore.
3. In the NodeDefaultKeyStore window, click Personal certificates.
4. In the Personal certificates window shown in Figure 5, check default, and then click Extract.

![SSL certificate and key management window](image)

*Figure 5  The Personal certificates window*
5. In the Extract personal certificate window shown in Figure 6, enter c:\temp\server.der in the File name field (for Windows®; for UNIX®, enter a path on the file system), and select Binary DER data in the Data type drop-down menu. Click OK.

The certificate will be extracted to the file system of the machine running WebSphere Application Server.

**Importing into the WebSphere Service Registry and Repository Studio key store**

To import the server certificate into WebSphere Service Registry and Repository Studio, copy the server.der certificate from the machine running WebSphere Application Server to the machine running WebSphere Service Registry and Repository Studio into c:\temp\server.der for Windows or a location on the file system for UNIX.

To perform the import into WebSphere Service Registry and Repository Studio, from a command prompt, change the directory to the Studio installation home, change the directory to JRE/bin, and run:

```
keytool -import -alias businessspacev7 -keystore ..\lib\security\cacerts -file c:\temp\server.der
```

The file parameter specifies the location of the certificate; alter this parameter to specify where you stored the certificate. The alias parameter specifies the name of the alias under which the certificate is stored. In this example, we use the name of our server, businessspacev7, as the alias. We recommend that you use the fully qualified host name of your WebSphere Service Registry and Repository server here.

The tool will prompt you for a password for the JVM keystore; when it does, enter changeit. The default key store password is “changeit”. The tool will then ask for confirmation to import; enter yes. An example import is shown in Example 1.

**Example 1  Importing a server certificate into Studio**

```
c:\wsrrStudio7002\Studio\jre\bin>keytool -import -alias businessspacev7 -keystore ..\lib\security\cacerts -file c:\temp\server.der
Enter keystore password: 
Owner: CN=businessspacev7, OU=businessspacev7Node01Cell, OU=businessspacev7Node01, O=IBM, C=US
```
Finally, restart WebSphere Service Registry and Repository Studio.

**Creating a report project**

Choose the Report Design perspective when working with reports so that all the correct views are available for use. You must specify a project within which the reports can be placed. The project can be either a new project created specifically to contain the reports, or an existing project that contains other assets. A report can then be created within the project.

Reports, report libraries, and the resources related to those reports are contained in a report project. To create a report project, perform these steps:

1. In WebSphere Service Registry and Repository Studio, click the + to open the Perspectives window. Click Report Design and click OK, as shown in Figure 7.
2. Select **File → New → Other**, as shown in Figure 8.

![Figure 8  Creating a report project](image)

3. Select **Business Intelligence and Reporting Tools → Report Project**, and click **Next**, as shown in Figure 9.

![Figure 9  Selecting Report Project](image)
4. Enter JKHLEnterprises_reports as the project name and click **Finish**, as shown in Figure 10.

![Figure 10 Entering the report project name](image)

After the project is created successfully, it is listed in the Navigator view, as shown in Figure 11.

![Figure 11 JKHLEnterprises project listed](image)

### Creating a report library

To create a report library, perform these steps:

1. Select **File → New → Folder**.
2. Click **JKHLEnterprises_reports** as the parent folder. Enter library in the Folder name field. Click **Finish**.
3. Select **File → New → Library**, as shown in Figure 12.

![Figure 12 Creating a report library](image)
4. Expand **JKHLEnterprises_reports**. Click **library** under the JKHLEnterprises_reports project as the parent folder, enter WSRRv7ReportLibrary.rptlibrary in the File name field, and click **Finish**, as shown in Figure 13.

![New Library window](image)

*Figure 13   The New Library window*

5. Click **OK** on the Library window that appears.

After the library is created successfully, it is listed in the Navigator view, as shown in Figure 14.

![Navigator view](image)

*Figure 14   The library shown in the navigator*

### Creating reports

In this section, we create some reports and populate the report library.

### Introduction to the case study

**JKHL Enterprises** is a fictitious supply company facing the typical challenges that arise when striving to reach the potential benefits of SOA solutions. JKHLE is an example that appears in other IBM materials.

JKHL Enterprises want to create a set of reports for use with the WebSphere Service Registry and Repository V7.0 governance enablement profile. They want to maximize maintainability and reuse for their reports and decide to use a report library to achieve this goal.
Service level agreement grouped by life cycle state

In this section, we create a report that lists service level agreement (SLA) entities in WebSphere Service Registry and Repository grouped by their life cycle state. The report should also group the SLAs by their last modified date, and display a count of the number of SLAs in each life cycle state.

Creating XML data sources in the report library
First, we create a data source that retrieves all SLAs in WebSphere Service Registry and Repository. Then we create a data source to retrieve all concepts from WebSphere Service Registry and Repository, so that later we can create a data set that obtains all the life cycle classifications from the concepts.

Data source to retrieve all SLA items
From the Report Design perspective, perform these steps:
1. From the Navigator view, expand library and double-click WSSRv7ReportLibrary.rptlibrary. The report library editor opens.
2. In the Data Explorer view, right-click Data Sources and click New Data Source, as shown in Figure 15.

Figure 15   Creating a new data source

3. In the New Data Source window, click XML Data Source and enter xml_SLA in the Data Source Name field, as shown in Figure 16. Click Next.

Figure 16   Entering values for the new data source
4. In the New XML Data Source Profile window, enter the URL shown in Example 2 in the 
   Enter the URL of the XML Source or browse to the file containing the data field, as shown 
   in Figure 17. The URL for the XML data source specifies the HTTP address of the 
   WebSphere Service Registry and Repository REST API, and specifies an XPath query 
   that retrieves all service level agreement items. For this paper, the WebSphere Service 
   Registry and Repository REST API is located on the host on which we used Studio, on 
   port 9082. You may need to change the host name and port to specify your WebSphere 
   Service Registry and Repository. You can enter the URL into a web browser and if 
   successful, you will see the XML response from the REST API.

**Example 2**  The URL for the xml_SLA data source

http://businessspacev7:9082/WSRR/7.0/Metadata/XML/GraphQuery?query=/WSRR/Generi 
cObject[classifiedByAnyOf(,'http://www.ibm.com/xmlns/prod/serviceregistry/pro 
file/v6r3/GovernanceProfileExtensions%23ServiceLevelAgreement')]

![New XML Data Source Profile](image)

**Figure 17**  Entering the URL for the XML data source

If you have security enabled on WebSphere Service Registry and Repository, whenever 
you specify a URL for an XML data source, you enter the URL to the HTTPS port where 
the WebSphere Service Registry and Repository REST API is located, using the host 
name. Click **Test Connection** and Studio will prompt for a user name and password the 
first time. The HTTPS URL for our WebSphere Service Registry and Repository is shown 
in Example 3.

**Example 3**  The HTTPS URL for the xml_SLA data source

https://businessspacev7:9445/WSRR/7.0/Metadata/XML/GraphQuery?query=/WSRR/Generi 
cObject[classifiedByAnyOf(,'http://www.ibm.com/xmlns/prod/serviceregistry/pro 
file/v6r3/GovernanceProfileExtensions%23ServiceLevelAgreement')]
For further information about WebSphere Service Registry and Repository query, refer to the WebSphere Service Registry and Repository Information Center found at the following address:

rvce_searchandquery02.html

5. Click **Test Connection**. Studio will connect to WebSphere Service Registry and Repository and display a success message, shown in Figure 18, if the connection succeeds.

![Figure 18 Successful connection message](image)

6. Click **Finish**. Select **File** → **Save** to save the changes to the report library.

After the XML data source is created successfully, it is listed in the Data Explorer view, as shown in Figure 19.

![Figure 19 The xml_SLA data source in the Data Explorer view](image)

**Data source to retrieve all concepts**

From the Report Design perspective, perform these steps:

1. From the Navigator view, double click `WSRRv7ReportLibrary.rptlibrary`. The report library editor opens.

2. In the Data Explorer view, right-click **Data Sources** and click **New Data Source**.

3. In the New Data Source window, click **XML Data Source** and enter `xml_GenericObject` in the Data Source Name field. Click **Next**.

4. In the New XML Data Source Profile window, enter the URL shown in Example 4 in the Enter the URL of the XML Source or browse to the file containing the data field. The URL for the XML data source specifies the HTTP address of the WebSphere Service Registry and Repository REST API, and specifies an XPath query that retrieves all Concept items.

   **Example 4 The URL for the retrieve all concepts data source**

   http://businessspacev7:9082/WSRR/7.0/Metadata/XML/GraphQuery?query=/WSRR/Generi
cObject

5. Click **Test Connection**. Studio will connect to WebSphere Service Registry and Repository and display a success message, shown in Figure 18, if the connection succeeds.

6. Click **Finish**. Select **File** → **Save** to save the changes to the report library.
After the XML data source is created successfully, it is listed in the Data Explorer view.

**Creating data sets in the report library**

First, we create a data set to map the SLA bsrURI, name and last modified date to report columns. Next, we create a data set to map the classifications on all concepts to the bsrURI of the concept they are on and the classification URI. The data set is filtered so only life cycle classifications are returned, so this data set will contain the life cycle states of all SLA items.

**Data set to map SLA properties to report columns**

This data set maps the SLA bsrURI, name and last modified date to report columns. From the Report Design perspective, perform these steps:

1. From the Navigator view, double-click `WSRRv7ReportLibrary.rptlibrary`. The report library editor opens.
2. In the Data Explorer view, right-click **Data Sets** and click **New Data Set**, as shown in Figure 20.

![Figure 20 Creating a new data set](image)

3. In the New Data Set window, click **xml_SLA** in the Data Source Selection field. Enter SLA in the Data Set Name field, as shown in Figure 21.

![Figure 21 Selecting the data source and data set name](image)

4. Click **Next**.
5. In the Sample XML Settings window, accept the defaults shown in Figure 22. Click **Next**.

![Figure 22 Sample XML Settings window](image)

6. In the Row Mapping window, click + next to resources, then click **resource**, as shown in Figure 23.

![Figure 23 Mapping rows in the XML](image)
7. Click >. The Select or edit the XPath expression window opens, as shown in Figure 24.

![Select or edit the XPath expression window]

Figure 24  The Select or edit the XPath expression window

8. Click OK. The Row Mapping window opens, as shown in Figure 25.

![Row Mapping window specifying the row XPath]

Figure 25  The Row Mapping window specifying the row XPath

The XPath expression specifies that each occurrence of the resource element within a parent element of resources will be a row in the report data set. In the WebSphere Service Registry and Repository REST API, each WebSphere Service Registry and Repository item returned by a graph query is contained in a resource element, within a parent resources element for all results. Therefore, a row in the data set maps to a WebSphere Service Registry and Repository item in the result XML.
For further information about the format of the WebSphere Service Registry and Repository REST response, see the WebSphere Service Registry and Repository Information Center at the following address:


9. Click Next.

In the Column Mapping window shown in Figure 26, the XML Structure section shows the structure of the results of running the query in the data source we used. It is possible to pick values from the XML Structure section and add them to the Column Mapping section on the right. However, we explicitly specify Column Mapping values.

![Column Mapping](image)

**Figure 26** The Column Mapping window

10. In the Column Mapping section, click Add....

11. In the Column Mapping window, enter bsrURI in the Column Name field, click String in the Data Type field and enter /@bsrURI in the XPath Expression field, as shown in Figure 27.

![Column Mapping](image)

**Figure 27** Mapping a single column to the bsrURI of the item

12. Click OK. This maps the bsrURI attribute on the XML to the data set column name bsrURI.
13. Repeat steps 10 to 12 for the values shown in Table 1.

