CentraSite

BIRT Tutorial

Version 9.0.1

June 2013

This document applies to CentraSite BIRT Tutorial Version 9.0.1.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: IINM-TUTORIAL-BIRT-901-20130618

Table of Contents

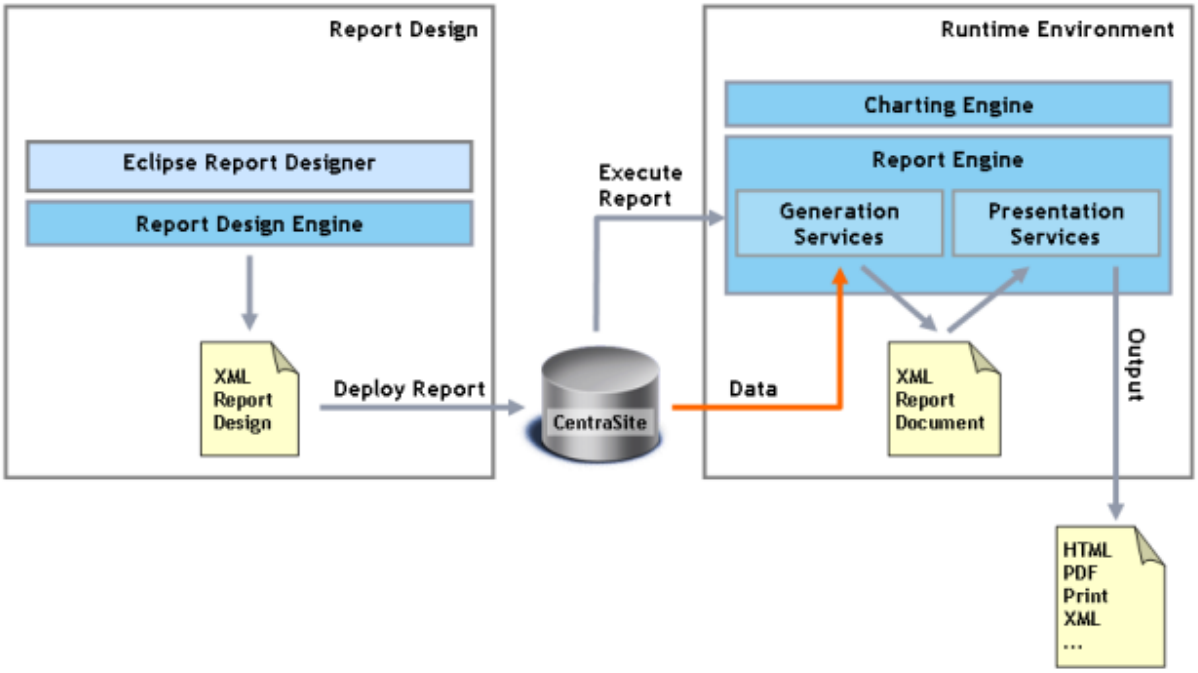
Preface

This document tells you how to use the reporting features in CentraSite that make use of “BIRT”, the “Business Intelligence and Reporting Tools”, which is an open source, Eclipse based reporting system. BIRT can generate both HTML and PDF reports. This description relates to CentraSite version 8.2.

1 BIRT Tutorial: Introduction

CentraSite includes a reporting feature that is based on BIRT, the Business Intelligence and Reporting Tools. This is an open source reporting system based on Eclipse.

The following diagram gives an overview of BIRT reporting in CentraSite:



Displaying a report in the BIRT/CentraSite world means executing a report template (the template is also referred to as an *rptdesign* file, since the filename suffix is "rptdesign"). A report template is an XML file that describes the appearance of the report, and how to retrieve the data with which the report should be populated when it is executed. The CentraSite distribution includes a number of predefined report templates; you can build your own templates using the BIRT Report Designer (in Eclipse). The Report Designer is shown on the left in the **diagram above**.

Newly-built templates can be deployed to CentraSite, i.e. stored in the CentraSite data storage. Executing report templates is the task of the BIRT runtime environment. The tasks of the BIRT runtime environment include establishing the report as described in the template, and also reading and including the data. In the case of CentraSite reports, the data is read from CentraSite.

The CentraSite distribution includes a set of predefined reports that are already loaded into the CentraSite data storage and can conveniently be executed either from CentraSite Control or from the CentraSite Eclipse Perspective. This is the easiest way to use BIRT and it satisfies many entry-level situations, since the predefined reports cover many requirements such as information about assets, information about the relationships between assets, information about assets' histories, and the like. If your interest is covered by one of the predefined reports alone, we suggest that you jump directly to the chapter **Predefined Reports in CentraSite**, for a detailed overview.

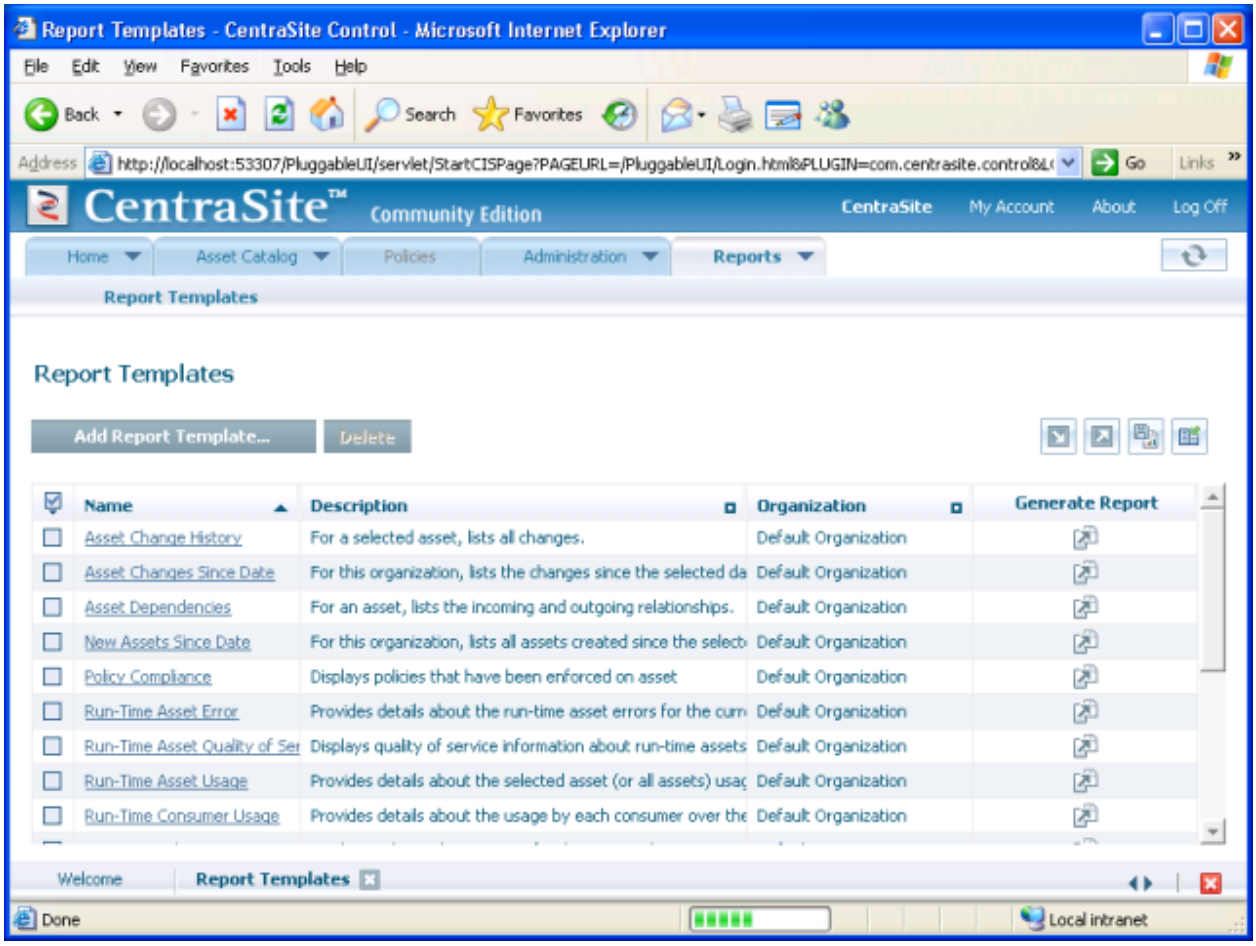
The chapter ***Migrating a Report*** explains how to migrate a report that was created using a previous version of CentraSite.

The subsequent chapters discuss how to modify the predefined reports, how to build new reports based on these predefined reports, and how to build totally new reports from scratch. These topics bring us into the world of Eclipse or, more precisely, to what the Software AG specific Eclipse version is called, the "Designer".

2 Predefined Reports in CentraSite

The CentraSite distribution includes a number of predefined reports; they are also referred to as report templates. In contrast to CentraSite version 3.1, where predefined reports were delivered as *rptdesign* files in the CentraSite install directory, starting with CentraSite version 8.0 the predefined reports are already loaded into the CentraSite data storage. More precisely, each predefined report is represented as a pair of assets, namely a registry entry in the registry and, attached to it, the report's *rptdesign* file in the repository.

You can see an overview of all the predefined reports that are included with the CentraSite distribution by selecting the **Reports** tab in CentraSite Control:



As with all object listings in CentraSite Control, you can sort the list by clicking on a column header (in the **screenshot**, the report is sorted alphabetically by name). There are 17 predefined reports; they are also referred to as report templates, since they need to be filled with data to become real reports.

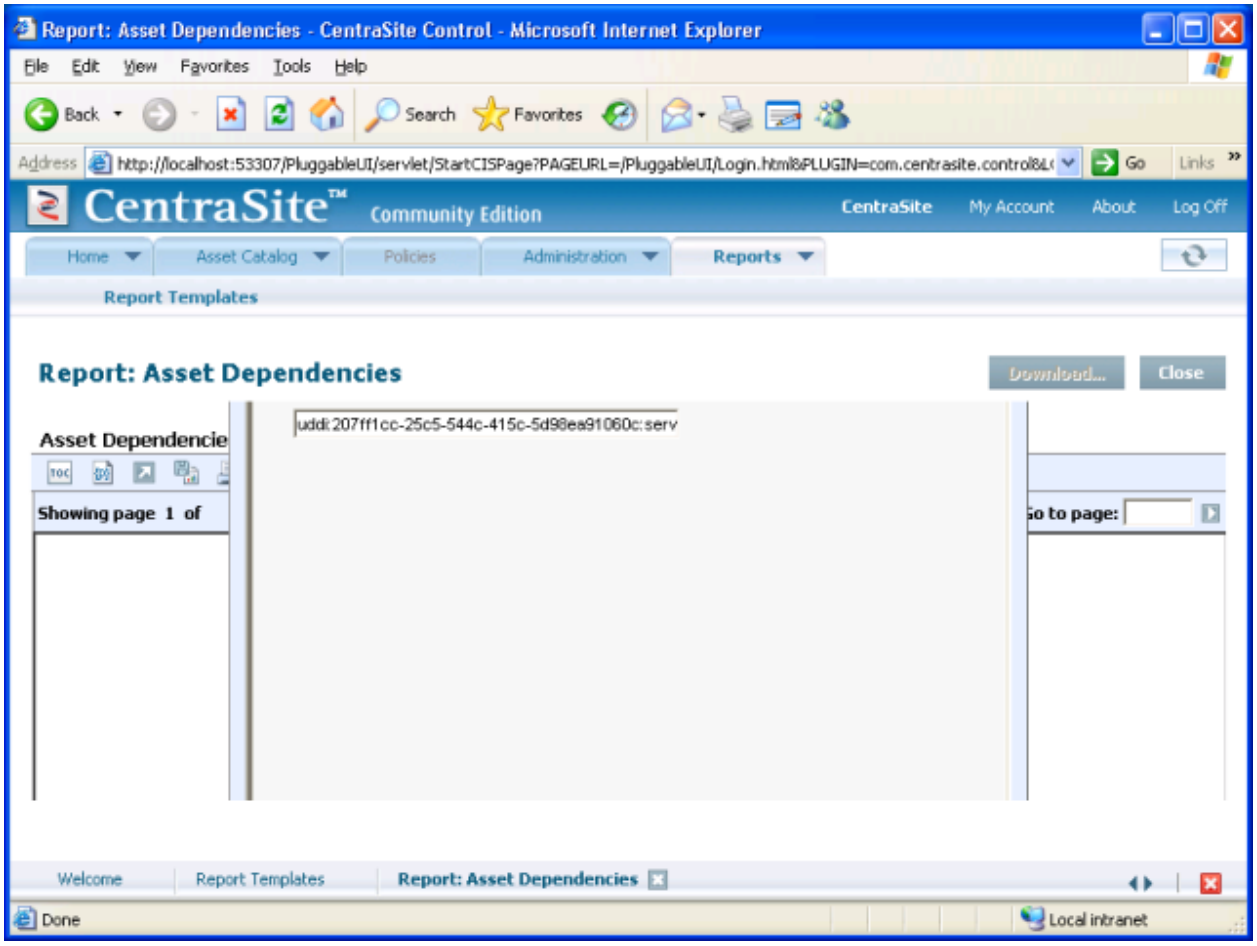
Before we describe how to execute reports (or more accurately how to execute one particular report, namely Asset Dependencies), take a look at the **following table**,which lists the names and descriptions of the predefined reports. An asterisk in the last column (“Runtime”) indicates report templates

that are specific to runtime issues; if you are using the Community Edition, reports generated from these templates are empty unless runtime-specific data has been added.

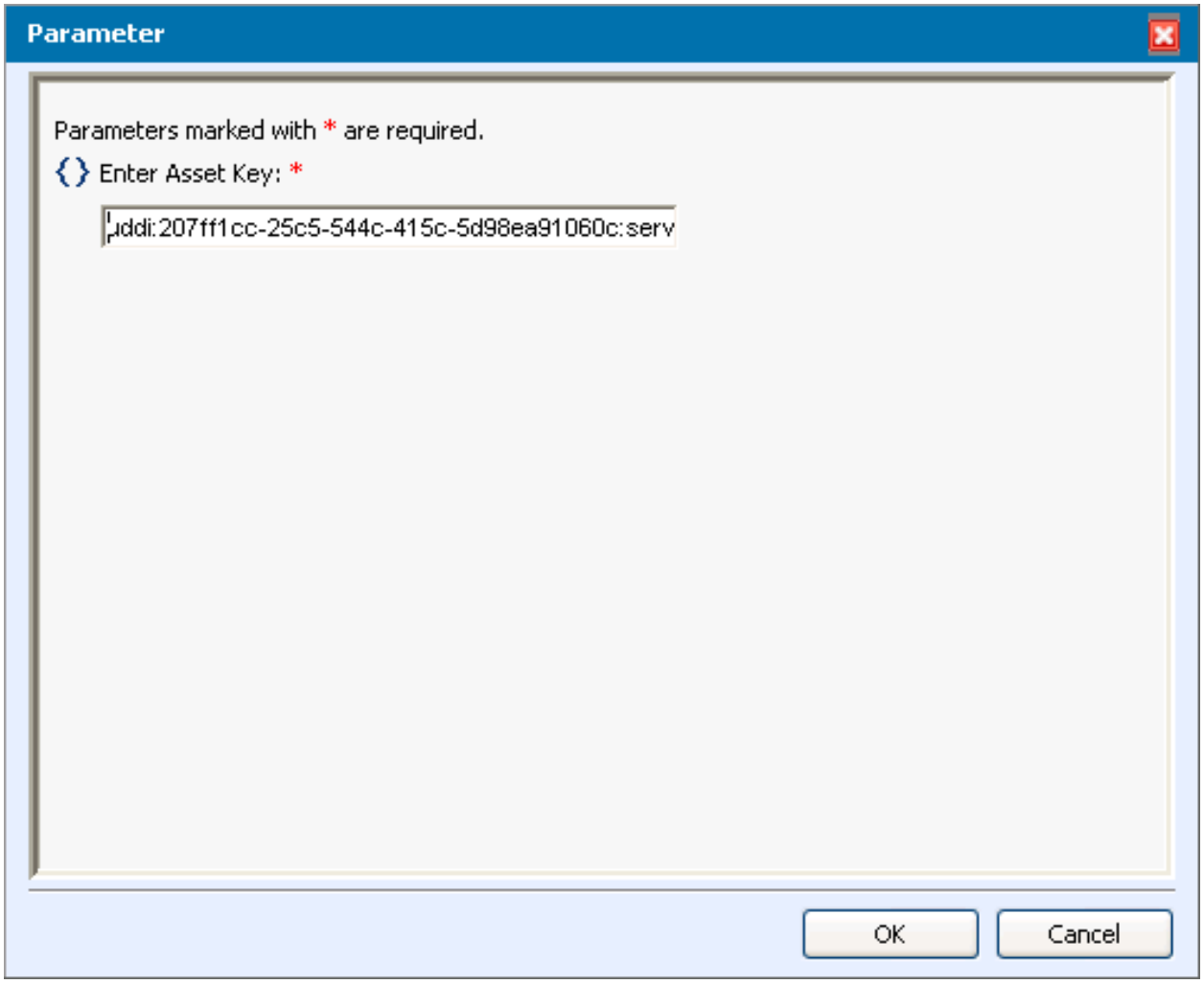
Number	Name	Description	Runtime
1	Asset Change History	For a selected asset, lists all changes.	
2	Asset Changes Since Date	For this organization, lists all changes made since the specified date.	
3	Asset Dependencies	For an asset, lists the incoming and outgoing relationships.	
4	New Assets Since Date	For this organization, lists all assets created since the specified date.	
5	Policy Compliance	Displays policies that have been enforced on asset.	
6	Run-Time Asset Error	Provides details about the runtime asset errors for the current day.	*
7	Run-Time Asset Quality of Service	Displays quality of service information about runtime assets.	*
8	Run-Time Asset Usage	Provides details of the usage of the selected asset (or all assets) for the selected period.	*
9	Run-Time Consumer Usage	Provides details about the usage by each consumer over the selected period.	*
10	Run-Time Policy Errors	Displays policy violation errors for the current day.	*
11	Run-Time Services	Displays summary information about virtual services.	*
12	Service Details	Displays the details of all services of an organization.	
13	SLA Violations	Provides details of SLA violations over the selected period.	*
14	SOA Maturity	For each month in the current and previous year, this report calculates the number of services available per month with the total number of usage references for those services. The result is rendered as a table and as a chart.	
15	Top Ten Consumers	Displays the top ten consumers in the SOA infrastructure.	*
16	Top Ten Services	Displays the top ten services running in the SOA infrastructure.	*
17	Unreferenced Assets	Shows all assets that have no incoming relationships.	

