



Cúram 8.1.3

Business Intelligence BIRT Developer Guide

Note

Before using this information and the product it supports, read the information in [Notices on page 53](#)

Edition

This edition applies to Cúram 8.1, 8.1.1, 8.1.2, and 8.1.3.

© Merative US L.P. 2012, 2024

Merative and the Merative Logo are trademarks of Merative US L.P. in the United States and other countries.

Contents

Note.....	iii
Edition.....	v
1 Developing Business Intelligence with BIRT.....	9
1.1 Introduction.....	9
1.2 Business Intelligence Overview.....	9
Business Features.....	10
Technical Features.....	10
1.3 BIRT System Landscape.....	10
BIRT System Environment.....	11
1.4 Build Process Reviewed.....	11
Build Process Note.....	11
1.5 Viewing and Integrating Report Content.....	11
Introduction.....	12
Viewing BIRT Reports Within Eclipse.....	12
Eclipse - BIRT Development.....	12
Viewing Report Content Within Eclipse.....	12
Developing Report Content/ BIRT Reports.....	13
Integrating BIRT Content.....	13
Writing Portable BIRT Reports across DB2 databases.....	18
1.6 Authentication and Entitlements.....	19
Overview.....	19
1.7 Developing New Reports.....	19
Eclipse Installation.....	19
Eclipse Post Installation.....	20
Creating your first Sample Report.....	21
Creating your first Sample Report for a reporting tab.....	22
Report Content Compliance.....	24
Data Sources and JDBC Connections.....	24
Internationalization and Localization.....	24
1.8 Design Guidelines.....	25
Real Time KPI Design Guidelines.....	25
Summary Table Design Guidelines.....	26
BIRT Development Standards.....	26
Drilling Down From a BIRT Report to a Reporting Application Page.....	27
Accessibility and BIRT Charts.....	28
1.9 BIRT Compliance.....	28
BIRT Reports.....	28

1.10 Creating a Datamart Data Source on WebLogic and WebSphere.....	28
WebSphere for OWB.....	29
WebLogic.....	32
1.11 Enabling WebSphere Application Server logging for BIRT.....	32
1.12 Troubleshooting.....	33
Cannot view BIRT content through Eclipse with H2.....	33
Build client.birt fails to execute.....	34
The library with the namespace CEFLibrary is not found.....	34
Error when executing a report.....	34
Eclipse dies when previewing a BIRT report.....	35
Report Page only displays one report when more that one report exists on the operational workspace.....	35
Tomcat closes on report preview or ODA SQL preview.....	36
BIRT Viewer does not run / List of reports not present.....	36
Exception thrown when running report.....	37
Unable to view all entities after creating dataset.....	38
1.13 Deployment Considerations.....	38
Default deployment.....	38
Non-Homogeneous Deployment for WebSphere and BIRT.....	38
1.14 Runtime Architecture.....	48
Default Runtime Architecture.....	49
BIRT Viewer Application.....	49
BIRT.....	49
1.15 Viewing Reports - Dependencies.....	50
Viewing Reports.....	50
Application Server.....	51
Tomcat - BIRT Viewer.....	51
BIRT Eclipse.....	51
Notices.....	53
Privacy policy.....	54
Trademarks.....	54

1 Developing Business Intelligence with BIRT

Use this information to learn how the application reporting tools are used. Business Intelligence and Reporting Tools is an open source component, which provides a development and execution environment for reporting. It is primarily a tool for the development and execution of charting and tabular data with the aim of providing decision support information for front line staff, line managers, and senior managers in the organization.

1.1 Introduction

The purpose of this guide is to describe how the Cúram Business Intelligence and Reporting Tools (BIRT) stack is used.

BIRT is an open source component, which provides a development and execution environment for report content. BIRT is used primarily as a tool for the development and execution of charting and tabular data with the aim of providing decision support information for front line staff (case workers), line managers, and senior managers in the organization.

This document describes how report content is developed, integrated, and run in a runtime environment. The following are considered the key benefits for the assessment of BIRT Engine:

- **Look and Feel:** The visual appeal of the product release is paramount. BIRT supports the production of content that can be seamlessly integrated into the application.
- **Licensing Terms and Cost:** An important element is the ability to include basic functionality in the "Out of the Box" product that uses an open source version of the product; including a licensed product was not acceptable for this purpose. The second element was the licensing options for customers for more advanced features. It is critical that they are able to license tools at a cost, which is proportional to the value they derive from its usage.
- **Technical Merit:** The technical merit of BIRT was evaluated against the three main areas of report functionality - Online Analytics, Reports, Dashboards and Charting.

Prerequisites

This guide is intended for any reader who uses BIRT to create and integrate report content into the application.

Audience

Readers must have a good working knowledge of Java™ and BIRT.

1.2 Business Intelligence Overview

Business Features

The application allows agencies to measure and monitor the performance of an organization, to detect gaps in processes, and to analyze the issues that are occurring. These agencies require a business intelligence solution that will provide them with the tools to support better decision making. The application provides a solution that covers the complete spectrum of reporting tools and technologies that enable organizations to make valuable business decisions. This includes decision support information for front line staff (case workers), line managers and senior managers in the organization. The following areas are provided for:

- **Embedded Analytics:** Embedded Analytics are representations of aggregated data that can be interacted with by the user to construct alternative views/sub-groupings of the data which were not envisaged at design time. The infrastructure to construct embedded analytics is provided and a number of these analytics have been added out of the box to the application. The purpose of this is to assist front line staff and managers in the decision making process. This is achieved by integrating the online application and the data warehouse which therefore provides interactive, summarized information in context.
- **Interactive Dashboards and Reports:** Interactive dashboards include the ability to publish graphically intuitive displays of information, including dials, gauges and traffic lights. These displays indicate the state of the performance metric, compared with a goal or target value. This data is a more in-depth view into the aggregated data in the business specific data warehouse. Reports provide the ability to create formatted and interactive reports with highly scalable distribution and scheduling capabilities. Infrastructure is provided in support of these and business specific reports will be added over time.
- **Ad-hoc Reporting:** The reporting infrastructure we provide, uses BIRT to render the charts as BIRT is an open sourced and low cost option for agencies that do not want an elaborate business intelligence offering.

Technical Features

The application is composed of a number of components.

- A BIRT Engine infra-structure component Cúram BIRT Viewer which executes BIRT charts and reports, this wraps the open source BIRT Engine, and is a Java Platform, Enterprise Edition application.
- The Cúram Platform BIRT Manager component is responsible for managing the look-up of report templates when the BIRT Engine is in use. A BIRT render is also available that simplifies the process of placing BIRT charts and reports on Report Pages.
- The Administration application has a report configuration section, which contains sections to configure the viewer and sections to configure BIRT reports.

1.3 BIRT System Landscape

The section gives an overview of the BIRT architecture and system environment and describes the dependencies between the reporting third-party tools that are supported.

BIRT System Environment

BIRT is an open source component, which provides a development and execution environment for reporting content. It was selected for use following detailed analysis of the alternatives. The list provides an overview of the key features.

- **The BIRT Report Designer:** The BIRT Report Designer is an Eclipse plug-in, which allows a developer to write a BIRT report, which can then be deployed into the BIRT Report Engine. This supports the use of a wide range of graphs (by using the BIRT Charting Engine) as well as data listing. This is built on the BIRT Report Design Engine, which can be used by any Java application to create or modify report designs.
- **The BIRT Charting Engine:** The BIRT Charting Engine has many built-in charts as well as support for user-defined charts. The user-defined charts are not constrained to any single technology, but all the built-in charts are implemented in Java.
- **BIRT Report Engine:** The BIRT Report Engine is the runtime component that renders a report design. It produces output in a number of formats including HTML and PDF.

1.4 Build Process Reviewed

The introduction of BIRT content has resulted in a new step being added to the build process. This additional step publishes BIRT content from the `BIContent` folder to the BIRT Viewer.

The BIRT Viewer executes BIRT content which is then rendered on a report page. The new build processes is reviewed in the next section:

Build Process Note

Important: When viewing the Client Application using Eclipse the follow process is now required:

Buid the server Change to the `EJBServer` folder and build the server (build server).

Buid the client Change to the `webclient` folder and build the client (build client).

Publish BIRT content for execution Change to the `BIContent` folder and publish your BIRT content (build client.birt).

Refresh Eclipse and Start Tomcat Start Apache Tomcat and view the client application.

1.5 Viewing and Integrating Report Content

Introduction

This section will guide you through the processes that are required to view BIRT content in the client application. The following sections will guide you through:

- How to import into Eclipse the projects that are required to view BIRT content.
- The post installation configuration steps required to view BIRT content (ensuring report content is rendered onto application pages).
- How integrate BIRT content into application pages, i.e. how to use the BIRT renderer and server side classes to embed BIRT content into report pages.

Viewing BIRT Reports Within Eclipse

See the Third Party Tools Guide for guidance on Eclipse versions and installation. The following instructions relate to the Eclipse environment used for report development.

See the following paragraph for a description of the development environment required for BIRT development.

Please note if you are viewing existing source database content you do not need to install a BIRT development environment. If you are using H2® as your RDBMS ensure it is started in remote mode.

Eclipse - BIRT Development

BIRT is an (Eclipse) open source component which provides a development and execution environment for report content.

The use of a second Eclipse instance for BIRT development is required because standard BIA Report development requires a different version of Eclipse than the Eclipse version that is used for development.. See the Third Party Tools Guide for a full explanation of which versions are required. Please note, that a BIRT development environment is not required to view BIRT content shipped with any delivery.

Viewing Report Content Within Eclipse

This section will guide you through the process of configuring the Eclipse environment used for report development.