Table 1  Values for column mappings for the SLA data set

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>XPath expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>/properties/property[@name='name']/@value</td>
</tr>
<tr>
<td>lastmodified</td>
<td>String</td>
<td>/properties/property[@name='lastModified']/@value</td>
</tr>
</tbody>
</table>

This action maps the values of the properties in the XML response with names *name* and *lastModified* to the specified columns.

14. In the Column Mapping window, click Show Sample Data. The Sample XML Data window opens, as shown in Figure 28. Click Close.

15. In the Column Mapping window, click Finish. The Edit Data Set - SLA window will open and show the data set, as shown in Figure 29.

16. Click OK. Select File → Save to save the changes to the report library.
After the XML data source is created successfully, it is listed in the Data Explorer view, as shown in Figure 30.

![Figure 30: The SLA data set in the Data Explorer](image)

**Data set to map concept life cycle classifications to report columns**

This data set contains the life cycle states of all WebSphere Service Registry and Repository entities with the bsrURI of the WebSphere Service Registry and Repository entity for each life cycle state. To create the data set, perform the following steps:

1. From the Navigator view, double click **WSRRv7ReportLibrary.rptlibrary**. The report library editor opens.
2. In the Data Explorer view, right-click **Data Sets** and click **New Data Set**.
3. In the New Data Set window, click **xml_GenericObject** in the Data Source Selection field. Enter **LifeCycleStates** in the Data Set Name field. Click **Next**.
4. In the Sample XML Settings window, accept the defaults. Click **Next**.
5. In the Row Mapping window, enter `resources/resource/classifications/classification` into the XPath Expression field. Click **Next**.
6. In the Column Mapping window, in the Column Mapping section, for each entry in Table 2, click **Add...** and enter the values.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>XPath expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject_bsrURI</td>
<td>String</td>
<td>../../@bsrURI</td>
</tr>
<tr>
<td>uri</td>
<td>String</td>
<td>/@uri</td>
</tr>
</tbody>
</table>

The XPath expression for the subject_bsrURI column specifies the bsrURI attribute from two levels above (using the .. notation). This is because we mapped each row in the data set to be an individual classification on an entity. The XPath virtual XML document structure for a single row is shown in Example 5. The current level is at the highlighted classification element.

**Example 5: The XPath document structure for a single row in the classification data set**

```xml
<resources>
  <resource bsrURI="value">
    <classifications>
      <classification uri="http://uri.com"/>
    </classifications>
  </resource>
</resources>
```

We can obtain the uri attribute at the current level with an XPath expression of `//@uri`, but to get the bsrURI of the entity, we have to traverse upward and get the bsrURI attribute from the resource element. Therefore, we specify `../..` to move upward two levels in the
XML hierarchy to the resource element, and then specify “@bsrURI” to get the value of the bsrURI attribute on the resource element.

7. In the Column Mapping window, click **Show Sample Data**. The Sample XML Data window opens, as shown in Figure 31. Click **Close**.

8. In the Column Mapping window, click **Finish**. The Edit Data Set - LifeCycleStates window opens and shows the data set.

Currently, the data set will return all classifications on concepts in WebSphere Service Registry and Repository. However, we only want the classifications that correspond to a state in the governance enablement profile life cycle system. To filter the data set, perform the following steps:

1. In the Edit Date Set - LifeCycleStates window, click **Filters**. This action displays the Filters panel on the right.

2. Click **New**.

3. In the New window, enter the expression shown in Example 6 into the Expression field. The expression returns -1 if the specified URI does not appear anywhere in the URI, or 0 or greater if it does.

   **Example 6** Filter expression to match on GEP life cycle states

   ```javascript
   BirtStr.indexOf('http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition', row["uri"])
   ```

4. Select **Greater than or Equal** in the Operation drop-down menu.
5. Enter 0 in the Value 1 field, as shown in Figure 32. The filter will only allow URIs that contain the GEP life cycle definition URI.

![Image of filter with expression: \text{Bir}5b: index0[http://www.ibm.com/arnhs/pn0/dserlces].](image)

\textbf{Figure 32} The LifeCycleState data set filter

6. Click **OK**.

The filter specifies that the URI returned in the uri row must start with the GEP life cycle URI to be allowed.

Finally, save the changes made to the data set and save the library:
1. In the Edit Data Set - LifeCycleStates window, click **OK**.
2. Select **File** → **Save** to save the changes to the report library.

After the XML data source is created successfully, it is listed in the Data Explorer view.

**Joining the data sets in the report library**

To report SLA items by life cycle state, we create a joint data set in the library between the SLA and the LifeCycleStates data sets called SLA_State. This joint data set needs to return rows where a row in the LifeCycleStates data set has a subject bsrURI that is in the bsrURI column of a row in the SLA data set, that is, it only returns SLA items that have a life cycle state. Therefore, it uses an inner join.

This will be the data set we use in the final report.

To create the joint data set, perform the following steps:
1. From the Navigator view, double-click \texttt{WSRRv7ReportLibrary.rptlibrary}. The report library editor opens.
2. In the Data Explorer view, right-click **Data Sets** and click **New Joint Data Set**, as shown in Figure 33.

![Figure 33 Creating a new joint data set](image)

3. In the New Joint Data Set window, click **SLA** in the first drop-down box. Click **LifeCycleStates** in the second drop-down box. Click **bsrURI** for the column for the SLA data set. Click **subject_bsrURI** for the column for the LifeCycleStates data set. Click **Inner Join** in the Join Types field. Enter SLA_State in the Data Set Name field. The completed window is shown in Figure 34. Click **Finish**.

![Figure 34 Setting the joint data set options](image)

4. In the Edit Data Set - SLA_State window, click **Output Columns**.
5. In the Define output columns: section, click **SLA::name** and click **Edit...**
6. In the Edit Output Column window shown in Figure 35, enter sla_name in the Alias field and enter SLA Name in the Display Name field. Click OK.

![Figure 35 Setting an alias for an output column](image)

7. Repeat steps 5 and 6 using the values shown in Table 3.

<table>
<thead>
<tr>
<th>Name</th>
<th>Alias</th>
<th>Display name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLA::lastmodified</td>
<td>sla_lastmodified</td>
<td>Last Modified Date</td>
</tr>
<tr>
<td>LifeCycleStates::uri</td>
<td>sla_state</td>
<td>State</td>
</tr>
<tr>
<td>SLA::bsrURI</td>
<td>sla_bsrURI</td>
<td>BsrURI</td>
</tr>
</tbody>
</table>

![Table 3 Values for output column for the SLA_State data set](image)

8. The completed window is shown in Figure 36. Click OK.

![Figure 36 The completed output columns for the SLA_State data set](image)

9. Select File → Save to save the changes to the report library.

**Creating the report**

Now that the data sets are created in the report library, we create the report to render the results. To create the report, perform the following steps:

2. In the New Report window, click the JKHLEnterprises_reports folder. Enter SLAByLifeCycleState.rptdesign in the File name field, as shown in Figure 37.

![New Report window](image)

Figure 37 New Report window

3. Click Finish. The report opens in the report editor and is listed in the Navigator, as shown in Figure 38.

![The report listed in the navigator](image)

Figure 38 The report listed in the navigator

Adding the report library to the report
Before the data sets defined in the report library can be used by the report, the report library must be added to the report. Perform the following steps:

1. From the navigator, double-click SLAByLifeCycleState.rptdesign.
2. Switch to the Outline view, as shown in Figure 39. If the Outline view is not shown, select Window → Show View → Outline.

![Figure 39 The SLAByLifeCycleState Outline view](image)

3. Select Library → Use Library, as shown in Figure 40.

![Figure 40 Using a library in a report](image)

4. In the Use Library window, expand library and click WSRRv7ReportLibrary.rptlibrary, as shown in Figure 41.

![Figure 41 Specifying the report library in the Use Library window](image)
5. Click **OK**. The report library is shown in the report Outline view, as shown in Figure 42.

![Figure 42](image1.png)  
Figure 42  The report library listed in the report Outline view

6. Select **File** → **Save** to save the changes to the report.

**Adding an external report library to the report**

You can create a report library in another project, and use it in the report. To do this task, you set the folder where the report library is stored as the project resource path. In our report project, this is unnecessary because the report library is stored inside the same project as the reports.

If you want to use a report in a different project, perform the following steps:

1. Select **Window** → **Preferences**. Expand **Report Design** and click **Resource**.
2. Click **Configure Project Specific Settings...**
3. In the Project Specific Configuration window, click **JKHLEnterprises_reports**, as shown in Figure 43. Click **OK**.

![Figure 43](image2.png)  
Figure 43  Selecting the report project for which to configure resources

4. In the Properties for **JKHLEnterprises_reports** (filtered) window, click **Enable project specific settings** and click **Select...**
5. In the Browse For Folder window, select the location where your report library is stored. Click **OK**.
6. In the Properties for **JKHLEnterprises_reports** (filtered) window, click **OK**.
7. In the Preferences window, click **OK**.
You then add the library to the project, as detailed in “Adding the report library to the report” on page 27.

**Note:** We chose to create our report library in the same project as the reports in this paper for the sake of simplicity. In the situation where multiple report projects need to use a single common library, we recommend locating the library in a stand-alone project and adding it as an external library to each report project.

Add the SLA_State data set to the report

To use the SLA_State data set in the report, you must first add it to the report. This action creates a reference to the data set in the library from the report; if the data set is updated in the library, the report will use the new data set.

To add the SLA_State data set to the report, follow these steps:

1. Switch to the Resource Explorer view, as shown in Figure 44. If the Resource Explorer view is not shown, select **Window → Show View → Resource Explorer**.

   ![Figure 44 The Resource Explorer view](image)

2. Expand **WSRRv7ReportLibrary.rptlibrary**.
3. Expand **Data Sets**.
4. Right-click **SLA_State** and select **Add to Report**. The data set is added to the report Outline, as shown in Figure 45.

   ![Figure 45 The SLA_State data set added to the report](image)

5. Select **File → Save** to save the changes to the report.
Add a table to the report
A table will render the report in rows and columns. To add a table to the report, perform the following steps:

1. Switch to the Palette view, as shown in Figure 46. If the Palette view is not shown, select Window → Show View → Palette.

![Figure 46 The Palette view](image)

2. In the Palette view, click **Table**.
3. In the report editor, click just inside the report page, as shown in Figure 47.

![Figure 47 Clicking in the report editor to add a table](image)

4. In the Insert Table window, enter 3 in the Number of columns field.
5. Enter 1 in the Number of details column.
6. Select **SLA_State** in the Data Set drop-down menu, as shown in Figure 48.

![Figure 48 The Insert Table window](image)
7. Click **OK**. The table is shown in the report editor, as shown in Figure 49.

![Figure 49 The SLA state report table](image)

8. Select **File** ➔ **Save**.

**Adding header labels**

A new table does not have any headings. To add the header labels, perform the following steps:

1. In the top left cell of the table, where it reads **Header Row**, right-click and select **Insert** ➔ **Label**.
2. Enter **State** into the label field.
3. In the Property Editor - Label view shown in Figure 50, click **General**. Click **B** to set the font to bold.

![Figure 50 The Property Editor - Label view for the State label](image)

4. Repeat steps 1 to 3 for the remaining two header cells, and enter **Last Modified Date** and **SLA Name** for the labels. The table is shown in Figure 51.

