One Design
One Server
One User Experience

Deploying to a BIRT iHub System

Contains information proprietary to:
Actuate Corporation, 951 Mariners Island Boulevard, San Mateo, CA 94404

www.actuate.com

Document No. 130131-2-430360 January 23, 2013

Contents

About Deploying to a BIRT iHub System

Deploying to a BIRT iHub System provides information for volume administrators, report developers, and data modelers regarding deploying reports and information objects to an iHub Encyclopedia volume.

Deploying to a BIRT iHub System includes the following parts and chapters:

- *About Deploying to a BIRT iHub System.* This chapter provides an overview of this guide.
- *Chapter 1. Deployment overview.* This chapter describes ODA drivers and the connection configuration file.
- *Chapter 2. Deploying BIRT reports to BIRT iHub.* This chapter describes how to publish BIRT reports and report resources to an Encyclopedia volume, how to deploy Java classes, and how to install JDBC and ODA drivers and plug-ins.
- *Chapter 3. Connecting to data sources.* This chapter describes how to use a connection profile, how to define environment variables, and how to connect to various data sources.
- *Chapter 4. Configuring fonts in BIRT iHub.* This chapter describes how to configure fonts for use with BIRT reports.
- *Chapter 5. Working with BIRT encryption.* This chapter describes how to deploy encryption plug-ins.
- *Chapter 6. Using custom emitters.* This chapter describes how to deploy custom emitters.
- *Chapter 7. Deploying information objects.* This chapter describes the specific procedures for deploying information objects to an Encyclopedia volume.

Part One

Introduction to deploying reports and information objects

Chapter

1

Deployment overview

This chapter contains the following topics:

- Deployment overview
- About accessing other types of data sources
- Using a connection configuration file

Deployment overview

Actuate users work with reports and information objects when they log in to an iHub Encyclopedia volume using Information Console. The volume administrator is responsible for working with the developer to deploy reports and information objects to the Encyclopedia volume and make them available to users.

Deploying reports

Actuate offers several report designers including:

- BIRT Studio
- BIRT Designer Professional
- e.Report Designer Professional
- BIRT Spreadsheet Designer

BIRT Studio is a web-based report design tool for business users. With BIRT Studio, a user designs a report and saves the report design in the Encyclopedia volume. It is not necessary to publish the report design to the Encyclopedia volume. For more information about designing reports with BIRT Studio and integrating BIRT Studio with iHub, see *Using BIRT Studio - iHub Edition.*

The other report designers reside on the desktop. A report developer designs a report in one of these report designers and publishes the report executable to the Encyclopedia volume. The volume administrator then performs the following tasks:

- Places the report executable in the appropriate folder
- Schedules a job to generate the report document in the appropriate format, for example PDF
- Assigns privileges on the report document for users and roles
- Notifies users that the report document is available

For more information about these tasks, see *Managing an Encyclopedia Volume.*

Before the volume administrator makes reports available to users, he may need to perform additional tasks, for example, define database connections and make fonts available to reports. These tasks are discussed in the following chapters.

Deploying information objects

An information object can be used as a data source in any type of report design. To deploy an information object, you must publish the information object to the resources folder for the appropriate Encyclopedia volume. If you want users to be

able to execute an information object without having execute privilege on its sources, apply the trusted execute privilege to the information object.

About accessing other types of data sources

A report or information object developer can access a variety of data sources using predefined data drivers. To access other types of data, you can create a custom data driver, known as an open data access (ODA) driver.

An ODA driver supports both design-time and run-time functionality. The report or information object designer uses the driver to build a connection to the data source, retrieve parameter and data row definitions, and compile these definitions for use at run time. At run time, BIRT iHub loads the driver. Then, the driver creates the connection and data source instance and fetches the requested data.

Each ODA driver supports one type of connection and can support multiple instances of that connection type. Each driver can support multiple data sources. The driver must be installed on the system where you design the report or information object and on BIRT iHub.