1. Import the CuramBIRTViewer project. To import the CuramBIRTViewer project into eclipse, go to *File->Import->General-> Existing Projects into Workspace* and click Next. Select the root directory to be *CURAM_DIR\BIApp\CuramBIRTViewer* and hit Finish.
2. Select the CuramBIRT Viewer project in eclipse, then *window->preferences->java-> compiler->compile compliance level*. and set to 1.5
3. Select the CuramBIRTViewer project in eclipse, right-click and select *Tomcat Project-Update context definition*.

4. From the Window Menu select *Preferences->Tomcat->JVM Settings* and on the classpath window select the directory button and add in the full path to the *CURAM_DIR \EJBServer \project\properties* directory from your installed location.
5. To view licensed content via the Reports Tab (ignore this section if you do not have a Reports tab), then from the Window Menu select *Preferences->Tomcat->JVM Settings* and on the classpath window select the directory button and add in the full path to the *CURAM_DIR\Reporting\project\properties* directory from your installed location.
6. You must now publish all report content to the BIRT Viewer, from the directory *CURAM_DIR\BICContent* execute the command *build client.birt*.
7. Start Tomcat, to verify your BIRT viewer is working execute the follow URL:
 1. <http://localhost:9080/CuramBIRTViewer/>
 2. Execute the report */birtsamples/test.rptdesign* to verify the viewer is serving static BIRT reports.
 3. Execute the report */components/core/birt/curamsamples/SampleBIReportPDF.rptdesign* to verify the viewer is serving dynamic BIRT
8. If you are using H2 as your RDBMS ensure it is started in remote mode.
9. If you are experiencing issues please refer to the Troubleshooting guide at the end of the document.
10. To publish content to the CuramBIRTViewer application execute the following steps.

If you are experiencing issues please refer to the Troubleshooting guide at the end of the document.

For Report Dependencies, please refer to the Appendices.

Developing Report Content/ BIRT Reports

The section entitled *Developing BIRT Reports* describes how to create or modify BIRT report design documents.

Integrating BIRT Content

The process for integrating BIRT content is described briefly in the following points and explained fully in the sections below, please note that the client must follow the *Cúram Server Developer's Guide* at all times.

1. See the section entitled *UIM Example* for an example UIM code snippet.
2. See the section entitled *Report Name* for an example report name.
3. See the section entitled *Report Configuration* for on guidance on how reports are configured and what *DMX* is required.
4. See the sections *POD Facade Implementation* or *General Facade Implementation* for example on how to embed BIRT in PODS and report pages respectively.

UIM Example

A new domain definition has been created for BIRT reports, i.e. *BIRT_REPORT*. For BIRT content to be displayed on a Report Page this domain definition must be referenced/configured for the struct field in question. When embedding a BIRT report in a POD the domain definition must be set to *BIRT_POD*. *BIRT_POD* is a POD styling definition which must have been previously set.

Report Name

The name logical BIRT Report name is coded to a constants file, e.g. see *Const.java*. The report name is a logical name with the physical report returned by a call using the CEF utility *BIHelper* component, the report DMX (Data Mining Extensions) configuration entries must be available in the database or this method will throw an exception. See the following section on DMX configuration for an example.

```
/** Investigation Summary report. */
public static final String gkInvestigationsSummaryReportName =
    "InvestigationsSummary";
```

Reporting Configuration - DMX

Once the BIRT report is fit for purpose you must create a DMX entry for your report. The report DMX entry must be present your custom initial demo data, for examples see the DMX file *BIREPORTCONFIGURATION.dmx*. Please note that the client must follow the *Cúram Server Developer's Guide* at all times, also note that the *BIHelper* class will throw a runtime exception if the report logical name cannot be found in the *BIReportConfiguration* entity. See the section below for an example DMX configuration entry.

```
<row>
  <attribute name="biReportConfigurationID">
    <value>2002</value>
  </attribute>
  <attribute name="reportName">
    <value>AuditPlanSummaryBarChart</value>
  </attribute>
  <attribute name="reportCategory">
    <value>RC2001</value>
  </attribute>
  <attribute name="reportFileName">
    <value>components\core\birt\CaseAudit\
      AuditPlanSummaryBarChart.rptdesign</value>
  </attribute>
  <attribute name="reportServlet">
    <value/>
  </attribute>
  <attribute name="width">
    <value>100%</value>
  </attribute>
  <attribute name="height">
    <value>280</value>
  </attribute>
  <attribute name="scrolling">
    <value>RS2003</value>
  </attribute>
  <attribute name="reportFrameborder">
```

```

        <value>0</value>
    </attribute>
    <attribute name="description">
        <value>Audit Plan Summary</value>
    </attribute>
    <attribute name="recordStatus">
        <value>RST1</value>
    </attribute>
    <attribute name="versionNo">
        <value>1</value>
    </attribute>
</row>

```

UIM Example

The following is a UIM snippet to include a BIRT report in a page.

```

<SERVER_INTERFACE
    CLASS="ProductDelivery"
    NAME="DISPLAY"
    OPERATION="getReassessmentResultsChart"
    PHASE="DISPLAY"
/>
<FIELD ALIGNMENT="CENTER">
    <CONNECT>
        <SOURCE
            NAME="DISPLAY"
            PROPERTY="report"
        />
    </CONNECT>
</FIELD>

```

POD Facade Implementation Example

The following is a java code snippet which includes a BIRT report in a report page.

```

protected Document getBIRTReportData()
    throws ApplicationException, InformationalException {
    ...

    reportData = biHelper.getReportData(
        PodsConst.kCaseloadSummaryBarChartBIReportName,
        reportParameters);

    return reportData;
}

```

The following is a java code snippet which creates a pod with links to new investigation and list of investigation pages.

- workspaceDocument: Document to which the investigation summary pod is added
- contexts: A map of contexts available to the Pod
- Node: represents the pod structure to be loaded

```

public Node createPod(final Document workspaceDocument, final Map
    <String, Object> contexts)
{

```

```

try {
    ...

    PodBuilder pod =
    PodBuilder.newPod(workspaceDocument,
    PODTYPE.INVESTIGATIONSSUMMARYPOD);

    pod.addContent(chart, rc);

    return pod.getWidgetRootNode();
} catch (Exception e)
{
    throw new AppRuntimeException(e);
}

```

The following is java code which gets the investigation summary details

- param selection: Selected range for which investigations summary details is required
- return investigations summary document
- throws AppException: Generic Exception Signature
- throws InformationalException: Generic Exception Signature

```

protected Document
getBIRTSummary(final String selection)
    throws AppException, InformationalException {
    ...

    //Building Report Parameters
    Date startDate = Date.getCurrentDate();
    Date endDate = Date.getCurrentDate();

    Calendar calendar = startDate.getCalendar();
    calendar.add(Calendar.MONTH, -CuramConst.gkOne);
    startDate = new Date(calendar);

    reportParameters.put(
    PodsConst.knvestigationSummaryStartDateParameter,
    startDate.toString());

    reportParameters.put(
    PodsConst.knvestigationSummaryEndDateParameter,
    endDate.toString());

    return biHelper.getReportData(
    CuramConst.gkInvestigationsSummaryReportName,
    reportParameters);
}

```

Non POD Facade Implementation

The following is a java code snippet which adds content to a page (e.g. a content panel) with format XML data for an employer work force tab details:

- parameter caseID: Employer concern role id
- return ContentPanelBuilder
- throws InformationException: Generic Exception Signature
- throws AppException: Generic Exception Signature

```
protected ContentPanelBuilder getBIRTReportDetails(
    final long concernRoleID)

    throws AppException, InformationalException
{

    ContentPanelBuilder contentPanelBuilder =
    ContentPanelBuilder.createPanel(
    CuramConst.gkEmployerWorkforceDetail);

    contentPanelBuilder.addRoundedCorners();

    Map <String, String> reportParameters =
    new HashMap<String, String>();

    reportParameters.put(CuramConst.gkParam_ConcernRoleID,
    String.valueOf(concernRoleID));

    WidgetDocumentBuilder reportBuilder =
    biHelper.getDocumentBuilder(
    CuramConst.gkBIRTProspectEmployerWorkforceReport,
    reportParameters);

    contentPanelBuilder.addWidgetItem(reportBuilder,
    CuramConst.gkStyle,
    CuramConst.gkStyleBirt); return contentPanelBuilder;

}
```

The following is a java code snippet which adds content to a page (e.g. a content panel) with formatted XML data for an employer work force tab details:

- parameter caseID: Employer concern role id
- return ContentPanelBuilder
- throws InformationalException: Generic Exception Signature
- throws AppException: Generic Exception Signature

```
protected ContentPanelBuilder getBIRTReportDetails(
    final long concernRoleID)
    throws AppException, InformationalException
{

    ContentPanelBuilder contentPanelBuilder =
    ContentPanelBuilder.createPanel( CuramConst.gkEmployerWorkforceDetail);

    contentPanelBuilder.addRoundedCorners();

    Map <String, String> reportParameters = new HashMap<String,
    String>();

    reportParameters.put(
```

```

        CuramConst.gkParam_ConcernRoleID,
String.valueOf(concernRoleID));

        WidgetDocumentBuilder reportBuilder =
biHelper.getDocumentBuilder(
        CuramConst.gkBIRTEmployerWorkforceReport, reportParameters);

        contentPanelBuilder.addWidgetItem(reportBuilder,
        CuramConst.gkStyle, CuramConst.gkStyleBirt);

        return contentPanelBuilder;
}

```

Writing Portable BIRT Reports across DB2 databases

The following steps need to be applied to BIRT reports if they need to be run on DB2 databases for both ZOS and non-ZOS platforms.