Executing Reports from CentraSite Control

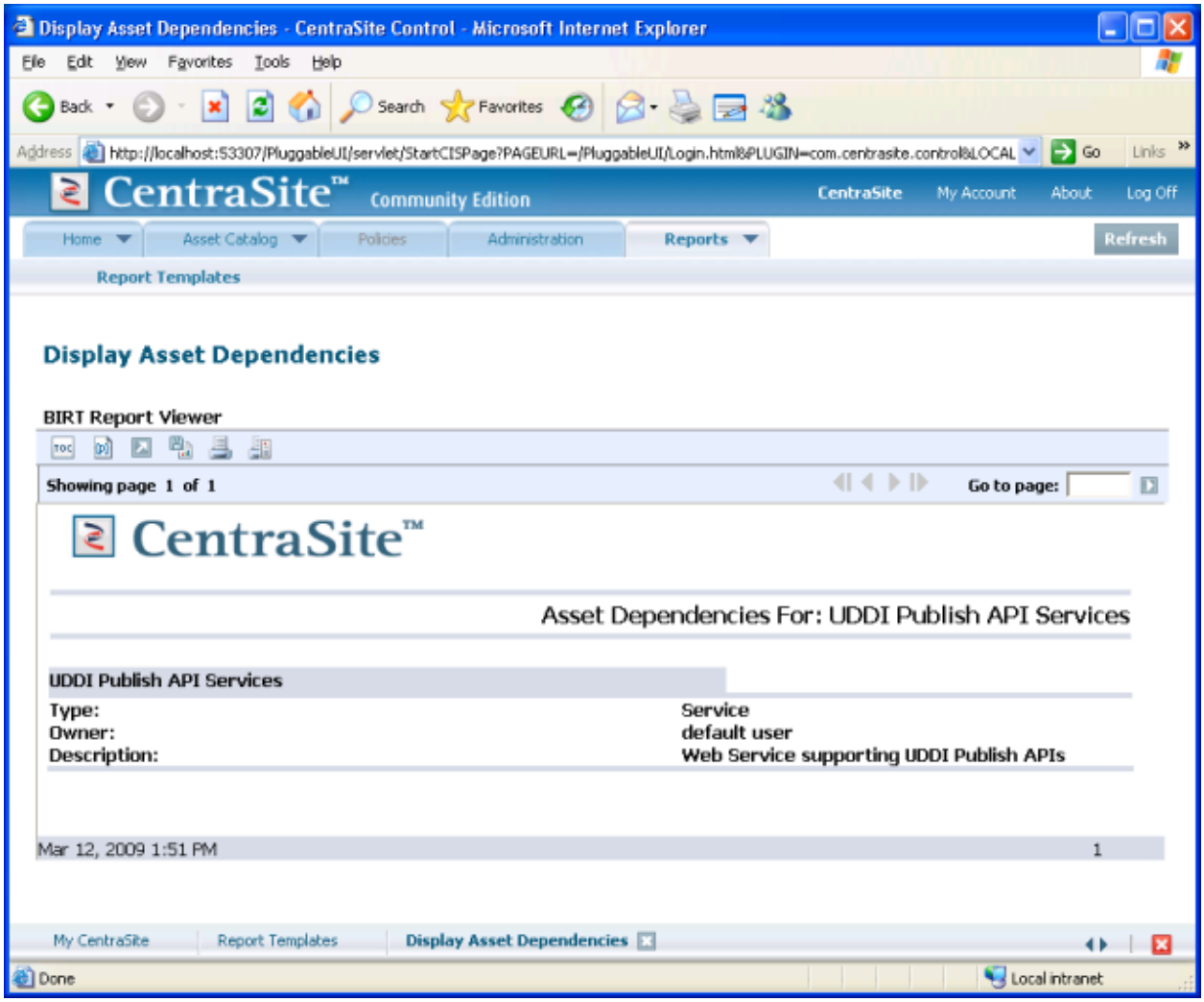
The first way to execute reports is directly from the reports listing, by clicking on the corresponding icon in the **Generate Report** column. The following screenshot shows this for the Asset Dependencies report:



This report is parameterized, so you are prompted to enter the input parameter. The prompt looks like a pop-up, but it isn't: the area within this Control window is completely given to the BIRT runtime component. If your window looks like the one above, enlarge the complete Control window (preferably, maximize it) so that you can see the complete parameter requirement:

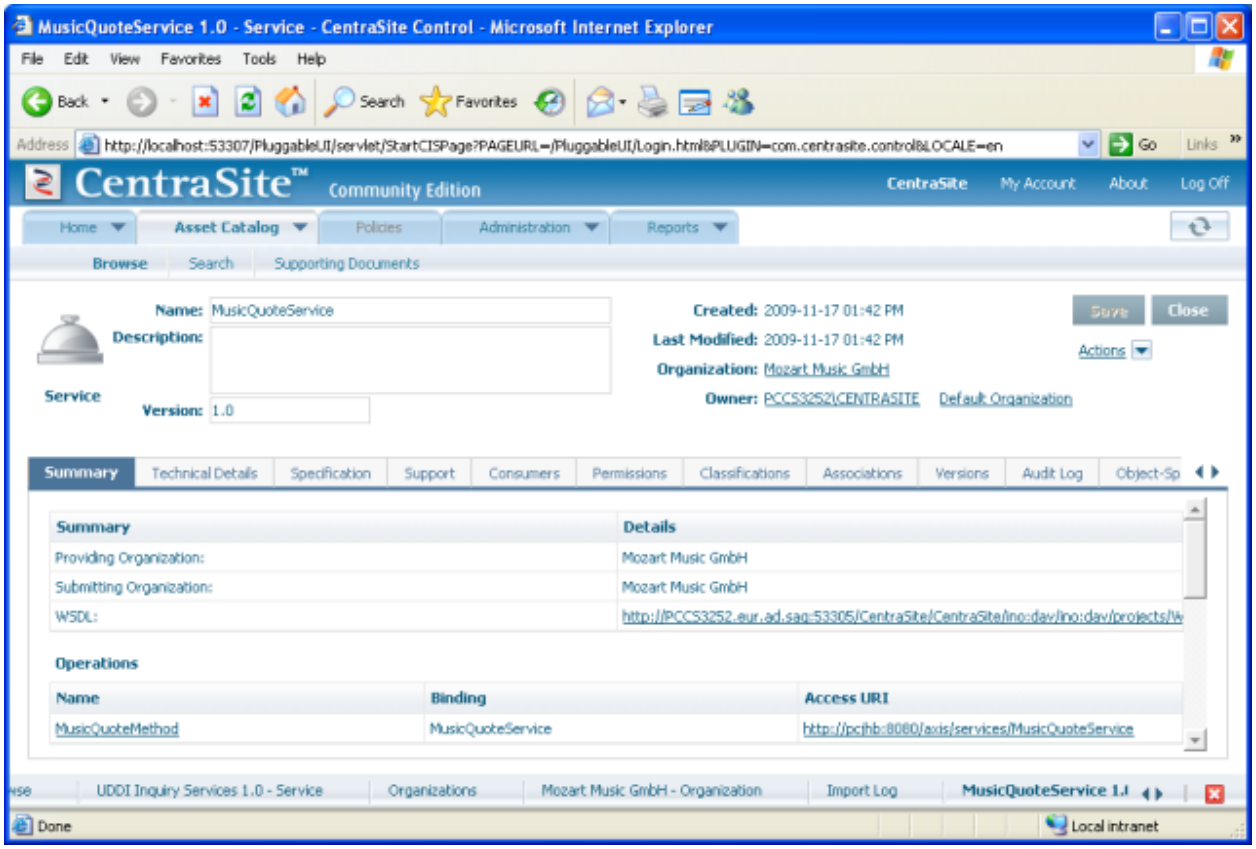


The parameter required by the report Asset Dependencies is a UDDI key identifying a registry entry. The report suggests a key that points to the predefined service “UDDI Publish API Services”. If you accept that key, i.e. click **OK** without changing anything, the report for this service is shown. Since the service is currently not associated with any other assets, the report doesn't show much:

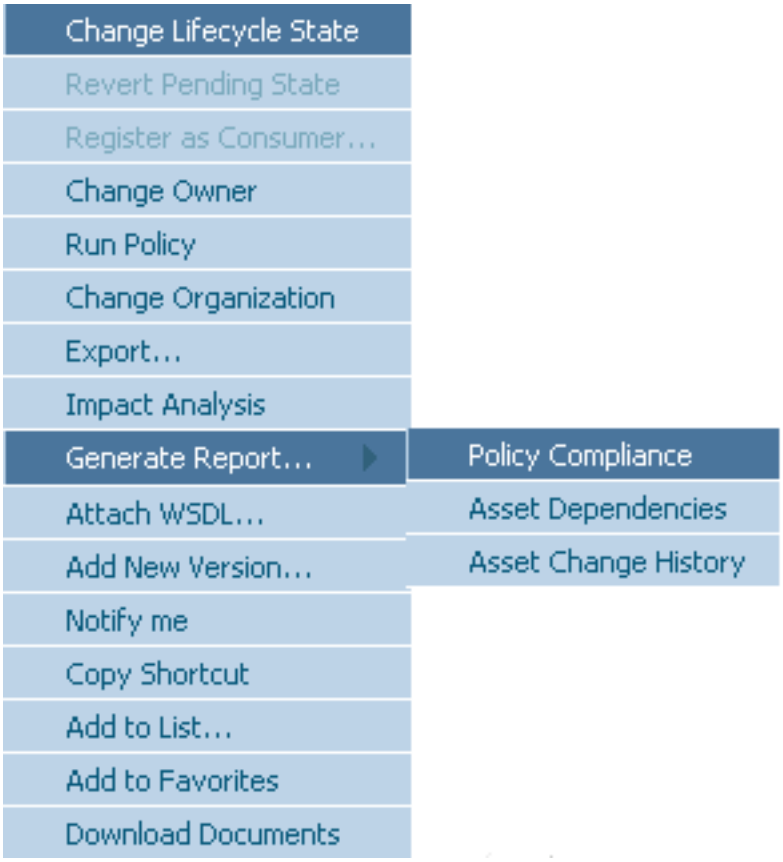


To get a more meaningful report, we should choose an asset that is somehow associated. Starting the report from here gives you the choice of either accepting the parameter's default setting or replacing it with another asset's UDDI key, which in general you will not know. Thus we now use another method to generate a report. To this purpose, we go to the detail view of the asset upon which we want to apply the report. For this we choose another non-predefined service that stems from the well-known music example, namely MusicQuoteService.

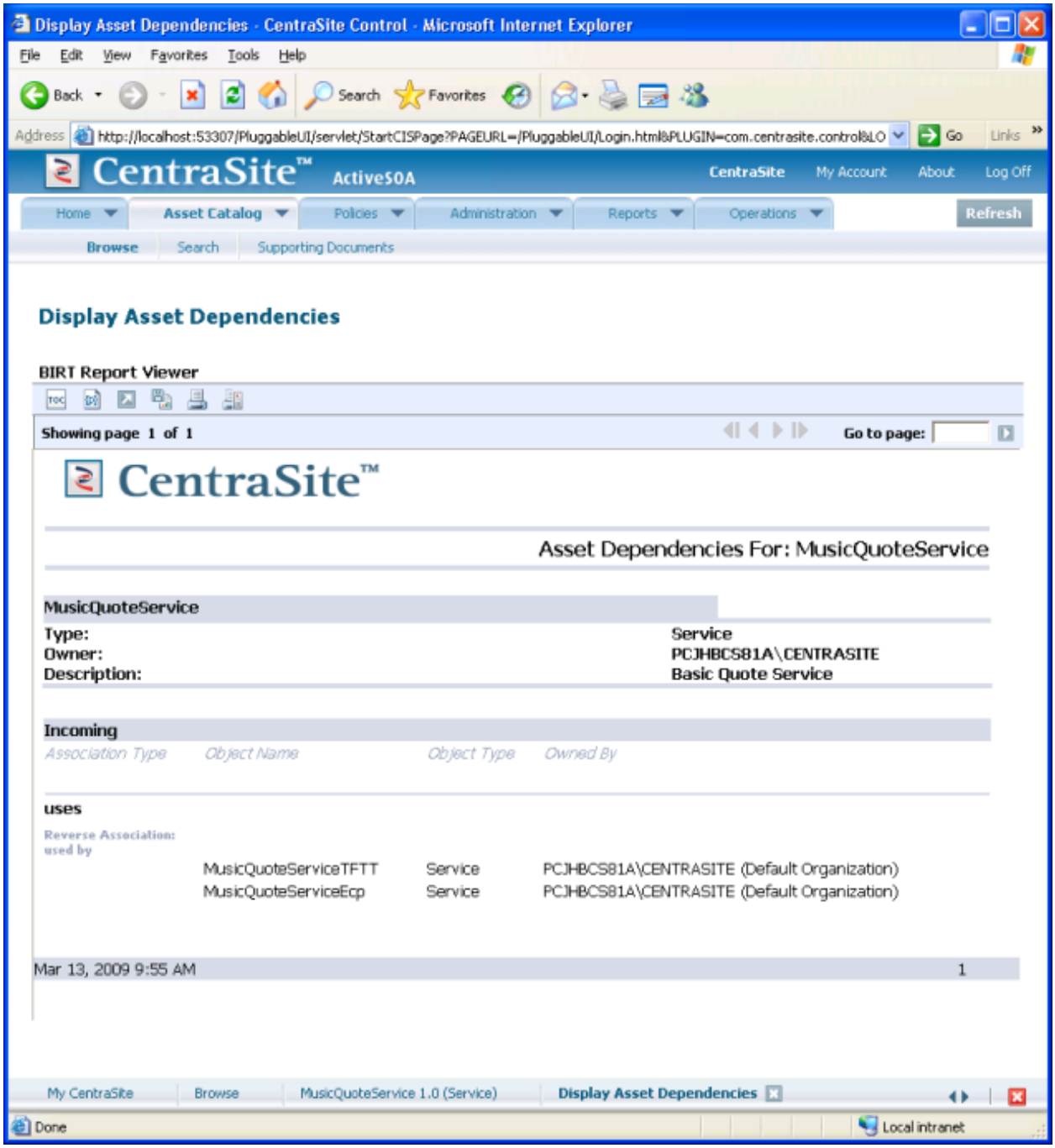
Go to the detail view of the registry entry upon which the report is to be applied:



Clicking on **Actions** in the upper right-hand corner of the detail view reveals a set of actions applicable to this registry entry; one of the actions is "Generate Report". Selecting this shows all the reports that are currently linked to the registry object's type, which in this case is "Service". Since the report "Asset Dependencies" is predefined as being linked to many object types, it is offered here:



Selecting the corresponding report ("Asset Dependencies") executes the report, the report's parameter being set to the UDDI key of the current object.



The report shows that the service "MusicQuoteService" is used (i.e. it is pointed to via the association "Uses") from two other services.

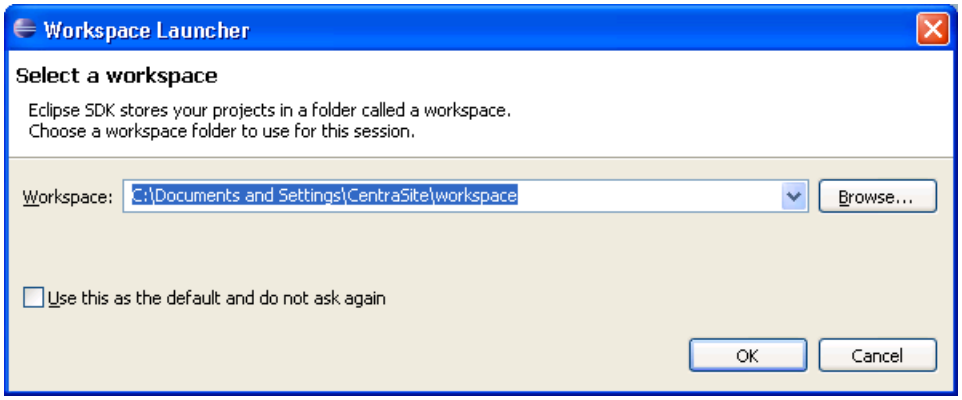
To link a report template to an additional object type, go to the detail view of the registry entry that represents the report. One way to reach such a detail view is by clicking on the report name in the report overview page (see above). In the "Report Template" details page, select one or more object types in the **Applicable to Object Types** panel and save.

When presenting the result of a report execution, CentraSite Control delivers the output page by page. Since the two results that we have seen so far each filled only one page, the icons that support paging, i.e. and , are greyed out.

Executing Reports from Eclipse

We now run a predefined report from within Eclipse or, more precisely, from the CentraSite Eclipse Perspective. Since this is our first close encounter with Eclipse in this tutorial, we start with some general remarks.

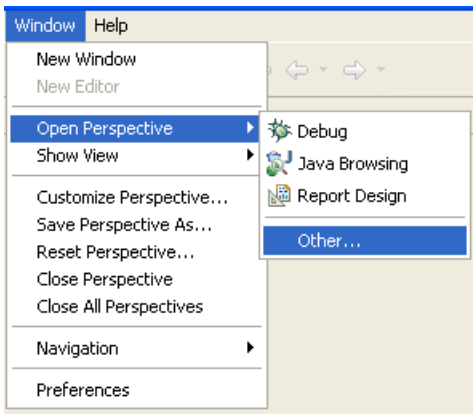
Starting the Designer for the first time opens a pop-up that prompts you to specify the workspace location:



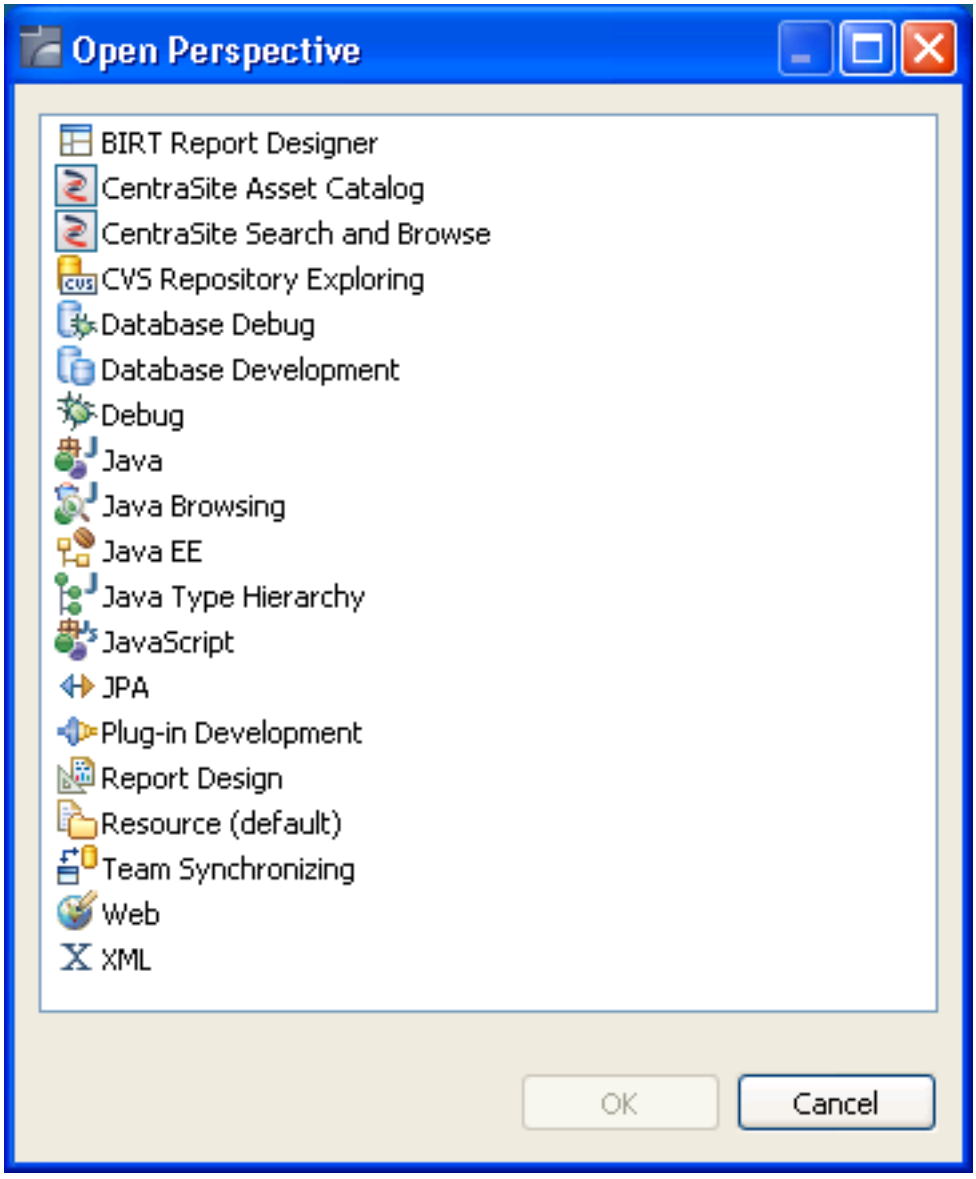
The workspace is the location where Eclipse stores its project data. It is not very important where this is, but you should make sure that enough free disk space is available at the specified location. Mostly, the default setting is acceptable. Additionally, click the checkbox to use this location automatically for future Eclipse sessions. Both the workspace location and the choice whether Eclipse shows this pop-up when starting can be modified at any time from the Eclipse workbench.

When starting Eclipse for the first time, it does not go immediately into the workbench (i.e. the frame hosting Eclipse perspectives and views), but displays a Software AG proprietary Welcome page for the Designer:

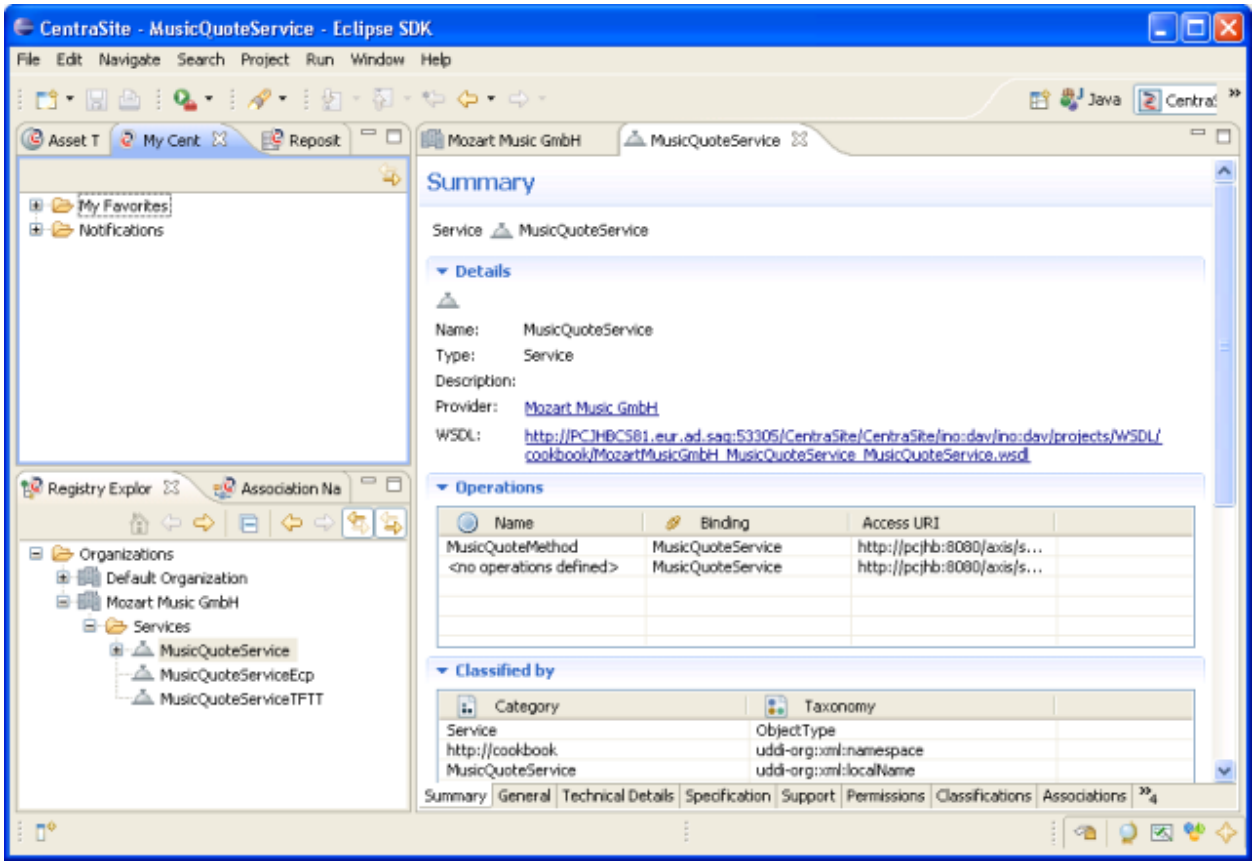
To change to the CentraSite perspective, use the **Window** menu item:



Installing the CentraSite plugins added two perspectives to the Eclipse workspace, namely "CentraSite Asset Catalog" and "CentraSite Search and Browse", as shown below:

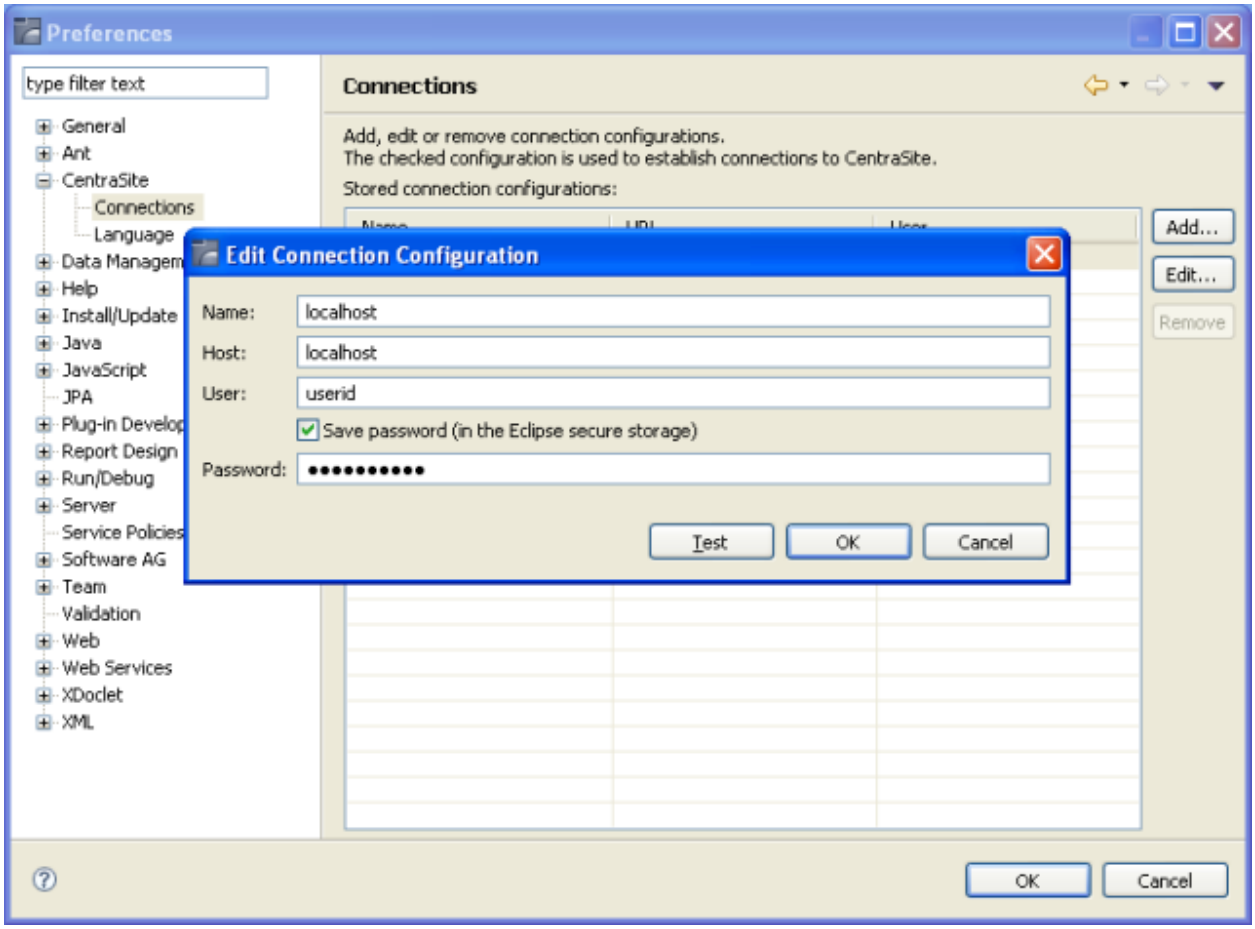


Choose "CentraSite Search and Browse". The precise appearance of this CentraSite perspective depends on your individual settings. The following screenshot shows the two views "My CentraSite" (in the upper left pane) and "Registry Explorer" (in the lower left pane), and also the detail view for the service MusicQuoteService:

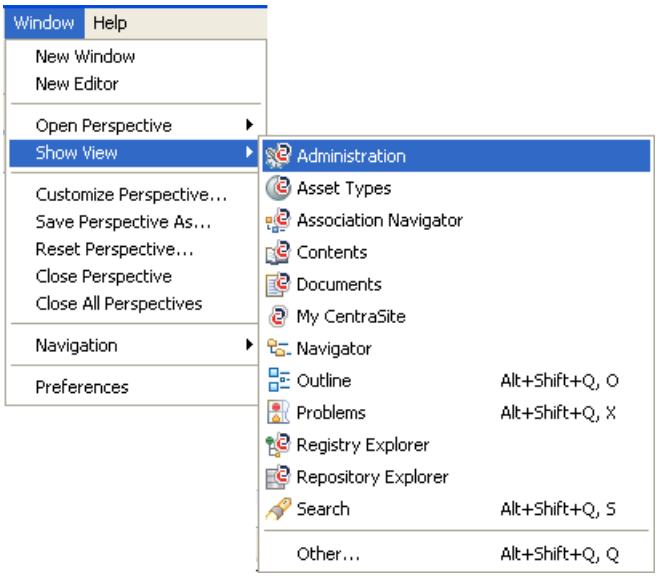


To be able to connect the CentraSite data storage, it is necessary to establish a connection configuration. This is achieved in the Eclipse Preferences Dialog (reachable from the **Window** menu entry). This dialog automatically pops up when connection to the data storage is first requested.

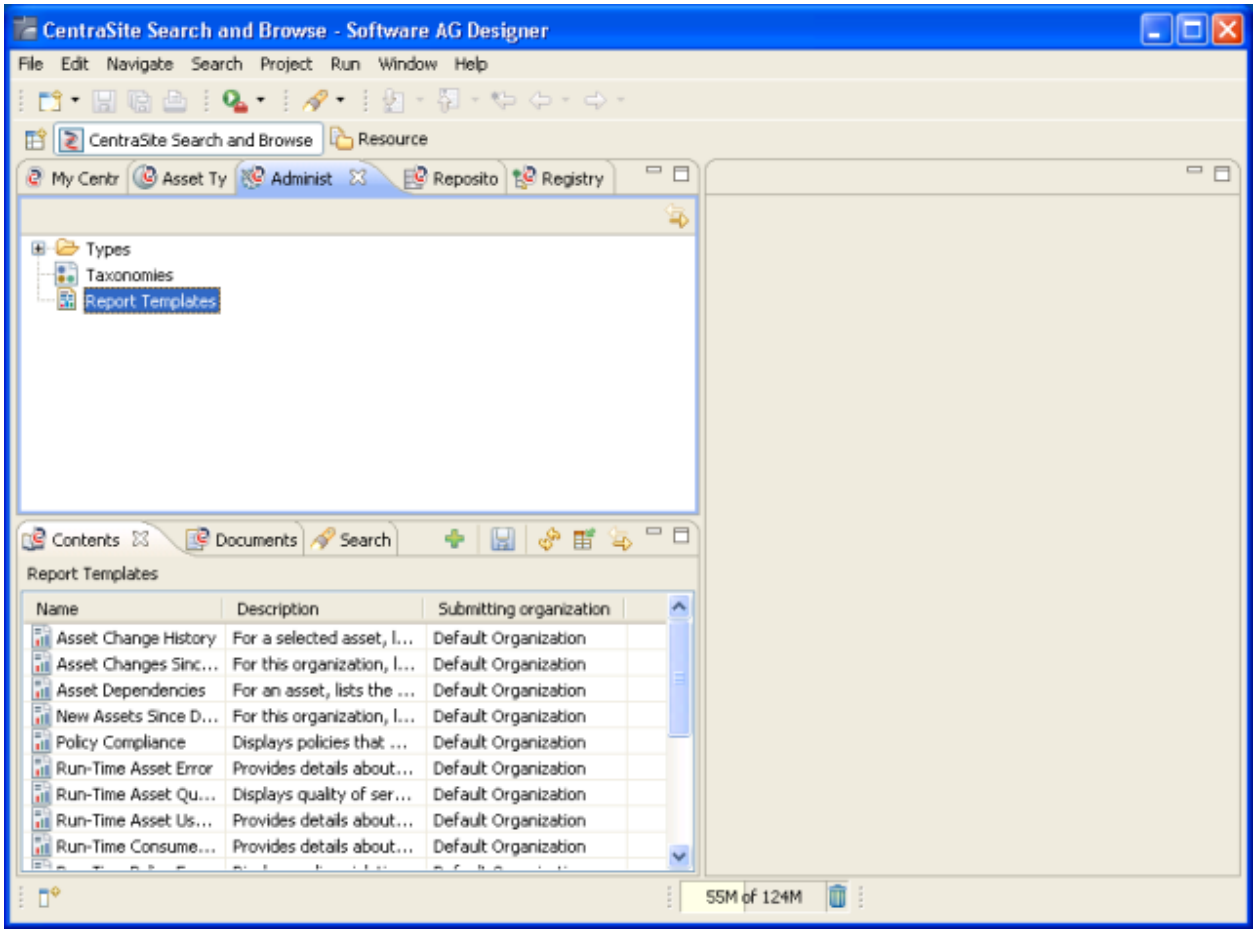
Click **Add** to create a new connection. In the next pop-up, enter "localhost" twice (if this what you want to connect to) and your user-ID and password. Putting a check (tick) in the checkbox "Save password" avoids having to enter your password for future Eclipse sessions, but is not recommended in a security-sensitive environment.



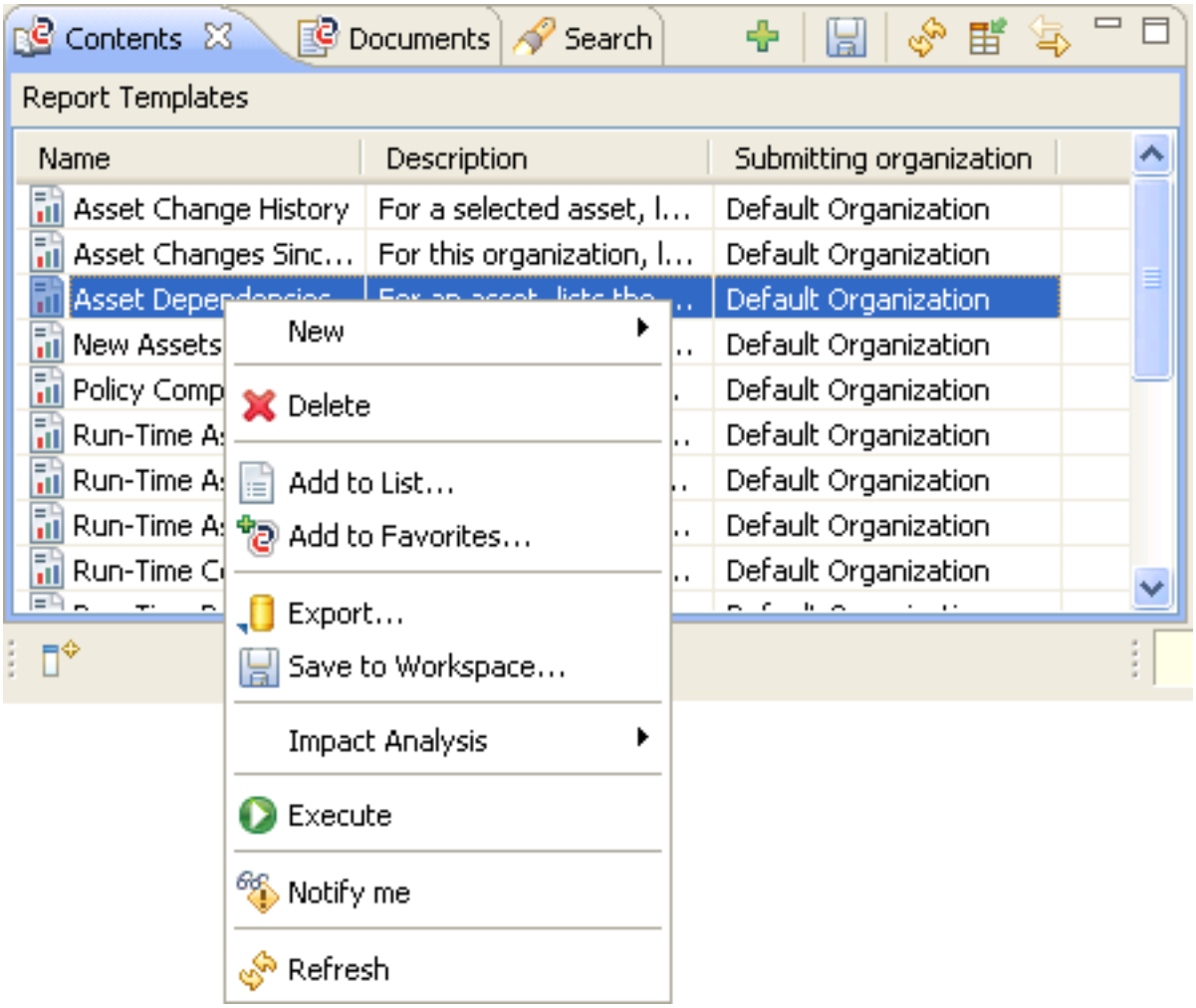
To display a list of all predefined reports, open the **Administration View**:



In the **Administration View**, double-click on the entry **Report Templates** to open the list:



To display the report, select **Execute** from the report's context menu:


Contents
Documents
Search
Report Templates
Name
Description
Submitting organization
Asset Change History
For a selected asset, l...
Default Organization
Asset Changes Sinc...
For this organization, l...
Default Organization
Asset Dependencies
Default Organization
New Assets
Default Organization
Policy Comp
Default Organization
Run-Time A
Default Organization
Run-Time A
Default Organization
Run-Time A
Default Organization
Run-Time C
Default Organization
New
Delete
Add to List...
Add to Favorites...
Export...
Save to Workspace...
Impact Analysis
Execute
Notify me
Refresh

3 Migrating a Report

This chapter discusses the following report migration topics:

Changes Due to the Introduction of BIRT version 2.6.1

A report that was created with an earlier version of CentraSite may require some adaption, since version 8.2 of CentraSite introduced BIRT version 2.6.1 (older versions of CentraSite used BIRT version 2.3.2). A known issue with the predefined reports that are supplied with CentraSite, and reports built from copies of them, is the `locale` property.

The beginning of the report template for the report *Asset Change History* as included with CentraSite version 8.0 is shown below. You can see this, for example, in CentraSite Control: go to the detail view of the report's registry entry and choose **Show Template File** from the **Action** drop-down menu.

```
<?xml version="1.0" encoding="UTF-8" ?>
<report id="1" version="3.2.17" xmlns="http://www.eclipse.org/birt/2005/design">
   <property name="createdBy">Eclipse BIRT Designer Version 2.3.2.r232_20090202 Build
     <2.3.2.v20090218-0730></property>
    <property name="units">in</property>
    <text-property name="title">Asset Change History</text-property>
    <property name="comments">CentraSite ActiveSOA Copyright
© 2005-2013 Software AG, Darmstadt, Germany
and/or Software AG USA, Inc., Reston, VA, United States of America,
and/or their licensors.</property>
    <property name="bidiLayoutOrientation">ltr</property>
    <html-property name="description">For a selected asset, lists all ↩
changes.</html-property>
    <list-property name="userProperties">
        <structure>
            <property name="name">locale</property>
            <property name="type">string</property>
            <property name="isVisible">false</property>
        </structure>
        <structure>
            <property name="name">KEY</property>
            <property name="type">string</property>
            <property name="isVisible">false</property>
        </structure>
    </list-property>
    <property name="locale">en-US</property>
```

BIRT no longer accepts the `locale` property, therefore you must delete the `structure` element that includes the `locale` property from the `list-property` element with `name="userProperties"`.

Installing CentraSite version 8.2 automatically adapts all reports, including user-defined reports, that it finds in the default report template file location, which is *http://localhost:53305/Centra-*

Site/CentraSite/ino:dav/ino:dav/projects/CentraSite/Reports. Any user-defined reports that contain the `locale` property and are in other locations must be modified manually.

For more information about BIRT's new support for multiple resource files, please see **New and Notable Features within BIRT 2.6**.

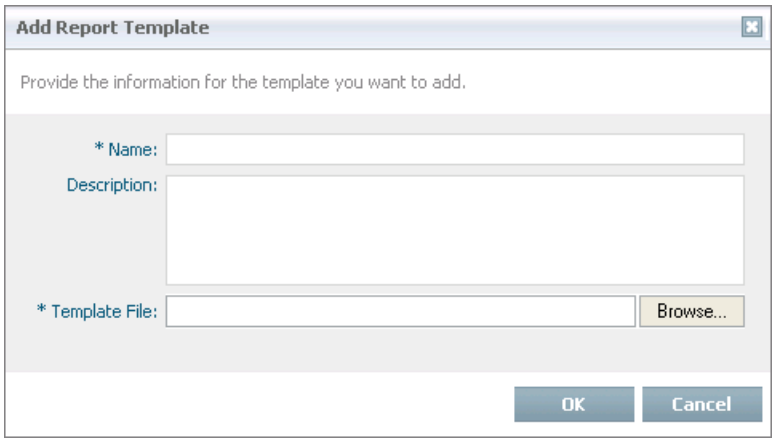
Migrating Reports into CentraSite

Apart from the incompatibility **discussed above**, a report that was created using an earlier version (or a combination of earlier versions) of CentraSite, Eclipse and/or BIRT can be migrated into the current version without loss of information. You can migrate the report in any of three ways:

- within a CentraSite GUI, using CentraSite Control;
- within a CentraSite GUI, using the CentraSite Eclipse perspective;
- using Eclipse BIRT functionality.

The first two options, working within a CentraSite GUI, involve creating a new report registry entry either by referring to an existing *rptdesign* file, or by importing an archive that contains a report. We do not discuss the latter method, because the steps are the same as when working with assets of other types.

If you prefer to use CentraSite Control to create a new report registry entry based on an existing *rptdesign* file, go to the CentraSite Control **Reports** tab. In the screen that shows the list of all report templates, choose the button **Add Report Template**, which is located in the upper left-hand corner. The following dialog window appears:



Choose **Browse...**, and navigate to the *rptdesign* file that you want to import. Specify the name and optionally a description of the new report. Choose **OK**. The *rptdesign* file is now added to the repository, a registry entry pointing to the file is created, and the new report template is added to the list of report templates.

If you prefer to use the CentraSite Eclipse Perspective to create a new report registry entry based on an existing *rptdesign* file, use the plus icon in the list of predefined reports. To open this list, open the **Administration View**, then double-click on the entry **Report Templates**.

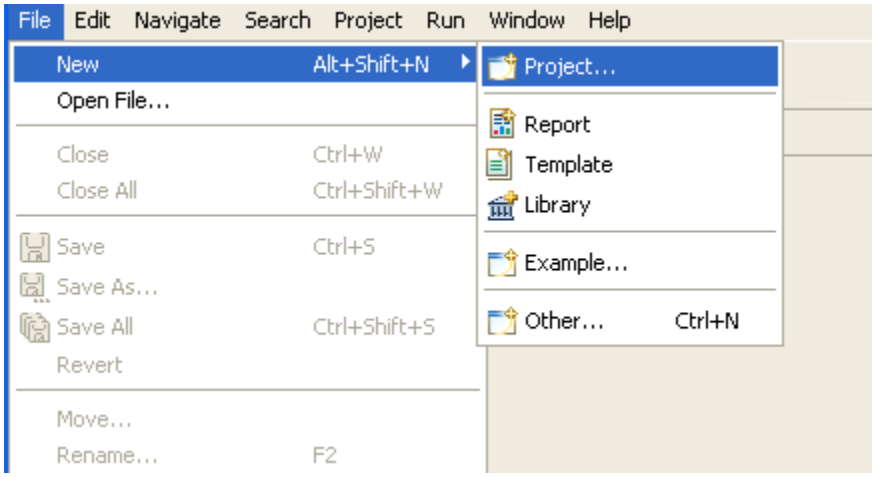
If you prefer to use Eclipse BIRT functionality, you can add an *rptdesign* file to any folder in the Eclipse workspace that represents a BIRT Reporting Project.

4 Modifying a Predefined Report

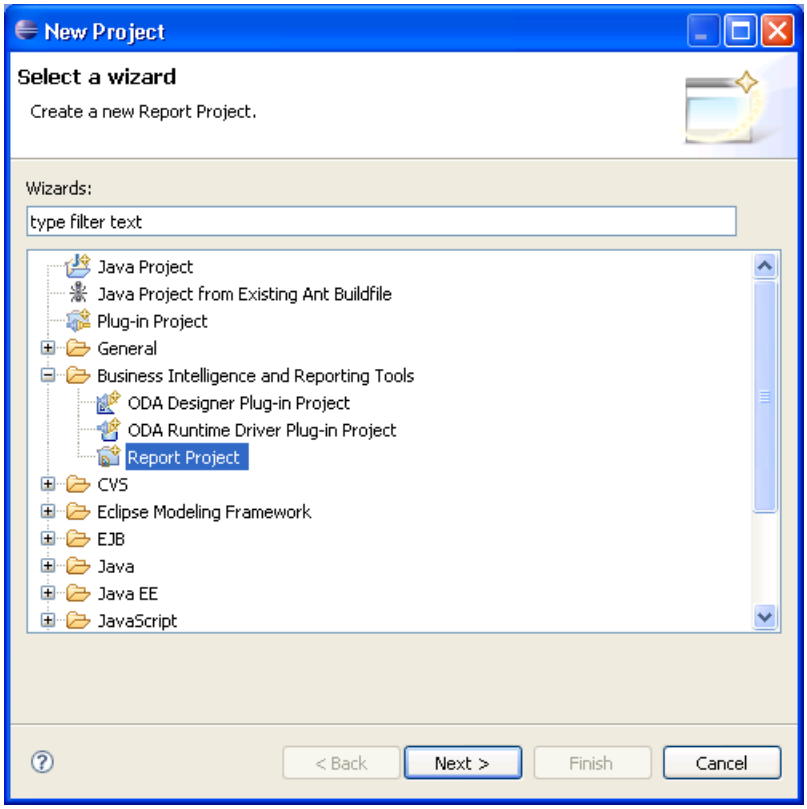
The predefined reports introduced above can be used as they stand for retrieving information about the contents of the CentraSite data storage, but in addition they can also serve as an easy introduction to writing your own reports: you can load one of the predefined reports into the Eclipse workspace and modify it to suit your own requirements.