For more information about installing ODA drivers on BIRT iHub, see *Configuring BIRT iHub.*

Using a connection configuration file

Connection configuration files enable deployment of reports and information objects to a production environment. You can use a connection configuration file to specify which data connections to use in the design environment. You can use the same data connections or specify different connections for use when BIRT iHub runs the report or information object in the production environment. By specifying both design-time and run-time connections, you do not have to change the report design or information object when deploying to a production environment. For more information about connection configuration files, see the documentation for the report or information object designer and *Configuring BIRT iHub.*

Part Two

Deploying BIRT reports

Chapter

2

Deploying BIRT reports to BIRT iHub

This chapter contains the following topics:

- About deploying BIRT reports
- Publishing a report to iHub
- Publishing a report resource to Actuate BIRT iHub
- Deploying Java classes used in BIRT reports
- Installing a custom JDBC driver
- Installing custom ODA drivers and custom plug-ins
- Configuring the BIRT design cache

About deploying BIRT reports

This chapter describes how to run and distribute BIRT reports in the Actuate business reporting system. To deploy BIRT reports, you need to understand the environment in which the reports run.

iHub provides a central location from which business users can access, run, and view reports. You can also use Actuate Information Console to run report executables, and to manage, generate, view, and print report documents.

Actuate BIRT Designer Professional, the tool that you use to develop BIRT reports, has built-in capabilities that facilitate the deployment process. The Actuate BIRT Designer integrates with iHub in several important ways to support performing the following tasks:

- Use an open data access (ODA) information object data source that resides on an Encyclopedia volume.
- Publish a report design to an Encyclopedia volume.
- Publish a resource to an Encyclopedia volume.
- Install a custom JDBC driver for use by BIRT reports running in the iHub environment.

A user accesses BIRT Studio from Actuate Information Console. BIRT Studio is a licensed option of iHub. To deploy templates and reports to BIRT Studio you use the deployment features available in Actuate BIRT Designer Professional. The following sections describe these capabilities.

Publishing a report to iHub

The purpose of publishing a report to iHub is to make it accessible to a large number of users. A published report is available to manage, meaning you can schedule re-running the report to include updates from the data sources. You can also choose who can access part or all of the report.

Actuate BIRT Designer Professional provides tools for easy deployment of reports, templates and their resources to iHub. The designer connects directly to an iHub and deploys the reports to selected iHub folders. The designer provides an iHub Explorer view for managing iHub connections. Using iHub Explorer, you can create iHub connection profiles to store the connection properties to a specific Encyclopedia volume. Figure 2-1 shows iHub Explorer displaying an iHub profile.

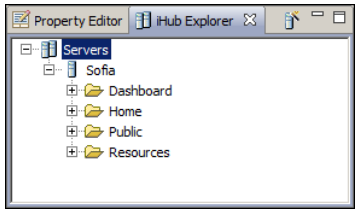

Figure 2-1 iHub Explorer view

How to create an iHub profile

1. In Actuate BIRT Designer, open iHub Explorer. If you do not see the iHub Explorer view in the designer, select Windows➛Show View➛iHub Explorer.
2. In iHub Explorer, right-click Servers, and choose New iHub Profile.
3. In New iHub Profile, specify the connection information. Figure 2-2 displays an example of connection properties provided for an iHub named Athena.