1. Create a new 'Viewer Configuration' parameter, via the CURAM Application, called 'db2numbertype' with a parameter value of 'bitdata'
2. Copy and paste the existing dataset for the relevant report in BIRT Eclipse and
 - Rename this new dataset to be 'zos' - all in lowercase
 - Edit the dataset and remove any parameters specified
 - Go to the query and remove any lines with question marks for parameters
3. Append the following code to the 'Before Factory' script area and enter the relevant chart name in the code below

```

if(params["db2numbertype"].value == "bitdata") {

    mytable = reportContext.getDesignHandle().
findElement("<ENTER CHART NAME HERE>");

    mytable.setProperty( "dataSet", "zos" );

}

```

4. Click on the 'zos' data set and then select the 'script' tab Tab (located at the bottom of the 'Report Designer' window) and choose the 'beforeOpen' Script option. Edit the following code snippet to match your query needs and paste it in the report.

```

importPackage(Packages.java.math);
importPackage( Packages.biapp );

var param_<append parameter name here> = BigDecimal.valueOf
(params["<ENTER Parameter name here>"].value);

param_<append parameter name here>_long_to_string =
UniqueIDUtil.longToString
(param_<append parameter name here>);

this.queryText = " <Enter the SQL select statement here>"

```

Note: The parameter name is case sensitive. You must enter it as it appears in the report.

E.g. The following is an example of a completed code section. You will need to create the relevant parameters based on the query given. In this example there is only 1 parameter called 'concernRoleID', this parameter is passed in to the report and then it is converted to be used in the SQL query as follows:

```
importPackage(Packages.java.math);
importPackage(Packages.biapp);

var param_concernroleid =
    BigDecimal.valueOf(params["concernRoleID"]
        .value);

param_concernroleid_long_to_string = UniqueIDUtil.longToString
    (param_concernroleid);

this.queryText = "SELECT numberPermanentStaff as numberStaff"
    + ", 'PermanentStaff' as staffType"
    + " FROM Employer"
    + " WHERE concernRoleID =" +
    param_concernroleid_long_to_string
    + " UNION"
    + " SELECT"
    + " numberCasualStaff as numberStaff,"
    + " 'CasualStaff' as staffType"
    + " FROM Employer"
    + " WHERE concernRoleID =" +
    param_concernroleid_long_to_string
```

1.6 Authentication and Entitlements

Overview

The BIRT Viewer application ensures that only authenticated users can access the BIRT Viewer application, therefore ensuring only authenticated users can execute BIRT reports.

The inclusion of an additional security measure in form of entitlement checking or authorization checking is not included in the BIRT Viewer application at present. It is critical this is reviewed if any steps are taken to include BIRT content in an externally facing (public facing) client application.

1.7 Developing New Reports

Eclipse Installation

BIRT is an (Eclipse) open source component which provides a development and execution environment for reporting content. We recommend the use of a second Eclipse instance for BIRT development. This second instance ensures that newer BIRT versions can be used without

changing the development process, which would otherwise be required should a single Eclipse environment be used.

An all in one Eclipse and BIRT installation is used for internal development.

See the Third Party Tools Guide and the BIRT Developer Guide which provide a full explanation. Please note the Eclipse environment for BIRT development must be installed into a folder called <eclipsebirt>, if you choose a different folder name see the trouble shooting section "Problem: build client.birt fails to execute" on how to configure the BIRT Eclipse location.

Eclipse Post Installation

It is recommended that you create a batch file to start your BIRT Eclipse development environment. e.g.

- birteclipse.bat this will be located at the root of your project.
- Please note that the BIRT eclipse environment is only used for the development of BIRT content, you must use your "normal" eclipse development for "standard" application development work.
- You will be prompted to name your work space, we recommend that you give this workspace a descriptive title, e.g. <workspaceBIRT>.

Import BIRT Projects

Import the following projects into BIRT Eclipse:

- Import the <Curam reporting content> project into eclipse, go to File->Import->General->Existing Projects into Workspace and click Next. Select the root directory to be <Curam_DIR>\BIContent and hit Finish. This is the project within which you will work.
- Import the <Curam BIRT Viewer Samples> project into eclipse from the <Curam_DIR>\BIApp\CuramBIRTViewer\components\BIContent directory. There are a number of sample reports contained within this project which may be helpful.
- The CuramBIRTViewer project also needs to be imported. To import the <CuramBIRTViewer> project into eclipse, go to File->Import->General->Existing Projects into Workspace and click Next. Select the root directory to be Curam_DIR\BIApp and hit Finish.
- Select the CuramBIRTViewer project in eclipse, right-click ->go to properties->Java Build Path-> Libraries-> Add variable->Configure variable->New-> TOMCAT_HOME (set to DEV_ENV\tomcat).
- Select the Cúram BIRT Viewer project in eclipse, window->preferences ->java->compiler ->compile compliance level-> set to 1.5.

Configure BIRT Projects

- From the menu -> Preferences -> Report Design -> Resource, Select the Workspace button.
- Select the "CuramBIRTViewer Samples" project and select the resources folder, Select OK.
- The resource folder text box should now contain the value `${workspace_loc:Curam BIRT Viewer Samples(BI Stream)/resources}`, Hit OK.
- Select Cúram BIRT Viewer Samples and right click and select properties.

- Select Report Design -> Resources -> Configure Workspace settings (top right hand side).
- Select OK and OK again (even though you are not changing anything this need to be completed).
- Verify your common resource are available:
 1. From the <Curam BIRT Viewer Samples> Project select the resources folder->library and double click the CEFLibrary file.
 2. The library will open in the resource explorer, (Window->show view->Resource Explorer). You can now use the standard library features.
- To verify your environment is correctly configured:
 1. Select the "CuramBIRTViewer Samples" project, within tests/curamsamples open the sample report SampleBIReportPDF.rptdesign or SampleBIReport.rptdesign.
 2. To view the report select the Preview tab or from the View Report short cut on the tool bar. You have opened and executed your first report.
- You are now ready to create your first report with data from the operational database.

Creating your first Sample Report

- Select <Curam BI Content> project.
- Go to File, new, Report and if creating an overpayment sample report save the report to here <Curam_DIR>\BICContent\components\core\birt.
- Verify that the library is still available in the Resource View (from the previous steps).
- Go to the Outline View->libraries->right click and select use library, select the "CEFLibrary" from the Shared Resources\resources\library folder in the popup.
- Go to the Resource Explorer, expand "CEFLibrary" in Shared Resources\resources\library and complete the following:
 1. Data Sources ->CuramDB, right click and select add to report.
 2. Select CuramDM if creating a datamart report.
 3. Select Data Sources and if no current data source exists, right click and choose new data source. Edit the new/current data source to point to your local source database, i.e right click on datasource, click edit and set the following values:
 - Driver Class: com.ibm.dbc.jcc.DB2Driver (v3.57)
 - Driver URL: jdbc:db2://silverstreak:50000/curam
 - Username: db2admin
 - Password: *****
- The settings within this panel will be used for the Eclipse embedded preview mode only. Test your connection.
- Create a new data set using the data source configured in the previous step, paste in the SQL as an example select casetypecode as status, count (casetypecode) as casecount from caseheader group by casetypecode. If you are unable to view all entities this is due to the fact that BIRT puts a cap on the number of tables displayed when creating a dataset. Please see Troubleshooting Guide for steps to change this setting.
- Add the report item to the report by, going to the Resource Explorer and expand "report items" in Shared Resources\library\ReportItemLibrary.rptlibrary and complete the following:

NOTE: This library contains a number of Curam Styled, PNG formatted, library report items for each of the chart types available. To avail of the predefined formatting for each report type, it is recommended that you only use the Curam Styled report items from this report library. These are the library items named 'xxxChartCuramStyled', e.g. HorizontalBarChartCuramStyled etc.

1. Right click on the required **Curam styled chart** and select 'Add to Report'.
 2. Link the report to the dataset, set the X and Y value series and apply any required ordering and filtering
 3. Format the chart as required by setting Axis Values, Legends, Titles etc.
- In order to test your report via the application pages, copy the report to be executed from the Cúram BIRT Viewer:
 1. If your Cúram BIRT Viewer is already configured (see 4.1 Configuring the Cúram BIRT Viewer), then execute the following command from the <Curam_DIR>/BICContent directory to make your reports available to the Cúram BIRT Viewer - appbuild client.birt.
 2. Run your report from the Cúram BIRT Viewers test window or view from the application page:
 - `http://localhost:9080/CuramBIRTViewer/`

Creating your first Sample Report for a reporting tab

About this task

Only follow these steps if you are creating a report for a licensed reporting tab.

Procedure

1. You will need to place a BIBootstrap.properties file in the <Curam_DIR>\Reporting\project\properties directory. Copy the sample BIBootstrap.properties.sample file and rename it to BIBootstrap.properties. Change the centraldm properties to point to the datamart you intend to work from. You only need to configure the properties for the datamart, i.e. centraldm. #

You will also need to encrypt the passwords before entering them into the BIBootstrap.properties file. Encrypting a password is done as follows

- a) Open a command prompt from Reporting\components
- b) Run the following: 'appbuild encrypt.password -Dpassword=<p>' where <p> is the assigned password to be encrypted
- c) Enter the full encrypted password returned, for example: qqnscP4c4+s== as the password in the BIBootstrap.properties file.

See example below.

```

db.type=<databasetype> (e.g. ORA or DB2 - just
put this in the property file once)
# ORACLE connection properties for the data
mart
centraldm.db.server=<machinename>
centraldm.db.port=1521

```

```

centraldm.db.name=<servicename>
centraldm.db.SID=<SID> (e.g. ORCL)
centraldm.db.username=datamart
centraldm.db.password=<encrypted password> (e.g.
qqnscP4c4+s==)

```

2. Select <Curam BI Content> project.
3. Go to Window Menu -> Preferences -> Tomcat -> JVM Settings. On the classpath widow select the button directory and add in the full path to the <Curam_DIR>\Reporting \project \properties directory for your stream.
4. Create a new report e.g. if creating a report for the CEF application which is associated with a license then save the report to this location:

<Curam_DIR>\BIContent\components\reportingForCEF.

Platform reports associated with a license should be saved here, etc.

<Curam_DIR>\BIContent\components\reportingForPlatform.