Loading a Predefined Report into the Workspace

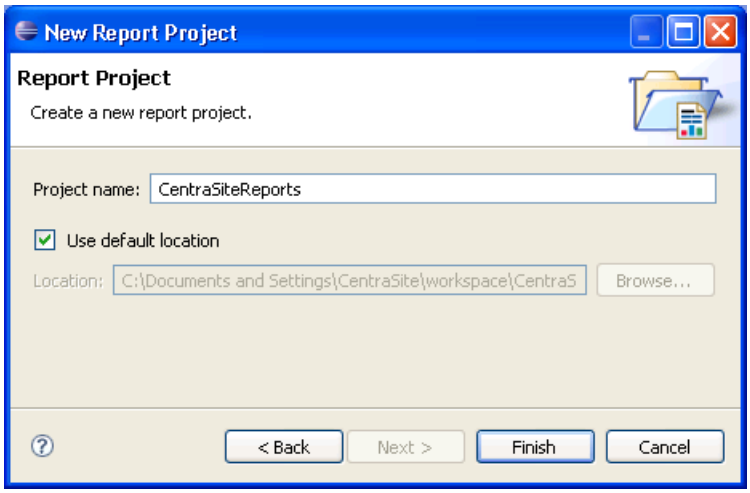
The first step is to load the appropriate *rptdesign* file into the Eclipse workspace. Start by creating a suitable Eclipse project. Select Project... from the Eclipse workbench's **File** > **New** menu item:



This opens a pop-up that offers several kinds of projects. Select "Business Intelligence and Reporting Tools" > "Report Project":



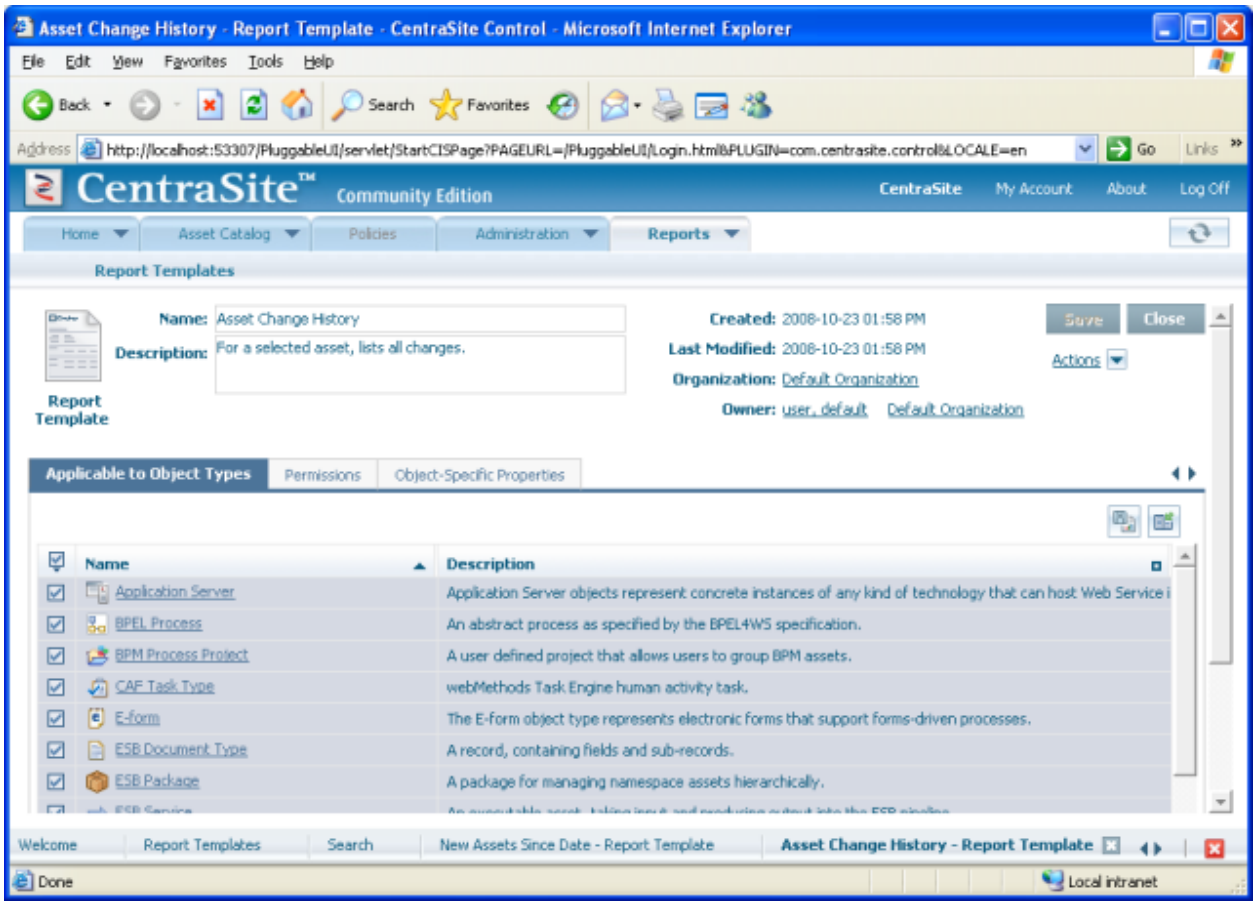
Choose **Next**. The following pop-up prompts you to enter a name for the project:



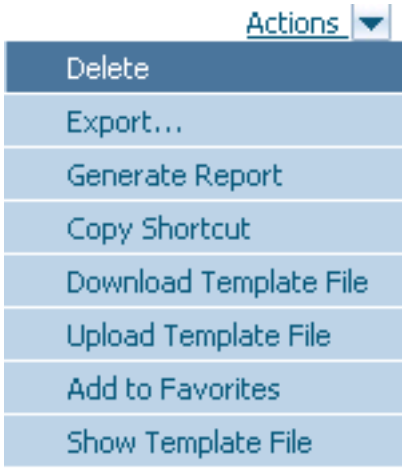
Choose **Finish** to create the new project.

Loading from CentraSite Control

You can load a report from CentraSite data storage into the Eclipse workspace from within CentraSite Control from the detail view of the report. Choose the report's name in the report object listing:



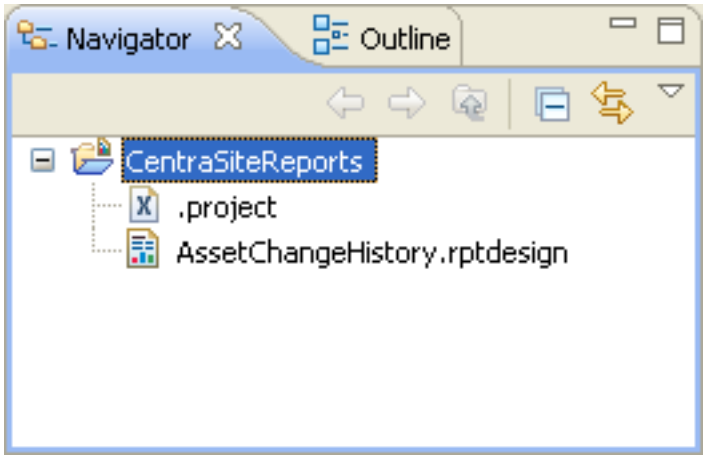
Choose the **Actions** button. You now see a list of actions that can be performed on the current registry object; in the case of a report, the list includes the action **Download Template File**.



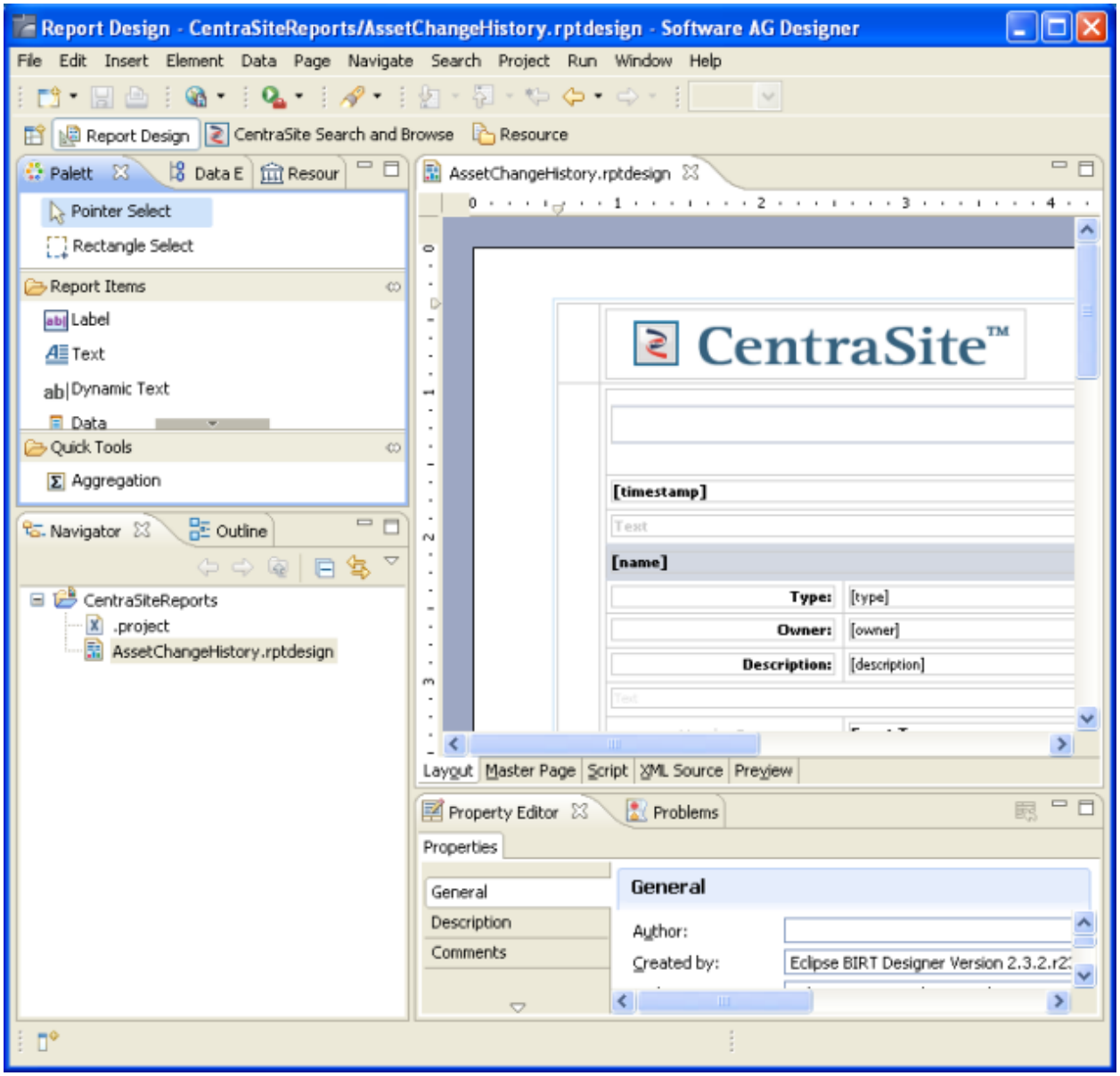
Choose **Download Template File**. A common file download dialog starts. Move the report's *rptdesign* file into the project folder of the Eclipse workspace that corresponds to the reporting project that you previously created. If you are unsure of the workspace's precise location, consult the **Switch Workspace** menu item in your Eclipse's **File** menu.

The system suggests the name *AssetChangeHistory_rptdesign* for the new template file. Change this to *AssetChangeHistory.rptdesign* before saving, in order to make the file recognizable as a template file.

It may be necessary to refresh the reporting project to show the new report; to do this, choose **Refresh** from the context menu of the report project's entry in the Eclipse Navigator view.



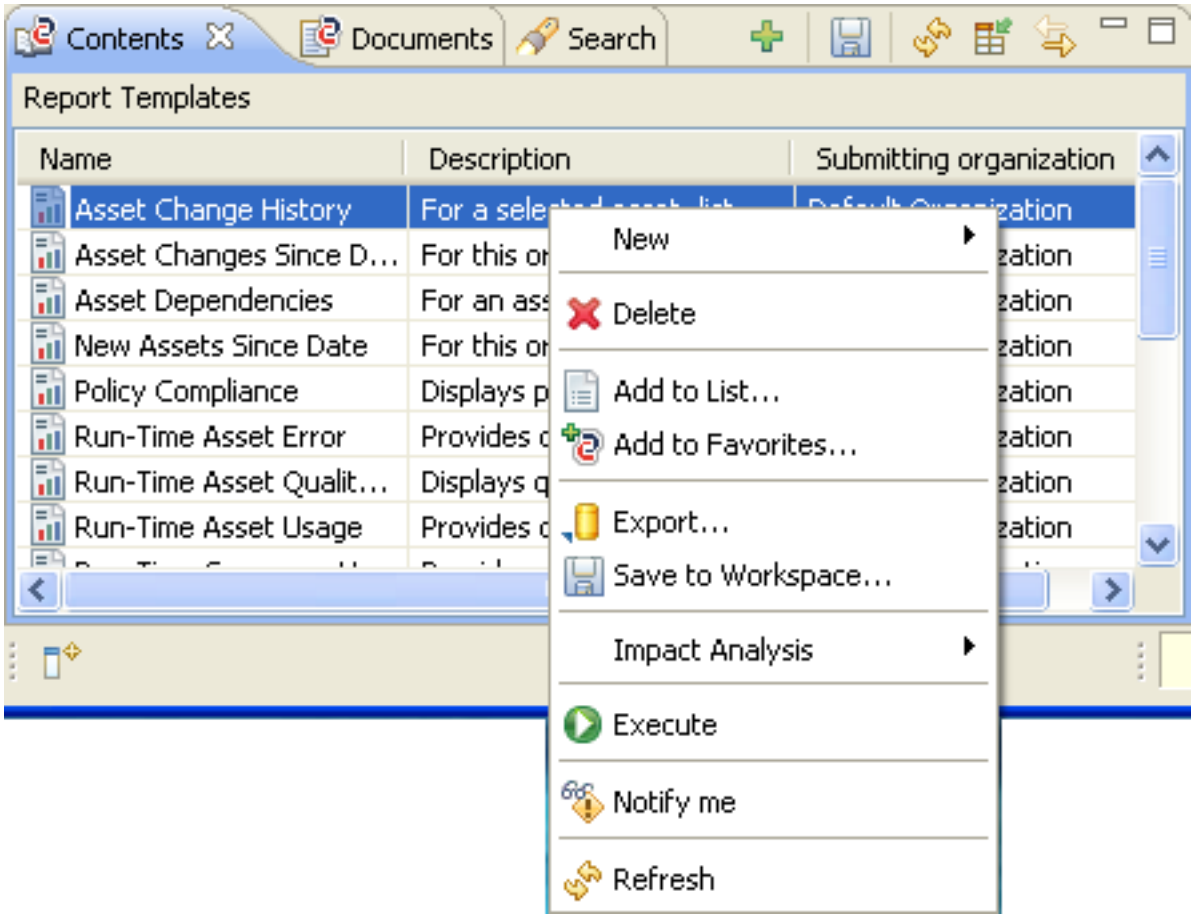
Double-click on the new report entry. The report is opened in the report editor:



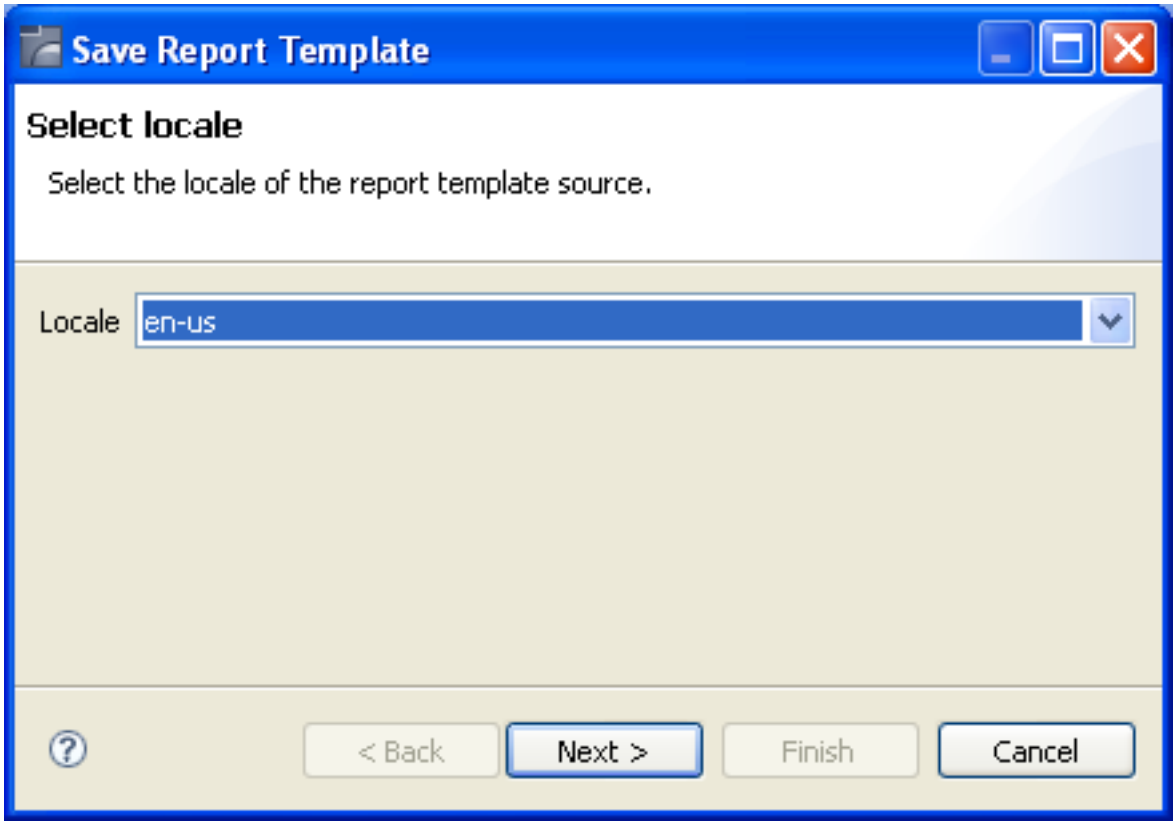
Proceed to the section **Modifying the Report**.

Loading from Eclipse

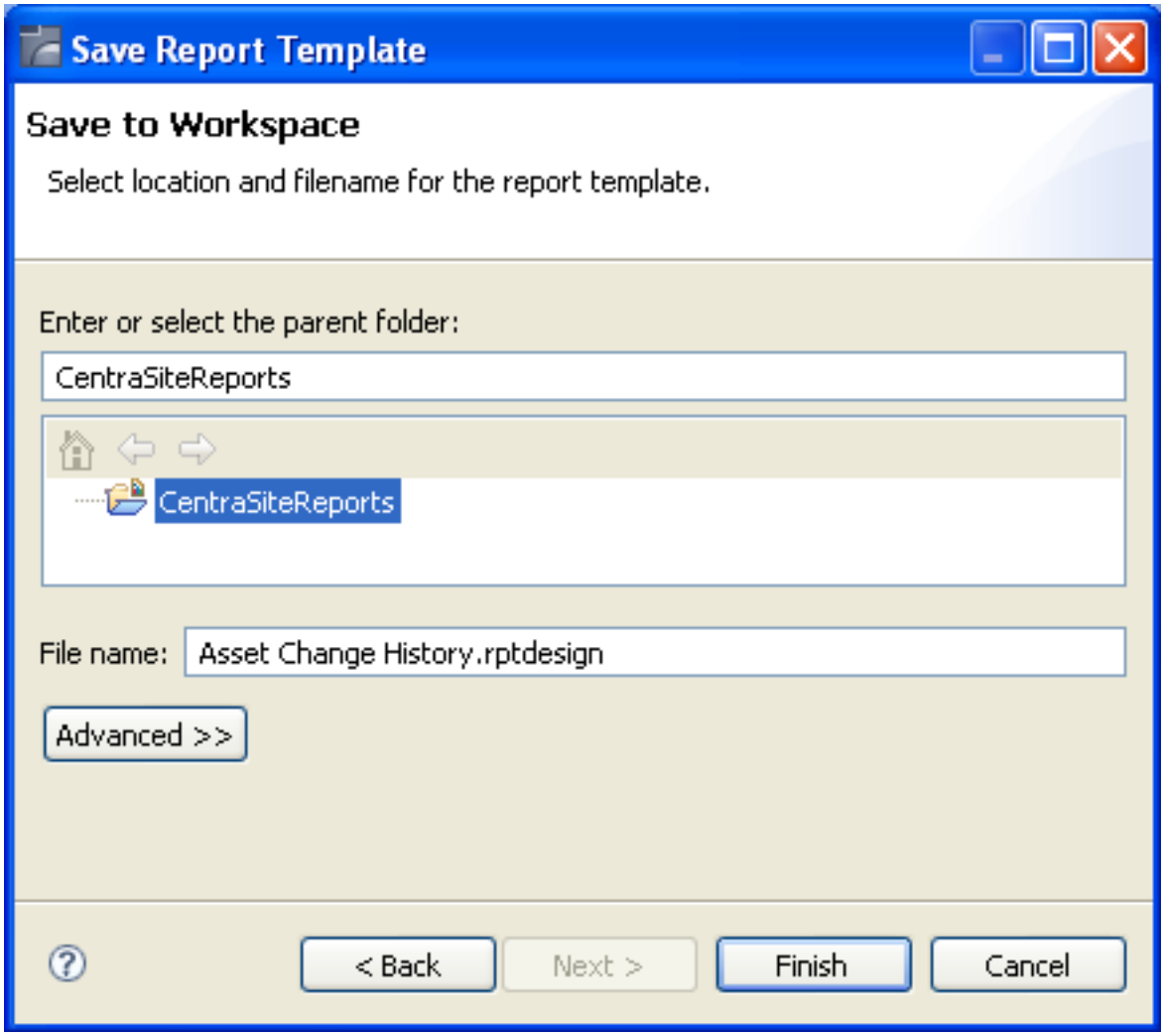
To save a report to the workspace from within Eclipse's CentraSite perspective, double-click on the entry **Report Templates** in the Administration View; the list of reports is now opened. Choose **Save to Workspace** from the appropriate list entry's context menu:



Since reports are localized, a pop-up menu now asks which version of the report should be downloaded. If no language packs have been installed, the only choice is `en-us`.