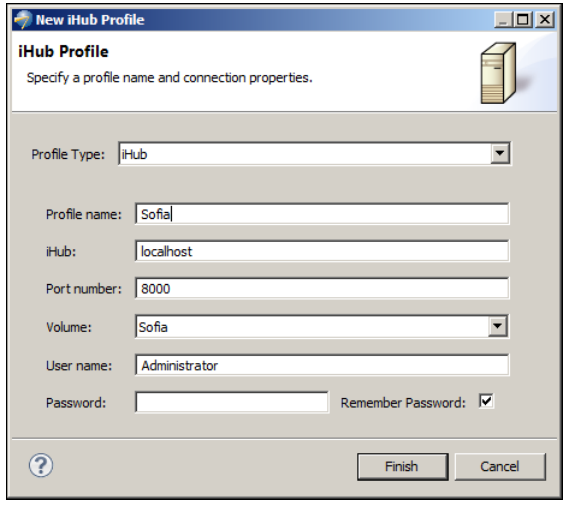

Figure 2-2 Setting properties for an iHub profile

1. In Profile type, select iHub.
2. In Profile name, type a unique name that identifies the new profile.
3. In iHub, type the name or IP address of the iHub.
4. In Port number, type the number of the port to access iHub.
5. In Volume, select the iHub Encyclopedia volume.
6. In User name, type the user name required to access the volume.
7. In Password, type the password required to access the volume.

8 Select Remember Password, you can choose to remember the password.

4 Choose Finish to save the iHub profile. The iHub profile appears in the iHub Explorer as shown in Figure 2-1.

How to publish a report design to iHub

1 Choose File➤Publish➤Publish to iHub.

2 On Publish to iHub, select a server profile. If there is no appropriate profile in the iHub profile list, create a new profile by choosing Add. In New iHub Profile, complete the information as shown in Figure 2-2. Then, choose Finish.

3 Select the project where the report you want to publish is located.

4 Select Publish Report Designs.

5 On Publish to iHub, select the report, as shown in Figure 2-3.

6 In Publish location, type or browse for the location on the Encyclopedia volume in which to publish the report design, as shown in Figure 2-3.

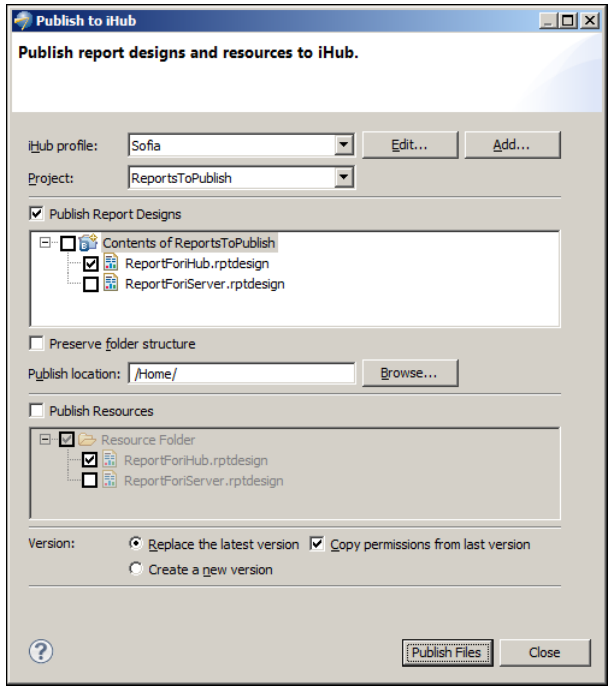

Figure 2-3 Selecting a server and location

7 Choose Publish Files. A new window, showing the file upload status, appears.

On Publishing, wait until the upload finishes, then choose OK as shown in Figure 2-4.

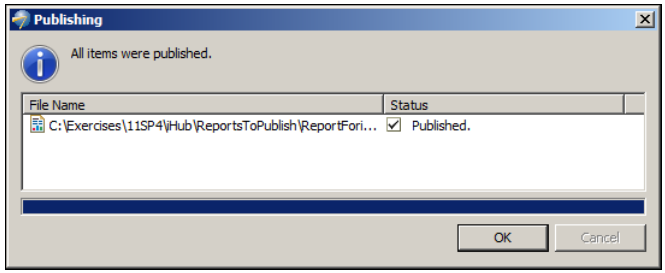

Figure 2-4 Confirming the report publishing

8 On Publish to iHub, choose Close.

Publishing a report resource to Actuate BIRT iHub

BIRT reports frequently use files with additional information to present report data to the users. A BIRT resource is any of the following items:

- Static image that a BIRT report design uses
- Report library
- Properties file
- Report template
- Data object
- CSS file
- External JavaScript file
- SWF file of a Flash object
- Java event handler class packaged as a Java archive (JAR) file

You can publish BIRT resources from the BIRT Designer Professional's local resource folder to iHub. By default, the Resource folder is the current report project folder. If you use shared resources with other developers and the resource files for your reports are stored in a different folder, you can change the default Resource folder. Use Windows➤Preferences➤Report Design➤Resource menu to set the Resource folder.