![Figure 51 The SLA state table with headers](image)

**Insert the SLA State group**

In order to group the results by the state of the SLA, you add a group to the table by performing the following steps:

1. Right-click the **Detail Row** cell in the table.
2. Click **Insert Group**.
3. In the New Group window, enter SLA_State_Group in the Name field.
4. Select sla_state in the Group On drop-down menu.
5. Accept the defaults. The completed window is shown in Figure 52.

![New Group window](image)

**Figure 52** The complete New Group window for the SLA_State_Group group

6. Click OK. The group is added to the table.
7. Select the table so the table scaffolding appears. Click the group 1 icon to the left of the group row, as shown in Figure 53.

<table>
<thead>
<tr>
<th>State</th>
<th>Last Modified Date</th>
<th>SLA Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[sla_state]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 53  Selecting the SLA_State_Group group in the table

8. The Property Editor - Row view shows properties for the row. Click **General**.

9. Click the button to the right of the Background color field. The Color window opens. Select a color and click **OK**.

10. In the Property Editor - Row view, click **B** and **I** to make the row bold and italic. The completed properties are shown in Figure 54.

![Property Editor - Row view for the SLA_State_Group row](image)

Figure 54  The Property Editor - Row view for the SLA_State_Group row

11. Select **File → Save**.

**Map the SLA state to messages**

By default, the SLA state column will show the URI of the state of the SLA entity. To show readable names for each state, we have to map each URI to a display string. To do this task, perform the following steps:

1. Click `[sla_state]` in the table.
2. In the Property Editor - Data view, click **Map** to switch to the Map tab.
3. Click **Add...**
4. In the New Map Rule window, select `sla_state` in the expression drop-down menu.
5. Select **Equal to** in the logic drop-down menu.
6. Enter the life cycle URI "http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLACreated" in the second expression field. Make sure you surround the URI with double quotes, because this syntax tells BIRT that the value is a string literal. Also ensure that the case is correct, because the match is case-sensitive.
7. Enter Created in the Then display the following value field, as shown in Figure 55.

![Figure 55 The SLA created state mapping rule](image)

8. Click OK.

Repeat steps 3 to 8 using the values given in Table 4. Make sure you surround the life cycle state URI with double quotes when you enter it into the field.

**Table 4 Values for the SLA state mapping**

<table>
<thead>
<tr>
<th>Life cycle state URI field</th>
<th>Display string</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLARequested">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLARequested</a></td>
<td>Requested</td>
</tr>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAINactive">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAINactive</a></td>
<td>Inactive</td>
</tr>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAActive">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAActive</a></td>
<td>Active</td>
</tr>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLATerminated">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLATerminated</a></td>
<td>Terminated</td>
</tr>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAREjected">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAREjected</a></td>
<td>Rejected</td>
</tr>
</tbody>
</table>
For the case that the SLA state is empty, add a mapping as follows:

1. Click **Add**.
2. In the New Map Rule window, select *sla_state* in the expression drop-down menu.
3. Click **Is Null** in the operator drop-down menu.
4. Enter Not Governed in the Then display the following value field.
5. Click **OK**. The completed Edit Properties - Data view is shown in Figure 56.

<table>
<thead>
<tr>
<th>Life cycle state URI field</th>
<th>Display string</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAIdentified">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAIdentified</a></td>
<td>Identified</td>
</tr>
</tbody>
</table>

6. Select **File → Save**.

**Insert the SLA last modified group**

To further group the results by the last modified date of the SLA, add a second group to the table by performing the following steps:

1. Right-click the **Detail Row** cell in the table.
2. Click **Insert Group**.
3. In the New Group window, enter **SLA_LastModified_Group** in the Name field.
4. Select *sla_lastmodified* in the Group On drop-down menu.
5. Accept the defaults. Click **OK**. The SLA_LastModified_Group group is inserted below the SLA_State_Group group.

---

**Figure 56**  The completed SLA state map
The last modified date is added into the first column by default. We want the last modified date to appear in the second column, under the Last Modified Date heading. To do this task, perform the following steps:

1. Click and drag \texttt{[sla\_lastmodified]} from the State column to the Last Modified Date column, in the same row, as shown in Figure 57.

![Table](image)

\textbf{Figure 57} \textit{The SLA state table with the last modified in the center column}

The last modified date is a digital time stamp and will currently display as a number. In order to display a readable date, alter the data binding for the column as follows:

a. Right-click \texttt{[sla\_lastmodified]}.

b. Click \textbf{Edit Value/Expression}....

c. In the Edit Data Binding window, select \textbf{Date} in the Data Type drop-down menu.

d. In the Expression field, enter \texttt{new Date(new Number(dataSetRow["sla\_lastmodified"]))}.

Alternatively, you can use the BIRT expression builder as follows:

a. In the Expression field, click \texttt{fx} to open the expression builder.

b. In the Expression Builder window, delete the text in the Expression field so it is empty.

c. Click \textbf{Native JavaScript Functions} in the Category field.

d. Click \textbf{Date} in the Sub-Category field.
e. Double-click **Date(milliseconds:number)** in the Double Click to insert field. The Expression field will have new `Date()` inserted into it and the cursor will be positioned inside the brackets, as shown in Figure 58.

![Expression Builder](image)

**Figure 58** Add a date expression in the Expression Builder window

2. Click **Native JavaScript Functions** in the Category field.
3. Click **Number** in the Sub-Category field.
4. Double-click **Number(value:Object)** in the Double Click to insert field. The Expression field will have new Number() inserted and the cursor will be positioned inside the brackets, as shown in Figure 59.

![Figure 59   Add a number expression in the Expression Builder window](image)

5. Click **Available Data Sets** in the Category field.
6. Click **SLA_State** in the Sub-Category field.
7. Double-click `sla_lastmodified` in the Double Click to insert field. The Expression Builder with the completed expression is shown in Figure 60.

![Figure 60 The completed Expression Builder window](image)

8. Click OK.

After the expression is complete in the Edit Data Binding window, as shown in Figure 61, perform the following steps:

1. Click OK.
2. Select File → Save.

![Figure 61 The completed data binding for SLA last modified](image)

**Adding an SLA count to the table**

The report should display a count of how many SLAs are in each life cycle state. To do this action, we add a count on the `SLA_State_Group` group so that a new total is created for each entry in the group. Perform the following steps:

1. Right-click **Group Footer Row (sla_state)**.
2. Select **Insert → Aggregation**, as shown in Figure 62.

![Figure 62](image)

3. In the Aggregation Builder window, select **Integer** in the Data Type drop-down menu.
4. Select **COUNT** in the Function drop-down menu.
5. Select **sla_bsrURI** in the Expression drop-down menu. The Expression field changes to read `row["sla_bsrURI"]`. 
6. In the Aggregate On field, click the Group radio button and select `SLA_State_Group` from the Group drop-down menu, as shown in Figure 63.

![Aggregation Builder](image)

*Figure 63 The SLA State aggregation*

7. Click **OK**. The aggregation is inserted into the table, as shown in Figure 64.

![Table](image)

*Figure 64 The count aggregation in the report table*

8. Select **File → Save**.

**Add the SLA name to the table**

Finally, we add the SLA name to the third column in the table by performing these steps:

1. In the Data Explorer view, expand **Data Sets → SLA_State**.
2. In the report editor, click the cell in the SLA Name column, in the Detail Row row, where the SLA Name will go. This will select the cell.
3. In the Data Explorer, right-click **SLA Name** and click **Insert in Layout**, as shown in Figure 65.

![Figure 65 Inserting the SLA Name into the table](image)

4. In the Edit Data Binding window, click **OK**.

5. Select **File** → **Save**.

**Running the report**

To run the report, in the Navigator view, right-click **SLAByLifeCycleState.rptdesign**. Select **Report** → **Run Report**. The results of the report are shown in Figure 66.

![Figure 66 The SLA by life cycle state report results](image)

**How to copy the life cycle state mappings**

When we added a mapping for the life cycle state, it was only added to the SLA by life cycle state report. In order to use the mapping in a different report, you have to re-enter the mapping. However, it is quicker if you copy the part of the report XML that specifies the mapping into a different report.

The XML that specifies a single mapping is shown in Example 7. The mappings are contained within a list-property element.

**Example 7 XML that specifies a single mapping**

```
<list-property name="mapRules">
  ...
  <structure>
    <expression name="testExpr">row["sla_state"]</expression>
    <property name="operator">eq</property>
    <simple-property-list name="value1">
      ...
    </simple-property-list>
  </structure>
</list-property>
```
In order to copy a mapping from a source report into a new report, perform the following steps:

1. In the new report, add a single mapping to a column using dummy values, for example, if "sentinel" Is Not Null then display "sentinel". This action will create an XML structure element.
2. In the new report editor, click XML Source to enter the XML view.
3. In the XML source, find the list-property element that contains the structure element containing the "sentinel" mapping you entered.
4. Delete the structure element under the list-property element.
5. In the source report, click XML Source to enter the XML view.
6. In the source report, find the mapping structure in the XML.
7. Copy all the structure elements that are inside the list-property element.
8. In the new report, paste the structure elements inside the list-property element.
9. Click Layout to return to the layout editor.

In Example 7 on page 43, the expression element specifies the sla_state row. If you copy a mapping into another report, ensure that the mapping uses the correct row name.

**Business capability by life cycle state grouped by type**

In this section, we create a report that lists business capabilities in WebSphere Service Registry and Repository grouped by their life cycle state. The report should also allow the viewer to select which type of business capability is shown, from business service, application, or process.

**Note:** You could improve performance of the report by creating a new data source that retrieves life cycle URIs for only business capabilities, rather than re-using the data source that retrieves life cycle URIs for all concepts.

**Creating XML data sources in the report library**

For this report, we create a data source that retrieves all business capabilities in WebSphere Service Registry and Repository. We can reuse the data source to retrieve all concepts from the WebSphere Service Registry and Repository installation we created earlier.

**Data source to retrieve all business capability items**

From the Report Design perspective, perform these steps:

1. From the Navigator view, expand library and double-click WSSRV7ReportLibrary.rptlibrary. The report library editor opens.
2. In the Data Explorer view, right-click Data Sources and select New Data Source.
3. In the New Data Source window, click **XML Data Source** and enter xml_BusinessCapabilities in the Data Source Name field. Click **Next**.

4. In the New XML Data Source Profile window, enter the URL in Example 8 in the Enter the URL of the XML Source or browse to the file containing the data field.

   **Example 8**  The URL for the xml_BusinessCapabilities data source

   http://businessspacev7:9082/WSRR/7.0/Metadata/XML/GraphQuery?query=/WSRR/Generi
ile/v6r3/GovernanceEnablementModel%23BusinessCapability')]

5. Click **Test Connection**. Studio will connect to WebSphere Service Registry and Repository and display a success message if the connection succeeds.

6. Click **Finish**. Select **File** → **Save** to save the changes to the report library.

After the XML data source is created successfully, it is listed in the Data Explorer view.

### Creating a report parameter in the report library

To allow the viewer to select the business capability type, we add a report parameter to the library that is later used to filter the results in a data set. Perform the following steps:

1. From the Navigator view, expand **library** and double-click **WSRRv7ReportLibrary.rptlibrary**. The report library editor opens.

2. In the Data Explorer view, right-click **Report Parameters** and select **New Parameter**.

3. Enter CapabilityType in the Name field.

4. Select **Radio Button** in the Display type drop-down menu.

5. Enter Select the type of business capability to report on in the Help text field.

6. In the List of value field, click **New...**

7. Enter http://www.ibm.com/xmlns/prod/serviceregistry/profile/v6r3/GovernanceEnablementModel#BusinessService in the Value field, as shown in Figure 67.