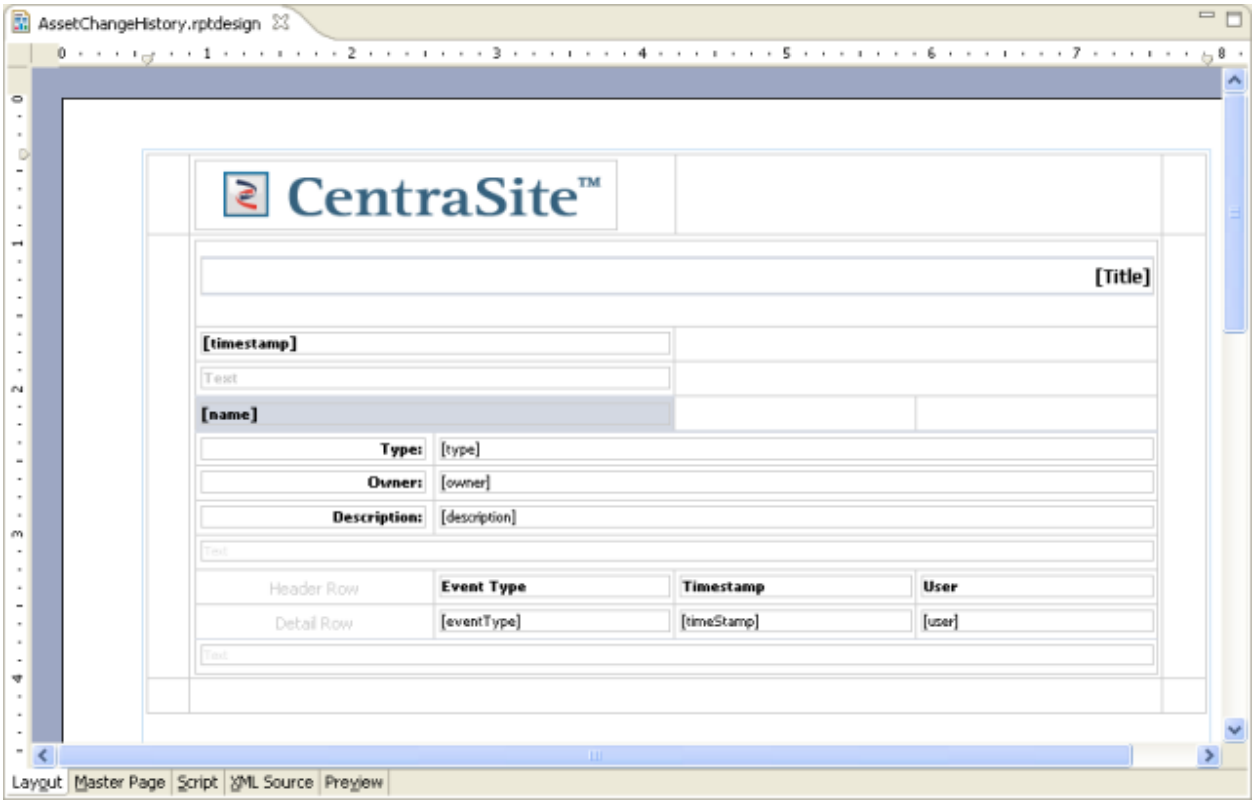
Finally, in the next pop-up menu, specify the workspace project into which the report should be stored:



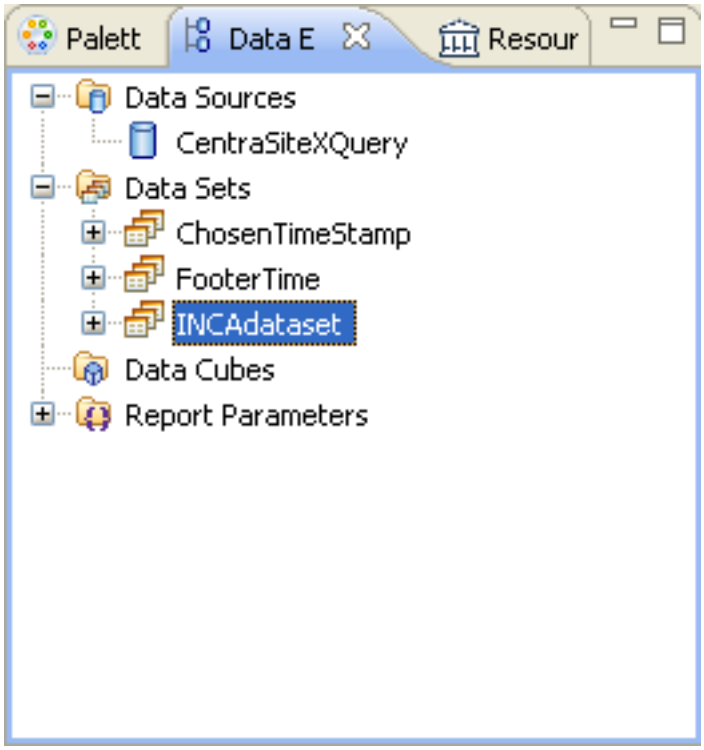
Modifying the Report

The report that we want to modify is named "Asset Change History". It reports all asset changes that were made after a specified time. You are going to change this report by removing the time parameter, so that the report will show all asset changes, irrespective of when they were made. Then you will add a chart that shows the number of asset changes made by each user.

Open the report by double-clicking its entry in the Navigator view. Doing this also enters the Eclipse BIRT perspective. The following screenshot shows an excerpt of this perspective, namely the report's layout as shown in the Eclipse BIRT perspective's report editor:



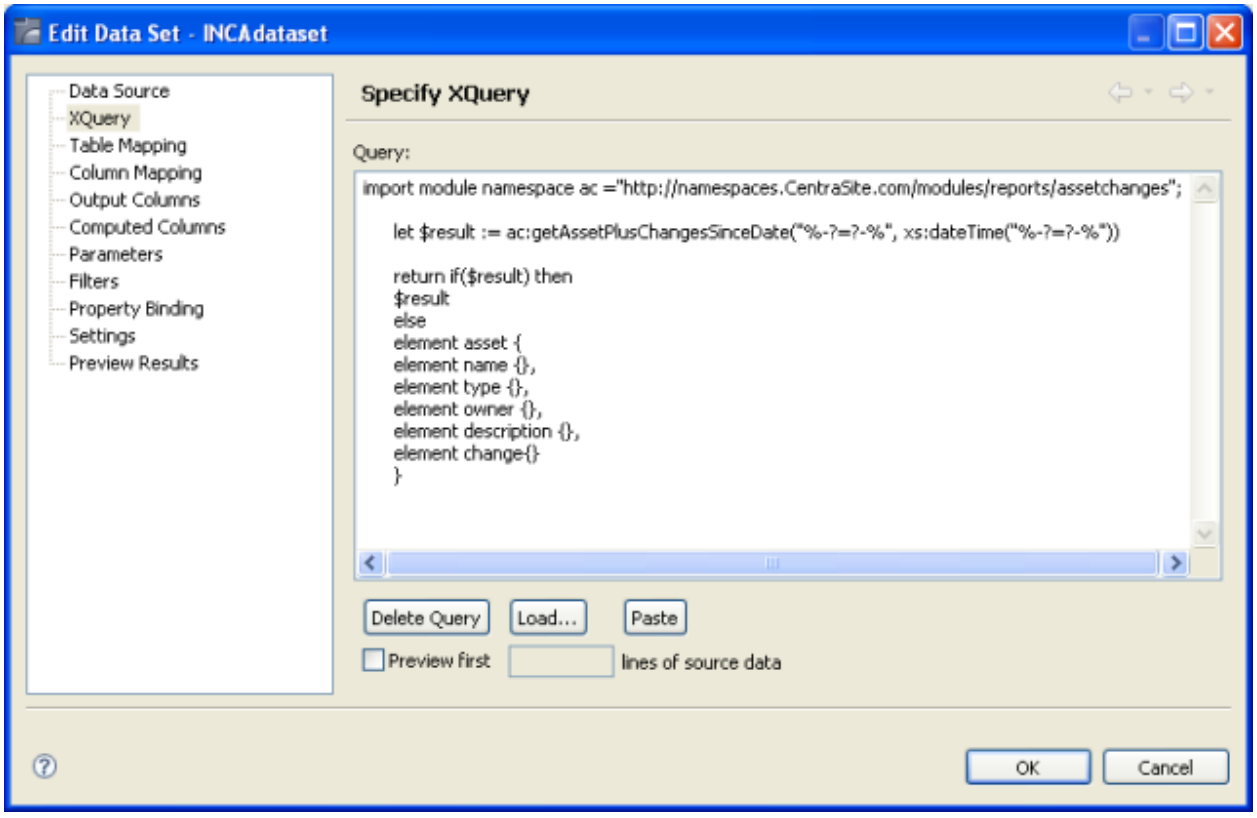
The report's **Data Explorer View** (see below) shows that the report's single **Data Source** is "CentraSiteXQuery". This is, the data that the report requests at runtime stem from a query addressed to the CentraSite data storage. Furthermore, there are three **Data Sets**, i.e. the report issues three different queries to obtain its data:



In the next section, we examine the data set *INCAdataset* that retrieves the changes that have been made to the asset.

Modifying a Data Set

Open the first data set, namely *INCAdataset*, by double-clicking on its entry in the data explorer:



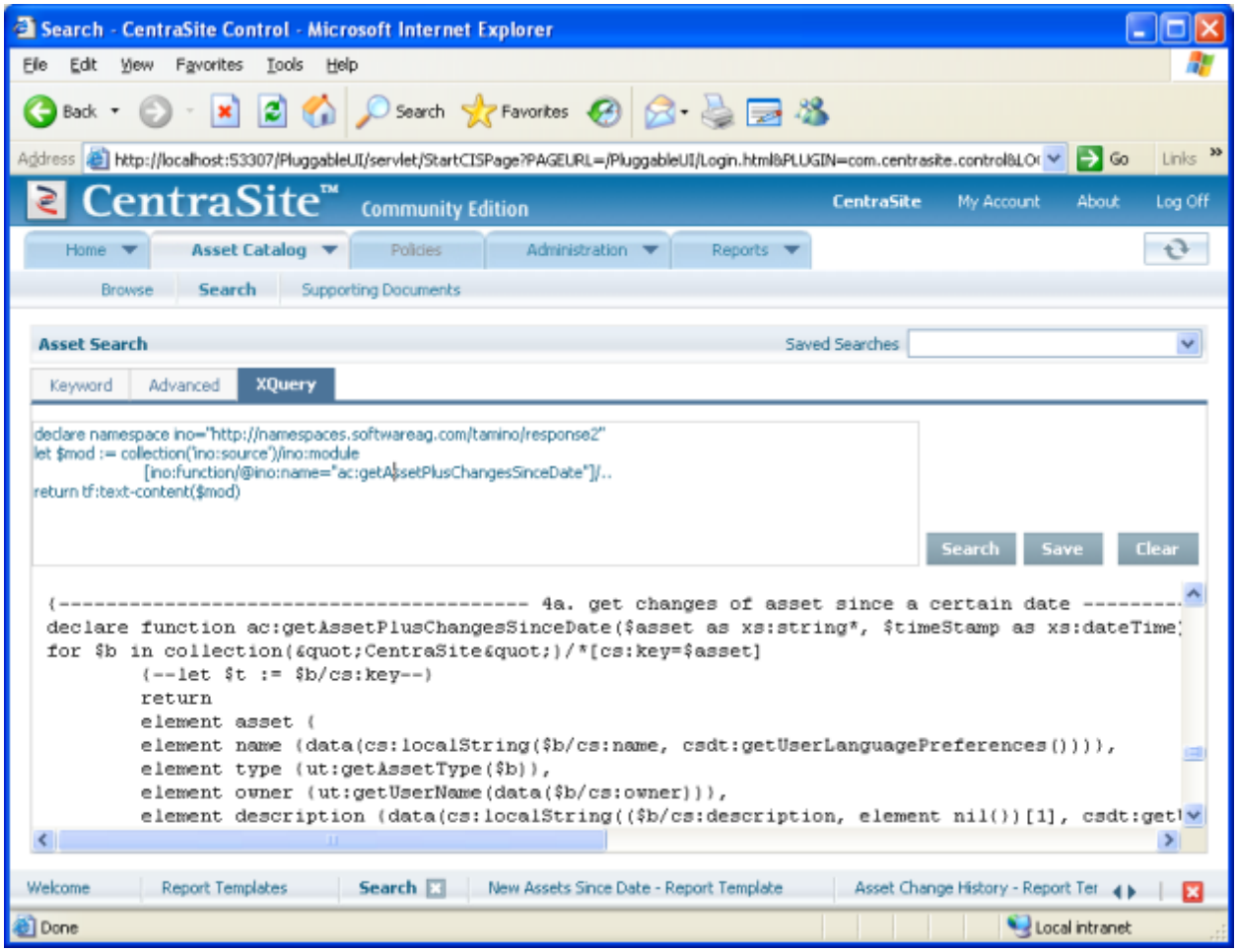
We want to retrieve all changes made to the asset irrespective of their timestamps, so we must eliminate the time parameter from the query. We can see straightaway that adapting the XQuery directly is not feasible, since the actual query is contained in a module and therefore is not visible in the data set view.

In order to see the actual function, we use CentraSite Control's XQuery features to access the XQuery module to which the data set refers.

In CentraSite Control, choose **Asset Catalog** > **Search** > **XQuery** and replace the query with:

```
declare namespace ino="http://namespaces.softwareag.com/tamino/response2"
let $mod := collection('ino:source')/ino:module
          [ino:function/@ino:name="ac:getAssetPlusChangesSinceDate"]/..
return tf:text-content($mod)
```

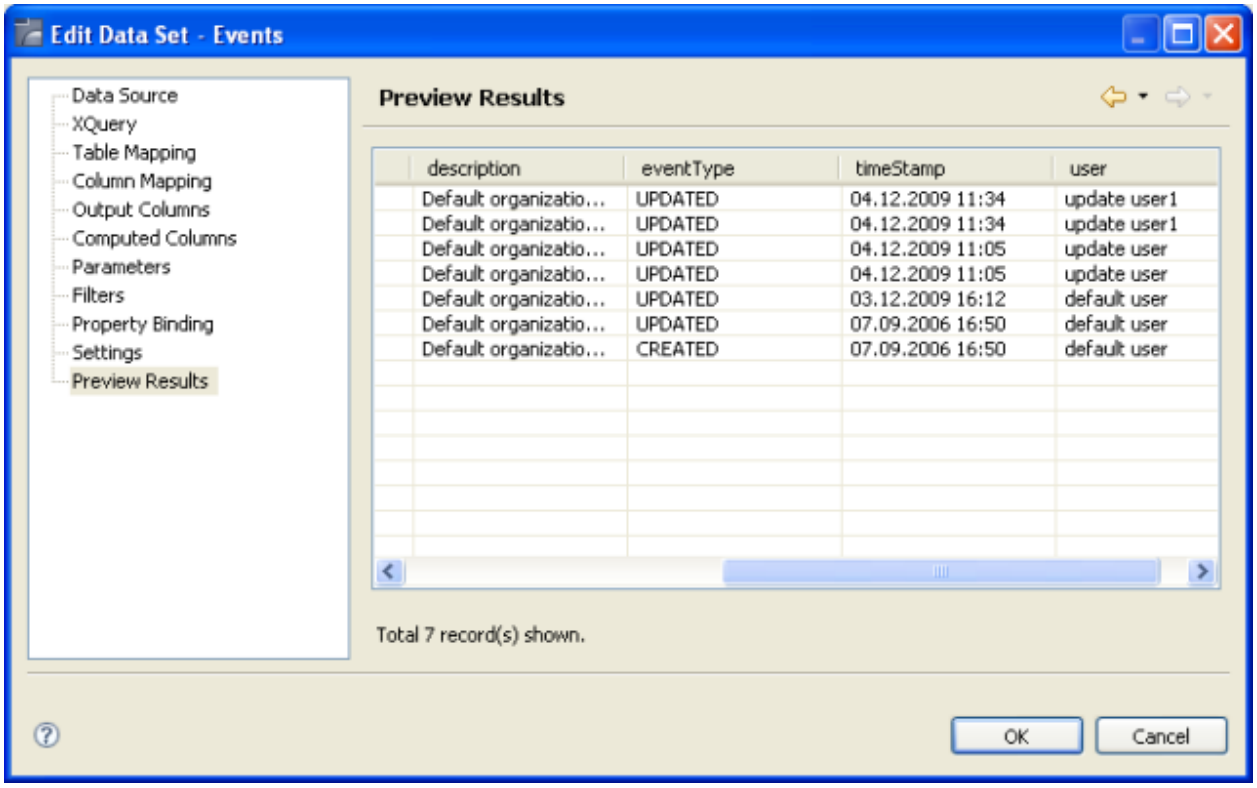
To see the actual function, scroll down:



Inspecting the predefined functions in the module, we see that the first function, `ac:getAssetChanges`, does what we need. Thus we change the query in the data set *INCAdataset*. The line that calls the function becomes `let $result := ac:getAssetChanges("%-?=uddi:207ff1cc-25c5-544c-415c-5d98ea91060c?-%")`, the key being the one that points to the predefined organization `Default Organization`. This key can be taken from the report's parameter default value. The actual value here is necessary in order to allow the data set preparation mechanism to perform a call to obtain the result format. The delimiters tell the report execution mechanism to replace the parameter by the actual value at runtime.

It is sometimes difficult to change a data set. Alternatively, it is possible to create a new data set with a different name and replace the data set in the `Binding` section of the data set using controls. In any case, ensure that the data set specifies exactly one parameter, namely `ROKEY`, to contain the asset's key.

After all this, the data set preview should show the proper result, i.e. all changes applied to the default organization, which might look like the following:



Since the time parameter is no longer relevant, you should now delete the line in the report table that shows the input value, and later on the report parameter.

Save the changes that you have made to the report. Run the report in BIRT preview mode, by choosing the **Preview** tab at the bottom of the report editor (instead of the usual **Layout** tab). The next screenshot shows an excerpt from the report:

Default Organization

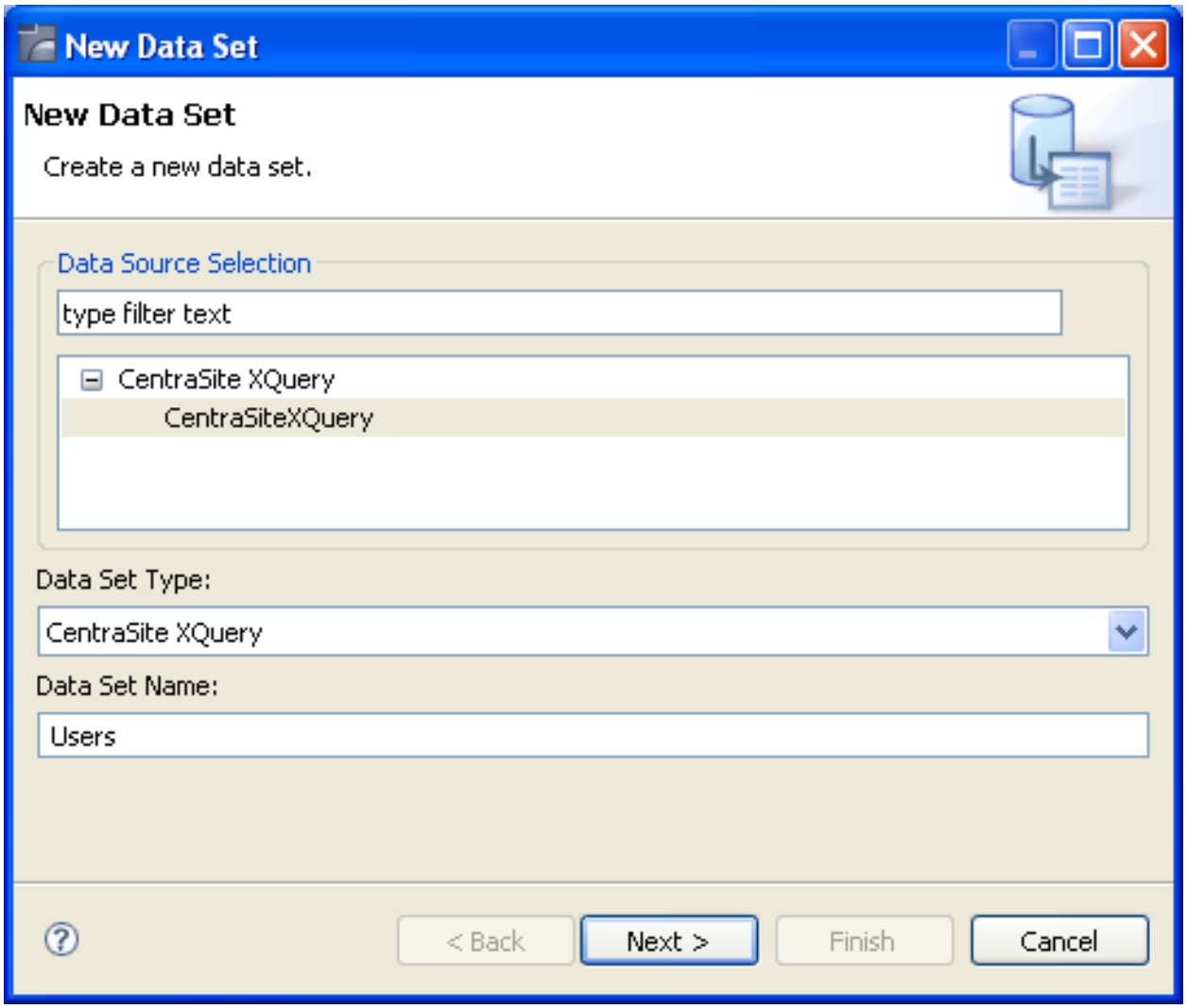
Type: Organization
Owner: default user
Description: Default organization of CentraSite.