In the Encyclopedia volume, the Resource folder is set to /Resources by default. In the sample Encyclopedia volume, the /Public folder contains sample reports. The libraries and templates used by these sample reports are stored in the /Resources folder.

If the Resource folder in the Encyclopedia volume is different from the default, before publishing a resource, you need to set up the Resource folder in the Encyclopedia volume.

How to change the Resource folder on an Encyclopedia volume

1. Open Management Console and log in to the Encyclopedia volume.
2. Create a folder to designate as a resource folder.
3. Choose Volume➛Properties.
4. On Properties➛General, in Resource folder, type or browse to the folder to which you want to publish BIRT resources. Choose OK.

How to publish a resource from the Resource folder to iHub

1. In Actuate BIRT Designer, choose File➛Publish➛Publish to iHub.
2. Select the iHub profile, as shown in Figure 2-5.
3. On Publish to iHub, you can publish reports and resources. Choose Publish Resources.
4. On Publish to iHub, expand the Actuate BIRT Designer's Resource Folder and select the resources to publish.

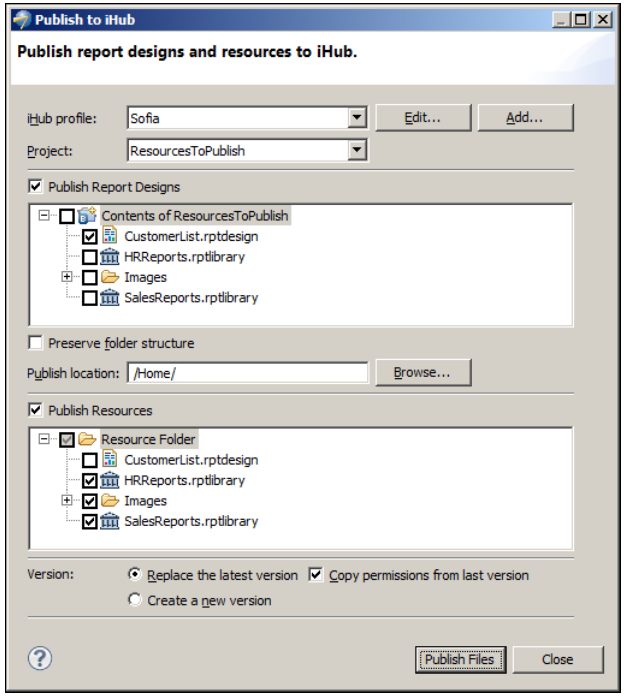

Figure 2-5 Publish to iHub

5 Choose Publish Files. A new window showing the file upload status appears.

6 Choose OK when the upload finishes.

7 In Publish to iHub, choose Close.

Deploying Java classes used in BIRT reports

A BIRT report uses scripts to implement custom functionality. For example, you can use scripts to create a scripted data set or to provide custom processing for a report element. When you deploy a BIRT report to an Encyclopedia volume, you must provide iHub with access to the Java classes that the scripts reference. You package these classes as JAR files that can be recognized and processed from an iHub Java factory process. There are two ways to deploy Java classes:

- Deploy the JAR files to the Encyclopedia volume.

  This method supports creating specific implementations for each volume in iHub. This method of deployment requires packaging the Java classes as a JAR file and attaching the JAR file as a resource to the report design file. You treat a JAR file as a resource in the same way as a library or image. Using this method, you publish the JAR file to iHub every time you make a change in the Java classes.