5. Verify that the library is still available in the Resource View (from the previous steps).
6. Go to the Outline View->libraries->right click and select use library, select the "CEFLibrary" from the popup.
7. Go to the Resource Explorer, select CEFLibrary and do the following:
 - a) Data Sources->CuramDM, right click and select add to report.
 - b) Select the data source and right click to edit the data source to point to your local source database by editing the values in the data source as follows:
 - Driver Class: com.ibm.db2.jcc.DB2Driver (v3.57)
 - Driver URL: jdbc:db2://silverstreak:50000/curam
 - User Name: db2admin
 - Password: *****

The settings within this panel will be used for the Eclipse embedded preview mode only. Test your connection.

8. Create a new data set using the data source configured in the previous step, paste in the SQL as an example

```
select casetypecode as status, count (casetypecode) as casecount from caseheader group by casetypecode.
```

9. Add the report item to the report by, going to the Resource Explorer and expand "report items" in Shared Resources\library\ReportItemLibrary.rptlibrary and complete the following:

- a) Right click on the required Curam styled chart and select 'Add to Report'.

NOTE: This library contains a number of Curam Styled, PNG formatted, library report items for each of the chart types available. To avail of the predefined formatting for each report type, it is recommended that you only use the Curam Styled report items from this report library. These are the library items named 'xxxChartCuramStyled', e.g. HorizontalBarChartCuramStyled etc.

- b) Link the report to the dataset, set the X and Y value series and apply any required ordering and filtering.

- c) Format the chart as required by setting Axis Values, Legends, Titles etc.
10. In order to test your report via the application pages, copy the report to be executed from the Cúram BIRT Viewer:
- 1. If your Cúram BIRT Viewer is already configured (see Eclipse Installation section), then execute the following command from the web client directory to make your reports available to the Cúram BIRT Viewer: `appbuild client.birt`.
 - 2. Run your report from the Cúram BIRT Viewer test window or view from the application page

`http://localhost:9080/CuramBIRTViewer/`

Report Content Compliance

Do not modify report designs directly, copy the report and then make any changes you need.

Data Sources and JDBC Connections

The BIRT Infrastructure defines the following data source for use:

- CuramDB: This is the data source name that must be used for all BIRT content when querying the application database.

The data-source must be manually create on the application servers, see the Third Party Tools Guides.

Internationalization and Localization

BIRT reports support localization and internationalization. Please follow the below steps to localize your BIRT report. When creating your BIRT Report also create corresponding properties files in the same location. See naming convention section below.

- Create report with corresponding default.properties file. These are the 2 files you will deliver e.g.
 1. SampleWithTextLocalised.rptdesign
 2. SampleWithTextLocalised.properties
- Run the client.birt command to publish these properties files so they are available to BIRT Eclipse.
- In Report Designer open the report you wish to localize.
- In the layout pane of the report click anywhere on the white space. Go to the Properties section and click on Resources.
- In the Resource file field click browse select the appropriate.properties file from the Properties folder.
- For each label, go the Localization tab in the Property Editor and choose the appropriate resource from the list for the Content key property.
- Test the report in BIRT Eclipse Click on the menu->Preferences->Report Design->Preview and choose your locale. Click ok and run the report. It should display the text for the corresponding locale you have chosen.

- Test the report running in Tomcat - When running report append the following parameter to the end of the report URL `&__locale=en / &__locale=en_US / &__locale=de_DE`.
 - E.G. `MyReport.rptdesign&__locale=en_US`
- These.properties file names should follow the convention "ReportName_lang_country.properties", where lang and country are two-letter ISO codes. For example for German translation use "myreport_de_DE.properties", for US English use "myreport_en_US.properties".
- The ".properties" files must be ASCII encoded. All characters with codes above 127 ASCII must be escaped using this pattern: `\uNNNN`, where NNNN is the four-hex-digit Unicode representation of the original character. In theory you could escape all characters, but really it is not needed for those under 128 ASCII.
- The contents of the properties files should follow standard reporting naming conventions:
 - `Help.PageDescription`=This page allows you to add a text translation to a localizable text record. A localizable text record allows for application text to be localized.
 - `PageTitle.StaticText1`=Add Text Translation.
 - `ActionControl.Label.Save`=Save.
 - `ActionControl.Label.Save.Help`=The Save action updates the record using the information entered on the page.
 - `ActionControl.Label.Cancel`=Cancel.
 - `ActionControl.Label.Cancel.Help`=The Cancel action dismisses the page.
 - `Cluster.Title.Details`=Details.
 - `Field.Title.Language`=Language.
 - `Field.Title.Language.Help`=The language type for the text translation e.g. en, en_US.
 - `Field.Title.Text`=Text.
 - `Field.Title.Text.Help`=The actual translated text.
 - `Report.Title` = Title of Report
 - `Report.chart.xaxis.title` = X-axis title, e.g. time
 - `Report.chart.yaxis.title` = Y-axis title, e.g. no. of people participating
 - `Report.chart.title` = Title of Chart.
 - `Report.chart.yseries1` = Series 1 of Y-values for comparing multiple rows

1.8 Design Guidelines

Real Time KPI Design Guidelines

On a case by case basis an assessment of each KPI must be undertaken to see:

1. If the KPI is performant in its own right?
2. Does the KPI adversely affect the performance of other report users?

If the answer to either of the previous questions is yes, it is recommended to take one of the below courses of action. In terms of finalizing the summary table design and next steps, these are the tactical options:

1. Review your BIRT content and apply all performance improvement techniques available to you and benchmark to verify your KPI is per formant.
2. If option 1 does not resolve your performance issues then design a summary table with as many de-normalized columns as needed to resolve your performance issues.

Summary Table Design Guidelines

Before starting the design of your summary table, please verify the following points still hold true:

- That the business need for this KPI to be real time is absolute.
- That the performance of the SQL for this KPI against the transaction tables is not acceptable or its impact is negative on other database usage.

Below are general design criteria for summary tables:

1. The summary table must have "de normalized" columns to help avoid table searches/scans, thus increasing the performance of record identification process, for example:
 - Adding Case Super, Case Owner and Case Type columns to a summary table might mean that the need to hit the Case header table to identify what case records are required is eliminated. The identification of the records is now carried out against the summary table.
2. The summary table should also contain other keyed columns that facilitate the retrieval of other related data items if needed, this also allows the summary table to be future proofed to some degree.
 - For example, a "CaseID" column on a summary table provides access to other case header data, but vitally the record set has already been identified in point 1 so joining to the case header table is as performant as possible.

BIRT Development Standards

Please follow the below standards when developing Reports in BIRT.

Naming Standard

Do not leave any spaces in the file name.

- **CCS Report Name Example:** P15_KPI1_PlacementStability.rptdesign
- **CEF Report Name Example:** OverpaymentCaseFinancialReport.rptdesign
- **Data Source Name:** curamdm
- **Data Set Name:** MonthlyPlacementStability (CCS e.g.)
- **Report Parameter Name:** P_Org
- **Data Set Parameter Name:** P_DS_Org

External Parameters

The following guidelines are for external parameters that are being passed into the BIRT report:

- **Operational IDs** (e.g. caseID, concernroleID etc.): Set parameter as 'Decimal' in report. Param is passed without any quotes.

- **Numbers** (excluding Operational IDs): Set parameter as Integer in report. Param is passed without any quotes.
- **Strings**: Set parameter as String in report. Param is passed without any quotes.
- **Dates**: Set parameter as Date in report. Param is passed without any quotes. Date must be in the form MM/DD/YYYY.

SQL

- All SQL should be written to return the minimum number of records back to BIRT.
- If possible, all calculations and aggregations should be performed at the database level.
- SQL should be portable, i.e. it should be able to run against both Oracle and DB2.
- Do not use vendor specific functions.

Chart Type

Please set the Output Format to be PNG. This is set in the Select Chart Type tab of the Edit Chart window.

Sorting Columns on X-Axis E.g. Displaying Month Name

If you need to rename the labels to something that does not exist on the database then the instructions below show how axis labels can be overridden.

If an Optional Y Series Grouping is added to a Report then whatever is displayed on the X-Axis is automatically defaulted in as the Sort Order. This means that if the X-Axis is displaying Month Names then they will be sorted alphabetically, which is incorrect.

Please follow these steps to resolve the problem:

- Put the month number on the x-axis.
- Sort it in Ascending order.
- On the Layout tab highlight the chart and click the Script tab
- Select the beforeDrawAxisLabel function
- See this URL for sample code:

<http://www.birt-exchange.org/org/forum/index.php/topic/10988-changing-x-axis-labels-only/>

Drilling Down From a BIRT Report to a Reporting Application Page

Follow the below development steps when drilling down to a Reporting Application page from a BIRT report.

The development process is:

- Pass in the page id as a parameter to the report.
- Add the script below to the onclick event.
- BIRT stores quotes as " in the xml file. Change the string back to quotes in the xml source tab in BIRT.

```
window.parent.dojo.require("curam.util.Navigation");
var emptyString = "";
```

```

        if (categoryData == "Completed")
        {
            window.parent.curam.util.Navigation.goToUrl(
                "CCSInvestigation_initialContactsCompletedPage.do?
                numberOfDays=" +
                emptyString + "complianceType" + categoryData);
        }

        else if (categoryData == "Pending")
        {
            window.parent.curam.util.Navigation.goToUrl(
                "CCSInvestigation_initialContactsPendingPage.do?
                numberOfDays=" +
                emptyString + "complianceType" + categoryData);
        }

        else if (categoryData == "Overdue")
        {
            window.parent.curam.util.Navigation.goToUrl(
                "CCSInvestigation_initialContactsOverduePage.do?
                numberOfDays=" +
                emptyString + "complianceType" + categoryData);
        }

```

Accessibility and BIRT Charts

The following paragraphs outline what Chart properties must be set to meet accessibility standards.