Event Type	Timestamp	User
UPDATED	04.12.2009 11:34	update user1
UPDATED	04.12.2009 11:34	update user1
UPDATED	04.12.2009 11:05	update user
UPDATED	04.12.2009 11:05	update user
UPDATED	03.12.2009 16:12	default user
UPDATED	07.09.2006 16:50	default user
CREATED	07.09.2006 16:50	default user

For more information about the XQuery language, please refer to the W3C XQuery page at **http://www.w3.org/standards/xml/query**.

Adding a New Data Set

Before you can add the chart that shows the number of times each asset has been changed, you need a data set that provides pairs of user names and numbers of changes. Create a data set, using either **New Data Set** from the context menu of the entry **Data Sets** in the Data Explorer, or the **Data** menu item. Set the name of the new data set to *Users*.

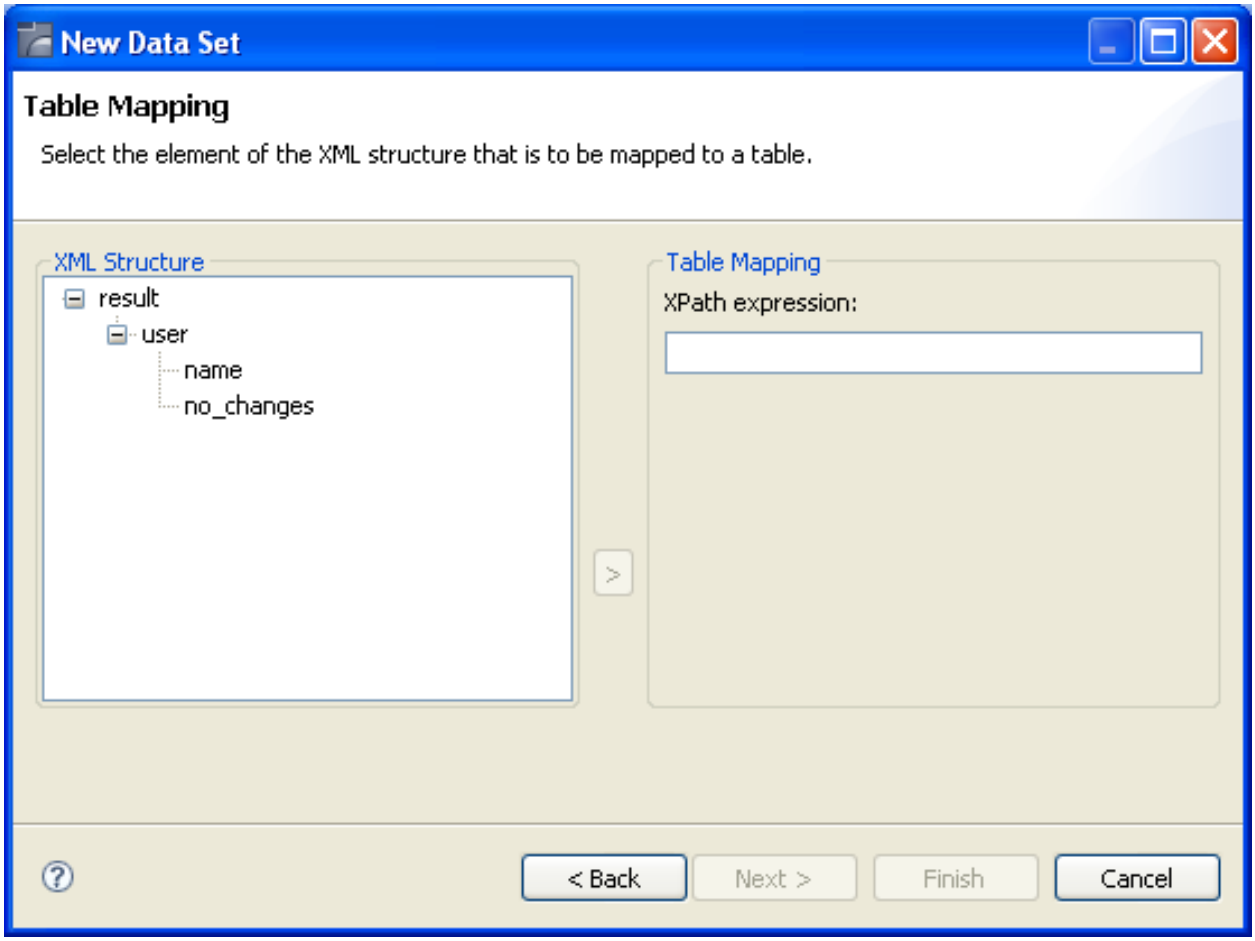


The following query retrieves the data required for building the chart:

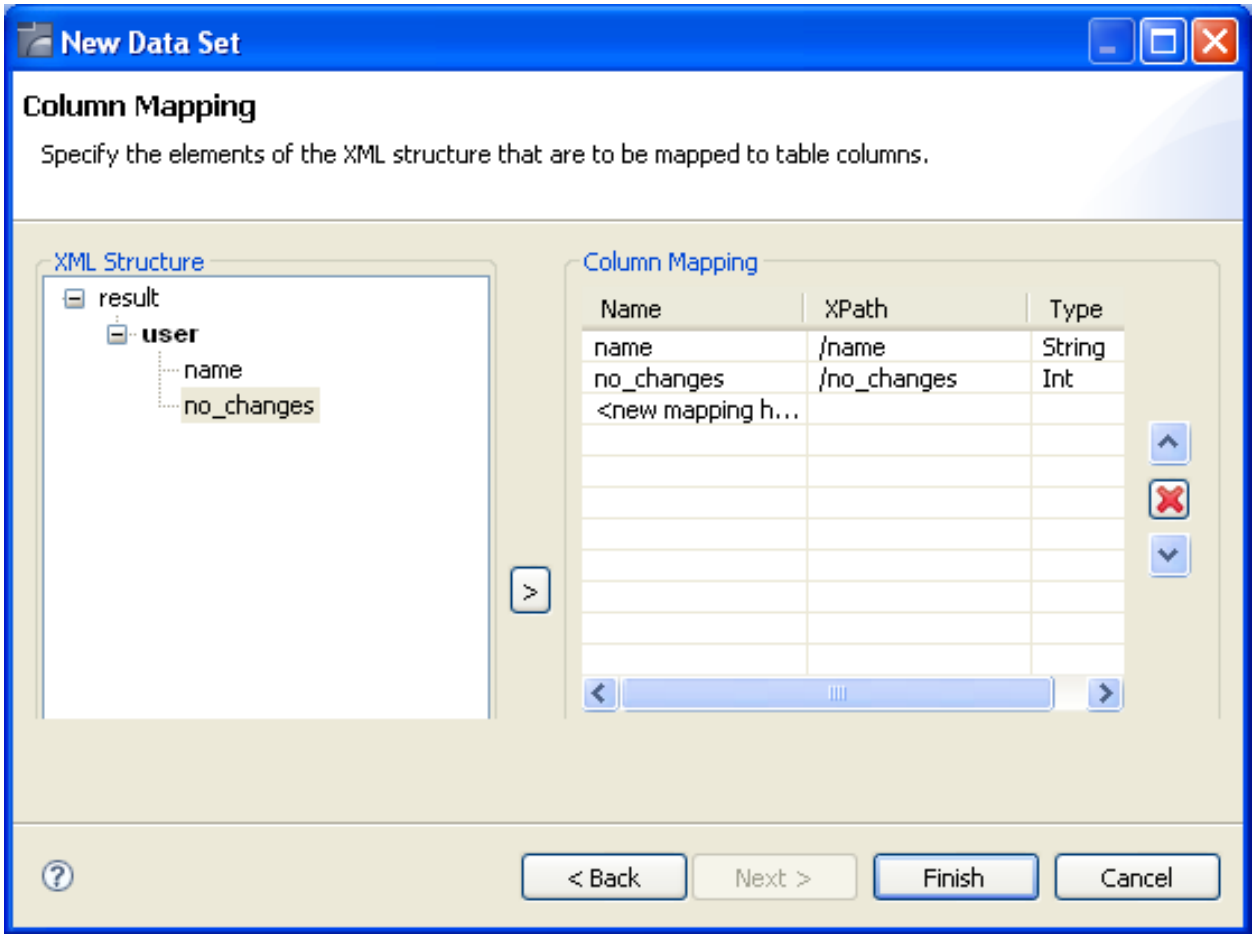
```
import module namespace ut = "http://namespaces.CentraSite.com/modules/util";
import module namespace cs = "http://namespaces.CentraSite.com/Schema/jaxr";

for $b in collection("CentraSite")/*
            [cs:key="%-?=uddi:207ff1cc-25c5-544c-415c-5d98ea91060c?-%"]
let $t := $b/cs:key
let $events := collection("CentraSite")/cs:auditableEvent[.//cs:registryObject=$t]
for $owner in distinct-values($events/cs:owner)
return <user>
         <name>{ut:getUserName(data($owner))}</name>
         <no_changes>{count($events[cs:owner = $owner])}</no_changes>
       </user>
```

Choose **Next** in the query specification step of the **New Data Set** dialog to proceed to the **Define Table Mapping** step. Expand the result **node** of the result tree, select **asset** and click on the arrow icon. The asset is now declared as the anchor of the new table:

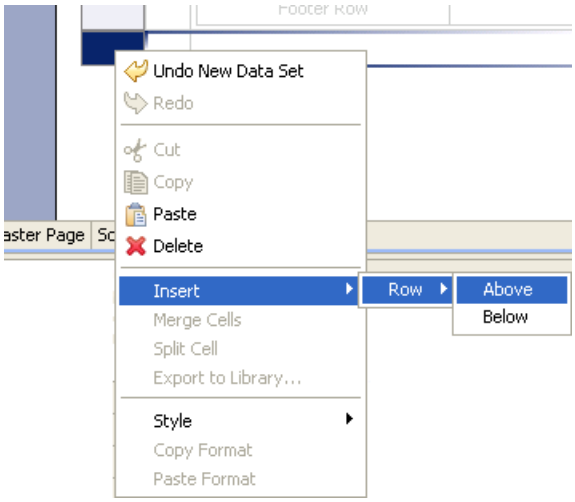


In the **Define Column Mapping** step, select both fields under `user` as columns in the new table. Be sure to declare `no_changes` as being of type `Int`:

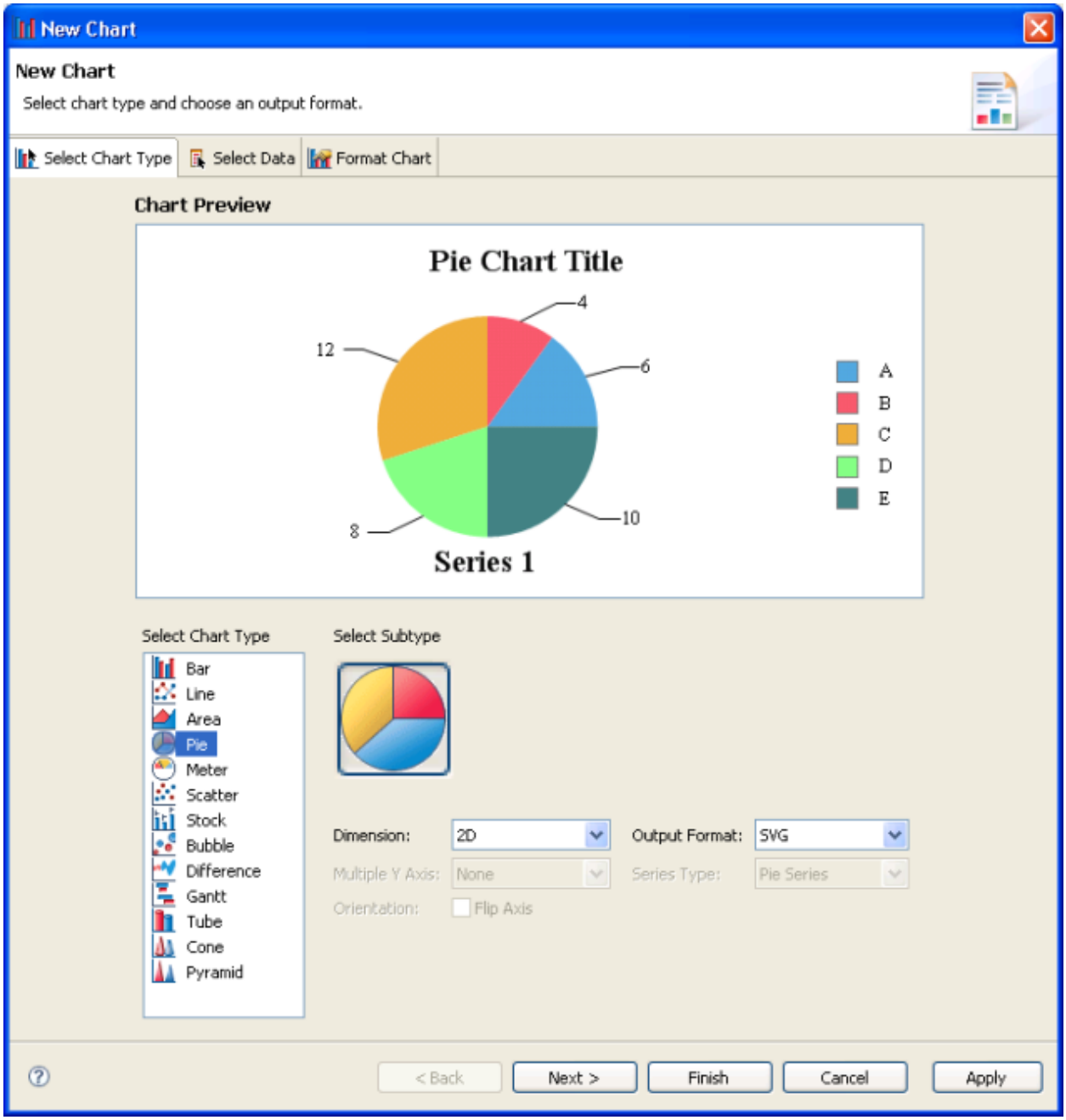


Adding a Chart

In order to disturb the existing appearance of the report as little as possible, insert a new row preceding the last row of the report's grid. To do this, right-click on the lower left-hand corner of the grid and choose **Insert** > **Row** > **Above**:



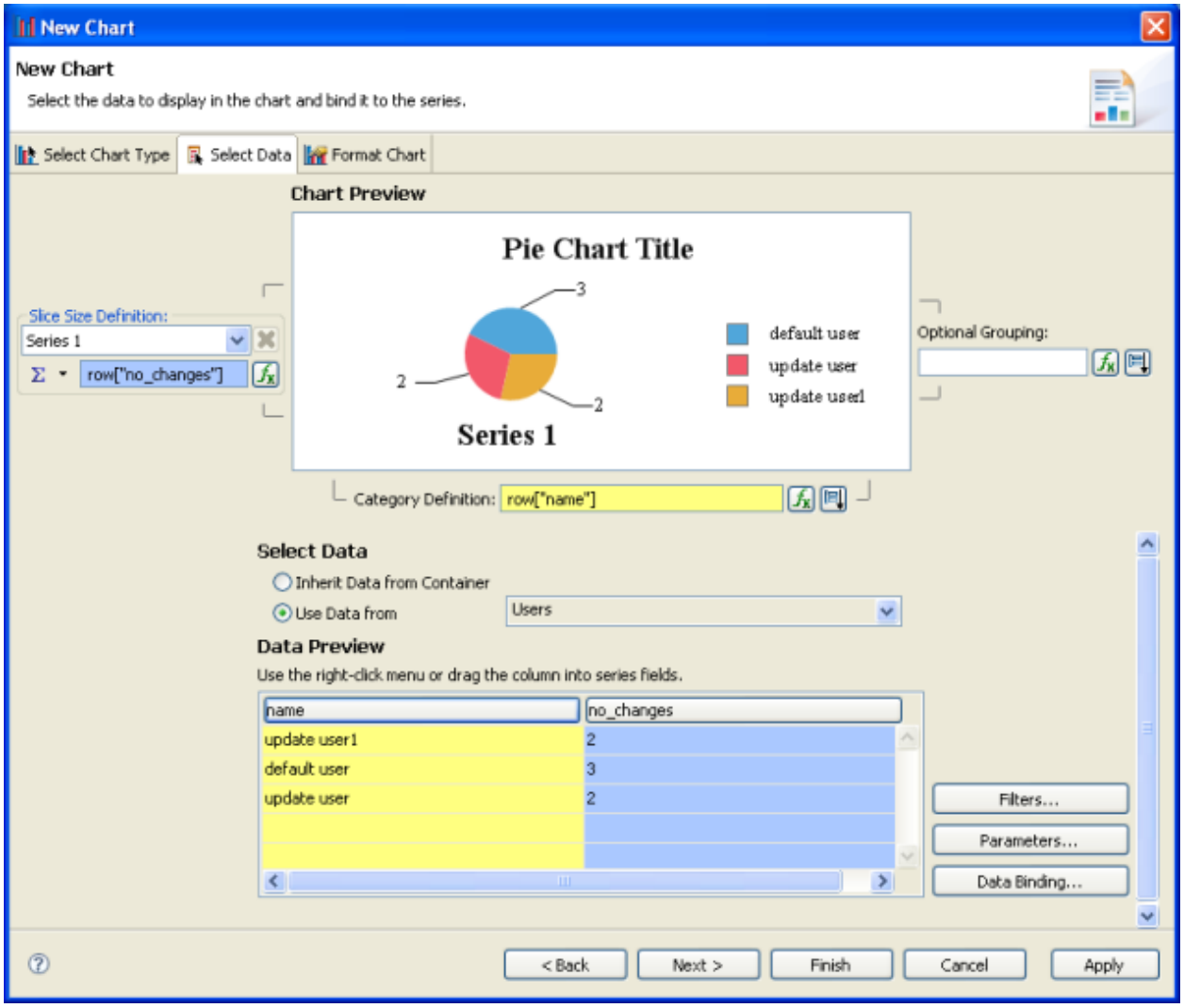
Now open the **Palette** view and drag the **Chart** tool into the report editor in the newly-created row below the table. This opens the **New Chart** dialog. Select "Pie" as the chart's type:



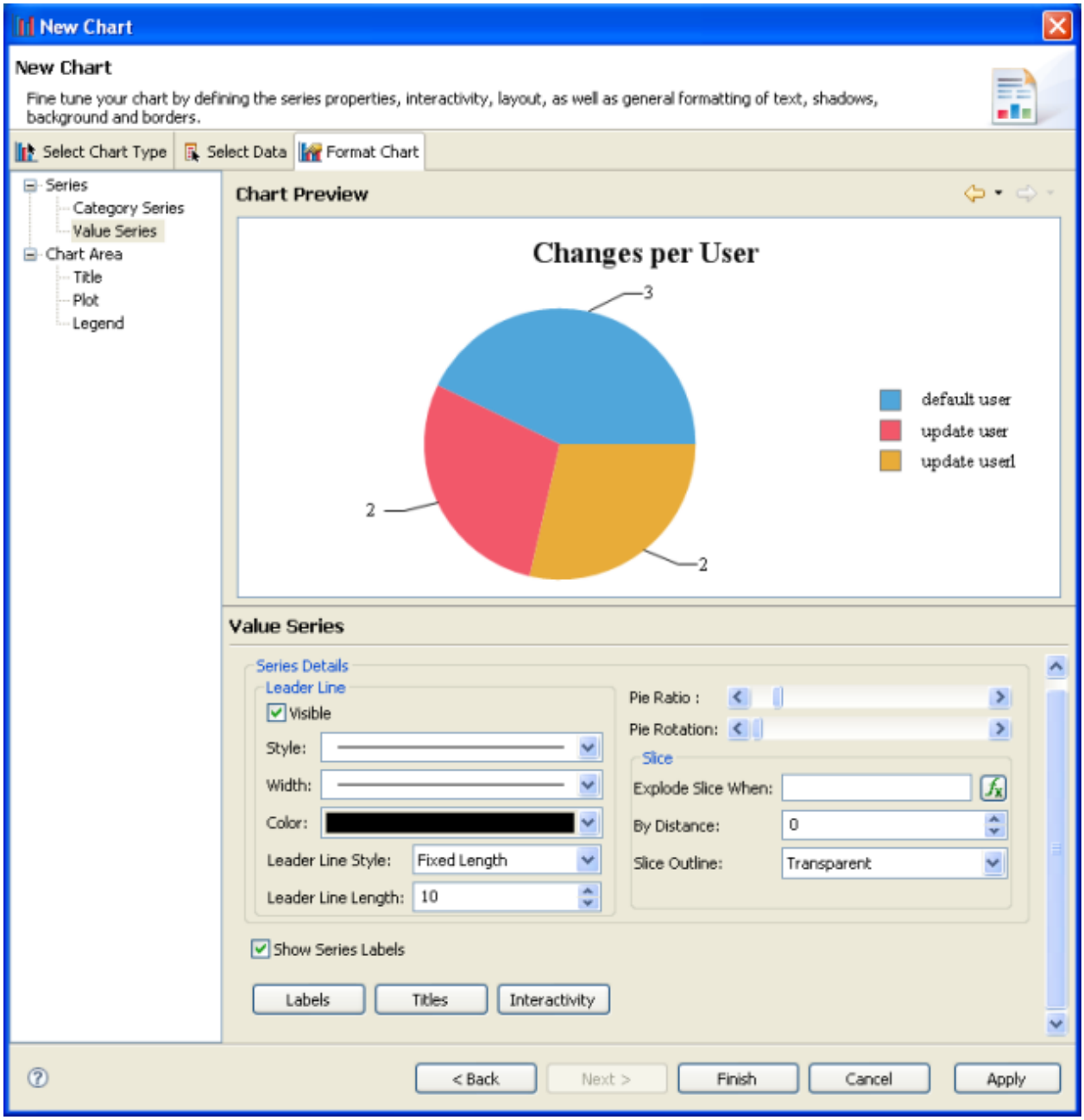
Choose **Next**. You now see the second chart specification screen, which specifies the data to be displayed by the chart.