- Deploy the JAR files to the following iHub subdirectory:

  ```
  $ACTUATE_HOME\iHub\resources
  ```

  This method uses the same implementation for all volumes in iHub. Actuate does not recommend deploying JAR files to an iHub /resources folder because iHub must be restarted after deploying the JAR file. Another disadvantage of this deployment technique is that the JAR file, deployed in the iHub /resources directory is shared across all volumes, which can cause conflicts if you need to have different implementations for different volumes. When using this method, you do not have to add the JAR file to the report design's Resource property.

How to configure BIRT reports and deploy a JAR file to an Encyclopedia volume

1 Package the Java classes as a JAR file and copy the JAR file to the Actuate BIRT Designer resource folder.

2 Open the report design in Actuate BIRT Designer.

3 In Outline, select the root report design slot and select Resources property group in the Property Editor window.

4 In Resources, in JAR files, choose Add and navigate through the Resource folder to select the JAR file, as shown on Figure 2-6.

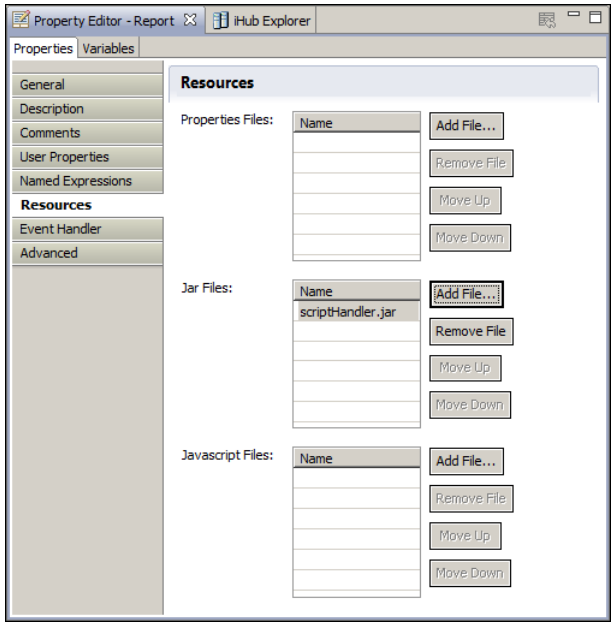

Figure 2-6 Add a JAR file as a resource to a report

When the report executes, the engine searches for the required classes and methods only in this JAR file.

5 Choose File➤Publish➤Publish to iHub to publish the report and the JAR file to the iHub.

1 Select the server profile.

2 Select Publish Report Designs, choose the report, and the folder on iHub where you want to copy the report.

3 Select Publish Resources and choose the JAR file. The JAR file is stored in the Encyclopedia volume's Resource folder.

6 Run the report from Information Console or Management Console.

How to deploy a JAR file to an iHub /resources folder

1 Copy the JAR file to the following iHub subdirectory:

```
$ACTUATE_HOME\iHub\resources
```

$ACTUATE_HOME is the folder where Actuate products install. By default, it is C:\Program Files\Actuate11 for version 11.

2 Publish the report to iHub.

3 Restart iHub.

4 Run the report from Information Console or Management Console.

Installing a custom JDBC driver

In order to run a BIRT application in the iHub environment when the BIRT application uses a custom JDBC driver, you must install the JDBC driver in the following location:

```
$ACTUATE_HOME\iHub\Jar\BIRT\platform\plugins
   \org.eclipse.birt.report.data.oda.jdbc_<VERSION>\drivers
```

Installing custom ODA drivers and custom plug-ins

All custom ODA drivers and custom plug-ins must be installed in the following folder:

```
$ACTUATE_HOME\iHub\MyClasses\eclipse\plugins
```

By default, Actuate iHub and Information Console load custom plug-ins from this folder. If your application uses a different location to store custom plug-ins, you must set this location in each product's link file. Actuate products use link files to locate folders where the custom plug-ins are deployed. The name of the link files are customPlugins.link and customODA.link. As the file names suggest, the customODA.link file stores the path for custom ODA drivers, and customPlugins.link is for all plug-ins used by BIRT reports and the BIRT engine, such as custom emitters, or Flash object library plug-ins. Typically, the link files are stored in a \links subfolder of the Eclipse instance of the product. For Actuate BIRT Designer, for example, the file is located in:

```
$ACTUATE_HOME\BRDPro\eclipse\links
```

You can change the path in customPlugins.link file and deploy the plug-ins to the new location.