8. Click **OK**.

9. Click **Business Service** to select the row and click **Set as Default**.
10. Repeat steps 5 to 8 using the values given in Table 5.

<table>
<thead>
<tr>
<th>Display text</th>
<th>Value</th>
</tr>
</thead>
</table>

f. Select Display Text in the Sort by drop-down menu.

The completed New Parameter window is shown in Figure 68.

To save the new report parameter:
1. Click OK.
2. Select File → Save.
Creating data sets in the report library

We create a data set to map the business capability bsrURI, name, and various other properties to report columns. We also add a mapping for the owning organization bsrURI so that in future we can use the data set in a joint data set to get details of the owning organization.

We then use the report parameter to filter the results and only return the specified business capability type.

Data set to map business capability properties to report columns

Perform the following steps:

1. From the Navigator view, double-click `WSRRv7ReportLibrary.rptlibrary`. The report library editor opens.
2. In the Data Explorer view, right-click **Data Sets** and select **New Data Set**.
3. In the New Data Set window, click `xml_BusinessCapabilities` in the Data Source Selection field. Enter `BusinessServiceFilteredCapabilityType` in the Data Set Name field. Click **Next**.
4. In the Sample XML Settings page, accept the defaults. Click **Next**.
5. In the Row Mapping window, enter `/resources/resource` into the XPath Expression field. Click **Next**.
6. In the Column Mapping page, in the Column Mapping section, for each entry in Table 6, click **Add...** and enter the values.

Table 6: Values for column mappings for the `BusinessServiceFilteredCapabilityType` data set

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>XPath expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>bsrURI</td>
<td>String</td>
<td>/@bsrURI</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>/properties/property[@name='name']/@value</td>
</tr>
<tr>
<td>lastmodified</td>
<td>String</td>
<td>/properties/property[@name='lastModified']/@value</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>/properties/property[@name='description']/@value</td>
</tr>
<tr>
<td>namespace</td>
<td>String</td>
<td>/properties/property[@name='namespace']/@value</td>
</tr>
<tr>
<td>version</td>
<td>String</td>
<td>/properties/property[@name='version']/@value</td>
</tr>
<tr>
<td>ownerEmail</td>
<td>String</td>
<td>/properties/property[@name='ale63_ownerEmail']/@value</td>
</tr>
<tr>
<td>owner</td>
<td>String</td>
<td>/properties/property[@name='owner']/@value</td>
</tr>
<tr>
<td>owningOrganization_bsrURI</td>
<td>String</td>
<td>/relationships/relationship[@name='ale63_owningOrganization']/@targetBsrURI</td>
</tr>
<tr>
<td>primaryType</td>
<td>String</td>
<td>/properties/property[@name='primaryType']/@value</td>
</tr>
</tbody>
</table>
7. In the Column Mapping window, click **Finish**. The Edit Data Set - BusinessServiceFilteredCapabilityType window opens and shows the data set.

Currently, the data set will return all business capabilities in WebSphere Service Registry and Repository. However, we only want the capabilities of the type specified in the CapabilityType report parameter. Therefore, we add a filter to the data set by performing the following steps:

1. In the Edit Date Set - BusinessServiceFilteredCapabilityType window, click **Filters**. This shows the Filters pane on the right.
2. Click **New**....
3. In the New window, select **primaryType** in the Expression drop-down menu.
4. Select **Equal to** in the Operator drop-down menu.
5. Enter `params["CapabilityType"].value` in the Value 1 field, as shown in Figure 69.

![Figure 69 The new filter for the BusinessServiceFilteredCapabilityType data set](image)

6. Click **OK**.
7. In the Edit Date Set - BusinessServiceFilteredCapabilityType window, click **OK**.
8. Select **File → Save** to save the changes to the report library.

### Joining the data sets in the report library

To report business capability items by life cycle state, we create a joint data set in the library between the BusinessServiceFilteredCapabilityType and the LifeCycleStates data sets. This joint data set needs to only return rows where a row in the LifeCycleStates data set has a subject `bsrURI`, which is in the `bsrURI` column of a row in the business service data set, that is, only return business capability items that have a life cycle state. Therefore, it uses an inner join.

To create the joint data set, perform the following steps:

1. From the Navigator view, double-click **WSRRv7ReportLibrary.rptlibrary**. The report library editor opens.
2. In the Data Explorer view, right-click **Data Sets** and select **New Joint Data Set**.
3. In the New Joint Data Set window, click **BusinessServiceFilteredCapabilityType** in the first drop-down menu. Click **LifeCycleStates** in the second drop-down menu. Click **bsrURI** for the column for the BusinessServiceFilteredCapabilityType data set. Click **subject_bsrURI** for the column for the LifeCycleStates data set. Click **Inner Join** in the Join Types field. Enter BusinessCapabilityFilteredCapabilityType_State in the Data Set Name field. Click **Finish**.

4. In the Edit Data Set - BusinessCapabilityFilteredCapabilityType_State window, click **Output Columns**.

5. In the Define output columns section, edit the output columns to set their alias and display name as shown in Table 7.

<table>
<thead>
<tr>
<th>Name</th>
<th>Alias</th>
<th>Display name</th>
</tr>
</thead>
<tbody>
<tr>
<td>LifeCycleStates::uri</td>
<td>bc_state</td>
<td>State</td>
</tr>
<tr>
<td>BusinessServiceFilteredCapabilityType::bsrURI</td>
<td>bc_bsrURI</td>
<td>BsrURI</td>
</tr>
<tr>
<td>BusinessServiceFilteredCapabilityType::name</td>
<td>bc_name</td>
<td>Business Capability Name</td>
</tr>
<tr>
<td>BusinessServiceFilteredCapabilityType::lastmodified</td>
<td>bc_lastmodified</td>
<td>Last Modified Date</td>
</tr>
</tbody>
</table>

6. The completed window is shown in Figure 70. Click **OK**.

7. Select **File** → **Save**.

   **Creating the report**
   To create the report, perform the steps in “Creating the report” on page 26, and enter BusinessCapabilityStateFilter.rptdesign in the File name field.
Adding the report library to the report
Perform the steps in “Adding the report library to the report” on page 27 to add the report library to the BusinessCapabilityStateFilter.rptdesign report.

Add the report parameter to the report
We need to add the CapabilityType report parameter to the report so that it can be used to filter the business capability data set. Perform the following steps:

1. Switch to the Resource Explorer view. If the Resource Explorer view is not shown, select Window → Show View → Resource Explorer.
2. Expand WSRRv7ReportLibrary.rptlibrary.
4. Right-click CapabilityType and select Add to Report. The report parameter is added to the report.

Add the business capability data set to report
To add the filtered business capability data set to the report, perform the steps in “Add the SLA_State data set to the report” on page 30 to add the BusinessCapabilityFilteredCapabilityType_State data set.

Add a table to the report
To add a table to the report, perform the steps in “Add a table to the report” on page 31. In the Insert Table window, select BusinessCapabilityFilteredCapabilityType_State in the drop-down menu.

Add labels
To add labels to the table, perform the steps in “Adding header labels” on page 32 and add labels of State, Last Modified Date, and Business Capability Name, as shown in Figure 71.

Add the business capability state group
To group the report by state, perform the steps in “Insert the SLA State group” on page 32. Set the group name to BusinessCapability_State_Group. Select bc_state in the Group On drop-down menu.

Map the business capability state to messages
By default, the State column will show the URI of the state of the business capability. In order to show readable names for each state, we have to map each URI to a display string. In the GEP, a business capability goes through the capability life cycle, whereas an SLA goes through the SLA life cycle, so we cannot reuse the mapping for the SLA state.

To create the mapping for the capability life cycle state, perform the steps in “Map the SLA state to messages” on page 34 and edit the mapping for the [bc_state] cell, using the values
given in Table 8. For all mappings, select **bc_state** in the first expression drop-down menu and ensure that the URIs are surrounded by double-quotes.

### Table 8  Values for the business capability state mapping

<table>
<thead>
<tr>
<th>Life cycle state URI</th>
<th>Display string</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityIdentified">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityIdentified</a></td>
<td>Identified</td>
</tr>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityApproved">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityApproved</a></td>
<td>Approved</td>
</tr>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityDeprecated">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityDeprecated</a></td>
<td>Deprecated</td>
</tr>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityRetired">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityRetired</a></td>
<td>Retired</td>
</tr>
<tr>
<td><a href="http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityRejected">http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityRejected</a></td>
<td>Rejected</td>
</tr>
</tbody>
</table>

When you add a map for the case when the business capability state is empty, in the New Map Rule window, select **bc_state** in the expression drop-down menu.

**Add the business capability last modified group**

In order to further group the results by the last modified date of the business capability, you add a second group to the table. Perform the steps in “Insert the SLA last modified group” on page 36, enter `BusinessCapability_LastModified_Group` as the group name, and select **bc_lastmodified** in the Group On drop-down menu. Move the last modified date to appear in the second column. Ensure you alter the data binding to make the last modified date appear as a readable date.

**Add a business service count to table**

To add a count for the number of business services in each state, perform the steps in “Adding an SLA count to the table” on page 40. Add the aggregation to Group Footer Row (bc_state) and select **bc_bsrURI** in the Expression drop-down menu.

**Add the business capability name to the table**

Add the business capability name to the third column in the table by performing the steps in “Add the SLA name to the table” on page 42. Add the Business Capability Name column from the data set to the cell in the Business Capability Name column and the Detail Row row.
Add a title to show the capability type

Add a title to the report that shows which capability type the report is for. Perform the following steps:

1. In the report editor, right-click an empty part of the report and select Insert → Text.
2. In the Edit Text Item window, select HTML in the type drop-down menu.
3. Enter `<value-of>params["CapabilityType"].displayText</value-of> by life cycle state` into the value field, as shown in Figure 72. This expression inserts the display value of the CapabilityType report parameter into the text field.

![Edit Text Item window](image)

*Figure 72   Entering values for the report title*

4. Click OK.

The text field is added below the report table. To move it above the report table, drag and drop the text field. To improve the formatting of the text field, perform the following steps:

1. Click the text field.
2. In the Property Editor - Text view, click General.
3. In the General view, click B to set the text to bold.
4. Click Margin.
5. In the Margin view, enter 10 in the Bottom field, as shown in Figure 73.

![Property Editor - Text window](image)

*Figure 73   Setting the margin for the business service title text*

Running the report
To run the report, in the Navigator view, right-click BusinessCapabilityStateFilter.rptdesign. Select Report → Run Report. BIRT will prompt you for the capability type, as shown in Figure 74.

![Report prompt for capability type](image1)

Click OK. The results of the report are shown in Figure 75.

![The business capability by life cycle state report results](image2)
Add date filtering to the report
It is desirable to be able to filter the report by the last modified date of the business capability. In order to do this task, you add two report parameters to the report itself, then specify filtering on the last modified group in the table. You add the parameters to the report rather than the library so that you can specify a meaningful name for the parameters.

First, make a copy of the report by performing the following steps:
1. In the Navigator view, right-click BusinessCapabilityStateFilter.rptdesign and click Copy.
3. In the Name Conflict window, enter BusinessCapabilityStateDateFilter.rptdesign in the name field.
4. Click OK.

Add two new report parameters
To add the new report parameters, from the Data Explorer view, perform the following steps:
1. Right-click Report Parameters.
2. Click New Parameter.
3. In the New Parameter window, enter Capability Last Modified Start Date in the Name field.
4. Select Date in the Data type drop-down menu.
5. Select Text Box in the Display type drop-down menu.
6. Enter 2010-01-01 in the Default value field.
7. The completed values are shown in Figure 76. Click OK.