Firstly, find the **Select Data** section of the screen, and select the radio button **Use Data**.In the associated selection box, select the data set *Users*. A preview of this data set's result now appears in the **Data Preview** section below. To specify how the chart will display the data, drag the left column

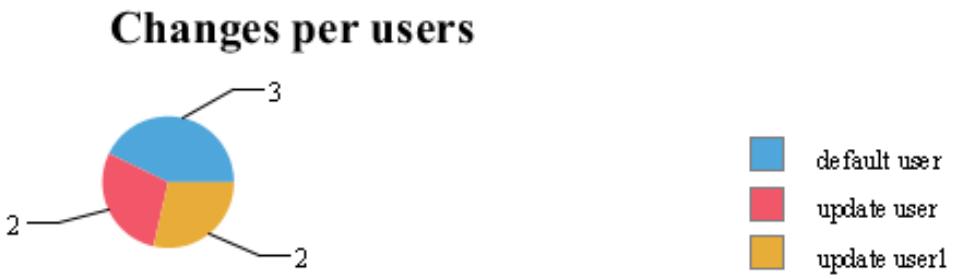
header (**name**) into the input field to the right of **Category Definition** in the **Chart Preview** section. Next, drag the right column header (**no_changes**) into the lower input field below **Slice Size Definition**, also in the **Chart Preview** section:



The third and last dialog box for chart creation is for fine tuning. In the tree view on the left, select **Chart Area** > **Title** and change the title to "Number of Changes". Select **Series** > **Value Series**, choose the **Titles** button at the bottom, and uncheck the **Visible** checkbox:

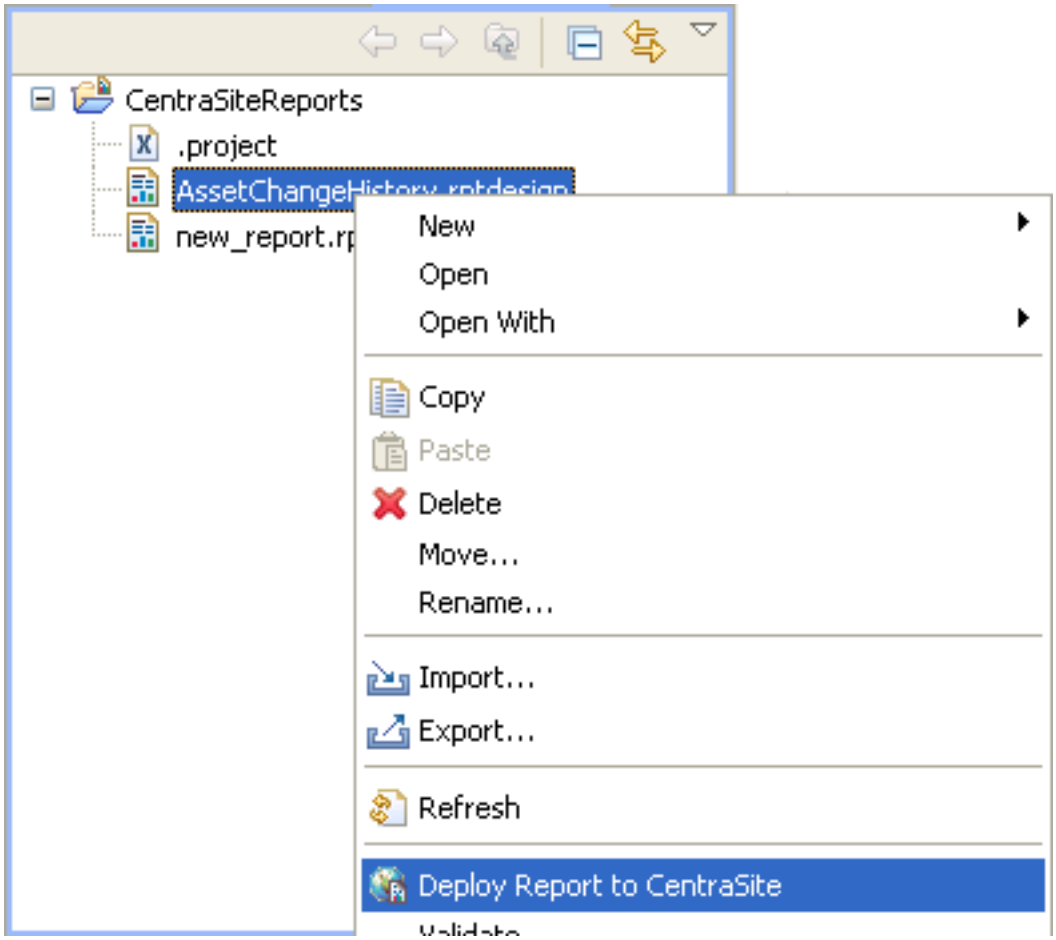


Choose **Finish**. The chart is now created in the report. Save the report, and preview it. The following is an excerpt from the bottom of the report, showing the new chart:

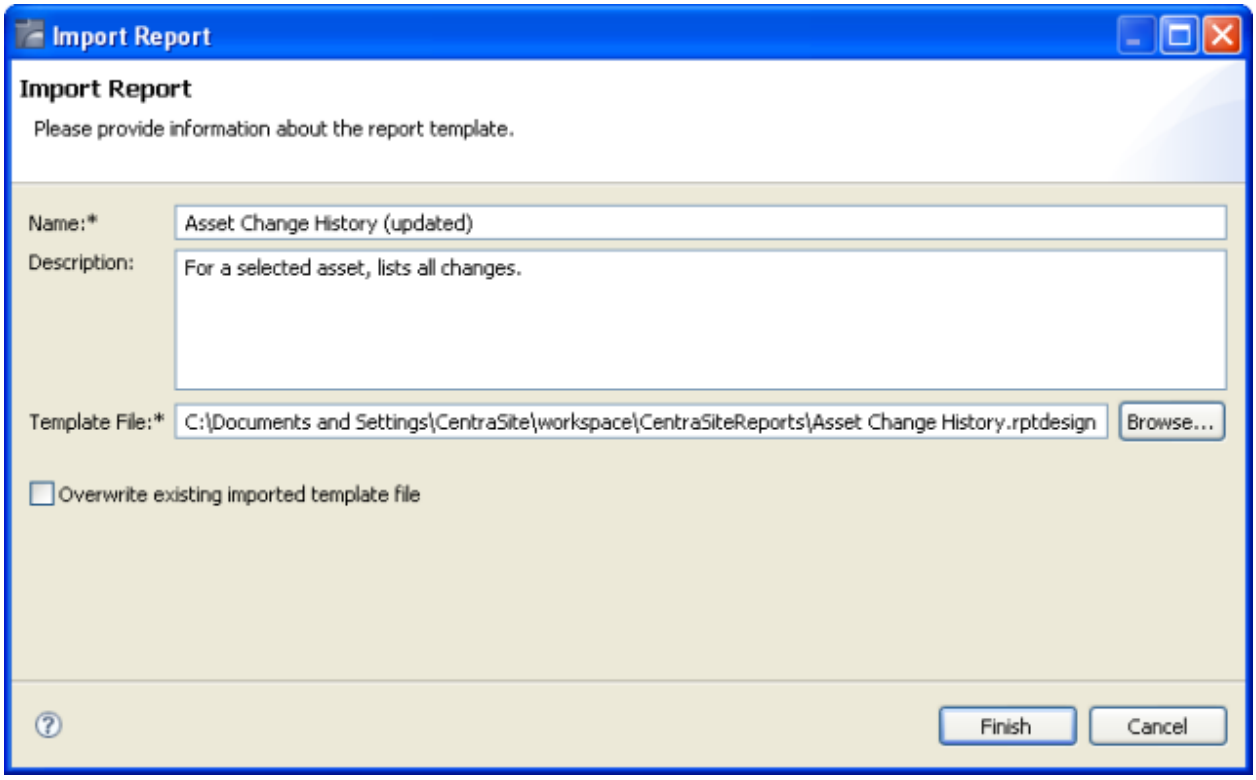


Writing the Modified Report Back to CentraSite

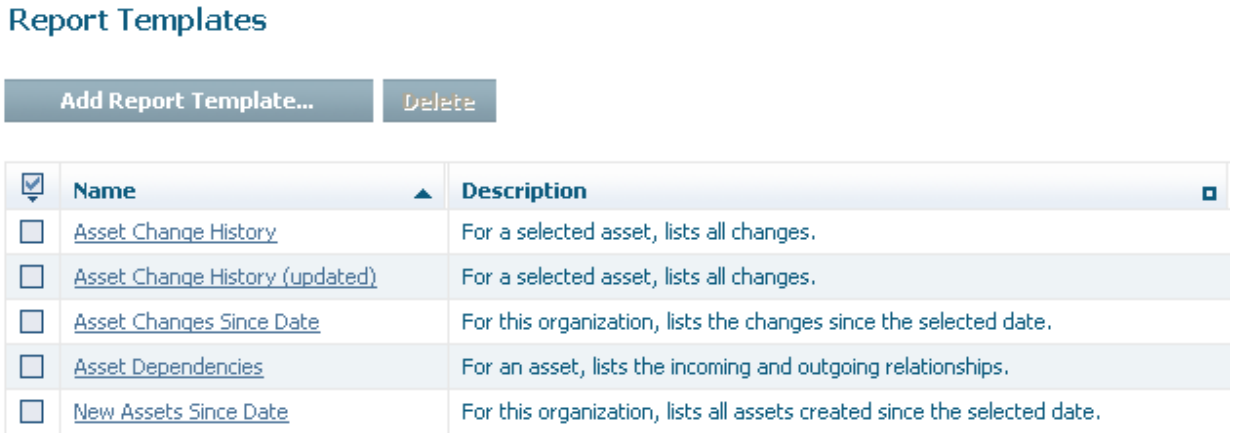
To make the report accessible to CentraSite Control, it must be deployed to CentraSite. To deploy the report, choose the option **Deploy Report to CentraSite** from the report's context menu in the **Navigator** view. Note that you can only do this if you have an ActiveSOA licence for your CentraSite installation.



A pop-up now prompts you to enter details about the report to be deployed. Add the suffix (updated) to the name of the report, so that you can distinguish it from the original:



The report now appears in CentraSite Control's report listing and can be executed from there. The next screenshot shows the beginning of the report listing (sorted alphabetically):



Alternatively, the modified report can be linked to any object type, just as the predefined reports can also be linked to any object type.

5 Building a Report From Scratch

This chapter explains how to create a report from scratch and introduces three examples. A special feature of the first example is that it uses the data source CentraSite, whereas all the predefined reports use the data source CentraSite XQuery. The CentraSite data source is GUI-based and is simpler to use because it does not involve XQuery.

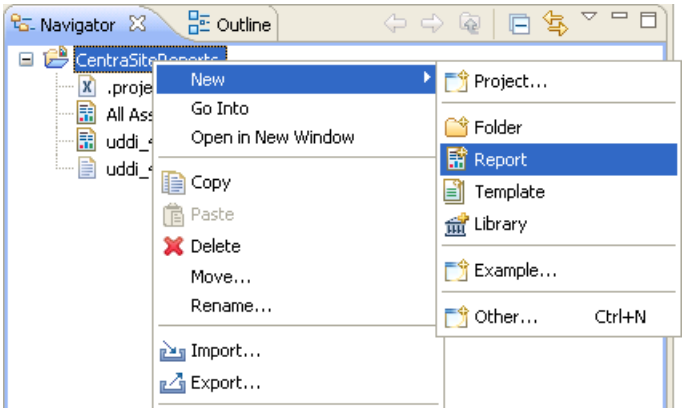
The second report displays a CentraSite taxonomy pertaining to the music example.

The third report has a parameter, namely a service's UDDI key, and displays the details of this service.

All Music Services

The CentraSite BIRT plugin extends the BIRT reporting facilities. It does so by adding two data sources, namely CentraSite and CentraSite XQuery. A data source in the sense of BIRT is a means of contacting a specific data provider. CentraSite XQuery allows data stored in CentraSite to be accessed by means of XQuery. The CentraSite data source allows you to specify the data stream via a GUI. We shall use the CentraSite data source to create a report that lists some details of specific services.

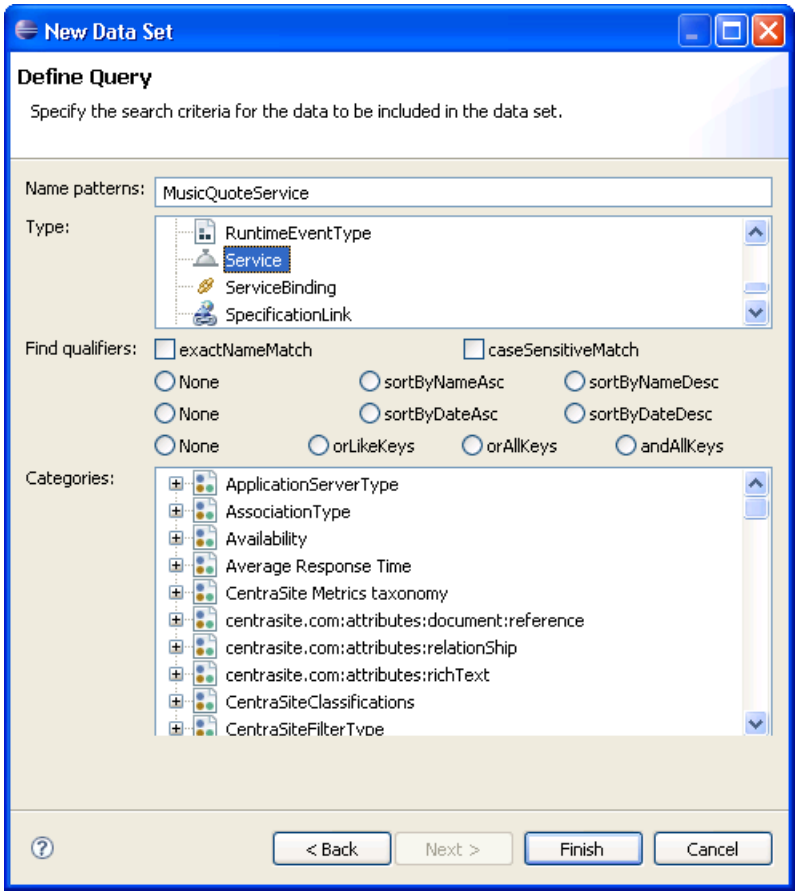
To create a report, choose **New** > **Report** from the CentraSiteReports project's context menu in the **Navigator** view:



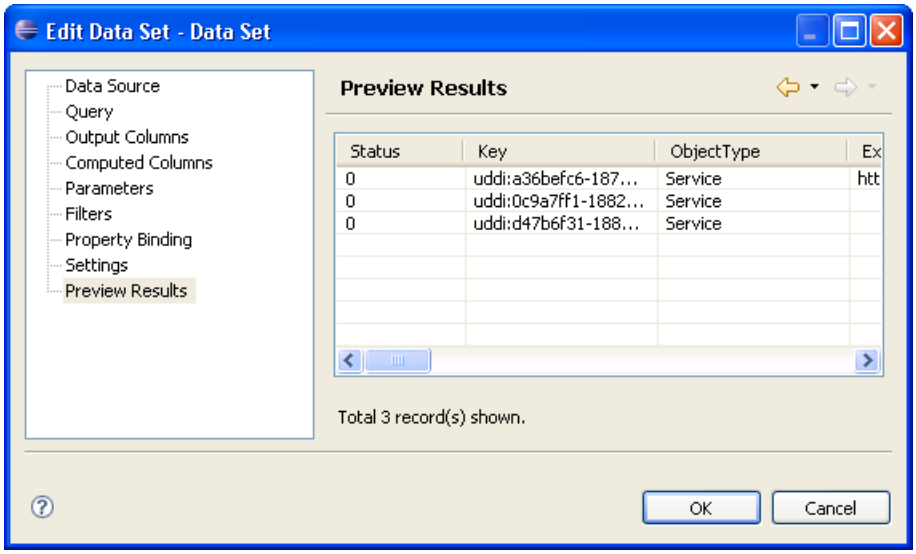
Give the report the name *MusicServices.rptdesign*. Choose **Finish** to accept the default settings for all parameters that are specified in subsequent steps.

Add a data source by opening the **Data Explorer** view and choosing **New Data Source** from the **Data Source** entry's context menu. Select CentraSite. Create a new data set by choosing **New Data Set** from the **Data Set** entry's context menu. The settings provided in the following pop-up are suitable, i.e. the new data set should be based upon our previously created data source named "Data Source", it should be named "Data Set", and the `Data Set Type` is "Business Query". Choose **Finish**. A pop-up appears, prompting you to enter a query. Specifying a query in this mode, however, is GUI-based, i.e. you specify the data to be selected by providing a name pattern and

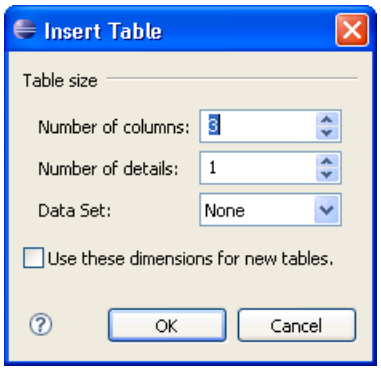
clicking on some options. As an example, specify the name pattern "MusicQuoteService" and select **Service** as the Object Type to be searched:



Choose **Finish**. A pop-up summarizing the specification options for this data set appears. Select the **Preview Results** option and verify that the query yields the three services in our CentraSite registry that contain that pattern.



Choose OK. To create a report from this query, open the **Palette** view and drag the **Table** tool into the layout area. Fill in the **Insert Table** dialog as shown below:



Open the pull-down list labeled **Data Set**, then drag the fields `Name`, *Description* and `Key` into the first, second, and third cells respectively of the table's detail row. Save the report.

The following screenshot shows the report as viewed in the BIRT Preview mode:

Name	Description	Key
MusicQuoteService		uddi:a36befc6-187d-11de-bcc4-fbda72ca3ed4
MusicQuoteServiceEcp	A specific quote for chamber music with piano	uddi:0c9a7ff1-1882-11de-bcc4-e760b576caa1
MusicQuoteServiceTFTT	A specific quote for brass ensemble music (trumpet, french horn, trombone, tuba)	uddi:d47b6f31-1881-11de-bcc4-b7be5c53deb2

Mar 26, 2009 5:06 PM

It is very convenient to specify a report in this manner; however, what can be done is rather limited. In particular, any query that can be created using the GUI front-end can, of course, also be created using pure XQuery. We now turn our attention to CentraSite XQuery for the following examples.