When you install the InformationConsole.war file to your own J2EE application server, the shared folder MyClasses is not available. In this case, custom plug-ins should be copied to this folder:

```
WEB-INF/platform/dropins/eclipse/plugins
```

The locations of the link files for each product are listed in Table 2-1.

Table 2-1 .link file locations

Product	Paths of .link files
Actuate BIRT Designer Professional	$ACTUATE_HOME\BRDPro\eclipse\links
Actuate BIRT iHub	$ACTUATE_HOME\iHub\Jar\BIRT \platform\links

Configuring the BIRT design cache

By default, iHub caches a BIRT design, including access privileges. Caching benefits users who access the design concurrently. Users who request access to the same design share the cached design if they have the required privileges. Performance can improve because iHub does not have to repeatedly load the design. Generally, the fewer number of designs iHub needs to load, the better the response time.

By configuring the cache timeout, the administrator can control how long the design remains in the cache. iHub removes the design from the cache when the time elapses. Increasing the timeout increases the time the design stays in the cache. Decreasing the timeout purges the design sooner.

The administrator can also configure a limit on the number of designs in the cache. When the cache reaches the limit, caching stops. To disable caching, set this limit to zero.

How to configure the BIRT design cache

1. In Configuration Console, on Server Configuration Templates, expand Viewing Service, BIRT, and BIRT Content Caches and choose Design Cache.
2. In Cache timeout for BIRT designs, accept the default, 1800 seconds or 30 minutes, as shown in Figure 2-7. Alternatively, type another value.

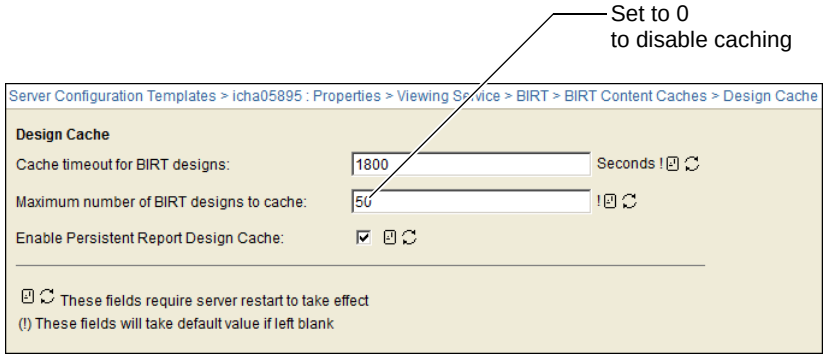

Figure 2-7 Configuring the BIRT design cache

3 In Maximum number of BIRT designs to cache, accept the default, 50, or type another value that limits the number of designs in the cache. To disable caching, set to 0. Choose OK.

4 Restart iHub.

Chapter

3

Connecting to data sources

This chapter contains the following topics:

- About data source connections
- About data source connection properties

About data source connections

Actuate Customer Support publishes the Supported Products and Obsolescence Policy document that describes the data sources, drivers, operating systems, and other software requirements for connecting iHub to data sources. Actuate Supported Products and Obsolescence Policy, available on the Actuate Support site at the following URL, also contains information about the required patches:

http://support.actuate.com/documentation/spm

iHub connects to data sources when generating reports and using the Actuate Caching service (ACS). The report designer specifies data source connection information in the report design, or you specify it in an external connection configuration file. In most cases, iHub and the database run on different computers for load-balancing purposes, but this division is not mandatory. Running iHub on the database host can improve performance.