The chart title must be set to ensure a chart is accessible, the charts accessibility is not related to the visibility of the chart title, the title can be set to visible or not-visible:

- Within Eclipse, double click on the chart to view the chart wizard.
- Navigate to the Format Chart tab and select the title property in the left navigation pane.
- Enter the chart title, depending on your user interface requirements make the title visible or not visible.

1.9 BIRT Compliance

BIRT Reports

A compliant process for changing report design document is to copy the report and to change the copy. Changing existing BIRT report design documents will result in a more difficult upgrade process.

1.10 Creating a Datamart Data Source on WebLogic and WebSphere

Reporting is supported on both Oracle and DB2.

WebSphere for OWB

The BIRT Viewer application has been tested using a non-XA data source and a XA data source. The recommendation for creating a Datamart Data Source on WebSphere® is to use the non-XA data source.

Creating the Data Source Login Alias

IBM® Db2®, IBM® Db2® for z/OS® and Oracle® Database are the databases supported. The Administrative Console can be used to configure a login alias for both the DB2 and Oracle data sources as follows:

- Navigate to Security -> Global Security.
- Expand the Java Authentication and Authorization Service option in the Authentication section and select the J2C authentication data option.
- Click New to open the Configuration screen.
- Set the following fields:

Alias = CuramReporting

User ID = <database username>

Password = <database password>

Description = The database security alias where <database username> and <database password> are set to the username and password used to login to the database.

- Click OK to confirm the changes.

Setup the Oracle Non-XA Database Driver

- Navigate to Resources->JDBC->JDBC providers.
- Choose the scope appropriate for your environment.
- Click New to add a new driver. This will open a configuration screen.
- Select Oracle from the the list in the Database type drop down supplied.
- Select Oracle JDBC Driver from the list in the Provider type drop down supplied.
- Select the Connection pool data source from the list in the Implementation type drop down supplied.
- Set the Name field to be Oracle JDBC Driver for Reporting, if not filled in automatically set the description.
- Click Next to continue.
- Review the Class path and set the environmental variable to the location of the ojdbc6.jar file, e.g. C:\CC\DevEnv\6.0\CuramSDEJ\drivers. Click next to continue.
- Review the properties on the configuration screen that opens. No changes should be required.

Setup the Oracle XA Database Driver

- Navigate to Resources->JDBC->JDBC providers.
- Choose the scope appropriate for your environment.
- Click New to add a new driver. This will open a configuration screen.

- Select Oracle from the the list in the Database type drop down supplied.
- Select Oracle JDBC Driver from the list in the Provider type drop down supplied.
- Select the XA data source from the list in the Implementation type drop down supplied.
- Set the Name field to be Oracle XA JDBC Driver For Reporting, if not filled in automatically set the description.
- Click Next to continue.
- Review the Class path and reset the environment variable to the location of the ojdbc6.jar file, e.g. C:\CC\DevEnv\6.0\CuramSDEJ\drivers. Click next to continue.
- Review the properties on the configuration screen that opens. There should be no need to change any of them.
- Click Finish to confirm the changes.

Set up the Oracle Non-XA Database Driver Data Source

- Navigate to Resources->JDBC->Data Source.
- Choose the scope appropriate for your environment
- Start the process for creating a new data source.
- Click New to add a new data source.
- Set the fields as follows:
 - Data source name: curamdm
 - JNDI Name: jdbc/curamdm
 - Click Next.
 - Select an existing JDBC driver, the driver created in Section "Setup the XA Database Driver" and hit next.
- Set the URL Value field to the value below where serverName is the name of the server hosting the database. Port is the port number the database is listening on and databaseName is the SID of the database. This is the URL to connect using Oracle SID name:
jdbc:oracle:thin:@serverName:port:databaseName
- Set the Data store helper class name to be Oracle.
- Leave all other fields untouched unless a specific change is required and click Next:
 - Set the Component-managed authentication alias drop down value to <none>.
 - Set the Mapping-configuration alias drop down value to <none>.
 - Set the Container managed-configuration alias drop down value to <Alias created in Section "Setup the non-XA Database Driver" >
- Leave all other fields untouched unless a specific change is required and click Next.
- Click Finish to confirm the changes and continue.

Setup the DB2 Environment Variable

- Navigate to Environment-WebSphere variables.
- Select the DB2UNIVERSAL_JDBC_DRIVER_PATH link from the list of environment variables. This will open the configuration screen for this variable.
- Set the Value field to point to the directory containing the Type 4/Type 2 drivers. This is normally the drivers directory under the SDEJ installation, e.g. D:\Curam\CuramSDEJ\drivers.
- Click OK to confirm the changes.

Set up the DB2 Database Driver Provider

- Navigate to Resources-JDBC-JDBC providers.
- Note: The appropriate scope where the data source is to be defined should be selected at this point.
- Click New to add a new driver. This will open a configuration screen.
- Select DB2 from the list in the database type drop down supplied.
- Select the DB2 Universal JDBC Driver Provider from the list in the Provider type drop down supplied.
- Select the XA data source from the list in the Implementation type drop down supplied.
- Click Next to continue.
- Review the properties on the configuration screen that opens. There should be no need to change any of them unless you are planning to connect to a zOS database. If so, verify that `#{DB2UNIVERSAL_JDBC_DRIVER_PATH}` field is pointing at the correct directory for your system. For example, it should point at the directory containing the DB2 Connect license jar, `db2jcc_license_cisuz.jar` provided by IBM® for zOS connectivity.
- Click Next and then Finish to confirm the changes.

Set up the DB2 Database Driver Data Source

- Select the DB2 Universal JDBC Driver Provider (XA) now displayed on the list of JDBC Providers. This will open the configuration screen for the provider.
- Select the Data sources link under Additional Properties.
- Click New to add a new data source.
- Set the fields as follows:
 - Data source name: `curamdm`
 - JNDI Name: `jdbc/curamdm`
- Set the Component-managed authentication alias drop down value to: `<valid for database>`; Set the Mapping-configuration alias drop down value to: `DefaultPrincipalMapping`. Set the Container-managed authentication alias drop down value to: `<valid for database>`; where the `<valid for database>` alias used is the one set up in the Creating the Data Source Login Alias section above.
- Leave all other fields untouched unless a specific change is required and click Next to continue.
- Click Finish to confirm the changes and continue.
- Select the newly created `DatasourceName` data source from the displayed list.
- Select the Custom Properties link under Additional Properties.
- Select the `fullyMaterializeLobData` entry.
- Set the value to be `false`.
- Click OK to confirm the change.

WebLogic

The BIRT Viewer application has been tested using a non-XA data source and a XA data source. The recommendation for creating a Datamart Data Source on WebLogic® is to use the non-XA data source.

Create a non-XA data source called <curamdm>, as below with the following settings, create a connection pool data source with no global transactions, add the Cúram node as a target.

Open the Administration Console as detailed in the previous section.

- Navigate to <DomainName>+Services+JDBC+Data Sources.
- Click the New button; Enter the following fields:
 - Name: curamdm
 - JNDI Name: jdbc/curamdm
- Change the Database Type to be Oracle and set the Database Driver to be Oracles Driver (Thin XA) for Instance connections.
- Click the Next button.
- Leave the default for Transaction Options and click the Next button.
- Set the URL Value field to the value below where serverName is the name of the server hosting the database. Port is the port number the database is listening on and databaseName is the SID of the database. This is the URL to connect using Oracle SID name:

```
jdbc:oracle:thin:@serverName:port:databaseName
```

- Leave all other fields untouched unless a specific change is required.
- Click the Next button.
- Review the settings and click the Next button.
- Select Targets, select the Cúram node.
- Click the Finish button.

It is advised to restart the AdminServer at this point, to ensure the changes are correct. To do this:

- Navigate to <DomainName>+Environment+Servers.
- Select the Control tab, then select AdminServer in the Server's list and click Shutdown+When work completes.
- Click the Yes button to shutdown the AdminServer.

1.11 Enabling WebSphere Application Server logging for BIRT

You can control which events are written to the log with the WebSphere Integrated Solution administrative console.

About this task

Set a logging level of **Severe** for the **org.eclipse.birt.*** component to output the appropriate amount of log information for BIRT in production environments.

Higher levels of logging such as `INFO` or `FINE` type log entries can be used on test systems to provide detailed logging information.

Note: The following steps require the `BIRT_VIEWER_LOG_LEVEL` property in `WEB-INF\web.xml` to be set to OFF.

Procedure

1. Open the WebSphere Integrated Solution administrative console in a browser.
2. Select **Servers > Server Types > WebSphere Application Servers**
3. Click **CuramServer**.
4. Click **Troubleshooting > Change Log Detail Levels**.
5. Expand **All Components**.
6. Scroll down and click **org.eclipse.birt.***. The **BIRT logging menu options** are displayed.
7. Click **Message and Trace Levels**.
8. Click **Severe**.
9. Click **Apply**.
10. Restart the application server.

1.12 Troubleshooting

This section details possible troubleshooting tips and fixes that the user may find helpful.

Cannot view BIRT content through Eclipse with H2

Condition

Recommended. The shortdesc element is optional and can be either on its own or inside an <abstract> element..

Cause

Cannot view BIRT content through Eclipse with H2.

<tsCauses>Optional. Clearly state the causes of the problem.</

tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the

steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the

problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

Build client.birt fails to execute.

Condition

Recommended. The shortdesc element is optional and can be either on its own or inside an <abstract> element..

Cause

Build `client.birt` fails to execute.

A build command exists which publishes BIRT content to the Cúram BIRT Viewer. This build command only executes cleanly if an environment variable `DEVENV` is created. This variable `DEVENV` must point to the parent directory of the BIRT Eclipse folder.

If the eclipse install folder is named `eclipsebirt` no further action is required. Otherwise, change the property `version-eclipsebirt=eclipsebirt`, for example, if birt eclipse is installed into directory named `BIEclipse` then update the property to `version-eclipsebirt= BIEclipse`. This property is located in the file `BIRTthird_party_version.properties` within the `BIApp` directory.