Repeat steps 1-7 for a parameter named Capability Last Modified End Date. Enter 2010-12-31 in the Default value field.

Figure 76 Completed values for the capability last modified start date parameter
**Add filtering to the BusinessCapability_LastModified_Group group**

To filter by the specified dates, add filtering conditions to the last modified group. Perform the following steps:

1. In the report editor, hover your mouse over the table and then click **Table** to select the table and display the table scaffolding, as shown in Figure 77.

![Figure 77 The selected table showing the scaffolding](image)

2. In the Property Editor - Table view, click **Groups** to switch to the Groups tab.
3. In the Groups tab, click **BusinessCapability_LastModified_Group**. Click **Edit...**
4. In the Edit Group window, in the Filters and Sorting field, click **Filters**.
5. Click **Add...**
6. In the New Filter Condition window, enter `new Date(new Number(dataSetRow["bc_lastmodified"]))` in the first filter condition field.
7. Select **Greater than or Equal** in the logic drop-down menu.
8. Enter `params["Capability Last Modified Start Date"].value` in the second filter condition field.
9. The completed filter is shown in Figure 78. Click **OK**.

![Figure 78 The completed start date filter expression](image)

The filter shows capabilities if their last modified date is after the date the user entered for the Capability Last Modified Start Date report parameter.
Repeat steps 5 to 9, enter `new Date(new Number(dataSetRow["bc_lastmodified"] - 86400000))` into the first filter condition field, select **Less than or Equal to** from the logic drop-down menu, and enter `params["Capability Last Modified End Date"].value` into the second condition field.

The second filter shows capabilities if their last modified date minus one day is before the date the user entered for the Capability Last Modified End Date report parameter. It is necessary to subtract one day from the capability last modified date because BIRT interprets the date that the user enters as the beginning of the day on that date, whereas the capability last modified date could be any time during the day of the displayed date.

In the Edit Group window, click **OK**. Select **File → Save**.

**Running the report with date filtering**

To run the report, in the Navigator view, right-click **BusinessCapabilityStateDateFilter.rptdesign**. Select **Report → Run Report**. BIRT will prompt you for the capability type and dates, as shown in Figure 79.

![Figure 79 Report prompt for capability type and dates](image.png)

Click **OK**.
The results of running the report with a start date of July 20th 2010 are shown in Figure 80. Note that the Eligibility service has been filtered out of the report because its last modified date is July 19th 2010.

<table>
<thead>
<tr>
<th>State</th>
<th>Last Modified Date</th>
<th>Business Capability Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved</td>
<td>Jul 22, 2010</td>
<td>Account creation business service</td>
</tr>
<tr>
<td></td>
<td>Jul 29, 2010</td>
<td>Ship Goods</td>
</tr>
<tr>
<td>Identified</td>
<td>Jul 31, 2010</td>
<td>Verify Credit</td>
</tr>
<tr>
<td>Rejected</td>
<td>Jul 31, 2010</td>
<td>Check customer for account service</td>
</tr>
<tr>
<td>Charter Review</td>
<td>Aug 1, 2010</td>
<td>Ordering Service</td>
</tr>
<tr>
<td></td>
<td>Aug 6, 2010 6:39 AM</td>
<td></td>
</tr>
</tbody>
</table>

Figure 80  Results of the business capability by life cycle state date filtered report

Note: There is not a date picker option in the version of BIRT in WebSphere Service Registry and Repository Studio, so users must enter a date string.

Business capability by capability version

This report shows business capabilities and their capability versions, with the capability version number.
To create the correct queries to retrieve the desired data, it is important to understand the entities and their properties and relationships in the GEP. This report uses the business capability and capability version, as shown in Figure 81.

![Figure 81 Governance enablement profile entities](image)

The report demonstrates how to create a data set with the desired data, when the entities in the final data set are related to each other by relationships.

Table 9 describes the new data sets we add to the report library.

<table>
<thead>
<tr>
<th>Data set name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessCapability</td>
<td>Business capabilities with important properties, such as name and bsrURI</td>
</tr>
<tr>
<td>CapabilityVersion</td>
<td>Capability versions with important properties</td>
</tr>
<tr>
<td>BusinessCapability_VersionRelationship</td>
<td>All the targets of the gep63_capabilityVersions relationship on all business capabilities, with the bsrURI of the business capability and the bsrURI of the capability version</td>
</tr>
</tbody>
</table>

We also create a new data source to return the capability version entities in WebSphere Service Registry and Repository.

Table 10 describes the new joint data set we will add to the report library.

<table>
<thead>
<tr>
<th>Joint data set name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessCapability_CVRef</td>
<td>Inner</td>
<td>Join of BusinessCapability and BusinessCapability_VersionRelationship on the bsrURI of the business capability</td>
</tr>
</tbody>
</table>
We chose a Left Outer join for the BusinessCapability_CapabilityVersion data set so that a business capability without capability versions would still appear in the report. To make the data set only show capabilities with one or more capability versions, set the join type to Inner.

Creating the XML data source in the report library
To create the data source in the report library, perform the steps in “Creating XML data sources in the report library” on page 44. Use the values given in Table 11 for the data source name and URL. Modify the URL to specify your WebSphere Service Registry and Repository server.

Table 11 Values for the new data source

<table>
<thead>
<tr>
<th>Data source name</th>
<th>URL</th>
</tr>
</thead>
</table>

Creating data sets in the report library
We create the data sets shown in Table 9 on page 58.

BusinessCapability data set
To create the business capability data set, perform the steps in “Data set to map SLA properties to report columns” on page 17. Use the values given in Table 12.

Table 12 Values for the BusinessCapability data set

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set name</td>
<td>BusinessCapability</td>
<td></td>
</tr>
<tr>
<td>Data source to use</td>
<td>xml_BusinessCapabilities</td>
<td></td>
</tr>
<tr>
<td>Row mapping XPath expression</td>
<td>/resources/resource</td>
<td>In the Row Mapping page of the New Data Set wizard</td>
</tr>
</tbody>
</table>

In the Column Mapping page, in the Column Mapping section, add mappings for the entries in Table 13.

Table 13 Values for column mappings for the BusinessCapability data set

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>XPath expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>bsrURI</td>
<td>String</td>
<td>/@bsrURI</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>/properties/property[@name='name']/@value</td>
</tr>
<tr>
<td>lastmodified</td>
<td>String</td>
<td>/properties/property[@name='lastModified']/@value</td>
</tr>
</tbody>
</table>
To create the capability version data set, perform the steps in “Data set to map SLA properties to report columns” on page 17. Use the values given in Table 14.

Table 14  Values for the CapabilityVersion data set

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set name</td>
<td>CapabilityVersion</td>
<td></td>
</tr>
<tr>
<td>Data source to use</td>
<td>xml_CapabilityVersion</td>
<td></td>
</tr>
<tr>
<td>Row mapping XPath expression</td>
<td>/resources/resource</td>
<td>In the Row Mapping page of the New Data Set wizard</td>
</tr>
</tbody>
</table>

In the Column Mapping page, in the Column Mapping section, add mappings for the entries in Table 15.

Table 15  Values for column mappings for the CapabilityVersion data set

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>XPath expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>String</td>
<td>/properties/property[@name='description']/@value</td>
</tr>
<tr>
<td>namespace</td>
<td>String</td>
<td>/properties/property[@name='namespace']/@value</td>
</tr>
<tr>
<td>version</td>
<td>String</td>
<td>/properties/property[@name='version']/@value</td>
</tr>
<tr>
<td>ownerEmail</td>
<td>String</td>
<td>/properties/property[@name='ale63_ownerEmail']/@value</td>
</tr>
<tr>
<td>owner</td>
<td>String</td>
<td>/properties/property[@name='owner']/@value</td>
</tr>
<tr>
<td>owningOrganization_bsrURI</td>
<td>String</td>
<td>/relationships/relationship[@name='ale63_owningOrganization']/@targetBsrURI</td>
</tr>
<tr>
<td>primaryType</td>
<td>String</td>
<td>/properties/property[@name='primaryType']/@value</td>
</tr>
</tbody>
</table>
To create the business capability related to capability version data set, perform the steps in “Data set to map SLA properties to report columns” on page 17. Use the values given in Table 16.

### Table 16  Values for the BusinessCapability_VersionRelationship data set

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set name</td>
<td>BusinessCapability_VersionRelationship</td>
<td></td>
</tr>
<tr>
<td>Data source to use</td>
<td>xml_BusinessCapabilities</td>
<td></td>
</tr>
<tr>
<td>Row mapping XPath expression</td>
<td>/resources/resource/relationships/relationship[@name='gep63_capabilityVersions']</td>
<td>In the Row Mapping page of the New Data Set wizard</td>
</tr>
</tbody>
</table>

In the Column Mapping page, in the Column Mapping section, add mappings for the entries in Table 17.

### Table 17  Values for column mappings for the BusinessCapability_VersionRelationship data set

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>XPath Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>businessCapability_bsrURI</td>
<td>String</td>
<td>.././@bsrURI</td>
</tr>
<tr>
<td>capabilityVersion_bsrURI</td>
<td>String</td>
<td>./targetBsrURI</td>
</tr>
</tbody>
</table>

**BusinessCapability_VersionRelationship**

Because there are multiple joint data sets, we do not specify any column aliases on any other than the data set used in the report.
**BusinessCapability_CVRef**

To create the BusinessCapability_CVRef joint data set, perform the steps in “Joining the data sets in the report library” on page 24. Use the values provided in Table 18 to specify the data set name, the two data sets that are joined, and the columns that are used for the join.

**Table 18  Values for the BusinessCapability_CVRef joint data set**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Name</td>
<td>BusinessCapability_CVRef</td>
</tr>
<tr>
<td>Join Type</td>
<td>Inner Join</td>
</tr>
<tr>
<td>Data set 1 name</td>
<td>BusinessCapability</td>
</tr>
<tr>
<td>Data set 1 column</td>
<td>bsrURI</td>
</tr>
<tr>
<td>Data set 2 name</td>
<td>BusinessCapability_VersionRelationship</td>
</tr>
<tr>
<td>Data set 2 column</td>
<td>businessCapability_bsrURI</td>
</tr>
</tbody>
</table>

The BusinessCapability_CVRef data set is shown in Figure 82.

**BusinessCapability_CapabilityVersion**

To create the BusinessCapability_CapabilityVersion joint data set, perform the steps in “Joining the data sets in the report library” on page 24. Use the values provided in Table 19 to specify the data set name, the two data sets that are joined, and the columns that are used for the join.

**Table 19  Values for the BusinessCapability_CapabilityVersion joint data set**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Name</td>
<td>BusinessCapability_CapabilityVersion</td>
</tr>
</tbody>
</table>
The BusinessCapability_CapabilityVersion data set is shown in Figure 83.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join Type</td>
<td>Left Outer</td>
</tr>
<tr>
<td>Data set 1 name</td>
<td>BusinessCapability_CVRef</td>
</tr>
<tr>
<td>Data set 1 column</td>
<td>BusinessCapability_VersionRelationship::capabilityVersion_bsrURI</td>
</tr>
<tr>
<td>Data set 2 name</td>
<td>CapabilityVersion</td>
</tr>
<tr>
<td>Data set 2 column</td>
<td>bsrURI</td>
</tr>
</tbody>
</table>

The BusinessCapability_CapabilityVersion data set is shown in Figure 83.