Displaying a Taxonomy

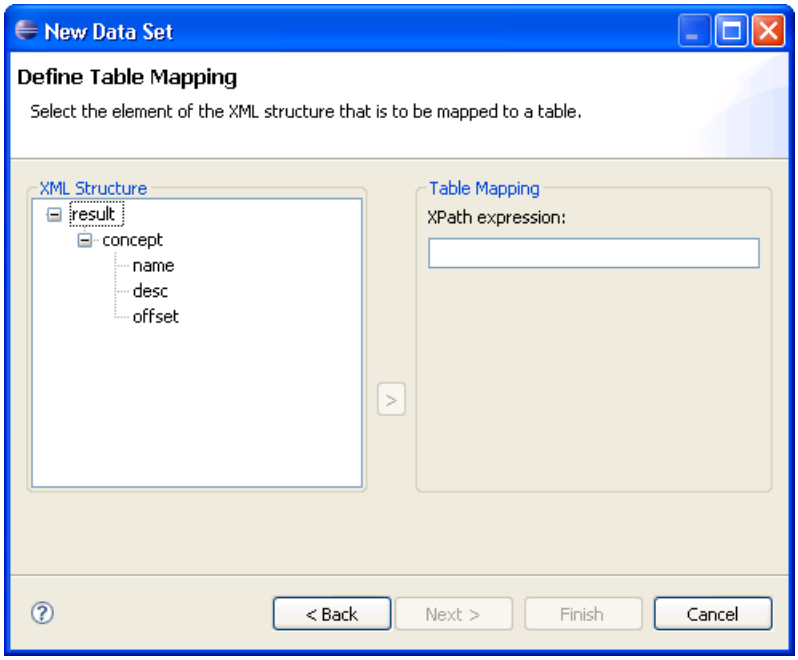
Your next exercise with the CentraSite BIRT reporting facility is to create a report that shows a complete taxonomy modeled after the page **2002 NAICS Codes and Titles** provided by the U.S. Census Bureau. See the **NAICS web pages** for more information.

Create a new report named *MusicTaxonomy.rptdesign*. Choose **New Data Source** to create a CentraSite XQuery data source via from within the **Data Explorer** view. Create a data set based on this data source.

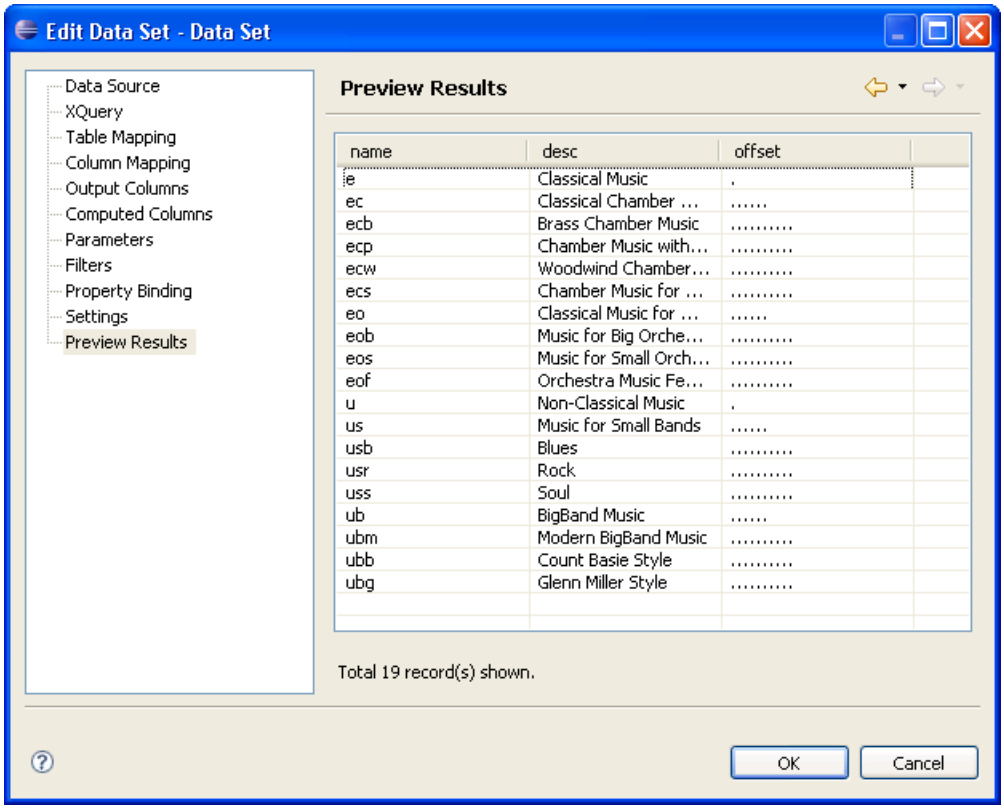
Enter the following query for the data set:

```
declare namespace jaxr = "http://namespaces.CentraSite.com/Schema/jaxr"
declare function local:getDescendants($node)
{
 for $child in collection("CentraSite")/jaxr:concept
                                [jaxr:parent=$node/jaxr:key]
 return ($child,local:getDescendants($child))
};
let $scheme := collection("CentraSite")/
  jaxr:classificationScheme[jaxr:name/jaxr:localString ="Music Classification"]
for $concept in local:getDescendants($scheme)
let $name := string(($concept/jaxr:name/jaxr:localString)[1])
return
 <concept>
  <name>{$name}</name>
  <desc> {
    string(($concept/jaxr:description/jaxr:localString)[1])
  } </desc>
  <offset> {
    substring(".........",1,1 + 5 * (string-length($name) -1))
  } </offset>
 </concept>
```

Then choose **Next**. This executes the query and displays the result's structure in the next step, namely **Define Table Mapping**:

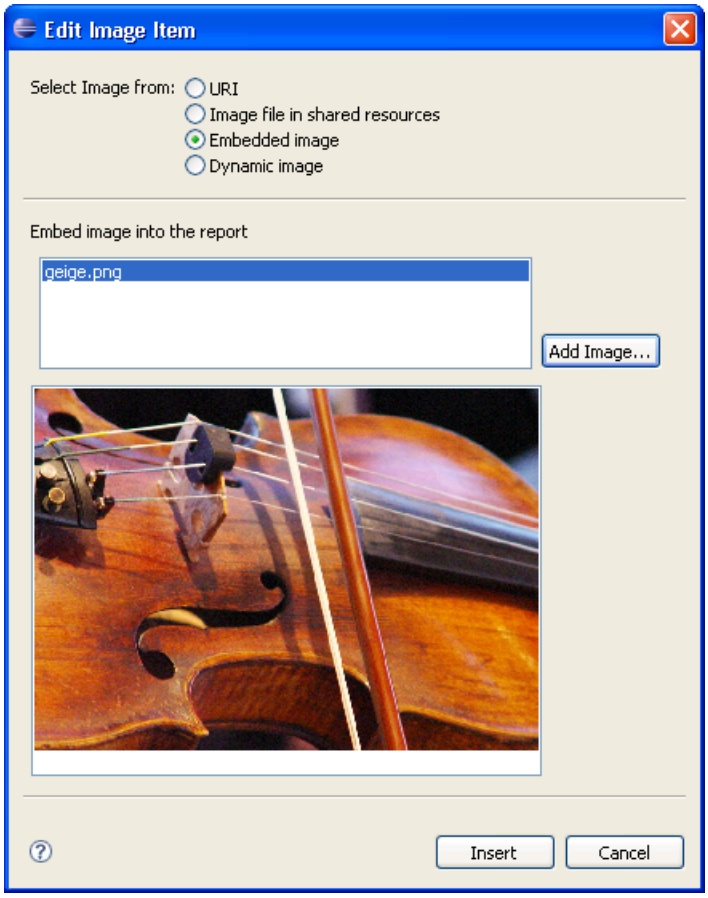


Select `concept` as the anchor. In the next step, select its children `name`, `desc` and `offset` as columns. Choosing **Preview Results** displays the following:

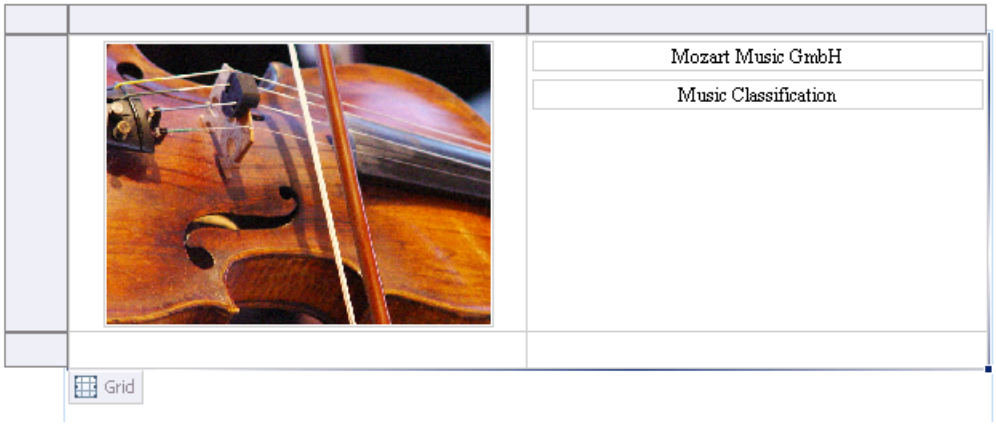


The `offset` field allows you to indent the concepts' descriptions to make the tree structure more readily apparent. The offset effect could have been achieved without additionally complicating the query. The data set editor offers a **Computed Columns** panel, where an additional data set column, i.e. a data item visible from outside, can be specified. You use JavaScript to specify how the values in this column are computed from other data items.

Now, having prepared the data provision, you can start designing the report. The report should be surrounded by a border; this is best done by putting all report elements into a `Grid` element. Drag the **grid** tool from the palette into the report layout area. Select "Number of Columns: 2 "and "Number of Rows: 2". Drag the **Image** tool into the upper left-hand cell of the grid. This pops up an image specification dialog, where you select **Embedded image** and include *geige.png*:



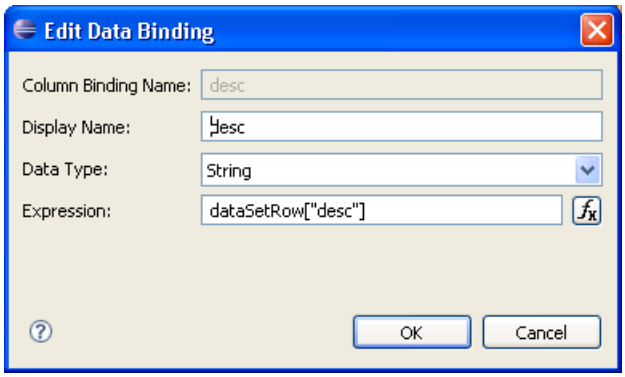
Drag two `Label` elements into the upper right-hand corner, and enter the texts "Mozart Music GmbH" and "Music Classification". Resize the grid cells accordingly. Moving the mouse over the grid occasionally makes a small square appear at the bottom. To modify the appearance of the grid, open its property editor by clicking on this square:



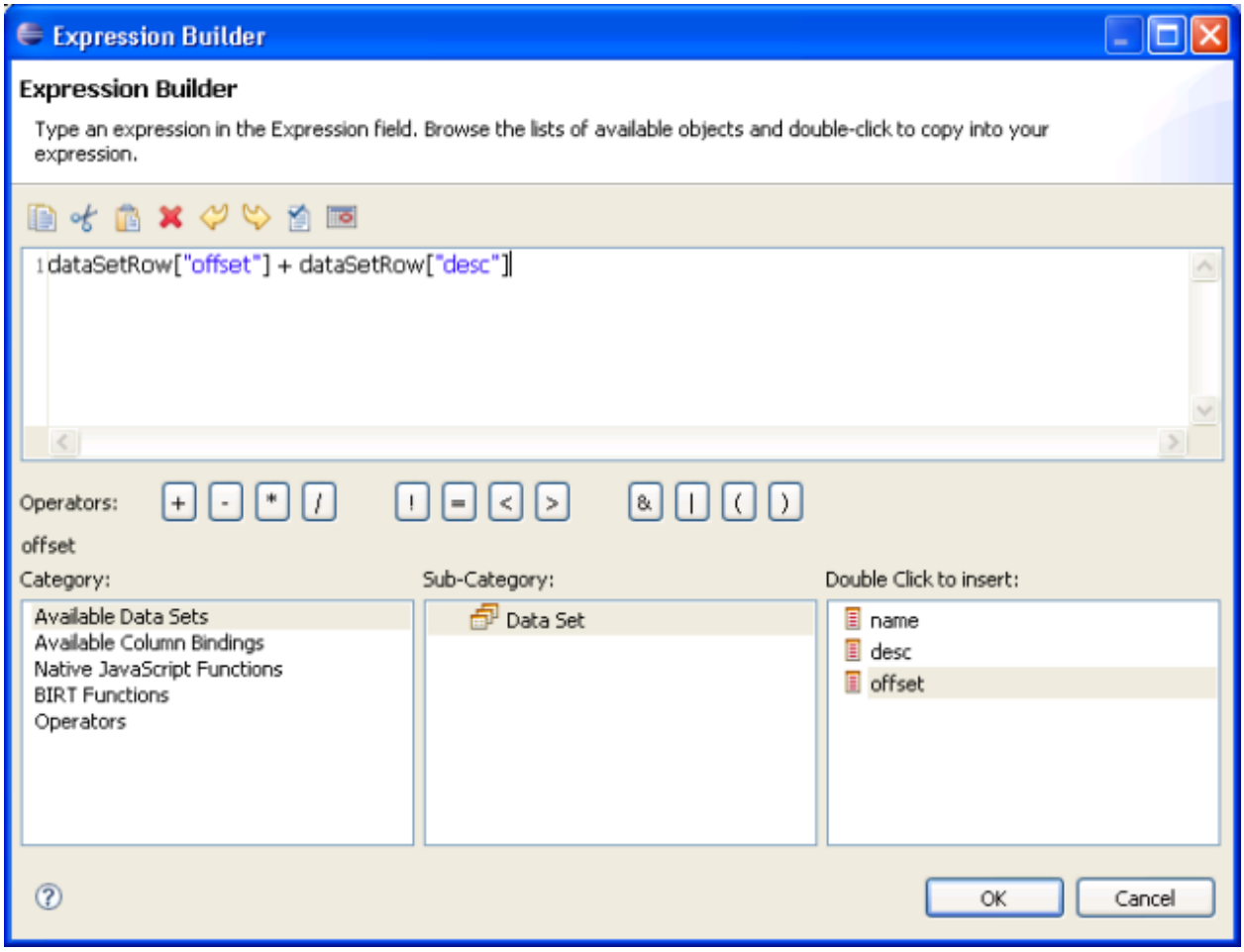
Use the grid property editor, the grid cell property editor and the label property editor to fine tune the grid and its contents. For most components, the relevant property editor is opened by double-clicking on the component.

To make room for the table, merge the two cells in the second row of the grid (i.e. click the left cell, hold down the Shift key and click the right cell, right-click in one of the cells and choose **Merge** from the context menu). Now drag a table from the palette into the lower row of the grid. The table has two columns and one detail row, i.e. one line per listed item. Switch back to the **Data Explorer**, expand the data set and drag "name" and "desc" into the first and second cells respectively of the detail row.

Double-click on the right cell of the table's detail row ([**desc**]) to update its contents:



Following the `Expression` input field, this pop-up includes the icon that opens the Expression Builder, a convenient BIRT tool for entering expressions. It offers all the items that an expression can contain, i.e. items defined in data sets, JavaScript functions, and BIRT-specific constructs as aggregation operators. Use the Expression Builder to change the field's content to `dataSetRow["offset"] + dataSetRow["desc"]`:



For visual enhancement, make appropriate changes to the text style and coloring of the table header and table body. Note that the table must have left-aligned text in order to render the tree structure correctly.

The following screenshot shows the report as a BIRT preview:

Mozart Music GmbH
Music Classification

Code	Description
e	.Classical Music
ec	.....Classical Chamber Music
ecb	.....Brass Chamber Music
ecp	.....Chamber Music with Piano
ecw	.....Woodwind Chamber Music
ecs	.....Chamber Music for Strings
eo	.....Classical Music for Orchestras
eob	.....Music for Big Orchestras
eos	.....Music for Small Orchestras
eof	.....Orchestra Music Featuring a Soloist
u	.Non-Classical Music
us	.....Music for Small Bands
usb	.....Blues
usr	.....Rock
uss	.....Soul
ub	.....BigBand Music
ubm	.....Modern BigBand Music
ubb	.....Count Basie Style
ubg	.....Glenn Miller Style

(To obtain a colored border, the report actually uses a grid with four columns, the outer columns being fixed to a certain width.)

More About Parameters

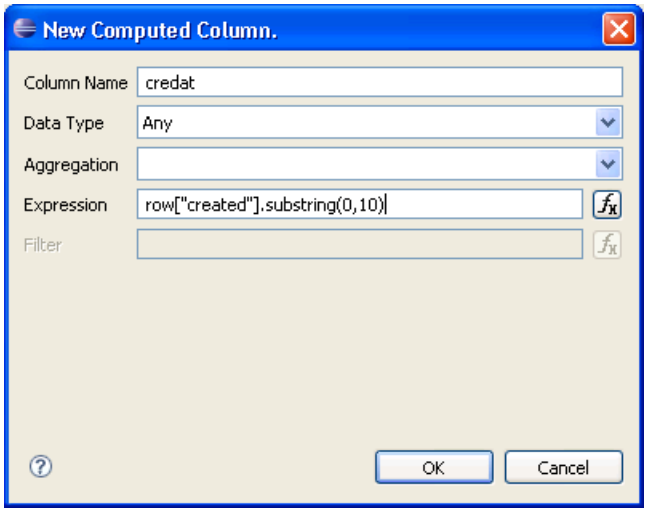
The last example uses two techniques, already encountered in the predefined report you modified previously, both having to do with report parameters. The first is a report parameter that is used in a query, and the second is a parameter that is automatically provided when the report is applied upon an asset from within CentraSite Control.

Create a new report named *ServiceDetails.rptdesign*. Create a new data source of type CentraSite XQuery and a new data set based on that data source. Enter the following query:

```
declare namespace jaxr = "http://namespaces.CentraSite.com/Schema/jaxr"
let $service := collection("CentraSite")/jaxr:service
       [jaxr:key="%-?=uddi:a36befc6-187d-11de-bcc4-fbda72ca3ed4?-%"]
return
 <service>
  <name_msg> {
    if ($service)
    then (data($service/jaxr:name/jaxr:localString[1]))
    else "Service not found"
  } </name_msg>
  <desc> {
    data($service/jaxr:description/jaxr:localString[1])
  } </desc>
  <provider> {
    let $org_id := $service/jaxr:providingOrganization
    let $org := collection('CentraSite')/*:organization[jaxr:key=$org_id]
    return data($org/jaxr:name/jaxr:localString[1])
  } </provider>
  <created> {
    if ($service)
    then
      let $events := collection('CentraSite')/jaxr:auditableEvent
                [jaxr:registryObject="%-?
                       = uddi:a36befc6-187d-11de-bcc4-fbda72ca3ed4?-%"]
      return string($events[jaxr:eventType="CREATED"]/jaxr:timeStamp)
    else "xxxxxxxxxx"
  } </created>
  <stability> { data($service/jaxr:stability) } </stability>
  <status> { data($service/jaxr:status) } </status>
  <wsdl> {
    let $conceptKey := $service//jaxr:serviceBinding
          //jaxr:specificationLink[1]/jaxr:specificationObject
    let $linkKey := collection('CentraSite')/jaxr:concept
          [jaxr:key = $conceptKey]//jaxr:externalLink[1]
    return string((collection('CentraSite')/jaxr:externalLink
          [jaxr:key = $linkKey]/jaxr:name/jaxr:localString)[1])
```

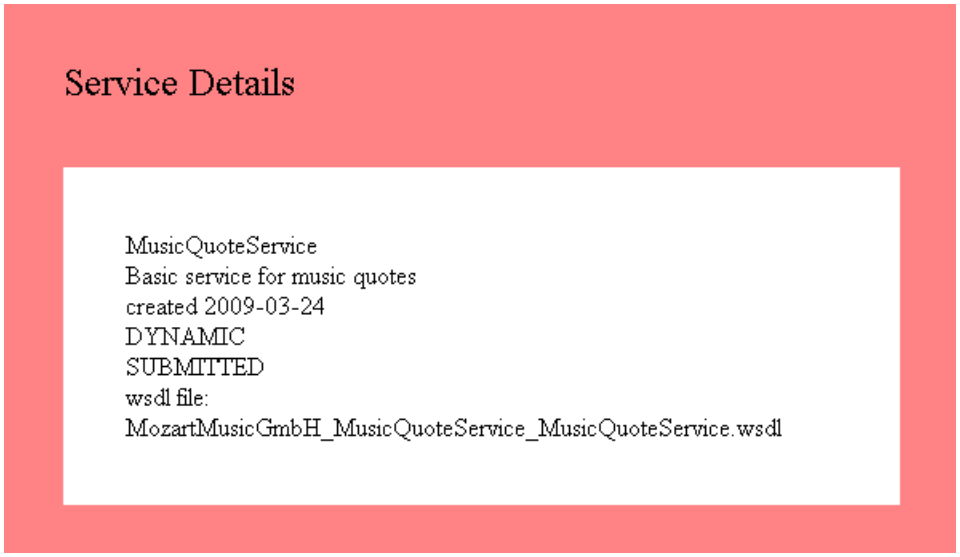
```
  } </wsdl>
</service>
```

Map the table to the field `service`, and select all its children as columns. Add a computed column `creationDate` and specify the expression `row["created"].substring(0,10)`. The ten "x"'s in the query result are necessary, otherwise applying `substring` in the case of a non-existing service causes the report to fail.



Add a parameter named ROKEY to the report. The parameter should have an existing service's UDDI key as its default value. Design the report using a list tool to contain the service's data and a grid to surround it with colored borders.

The following screenshot shows the report applied to **MusicQuoteService**:



6 Links

- **Eclipse BIRT project page**: (*http://www.eclipse.org/birt/phoenix/*)
- **New and Notable Features within BIRT 2.6**: (*http://www.eclipse.org/birt/phoenix/project/notable2.6.php*)
- **W3C XQuery web page**: (*http://www.w3.org/standards/xml/query*)
- **2002 NAICS Codes and Titles**: (*http://www.census.gov/epcd/naics02/naicod02.htm*)
- **NAICS Page**: (*http://www.census.gov/eos/www/naics/*)