The library with the namespace CEFLibrary is not found.

Condition

Cause

The library with the namespace CEFLibrary is not found.

In Eclipse using the view outline model, right click on the report title and select refresh library. Please ensure your workspace is correctly configured, ensure all the steps in section Develop New Reports have been completed.

Remedy

Procedure

SSS

Error when executing a report.

Condition

Cause

If you get a message (the message text may be vague) that states you cannot connect to the database or connection is null or oracle message file is missing, the first task is to verify that the JDBC drivers are present in the ODA drivers directory.

Verify that the JDBC jar files are present in the ODA drivers directory, if this occurs within Eclipse copy the drivers to the ODA plugins directory within the Eclipse install path. If you find this error when viewing the application through tomcat then verify that the drivers have been copied by the client.birt build target which is executed from the BICContent directory.

Eclipse dies when previewing a BIRT report.

Condition

Cause

Eclipse dies when previewing a BIRT report.

This was found to be an issue of the JVM used by Eclipse.

You change the *birteclipse.bat* file to use the following setting *-vm %DEVENV%\eclipsejre\bin\javaw.exe*.

Remedy

Procedure

SSS

Report Page only displays one report when more that one report exists on the operational workspace.

Condition

Cause

Report Page only displays one report when more that one report exists on the operational workspace.

There may be a race condition for loading the JDBC drivers resulting in one report failing.

A fix will be provided in due course, the current solution is to initialize your BIRT environment before your Report Page loads. Do this by going to your CuramBIRTViewer\List page and execute any report (this will bootstrap BIRT).

Tomcat closes on report preview or ODA SQL preview.

Condition

Recommended. The shortdesc element is optional and can be either on its own or inside an <abstract> element..

Cause

Required. Clearly state the symptoms of the problem.

Tomcat closes on report preview or ODA SQL preview.

The addition of BIRT may result in Tomcat having memory issues for the permanent generation heap.

<tsCauses>Optional. Clearly state the causes of the problem.</

tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the

steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

Remedy

Procedure

1. Add the following parameter to Tomcat "-XX:MaxPermSize=256m"
2. Go to: Menu ->Preferences->Tomcat->JVM Settings->Append to JVM Parameters->Add -> "-XX:MaxPermSize=256m"

<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse>

BIRT Viewer does not run / List of reports not present.

Condition

Cause

Required. Clearly state the symptoms of the problem.-->BIRT Viewer does not run / List of reports not present.

Remedy

To ensure that everything is pulled down correctly, follow these steps:

Procedure

1. Delete the BIAppBIRTInfrastructureVersion.txt file from `<CURM_DIR>/ActualVersions`
2. Run `appbuild client.birt` from `<CURM_DIR>/BICContent`
3. Verify the `<CURM_DIR>\BIApp\CuramBIRTViewer\WebContent\WEB-INF\classes\biapp` folder exists.
4. Start tname server, start operational database server, start tomcat.
5. View BIRT Viewer from: localhost:9080/CuramBIRTViewer/

Exception thrown when running report

Condition

Cause

The below error is returned when trying to run a report:

```
SEVERE: An error happened while running the report. Cause:
java.lang.NullPointerException
at java.util.Hashtable.put(Hashtable.java:396)
at java.util.Hashtable.put(Hashtable.java:396)
at java.util.Hashtable.putAll(Hashtable.java:470)
at org.eclipse.birt.data.engine.executor.DataSource.<init>(DataSource.java:76)at org.eclipse.birt.data.engine.executor.DataSourceFactory.getDataSource(DataSourceFactory.java:75)
at org.eclipse.birt.data.engine.impl.PreparedOdaDSQueryOdaDSQueryExecutor.getDataSource(PreparedOdaDSQuery.java:241)
```

Ensure that you have completed this below step:

- Go to Menu -> Preferences -> Tomcat -> JVM Settings. On the classpath widow select the button directory and add in the full path to the EJBServer\project\properties directory for your stream.

Remedy

Procedure

SSS

Unable to view all entities after creating dataset.

Condition

Recommended. The shortdesc element is optional and can be either on its own or inside an <abstract> element.

Cause

Unable to view all entities after creating dataset.

There is a cap on the number of tables that BIRT displays when creating a Data Set.

<tsCauses>Optional. Clearly state the causes of the problem.</tsCauses><tsEnvironment>Optional. Describe any environmental details that are not already in the title or short description.</tsEnvironment><tsDiagnose>Optional. Clearly state the steps necessary to diagnose the problem. Optionally, include appropriate response role elements. Alternatively, imbed a task topic or conref that provides the steps to diagnose the problem.<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse></tsDiagnose>

You can change the number of tables and schemas that BIRT pulls back from the Database. Go to Menu -> Preferences -> Report Design -> Data Set Editor -> JDBC Data Set and increase the number of tables.

<tsUserResponse>Optional. When you have particular actions that are performed by particular users, use one or more of the ts*Response elements.</tsUserResponse>

1.13 Deployment Considerations

There are a number of deployment options for the Curam BIRT Viewer application.

1. Deploying the BIRT Viewer application on the same cluster as the Curam application. This configuration is the default configuration, where the default build scripts deploy both the Curam application and BIRT Viewer application on the same server.
2. Deploying the BIRT Viewer application on a separate application server instance to the Curam application. This configuration supports greater fault tolerance and allows both the Curam and BIRT application to be scaled independently of each other.

Default deployment

The installation of a standard deployment where BIRT is co-located on a single application server instance, is the standard installation for the application. Other deployment configurations will require manual steps, but these will not be supported by the installer.

Non-Homogeneous Deployment for WebSphere and BIRT

How to Install the CuramBIRTViewer application on a BI Cluster

Assumptions are as follows:

1. That the BI Application is currently installed and working correctly on the clients environment within a Curam Cluster of WebSphere application servers.
2. That all JDBC resources are in use and working and are defined at the Curam Cluster level.
3. That all steps associated with Application Server configuration are relevant to application servers to be used within a BI Cluster.

Uninstall the CuramBIRTViewer application from the Curam Cluster

1. Navigate to *Applications->Application Types* and click on the **WebSphere enterprise applications** link.
2. Select the **CuramBIRTViewer** application and click **Uninstall**
3. Confirm the removal of the application and save all of the changes to the configuration
4. Note: OOTB scripts automatically redeploy the CuramBIRTViewer. In production these will need to be manually deleted

Create a BI Cluster and Add the Application Servers

To create the BI cluster and register an associated application server it is necessary to follow the steps presented below within the Administration console.

1. Navigate to *Servers ->Clusters ->WebSphere application server clusters*.
2. Click on the **New** button to create the cluster
3. Enter the **Cluster Name**, for this example enter *BICluster*
4. Ensure the **Prefer Local** check box is checked
5. Ensure that the **Configure HTTP Session memory-to-memory replication** is unchecked for the moment. If this is required, it can be configured after the initial configuration steps have been taken
6. Click **Next** to continue
7. You are then given the opportunity to create the cluster members. Since a server was already created when the cell was created, select the option to create the cluster and convert the existing application server server1 as a member
8. Check the summary details and click on the **Finish** button to create the cluster
9. Click on the **Save** link on the screen displayed
10. Click on the **Save** button
11. Click on the **Ok** button

Database Drivers and associated environment variables

A number of JDBC data sources are defined for use by the Curam and CuramBIRTViewer applications and should have already been defined at either Curam Cluster or WebSphere Cell scope level.

These and any associated environment variables must be available for use by the CuramBIRTViewer application to be installed on the new BI Cluster so if not already defined at Cell level then the resources should again be defined at the BI Cluster level.

The steps to define the related environment variables and the JDBC data sources themselves are now presented within this document.

1. Navigate to *Environment->WebSphere Variables*.
2. Set the scope of the variable using the drop down list and selecting the **BICluster** (based on the scope they are to be defined at).
3. Click on the **New** button to add a new environment variable.
4. If connecting WebSphere to a **DB2 database instance**, enter the following Information and Click on the OK button:
 - Name = DB2UNIVERSAL_JDBC_DRIVER_PATH
 - Value = <JDBC Driver Installation> (In this example the value is c:\Curam\runtime\CuramSDEJ\drivers)

Then enter the following Information for the second variable and Click on the **OK** button.

- Name = UNIVERSAL_JDBC_DRIVER_PATH
- Value = <JDBC Driver Installation> (In this example the value is c:\Curam\runtime\CuramSDEJ\drivers)

Else If connecting WebSphere to an **Oracle database instance**, enter the following Information and Click on the OK button.

- Name = ORACLE_JDBC_DRIVER_PATH
- Value = <JDBC Driver Installation> (In this example the value is c:\Curam\runtime\CuramSDEJ\drivers)

It is important to note that as configuration settings set at Node level override Cluster level or Cell level definitions, if it is intended that the Cell or Cluster level configuration is to be used then it is necessary to delete the environment variables at Node level if they exist and only set them when the intention is to override the Cluster Configuration. Similarly, if the definitions are at cell level then delete the cluster level values.

1. Navigate to *Environment->WebSphere Variables*
2. Click on the **Browse Nodes** button
3. Select DB2UNIVERSAL_JDBC_DRIVER_PATH and delete
4. Select UNIVERSAL_JDBC_DRIVER_PATH and delete.

If existing JDBC data sources used by the Curam and CuramBIRTViewer application are defined at the WAS Cell level then they are visible to applications in both the Curam and BI related application server clusters. If they are defined at the cluster level then the same JDBC data sources should again be defined at the BICluster level.