Because this is the data set that will be used in the report, we specify column mappings. Perform the steps in “Joining the data sets in the report library” on page 48 and specify the mappings in Table 20.

<table>
<thead>
<tr>
<th>Name</th>
<th>Alias</th>
<th>Display name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessCapability_CVRef::BusinessCapability::name</td>
<td>bc_name</td>
<td>Business Capability Name</td>
</tr>
<tr>
<td>BusinessCapability_CVRef::BusinessCapability::lastmodified</td>
<td>bc_lastmodified</td>
<td>Business Capability Last Modified</td>
</tr>
<tr>
<td>BusinessCapability_CVRef::BusinessCapability::bsrURI</td>
<td>bc_bsrURI</td>
<td>Business Capability BsrURI</td>
</tr>
</tbody>
</table>
Creating the report
To create the report, perform the steps in “Creating the report” on page 26 and enter BusinessCapabilityByCapabilityVersion.rptdesign in the File name field.

Adding the report library to the report
Perform the steps in “Adding the report library to the report” on page 27 to add the report library to the BusinessCapabilityByCapabilityVersion.rptdesign report.

Add the BusinessCapability_CapabilityVersion data set to report
To add the BusinessCapability_CapabilityVersion data set to the report, perform the steps in “Add the SLA_State data set to the report” on page 30 and add the BusinessCapability_CapabilityVersion data set.

Add a table to the report
To add a table to the report, perform the steps in “Add a table to the report” on page 31. In the Insert Table window, select BusinessCapability_CapabilityVersion in the drop-down menu, and enter 4 in the Number of columns field.

Add labels
To add labels to the table, perform the steps in “Adding header labels” on page 32 and add the labels Capability Type, Capability Name, Version Name, and Version Number, as shown in Figure 84.

<table>
<thead>
<tr>
<th>Name</th>
<th>Alias</th>
<th>Display name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessCapability_CVRef::BusinessCapability::primaryType</td>
<td>bc_primaryType</td>
<td>Business Capability Type</td>
</tr>
<tr>
<td>CapabilityVersion::name</td>
<td>cv_name</td>
<td>Capability Version Name</td>
</tr>
<tr>
<td>CapabilityVersion::version</td>
<td>cv_version</td>
<td>Capability Version Version</td>
</tr>
<tr>
<td>CapabilityVersion::bsrURI</td>
<td>cv_bsrURI</td>
<td>Capability Version BsrURI</td>
</tr>
<tr>
<td>CapabilityVersion::lastmodified</td>
<td>cv_lastmodified</td>
<td>Capability Version Last Modified</td>
</tr>
<tr>
<td>CapabilityVersion::versionTerminationDate</td>
<td>cv_terminationDate</td>
<td>Capability Version Termination Date</td>
</tr>
<tr>
<td>CapabilityVersion::versionAvailabilityDate</td>
<td>cv_availabilityDate</td>
<td>Capability Version Availability Date</td>
</tr>
<tr>
<td>CapabilityVersion::primaryType</td>
<td>cv_primaryType</td>
<td>Capability Version Type</td>
</tr>
</tbody>
</table>

Figure 84  Heads for the report

Insert the business capability type group
To group the report by business capability type, perform the steps in “Insert the SLA State group” on page 32. Set the group name to BusinessCapability_Type_Group. Select bc_primaryType in the Group On drop-down menu.
Map the business capability type to messages

By default, the Capability Type column will show the URI of the type of the business capability. In order to show readable names for each type, we have to map each URI to a display string.

To create the mapping for the capability type, perform the steps in “Map the SLA state to messages” on page 34, edit the mapping for the [bc_primaryType] cell and using the values given in Table 21. For all mappings, select bc_primaryType in the first expression drop-down menu and ensure the URIs are surrounded by double-quotes.

Table 21  Values for the business capability type mapping

<table>
<thead>
<tr>
<th>Type URI</th>
<th>Display string</th>
</tr>
</thead>
</table>

Insert the business capability name group

In order to further group the results by the name of the business capability, add a second group to the table. Perform the steps in “Insert the SLA last modified group” on page 36, enter BusinessCapability_Name_Group as the group name, and select bc_bsrURI in the Group On drop-down menu. Move the [bc_bsrURI] item to appear in the second column.

BIRT inserts a data item for the business capability bsrURI. We grouped on BsrURI because this is unique to a specific instance of a business capability, whereas the name is not. We need to change the data item to display the name. Perform the following steps:

1. Right-click [bc_bsrURI] and select Edit Value/Expression....
2. In the Edit Data Binding window, enter dataSetRow["bc_name"] in the Expression field, as shown in Figure 85.

   ![Figure 85](image.png)

   Figure 85  Editing the business capability bsrURI field to show the name

3. Click OK.
Add a count of capability versions to the table

To add a count for the number of capability versions for each business capability type, perform the instructions in “Adding an SLA count to the table” on page 40. Add the aggregation to Group Footer Row (bc_primaryType). Select cv_bsrURI in the Expression drop-down menu and enter BC_Aggregation in the Column Binding Name field.

Adding a label to the count

The count is currently shown as a plain number. You add a label to give the count some explanatory text by performing the following steps:

1. Right-click BC_Aggregation. Click Delete. This action removes the cell from the table but leaves the aggregation row in the project.

2. Right-click the cell Group Footer Row (bc_primaryType). Select Insert → Text.

3. Select HTML from the type drop-down menu.

4. Enter <value-of>row["BC_Aggregation"]</value-of> version(s) in the text field. This shows the value of the BC_Aggregation row and adds some text to the end.

5. Click OK.


Add the columns to the table

Next, add the last two columns to the table by performing the instructions in “Add the SLA name to the table” on page 42. Add the data in Table 22 to the detail row, where column 1 is the first column, going left to right.

| Table 22  Columns for the report table |
|---|---|---|
| Column name | Table column number | Data set column |
| Version Name | 3 | Capability Version Name |
| Version Number | 4 | Capability Version Version |
Running the report
To run the report, in the Navigator view, right-click **BusinessCapabilityByCapabilityVersion.rptdesign**. Select **Report → Run Report**. The results of the report are shown in Figure 86.

<table>
<thead>
<tr>
<th>Capability Type</th>
<th>Capability Name</th>
<th>Version Name</th>
<th>Version Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Application</strong></td>
<td>Footwear ordering application</td>
<td>Footwear ordering version</td>
<td>1.0</td>
</tr>
<tr>
<td>1 version(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Business Process</strong></td>
<td>Customer Care business process</td>
<td>Customer Care business process (1.0)</td>
<td>1.0</td>
</tr>
<tr>
<td>1 version(s)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Business Service</strong></td>
<td>Ship Goods</td>
<td>Ship Goods</td>
<td>2.0.1</td>
</tr>
<tr>
<td></td>
<td>Account creation business service</td>
<td>Account creation business service</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Account creation business service</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>Verify Credit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check customer for account service</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ordering Service</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eligibility service</td>
<td>Eligibility service (1.0)</td>
<td>1.0</td>
</tr>
<tr>
<td>4 version(s)</td>
<td>Aug 13, 2010 2:01 PM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 86  Results of the business capability by capability version report*

Reporting against XML files

In the XML data sources in the report library, we specified the URL of the WebSphere Service Registry and Repository REST API to retrieve the XML to generate the reports. However you can specify an XML file on the file system to use in an XML data source; the reports will then run against the XML contained in the XML file.

By querying the WebSphere Service Registry and Repository REST API once to retrieve and save the resulting XML per XML data source, then altering the XML data sources to use the XML file, you can produce a report library that does not need a running WebSphere Service Registry and Repository to produce reports.

First, you run the queries against the WebSphere Service Registry and Repository REST API and save the resulting XML to a directory on the file system on the machine where WebSphere Service Registry and Repository Studio runs, and then you make a copy of the report project and library. Finally, you alter the XML data sources to use the XML files.
Saving the WebSphere Service Registry and Repository REST XML

To save the WebSphere Service Registry and Repository REST XML, use Firefox or another web browser. Use the URLs from the XML data sources you created in “Creating reports” on page 13. The example URLs we gave are shown in Table 23; you should specify your host name and port.

Table 23  URLs and XML file names for the WebSphere Service Registry and Repository REST queries

<table>
<thead>
<tr>
<th>URL</th>
<th>File name</th>
<th>XML data source name</th>
</tr>
</thead>
</table>
xml_SLA.xml
xml_GenericObject.xml
xml_BusinessCapabilities.xml
xml_CapabilityVersion.xml | xml_SLA
xml_GenericObject
xml_BusinessCapabilities
xml_CapabilityVersion |

Perform the following steps in Firefox:

1. Select File → Open Location.
2. In the address bar, enter the URL from Table 23.
3. Press the Enter key.
4. After the XML is loaded, select File → Save Page As.
5. Navigate to a directory on the local file system where you will save the XML, for example, E:\work\xml.
6. In the File name field, enter the File name from Table 23 on page 68, as shown in Figure 87.

7. Click **Save**.

Repeat steps 1 to 7 for each URL in Table 23 on page 68.

**Copy the report project and library**

Next, we copy our report project, which contains the report library and the reports into another report project, so that the changes to the report library do not affect our existing reports.

If you are using a separate project to contain the report library, you need to copy just the report library into a new project, then change the resource location of your reports, as detailed in “Adding an external report library to the report” on page 29. This action demonstrates a benefit of using an external report library project.

To copy the reports and library, perform the following steps:

1. In the Navigator view, right-click **JKHLEnterprises_reports** and click **Copy**.
2. Right-click **JKHLEnterprises_reports** and click **Paste**.
3. In the Copy Project window, enter OfflineReports in the Project name field, as shown in Figure 88.

![Figure 88 Copying the JKHLEnterprises_reports project](image)

4. Click OK.

The report project is copied and the reports now use the report library in the new project.

**Set the XML data sources to use XML files**

Finally, we set the XML data sources in the report library to use the XML files rather than querying WebSphere Service Registry and Repository. Table 24 shows the XML data source name and the corresponding XML file name.

<table>
<thead>
<tr>
<th>XML data source name</th>
<th>XML file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml_SLA</td>
<td>xml_SLA.xml</td>
</tr>
<tr>
<td>xml_GenericObject</td>
<td>xml_GenericObject.xml</td>
</tr>
<tr>
<td>xml_BusinessCapabilities</td>
<td>xml_BusinessCapabilities.xml</td>
</tr>
<tr>
<td>xml_CapabilityVersion</td>
<td>xml_CapabilityVersion.xml</td>
</tr>
</tbody>
</table>

Perform the following steps in WebSphere Service Registry and Repository Studio:

1. Expand OfflineReports → library.
2. Right-click WSRRv7ReportLibrary.rptlibrary and select Open.
3. In the Data Explorer view, expand Data Sources.
4. Right-click xml_SLA and select Edit.
5. In the Enter the URL of the XML source or browse to the file containing the data field, click Browse.
6. In the Open window, navigate to the file system location where you saved the XML files in “Saving the WebSphere Service Registry and Repository REST XML” on page 68. For example, navigate to E:\work\xml.
7. Click xml_SLA.xml and click Open.
8. In the Edit Data Source - xml_SLA window, shown in Figure 89, click **OK**.

![Edit Data Source - xml_SLA](image)

Figure 89   The xml_SLA data source set uses an XML file

9. Repeat steps 4 to 8 for the other data sources listed in Table 24 on page 70.

10. Select **File** → **Save**.

    **Note:** You can optionally import the XML files into the report project and then specify the file system location of your workspace.

The reports that use the report library will now use the data in the XML files rather than querying the WebSphere Service Registry and Repository REST API. To run the reports, perform the steps in “Running the report” on page 43, “Running the report” on page 53, and “Running the report” on page 67.

**Using the sample reports in WebSphere Service Registry and Repository Studio**

The additional materials supplied with this paper (see “Additional material” on page 89) supply some sample reports.

**Capability by life cycle state filtered by type**
This report shows business capabilities grouped by their life cycle state, with their last modified date and name. The report prompts for a capability type and only shows business capabilities of that type.

The report is called BusinessCapabilityStateFilter.rptdesign.

**Capability by life cycle state filtered by type and last modified date**
This report shows business capabilities grouped by their life cycle state, with their last modified date and name.

The report prompts for a capability type and a start and end date, and only shows business capabilities of that type that have a last modified date that is between the two dates.
The report is called BusinessCapabilityStateDateFilter.rptdesign.

**Capability and capability versions grouped by type**
This report shows business capabilities grouped by their type (application, process, or service), then by each business capability, and then by name and version number of any service versions.

The report is called BusinessCapabilityByCapabilityVersion.rptdesign.

**Capability and capability versions grouped by type and filtered**
This report shows business capabilities grouped by their type (application, process, or service) then for each business capability, and then by the name and version number of any service versions.

The report prompts for a start and end date, and only shows capability versions that have a last modified date that is between the two dates.

The report is called BusinessCapabilityByCapabilityVersionDateFilter.rptdesign.

**Provider consumer grouped by organization**
To understand the structure of GEP entities used to create this report, refer to Figure 81 on page 58.

This report shows capability versions (the providers) that have an SLD that is subscribed to by an SLA from another capability version (the consumer), along with details of all consuming capability versions per each providing capability version. The report shows the provider capability version name, version, and contact email and the consuming capability version name, version, and contact email.

The report groups the provider capability versions by organization. It is called ProviderConsumer.rptdesign.

**Provisioning report**
The provisioning report is similar to the JKHLE provisioning report described in the JKHLE provisioning report in *Service Lifecycle Governance with IBM WebSphere Service Registry and Repository*, SG24-7793.

The report displays information about a service’s available service level definition (SLD) and the service level agreement (SLA) that are subscribed to it. The report shows business capabilities, their capability versions, and the SLD provided. For each SLD, it shows the SLAs using the SLD and various properties of each SLA. It then sums the maximum messages per day of all SLAs that use an SLD.

The report is called ProvisioningReport.rptdesign.

**SLA by life cycle state**
This report shows SLAs grouped by their life cycle state, with their last modified date and name.

The report is called SLAByLifeCycleState.rptdesign.

**SLA by life cycle state filtered by last modified date**
This report shows SLAs grouped by their life cycle state, with their last modified date and name.
The report prompts for two dates and only shows SLAs that have a last modified date between the two dates.

The report is called SLAByLifeCycleStateDateFilter.rptdesign.

**SLD by life cycle state and their endpoints with life cycle state**
This report shows SLDs grouped by their life cycle state, with their service endpoints sorted by the state of the endpoint. The report shows the SLD name, and the endpoint name, type, and state.

The report is called SLDByLifeCycleStateWithEndpoints.rptdesign.

**SLD by life cycle state and their endpoints with life cycle state filtered**
This report shows SLDs grouped by their life cycle state, with their service endpoints sorted by the state of the endpoint. The report shows the SLD name and last modified date, and the endpoint name, type, and state.

The report prompts for a start and end date and will only show SLDs that have a last modified date between the two dates.

The report is called SLDByLifeCycleStateWithEndpointsDateFilter.rptdesign.

**Schema specification by life cycle state and version**
This report shows schema specifications grouped by their life cycle state. It then groups by their version and shows the schema specification name, version, and namespace.

The report is called SchemaSpecificationByLifeCycleStateAndVersion.rptdesign.

**Schema specification by life cycle state and version filtered by date**
This report shows schema specifications grouped by their life cycle state. It then groups by their version and shows the schema specification name, version, namespace, and last modified date.

The report prompts for a start and end date, and only shows schema specifications that have a last modified date that is between the two dates.

The report is called SchemaSpecificationByLifeCycleStateAndVersionDateFilter.rptdesign.

**Service endpoints by life cycle state and environment**
This report shows service endpoints grouped by life cycle state and then further grouped by environment. It shows the endpoint name, namespace, and type.

The report is called ServiceEndpointByLifeCycleStateAndEnvironment.rptdesign.

**Service endpoints by life cycle state and environment filtered**
This report shows service endpoints grouped by life cycle state and then further grouped by environment. It shows the endpoint name, namespace, and type.

The report prompts for a start and end date, and only shows service endpoints that have a last modified date that is between the two dates.

The report is called ServiceEndpointByLifeCycleStateAndEnvironmentDateFilter.rptdesign.
WSDL documents by version
This report shows all WSDL documents grouped by their version number. It shows the WSDL document name, namespace, and version.

The report is called WSDLDocumentByVersion.rptdesign.

XSD documents by version
This report shows all XSD documents grouped by their version number. It shows the XSD document name, namespace, and version.

The report is called XSDDocumentByVersion.rptdesign.

Import the sample reports

To import the sample reports, create a report project, as described in “Creating a report project” on page 10. Perform the following steps:

1. In the Navigator view, right-click JKHLEnterprises_repos, and select Import, as shown in Figure 90.

Figure 90  Selecting import
2. Expand **General**, select **File System**, and click **Next**, as shown in Figure 91.
3. Click **Browse** to locate the `sample_reports` directory. Check `sample_reports`, and click **Finish**, as shown in Figure 92.
After the files import, they are listed under the JKHLEnterprises_reports project in the Navigator view, as shown in Figure 93.

![Figure 93 Sample reports listed in the JKHLEnterprises_reports project](image)

**Configuring the sample report library**

To use the sample reports, you must configure the URL of the WebSphere Service Registry and Repository REST service and, optionally, configure security in WebSphere Service Registry and Repository Studio.

If you are using SSL security in WebSphere Service Registry and Repository, perform the steps in “Configuring WebSphere Service Registry and Repository connectivity in WebSphere Service Registry and Repository Studio” on page 8.

To configure the REST URL in the report library, perform the following steps:

1. In the Navigator view, expand JKHLEnterprises_reports.
2. Expand library.
4. In the Library window, check Don’t show this message again.
5. Click OK.
6. In the Data Explorer view, expand Data Sources.
7. Right-click xml_BusinessCapabilities and select Edit.
8. In the Edit Data Source - xml_BusinessCapabilities window, change the URL of the XML data source to specify your WebSphere Service Registry and Repository, using the format in Example 9.

   **Example 9   The format for the xml_BusinessCapabilities data source URL**

   ```
   <protocol>://<host name>
   :<port>/WSRR/7.0/Metadata/XML/GraphQuery?query=/WSRR/GenericObject[classifiedBy
   nablementModel%23BusinessCapability')]
   ```

   The protocol is either http or https depending on whether SSL is enabled. The host name is the host name of the WebSphere Service Registry and Repository server. The port is
the port on the WebSphere Service Registry and Repository server where the REST API is available.

9. Click **Test Connection**. If SSL is enabled, the Password Required window shown in Figure 94 will prompt you for a user name and password for WebSphere Service Registry and Repository.

![Password Required](image)

*Figure 94  Prompt for user name and password*

Studio will connect to WebSphere Service Registry and Repository and display a success message, as shown in Figure 95.

![Success](image)

*Figure 95  Successful connection message*

10. Click **OK**.

Repeat steps 6 to 10 for the other data sources in the report library. After you are done, select **File → Save**.

**Additional environments**

In the Environments data set, there is a filter that specifies explicitly which governance profile taxonomy URI values are environments. If you have added any extra environment classifications and used them in WebSphere Service Registry and Repository, you must add the URI values to the filter on the Environments data set.

You must also add additional environment URIs to the mapping for the [endpoint_environment] in the reports ServiceEndpointByLifeCycleStateAndEnvironment.rptdesign and ServiceEndpointByLifeCycleStateAndEnvironmentDateFilter.rptdesign.

**Viewing the sample reports**

To view a sample report, right-click the rptdesign file and select **Report → Run Report**.
Using the WebSphere Service Registry and Repository data source

The WebSphere Service Registry and Repository data source and data set can be used to create reports in a similar manner as we detail in this paper.

The key difference is that although the WebSphere Service Registry and Repository data source only specifies the WebSphere Service Registry and Repository server to use, the actual XPath queries are specified in the WebSphere Service Registry and Repository data sets. In the next section, we detail how to create a report library that uses the WebSphere Service Registry and Repository data source rather than the XML data source to create a business capability by capability version report similar to the report detailed in “Business capability by capability version” on page 57.

Create the report project and library

To create the project and library, perform the steps in “Creating a report project” on page 10 and “Creating a report library” on page 12. Use WSRRv7DSLibrary.rptlibrary for the library name.

Configure WebSphere Service Registry and Repository Studio to connect to WebSphere Service Registry and Repository

From the WebSphere Service Registry and Repository Studio Report Design perspective, perform the following steps:

1. Select Window → Preferences.
2. In the Preferences window, click WebSphere Service Registry and Repository (WSRR).
3. Click Add.
4. In the Add WebSphere Service Registry and Repository Location window, enter WSRR in the Alias name field.
5. Specify the protocol, host, and port where the WebSphere Service Registry and Repository Web UI is located.
6. If security is enabled, perform the following steps:
   a. Check Security is enabled on WSRR Server.
   b. Enter the user name and password in the User ID and Password fields.
c. Click **Test Connection**. Studio will connect to WebSphere Service Registry and Repository. Studio may prompt you to accept the WebSphere self-signed certificate, as shown in Figure 96. Click **Accept**.

![Figure 96  Studio prompt to accept the WebSphere self-signed certificate](image)

Studio will display a success message if the test connection worked, as shown in Figure 97.

![Figure 97  Studio connection test successful message](image)
7. In the Test Connection window, click **OK**. The completed window is shown in Figure 98.

Figure 98   The completed General Properties page

8. Click **Finish**.

**Create the WebSphere Service Registry and Repository data source**

From the WebSphere Service Registry and Repository Studio Report Design perspective, perform the following steps:

1. Expand **library**. Double-click **WSRRv7DSLibrary.rptlibrary**.
2. Right-click **Data Sources** and select **New Data Source**.
3. In the New Data Source window, click **WebSphere Service Registry and Repository (WSRR) Data Source**.
4. Enter WSRR in the Data Source Name field.
5. Click **Next**.
6. Select **WSRR** in the server list.
7. Click **Finish**.
8. Select **File → Save**.
Create the WebSphere Service Registry and Repository data sets

The data sets required for the business capability by capability version report are the same as those required for “Business capability by capability version” on page 57. Table 25 lists the WebSphere Service Registry and Repository data set names, the XPath used, and the user-defined properties required.