The following steps describe how to define a new JDBC Provider and the various JDBC data sources at the BI cluster level. Using the WebSphere administration console:

1. Navigate to *Resources->JDBC->JDBC providers*
2. Set the scope using the drop down list and selecting the BICluster.
3. Click on the **New** button
4. Enter the following information and click on the **Next** button, If employing a DB2 database instance

- Database Type = DB2
- Provider Type = DB2 Universal JDBC Driver Provider
- Implementation Type = XA data source

Else if employing an Oracle database instance

- Database Type=Oracle
- Provider Type=Oracle JDBC Driver
- Implementation Type = XA data source

5. Click on the **Next** button.
6. The value of the environment variable `DB2UNIVERSAL_JDBC_DRIVER_PATH` (or `ORACLE_JDBC_DRIVER_PATH`) which was set in the previous step should now be employed by default as the value for the **Directory location for "db2jcc.jar, db2jcc_license_cisuz.jar" or "ojdbc6.jar"**. It must refer to a valid folder (on each node in the cluster) that contains the driver jar files.
7. Click on the **Next** button to view the summary screen before clicking on the **Finish** button.

It is then possible to define the required JDBC data sources using the following steps for each data source:

1. Navigate to *Resources->JDBC->JDBC Providers* and click on **DB2 Universal JDBC Driver Provider (XA)**.
2. From the scope drop down list select the **BICluster** from the list
3. Click on the **Data Sources** link under **Additional Properties**.
4. Press the New button to add a new data source and set the fields as follows before clicking on the Next button:
 - Data source name: `<DatasourceName>`
 - JNDI name: `jdbc/<DatasourceName>`
5. Check the **Select an existing JDBC provider** and select **DB2 Universal JDBC Driver Provider (XA)**.
6. Click Next to continue and set the fields as follows:
 - Database Name: The name of the DB2 database instance.
 - Driver type: 2 or 4 as required.
 - Server Name: The name of the DB2 database server.
 - Port Number: `<PORT NUMBER>` - Note: The default port number for DB2 on windows is 50000. Identifying the port DB2 is running on AIX can be done by executing the command: `db2 get dbm cfg | grep 'SVCENAME'`.
 - Leave the **Use this data source in container managed persistence** option checked and click the Next button.
7. Set the following drop down options to the name of the **J2C Authentication Alias for the Application Database** that was created previously in this section e.g. `databaseAdmin`.
 - Authentication alias for XA recovery.
 - Component-managed authentication alias.
 - Container-managed authentication alias.

8. Review the settings on the summary screen and then click the Finish button to conclude the creation of the data source.
9. Select the newly created data source from the displayed list.
10. Select the Custom Properties link under the Additional Properties section.
11. Select the **fullyMaterializeLobData** entry and set its value to **false**.
12. Click the OK button to confirm the change.

It is then necessary using the administration console to save the changes that have been made to the JDBC Configuration.

1. Click on the **Save** link on the screen displayed.
2. Ensure that the **Synchronize changes with Nodes** checkbox is **checked**.
3. Click on the **Save button**.
4. Check the notifications on the screen to verify that all the nodes have been updated.

The changes made must now be propagated to each node in the environment before the changes to the JDBC provider and data source can be tested.

1. Navigate to *System Administration->Nodes*
2. Select the nodes to synchronize the changes with. Click on the **Synchronize** or **Full Resynchronize** button.
3. Typically a success message is then presented. If there are problems with the synchronization process a message is displayed which will typically point to the SystemOut.log of the deployment manager.

It is then possible to test the data source configuration.

1. Navigate to *Resources->JDBC->JDBC providers* and select the appropriate scope value from the drop down list e.g. BICluster.
2. Select the JDBC Provider that has been created. In this example it should be DB2 Universal JDBC Driver Provider (XA).
3. Click on the **Data Sources** link under **Additional Properties**.
4. Select the Data Source that you want to test.
5. Click on the **Test connection** button.
6. You should get a successful message for each Node in the Cell as WebSphere will test the connection configuration from each node to the database.

Application Servers and required configuration steps

When a configuration scope is available in WebSphere it should be set to *cluster*. In this document everything prefixed by e.g. means a value must be provided but the one identified is an example. The following steps should be performed for each application server within the BI Cluster using the Administration console.

ClassLoader Configuration

Classloaders are responsible for locating and loading classes within a JVM. Within WebSphere, there are a number of settings for the classloader to be made when configuring the BI application within a cluster of application servers.

Using the Administration console:

1. Browse to *Servers->Server Types->WebSphere application servers*
2. Select the application server from the list. In the case of this example, the application server is **server1**
3. Set the **Classloader policy** to be **Multiple** Depending on the application classloader policy, an application classloader can be shared by multiple applications (SINGLE) or unique for each application (MULTIPLE). The application classloader policy controls the isolation of the applications that are running in the system. To ensure that the BI application that is installed within the cluster is isolated, the classloader policy is set to MULTIPLE
4. Set the **Class loading mode** to be **Parent first**. In WebSphere, this is the default setting but there is no harm verifying that this in the case. For the BI application, the order in which classes are loaded is important. For this reason, it is necessary to specify the **Parent first** classloader mode. This mode delegates the loading of classes to the application servers parent classloader before attempting to load the class from its own local classpath
5. These settings should appear as the defaults. If it is necessary to change them, click **Apply**, then save the changes

Web Container Configuration

1. Navigate to *Servers->Server Types->WebSphere application servers*
2. Select the application server from the list
3. Expand Web Container settings:
 - Select Web Container. Under general properties set the Default Virtual host to be default_host.
 - Select Web Container transport chains:
 - For each of the transport chains
 - For each TCP channel
 - Set the Tread pool to Web Container
4. Click **OK** to apply the changes

Save the changes made to the master configuration

Configure Port Access

The ports specified in the steps below must be unique if there are other existing application servers running on the same WebSphere node. The values that follow are just examples.

1. Navigate to *Servers->Server Types->WebSphere application servers*
2. Select the application server from the list;
3. Select the Ports link in the Communications box;
4. Select the details box;
5. Update the value of the **BOOTSTRAP_ADDRESS** port to the value specified in the Curam AppServer.properties file (e.g. curam.server.port=9810) and click OK.
6. Click the New button and set the following fields for the **Client TCP/IP port**:
 - User Defined Port Name: **CuramClientEndPoint**
 - Host: *
 - Port: **9049**

7. Click the New button and set the following fields for the **WebServices TCP/IP port**:
 - User Defined Port Name: **CuramWebServicesEndPoint**
 - Host: *
 - Port: **9082**
8. Navigate to *Servers->Server Types->WebSphere application servers*
9. Select the relevant server from the list.
10. Expand the Web Container branch in the Container Settings box.
11. Select the Web Container transport chains link
12. Click the New button and set the following fields for the Client transport chain:
 - Name: **CuramClientChain**
 - Transport Chain Template: WebContainer-Secure
 - Click Next
 - Use Existing Port: **CuramClientEndPoint**
 - Click Next and Finish
13. Click the New button and set the following fields for the WebServices transport chain:
 - Name: **CuramWebServicesChain**
 - Transport Chain Template: WebContainer
 - Click Next
 - Use Existing Port: **CuramWebServicesEndPoint**
 - Click Next and Finish
14. Select the newly created **CuramClientChain**
15. Select the HTTP Inbound Channel link
16. Ensure the User persistent (keep-alive) connections check-box is checked
17. Click the OK button to confirm the addition
18. Navigate to *Environment->Virtual Hosts*
19. Click the New button to add a new Virtual Host by setting the following fields:
 - Name = **client_host**
 - Repeat this step using the replacing *client_host* with *webservices_host*;
20. Select the *client_host* link from the list of virtual hosts.
21. Select the Host Aliases link in the Additional Properties box
22. Click the New button to add a new alias by setting the following fields:
 - Host Name = *
 - Port = 9049

where 9049 is the port used in step 5. Repeat this step for the other Virtual Host and port used (e.g. **webservices_host**, 9082). Note: If there are a number of server instances running on the same machine within a node in the cluster then the port for the CuramClientChain and CuramWebServicesChain will be different to 9049 and 9082 respectively for a second or third etc application server instance. In this case add a corresponding Host Alias to both the *client_host* and *webservices_host* (9050 and 9083 for example for a second application server instance).

23. Click the OK button to confirm the addition

24. Save the changes to the master configuration when prompted

WebSphere Logs

It is often worth considering to enable WebSphere to output logs to the fast disks using the following steps.

1. Navigate to *Servers->Server Types->WebSphere application servers*.
2. Select the application server from the list. In the case of this example, this will be **server1**
3. Expand **Container settings** and click on **Transaction Service**.
4. Enter the path to the fast disks.

ORB Container Configuration

Using the Administration console:

1. Navigate to *Servers->Server Types->WebSphere application servers*
2. Select the application server from the list. In the case of this example this will be **server1**.
3. Expand **Container Services** and select **ORB Service**. Ensure the checkbox **Pass by reference** has been selected.
4. Click **OK** to apply the changes.
5. Save the changes made to the master configuration.

JVM Configuration

Note: the values below are just examples and a client may consider to use alternative values or alter these following tuning.

Using the Administration console:

1. Navigate to *Servers->Server Types->WebSphere application servers*
2. Select the application server from the list. In the case of this example this will be **server1**.
3. In the **Server Infrastructure** section expand **Java and Process Management** and **Process Definition**. Under **Additional Properties** select **Java Virtual Machine** before making the following settings:
 - Verbose garbage collection = true
 - Initial Heap Size = e.g. 1024
 - Maximum Heap Size = e.g. 1024
 - Generic JVM arguments = e.g. `-Xgcpolicy:gencon -Xmn700m -Xverbosegclog:d:\MyLogs\gc.log`

"-Xgcpolicy:gencon" refers to Generational concurrent and handles short-lived objects differently than objects that are long-lived. Applications that have many short-lived objects can see shorter pause times with this JVM policy while still producing good throughput. "-Xmn700m" sets the initial and maximum size of the new (nursery) heap to the specified value when using the `-Xgcpolicy:gencon` setting.

4. For the **Java Virtual Machine** select **Custom Properties** in the **Additional Properties** section.
5. Click the *New* button and set the properties as follows:

- Name: **com.ibm.websphere.security.util.authCacheCustomKeySupport**
 - Value: **false**
6. Click the *OK* button to add the property and then save all changes to the master configuration when prompted.
 7. For non-Windows platforms it is also recommended to create the follow property. For the **Java Virtual Machine** select **Custom Properties** in the **Additional Properties** section. Click the New button and set the properties as follows:
 - Name: **java.awt.headless**
 - Value: **true**
 8. Click the *OK* button to add the property and then save all changes to the master configuration when prompted.

Install the CuramBIRTViewer application

It is assumed that an appropriate CuramBIRTViewer.ear file exists and is ready to be installed on the environment that has been configured. Once any necessary pre-install configurations have been made it is then time to install the application using the Administration console on the deployment manager:

1. Navigate to *Applications->New Application* and click on the **New Enterprise Application** link
2. Check the **Local File System** option before providing the absolute path to the *CuramBIRTViewer.ear* file.
3. Click on the **Next** button. The EAR file will then be loaded. On some machines this may take some time.
4. Proceed by selecting **Detailed - Show all installation options and parameters** when asked **How do you want to install the application?** and Click Next. It is important to target any HTTP Server and the BI Cluster to deploy the application on during this sequence of installation steps.
5. Proceed using the default values until the Curam BIRT Viewer application has been installed. It is necessary to updating the applications class loading policy once it has been installed using the following steps in the administration console.
 1. Navigate to *Applications->Application Types->WebSphere enterprise Applications->Curam*
 2. Click the **Class loading and update detection** link under Detail Properties.
 3. Set the **Class loader order** to be **Classes loaded with local class loader first (parent last)**. This is an important step to avoid encountering class loader-related exceptions when the application server instance is restarted.
6. If a HTTP Server instance has already been installed then choose to map the Curam BIRTViewer module to this server (along with the specific BIRT cluster) on the Map Modules to Servers page by selecting each module and hitting the Apply button. The installation can be verified by examining the contents of the following folder on each of the application server machines:
 - `%WAS_HOME%\profiles\<NodeName>\installedApps\<CellName>`

This folder should contain a *CuramBIRTViewer.ear* folder and all related jar files for the Curam BIRTViewer application.

When the CuramBIRTViewer application is deployed in a standalone environment the JSESSIONID cookie that is defined at the Application Server level is shared by the various applications that are deployed on that application server.

Within a WebSphere cell where each application can be installed separately to application servers within independently defined clusters it is necessary to override the session management settings at the application level and specify the cookie paths or the cookie name to be unique for that specific application on that cluster.

For both the Curam and CuramBIRTViewer application update the session management and cookie details as per the steps below:

1. Navigate to *Applications->Application Types->WebSphere enterprise Applications-><application-name>*
2. Under **Web Module Properties** select **Session Management** and on the resulting screen ensure that the check box beside **Override session management** is selected.
3. Also on that page ensure that the check box beside **Enable cookies** is selected and then click on the **Enable cookies** link itself.
4. Under **General Properties** enter the following details for the Curam application
 - Cookie Name: JSESSIONID
 - Cookie Path: /Curam

Enter the following details for the CuramBIRTViewer application

- Cookie Name: BIRTJSESSIONID
 - Cookie Path: /CuramBIRTViewer
5. Click **Apply** and Save changes to the master configuration in the usual manner and attempt to synchronize the nodes within the cluster.

Update the HTTP Server Plugin and restart HTTP Server

It is assumed that during the deployment of the CuramBirtViewer application it was mapped correctly to a HTTP Server instance in addition to the BI Cluster. To ensure that requests for the CuramBIVIEWER application are successfully propagated to the application server cluster within the WebSphere cell the following steps should be employed to regenerate and redeploy the HTTP server plugin. The HTTP Server should also be restarted following these steps.

It is assumed that an appropriate CuramBIRTView.ear file exists and is ready to be installed on the environment that has been configured.

Once any necessary pre-install configurations have been made it is then time to install the application using the Administration console on the deployment manager:

1. Navigate to *Applications->New Application* and click on the **New Enterprise Application** link
2. Check the **Local File System** option before providing the absolute path to the *CuramBIRTViewer.ear* file.

3. Click on the **Next** button. The EAR file will then be loaded. On some machines this may take some time.
4. Proceed by selecting **Detailed - Show all installation options and parameters** when asked **How do you want to install the application?** and Click Next.
5. Proceed using the default values until the Curam BIRT Viewer application has been installed.
6. If a HTTP Server instance has already been installed then choose to map the Curam BIRTViewer module to this server (along with the specific BIRT cluster) on the **Map Modules to Servers** page by selecting each module and hitting the Apply button. The installation can be verified by examining the contents of the following folder on each of the application server machines:

- %WAS_HOME%\profiles\<<NodeName>\installedApps\<<CellName>

This folder should contain a CuramBIRTViewer.ear folder and all related jar files for the Curam BIRTViewer application.

When the CuramBirtViewer application is deployed in a standalone environment the JSESSIONID cookie that is defined at the Application Server level is shared by the various applications that are deployed on that application server.

Within a WebSphere cell where each application can be installed separately to application servers within independently defined clusters it is necessary to override the session management settings at the application level and specify the cookie paths or the cookie name to be unique for that specific application on that cluster.

For both the Curam and CuramBIRTViewer application update the session management and cookie details as per the steps below:

1. Navigate to *Applications->Application Types->Webshpere enterprise Applications-><application-name>*
2. Under **Web Module Properties** select **Session Management** and on the resulting screen ensure that the check box beside **Override session management** is selected.
3. Also on that page ensure that the check box beside Enable cookies is selected and then click on the **Enable cookies** link itself.
4. Under **General Properties** enter the following details for the Curam application
 - Cookie Name: JSESSIONID
 - Cookie Path: /Curam

and the following details for the CuramBIRTViewer application

- Cookie Name: BIRTJSESSIONID
 - Cookie Path: /CuramBIRTViewer
5. Click **Apply** and Save changes to the master configuration in the usual manner and attempt to synchronize the nodes within the cluster.

1.14 Runtime Architecture

Default Runtime Architecture

The runtime architecture would be structured in the following order:

1. Database
2. Combined with the following:
 - Server Application
 - BIRT Server Application
3. Into:
 - Client Application

which is selected from based on configuration

BIRT Viewer Application

This is a new component which will wrap the open source BIRT Engine.

BIRT Engine Abstraction Layer

This component is responsible for managing the look-up of report templates when the BIRT Engine is in use.

BIRT Viewer

This component represents the BIRT Report Engine, which will run and render the selected template. This will serve reports which in turn access the Source Database.

BIRT

This section outlines how BIRT interacts with other application components. It also outlines some other areas which require consideration.

Rendering report content using BIRT

When a user opens a page containing report content, on a system where BIRT is the deployed Engine, the functional components consisting of the following:

1. **Client Browser**
2. **Curam Client Application**
3. **Curam Server Application**
4. **Curam Database**
5. **Curam BIRT Server App**

interact with each other in the following order:

- 2 Requests a page from 1
- 3 Calls a Facade from 2
- 4 Requests Data from 3
- 4 Returns Data to 3
- 3 Returns Data to 2

- 5 Requests Data from 2
- 4 Requests Data from 5
- 4 Returns Data to 5
- 5 Returns Report to 2
- 2 Renders Page to 1

BIRT Performance and Scalability

The key elements are:

- Well designed SQL queries influencing performance.
- The option of using summary tables within the transactional database, where needed to ensure acceptable performance.
- The option of using data from the data warehouse where no acceptable performant alternative can be found.

Security

There are potentially additional issues here, where the security of data within aggregated views will need to be considered when developing report content. As a basic principle the assumption is that where data has been aggregated; so that individual details can no longer be discerned, then it is acceptable to ignore security based on the underlying data. In switching to views where the underlying data is accessible, then the security must be enforced.

To this end we are recommending two approaches for constructing reports:

- Where the data will only be presented in the aggregate and anywhere that underlying data is accessible only through the application, the report should query the application database directly.
- Where the data will be presented in a way that exposes the underlying data, or where very specific data security requirements exist, the report should make EJB calls to facades in the application to retrieve the data.

1.15 Viewing Reports - Dependencies

Viewing Reports

In terms of features provided by the BIRT infrastructure when viewing reports, it is important to note that there are 3 distinct environments, each of which has its own dependencies.

1. Application Server
2. Tomcat - BIRT Viewer
3. BIRT Eclipse

Application Server

When testing the reports using an Application server, the BIRT infrastructure automatically sets the JNDI data source property name for all reports. The JNDI data source must already have been created on the Application Server. This will override any settings within the reports.

Tomcat - BIRT Viewer

When testing the reports using Tomcat, the developer must first, manually enter data source properties within the `SERVER_DIR\project\properties\Bootstrap.properties` file. This will override any settings within the reports. The reports can then be tested successfully

BIRT Eclipse

When testing the reports locally within BIRT Eclipse, the developer must manually enter data source properties for each report as explained in the 'Developing New Reports' section. The developer will then use the 'Preview' tool within the report itself.

Notices

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the Merative website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of Merative

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Merative.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

Merative reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by Merative, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

MERATIVE MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

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

Any references in this information to non-Merative websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those

websites are not part of the materials for this Merative product and use of those websites is at your own risk.

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 actual people or business enterprises is entirely coincidental.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

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 Merative, 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. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

Privacy policy

The Merative privacy policy is available at <https://www.merative.com/privacy>.

Trademarks

Merative™ and the Merative™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Adobe™, the Adobe™ logo, PostScript™, and the PostScript™ logo are either registered trademarks or trademarks of Adobe™ Systems Incorporated in the United States, and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, 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, and service names may be trademarks or service marks of others.