Table 25  Details for the WebSphere Service Registry and Repository data sets

<table>
<thead>
<tr>
<th>WebSphere Service Registry and Repository data set name</th>
<th>XPath</th>
<th>User-defined properties</th>
<th>Relationships or Classifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapabilityVersion</td>
<td>/WSRR/GenericObject[classifiedByAnyOf(.,'<a href="http://www.ibm.com/xmlns/prod/serviceregistry/profile/v6r3/GovernanceEnablementMode%23CapabilityVersion">http://www.ibm.com/xmlns/prod/serviceregistry/profile/v6r3/GovernanceEnablementMode%23CapabilityVersion</a>')]</td>
<td>primaryType,lastModified,g63_versionTerminationDate,g63_versionAvailabilityDate</td>
<td></td>
</tr>
</tbody>
</table>

To create the BusinessCapability data set, perform the following steps from the WebSphere Service Registry and Repository Studio Report Design perspective:

1. Expand library. Double-click WSRRv7DSLibrary.rptlibrary.
2. Right-click Data Sets and select New Data Set.
3. In the New Data set window, click WSRR.
4. Select WSRR Free Form Graph Query Data Set in the Data Set Type drop-down menu.
5. Enter BusinessCapability in the Data Set Name field.
6. Click Next.
7. Enter the XPath given in Table 25 for the BusinessCapability data set into the Enter the XPath query to be used for this data set field.
8. Enter primaryType into the Enter comma-separated list of user-defined properties to be returned field.
10. Click **Finish**. The Edit Data Set window opens, as shown in Figure 99.

![Image of Edit Data Set window](image)

**Figure 99** The new BusinessCapability WebSphere Service Registry and Repository data set

11. Click **OK**.

12. Repeat steps 2 to 11 for the other data sets listed in Table 25 on page 82. Only the BusinessCapability_VersionRelationship data set has Relationships checked; the others have Relationships and Classifications unchecked.

13. Select **File** → **Save**.

One difference between using the XML data set and the WebSphere Service Registry and Repository data set is that the BusinessCapability_VersionRelationship WebSphere Service Registry and Repository data set returns information about all relationship targets, with one row per relationship target, whereas the XML data set specifies only the gep63_capabilityVersions relationship. Therefore, we need to add a filter to the BusinessCapability_VersionRelationship WebSphere Service Registry and Repository data set to only return the gep63_capabilityVersions relationship rows. To accomplish this task, perform the following steps:

1. Right-click **BusinessCapability_VersionRelationship** and click **Edit**.
2. Click **Filters**.
3. Click **New**.
4. In the New window, select **relationship name** in the Expression drop-down menu.
5. Ensure **Equal to** is selected in the Operator drop-down menu.
6. Enter "gep63_capabilityVersions" in the Value 1 field. Ensure you use the double-quotes; this tells BIRT that the value is a string.
7. The New window opens, as shown in Figure 100. Click OK.

8. Click Preview Results. The preview is shown in Figure 101. Comparing it against the XML data set, the bsrURI of the business capability and the capability version are shown in the results, in the columns bsrURI and targetBsrURI.

9. Click OK.

10. Select File → Save.

Create the joint data set

Next, we create the joint data sets. Because there are multiple joint data sets, we do not specify any column aliases on any other data set than the one used in the report.
BusinessCapability_CVRef
To create the BusinessCapability_CVRef joint data set, perform the steps in “Joining the data sets in the report library” on page 24. Use the values provided in Table 26 to specify the data set name, the two data sets that are joined, and the columns that are used for the join. We use a left outer join because the BusinessCapability_VersionRelationship only contains entries for business capabilities with capability versions.

Table 26  Values for the BusinessCapability_CVRef joint data set

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Name</td>
<td>BusinessCapability_CVRef</td>
</tr>
<tr>
<td>Join Type</td>
<td>Left Outer</td>
</tr>
<tr>
<td>Data set 1 name</td>
<td>BusinessCapability</td>
</tr>
<tr>
<td>Data set 1 column</td>
<td>bsrURI</td>
</tr>
<tr>
<td>Data set 2 name</td>
<td>BusinessCapability_VersionRelationship</td>
</tr>
<tr>
<td>Data set 2 column</td>
<td>bsrURI</td>
</tr>
</tbody>
</table>

The BusinessCapability_CVRef data set is shown in Figure 102.

Figure 102  The BusinessCapability_CVRef data set
**BusinessCapability_CapabilityVersion**

To create the BusinessCapability_CapabilityVersion joint data set, perform the steps in “Joining the data sets in the report library” on page 24. Use the values provided in Table 27 to specify the data set name, the two data sets that are joined, and the columns that are used for the join.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Name</td>
<td>BusinessCapability_CapabilityVersion</td>
</tr>
<tr>
<td>Join Type</td>
<td>Left Outer</td>
</tr>
<tr>
<td>Data set 1 name</td>
<td>BusinessCapability_CVRef</td>
</tr>
<tr>
<td>Data set 1 column</td>
<td>BusinessCapability_VersionRelationship::target BsrURI</td>
</tr>
<tr>
<td>Data set 2 name</td>
<td>CapabilityVersion</td>
</tr>
<tr>
<td>Data set 2 column</td>
<td>bsrURI</td>
</tr>
</tbody>
</table>

The BusinessCapability_CapabilityVersion data set is shown in Figure 103.

![Figure 103: The BusinessCapability_CapabilityVersion data set](image)
Because this is the data set that will be used in the report, we specify column mappings. Perform the steps in “Joining the data sets in the report library” on page 48 and specify the mappings in Table 28.

Table 28  Mappings for the BusinessCapability_CapabilityVersion joint data set

<table>
<thead>
<tr>
<th>Name</th>
<th>Alias</th>
<th>Display name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessCapability_CVRef::BusinessCapability::name</td>
<td>bc_name</td>
<td>Business Capability Name</td>
</tr>
<tr>
<td>BusinessCapability_CVRef::BusinessCapability::lastModified</td>
<td>bc_lastmodified</td>
<td>Business Capability Last Modified</td>
</tr>
<tr>
<td>BusinessCapability_CVRef::BusinessCapability::bsrURI</td>
<td>bc_bsrURI</td>
<td>Business Capability BsrURI</td>
</tr>
<tr>
<td>BusinessCapability_CVRef::BusinessCapability::primaryType</td>
<td>bc_primaryType</td>
<td>Business Capability Type</td>
</tr>
<tr>
<td>CapabilityVersion::name</td>
<td>cv_name</td>
<td>Capability Version Name</td>
</tr>
<tr>
<td>CapabilityVersion::version</td>
<td>cv_version</td>
<td>Capability Version Version</td>
</tr>
<tr>
<td>CapabilityVersion::bsrURI</td>
<td>cv_bsrURI</td>
<td>Capability Version BsrURI</td>
</tr>
<tr>
<td>CapabilityVersion::lastModified</td>
<td>cv_lastmodified</td>
<td>Capability Version Last Modified</td>
</tr>
<tr>
<td>CapabilityVersion::gep63_versionTerminationDate</td>
<td>cv_terminationDate</td>
<td>Capability Version Termination Date</td>
</tr>
<tr>
<td>CapabilityVersion::gep63_versionAvailabilityDate</td>
<td>cv_availabilityDate</td>
<td>Capability Version Availability Date</td>
</tr>
<tr>
<td>CapabilityVersion::primaryType</td>
<td>cv_primaryType</td>
<td>Capability Version Type</td>
</tr>
</tbody>
</table>

Creating the report

Because we defined the same aliases in the joint data sets as in “Business capability by capability version” on page 57, the steps required to create the report are identical to those detailed in the following sections:

1. “Creating the report” on page 64.
2. “Adding the report library to the report” on page 64.
4. “Add a table to the report” on page 64.
5. “Add labels” on page 64.
6. “Insert the business capability type group” on page 64.
7. “Map the business capability type to messages” on page 65.
8. “Insert the business capability name group” on page 65.
9. “Add a count of capability versions to the table” on page 66.
10. “Adding a label to the count” on page 66.
11. “Add the columns to the table” on page 66.
The results of the report are shown in Figure 104.

<table>
<thead>
<tr>
<th>Capability Type</th>
<th>Capability Name</th>
<th>Version Name</th>
<th>Version Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Application</td>
<td>Footwear ordering application</td>
<td>Footwear ordering</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>version</td>
<td>version</td>
<td></td>
</tr>
<tr>
<td>Business Process</td>
<td>Customer Care business process</td>
<td>Customer Care</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>business process</td>
<td>business process</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1.0)</td>
<td>(1.0)</td>
<td></td>
</tr>
<tr>
<td>Business Service</td>
<td>Ship Goods</td>
<td>Ship Goods</td>
<td>2.0.1</td>
</tr>
<tr>
<td></td>
<td>Account creation business service</td>
<td>Account creation</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>business service</td>
<td>business service</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1.0)</td>
<td>(1.0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Verify Credit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check customer for account service</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ordering Service</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Eligibility service</td>
<td>Eligibility service</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>(1.0)</td>
<td>(1.0)</td>
<td></td>
</tr>
</tbody>
</table>

Troubleshooting hints and tips

At times you may have a need to adjust various timeout parameters provided in WebSphere Application Server components from their default values. For SOAP timeout, Transaction timeout, and HTTP timeout tuning, refer to the following address:


For HTTP Session timeout and LTPA token timeout tuning, refer to the following address:


If you see java.lang.OutOfMemoryError while trying to generate reports in WebSphere Service Registry and Repository Studio and Report BIRT Viewer, you may need to adjust the starting and maximum heap sizes for the Eclipse workbench JVM. Refer to the following address:

http://wiki.eclipse.org/FAQ_How_do_I_increase_the_heap_size_available_to_Eclipse%3F

Depending on the complexity of reporting logic and volume of metadata present in WebSphere Service Registry and Repository, you may need to adjust heap size of JVM running WebSphere Service Registry and Repository instance. To perform this task, refer to the following address:

Additional material

This section refers to additional material that can be downloaded from the Internet.

The material that is associated with this paper is available in softcopy on the Internet from the IBM Redbooks® web server. Point your web browser at:

ftp://www.redbooks.ibm.com/redbooks/REDP4687

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the Additional materials and open the directory that corresponds with the IBM paper form number, REDP4687.

The additional material includes a WebSphere Service Registry and Repository export file containing business capabilities, capability versions, SLA, SLD, and DOU in various life cycle states. To import the file into WebSphere Service Registry and Repository, it is necessary to use the WebSphere Service Registry and Repository Web UI. Perform these steps:

1. Log in to WebSphere Service Registry and Repository as an administrator.
2. Select the Administrator perspective.
3. Select Actions → Import.
4. Click Browse and select the whole_wsrr_export.zip file.
5. In the Preparing to Import Documents page, click OK.

The author who wrote this paper

This paper was produced by a technical specialist working at the International Technical Support Organization, Raleigh Center.

David Seager is a User Interface Developer for the WebSphere Service Registry and Repository development team based at the IBM Hursley Development Lab in the United Kingdom. He has 13 years of experience in middleware development at IBM. David holds a Masters degree in Software Engineering from Oxford University and a Masters degree in Physics from Oxford University. His areas of expertise include WebSphere Application Server, AJAX, JavaScript, Java™, and Web UI design. David contributed to the IBM Redbooks publication CICS Transaction Gateway V5 The WebSphere Connector for CICS, SG24-6133. David is an IBM developerWorks® contributing author and has written several IBM developerWorks articles in his area of expertise.

Thanks to the following people for their contributions to this project:

Michael Ellis, Software Services for WebSphere
IBM Canada

Ian Heritage, Worldwide WebSphere Technical Sales Support & Competitive Analysis - SOA Governance
Martin Rowe, Service Registry development
IBM UK
Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at: ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

➤ Find us on Facebook:
  http://www.facebook.com/IBMRedbooks
➤ Follow us on Twitter:
  http://twitter.com/ibmredbooks
➤ Look for us on LinkedIn:
  http://www.linkedin.com/groups?home=&gid=2130806
➤ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
➤ Stay current on recent Redbooks publications with RSS Feeds:
  http://www.redbooks.ibm.com/rss.html
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

developerWorks®  Redbooks®  Redbooks (logo)  ©
IBM®  Redpaper™  WebSphere®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.