The iHub installation process installs and configures DataDirect Connect for ODBC drivers and JDBC drivers. You can also use third-party drivers to connect to data sources, but you must license, install, and configure them.

To connect BIRT reports to other JDBC data sources, place the .jar files for the custom database driver in:

```
AC_SERVER_HOME\iHub\Jar\birt\platform\plugins
   \org.eclipse.birt.report.data.oda.jdbc<version>\drivers
```

About data source connection properties

Every BIRT data source object specifies the connection properties required to connect to an underlying data source. Typically, many report designs access the same data source. Rather than typing the same connection properties for each design, you can create a connection profile to reuse the same connection properties across multiple designs. Usually you change database connection properties used in the development environment when you publish the reports to Actuate iHub.

To change the connection properties dynamically when you design or deploy your reports, you can use one of two approaches, connection configuration file or connection profile. The connection profile approach is the recommended method of managing database connections. The following sections describe these two approaches.

Using a connection profile

The connection profile includes information about the database driver, a user ID, password, port, host, and other properties related to the type of data source. BIRT

supports using a connection profile when creating a data source in a report design. When a connection profile changes, the BIRT report picks up those changes. This behavior makes migration from a test to a production environment straightforward.

You can use connection profiles for all data source types, except SQL Query Builder data sources. If you have to use connection profiles for this type of data source, you must define a unique connection profile in each report.

Connection profiles are stored in text files called connection profile stores. Connection profile stores can contain multiple connection definitions for various ODA data sources. The default connection profile store is ServerProfiles.dat, located in the .metadata folder in your workspace.

You can also define your own connection profile store, and choose an absolute or a relative file path to store it. It is a good practice to create and use your own profile store file, instead of the default store. Using the default store requires using absolute file paths for a profile location and involves additional procedures of exporting a profile.

Using the Data Source Explorer to create a connection profile limits the connection profile location definition to an absolute file path only, while Data Explorer allows a relative and absolute file path definition. When using a relative file path, the Resource folders in the designer and iHub are used as the base folders. At design time, the BIRT Resource folder points to either the project root or an item in the workspace or a folder on the file system. This setting is available under Report Design➤Resources node in the Preferences view. At runtime, the BIRT Resource folder points to the Resource folder on the iHub.

Like other BIRT resource files, you must deploy the connection profile store to the iHub when you deploy the report that uses it. By default, BIRT Designer deploys relative path connection profiles to the iHub resource folder.

The connection profile store file can be encrypted using BIRT secured encryption framework. The default extension for the connection profile is .acconnprofiles. This extension is tightly integrated with the default out-of-the-box encryption.

Deploying a connection profile

Connection profiles that use relative paths are deployed the same way as report resources, and by default they are saved to the iHub resource folder. For more details on how to publish resources to iHub, see “Publishing a report resource to iHub,” in Chapter 26 of *Using Actuate BIRT Designer Professional.*

When deploying reports that use absolute connection profiles, you must deploy the connection profile to the correct folder in the file system on the iHub machine. For example, if a report uses a connection profile stored in folder C:\ConnProfile\MySQL.acconnprofiles, you must manually create the same folder C:\ConnProfile on the iHub machine and copy the MySQL.dat file there.

To find more information about how to create and manage connection profiles, refer to *Using Actuate BIRT Designer Professional.*

Using a connection configuration file

A connection configuration file is an XML file, such as the one shown in Listing 3-1, in UTF-8 or ASCII encoding. The file specifies the data source connection properties to use when iHub runs a report. Having the data source connection information for a report in an external file makes it convenient to modify. You change the connection information without altering the report design. You specify the location of the file using Configuration Console.

You can use an external connection configuration file to define a data source for the Actuate Caching service and for a data connection definition (.dcd) file, which contains information object connection properties for a data source. You can also use an external connection configuration file for connecting data sources to reports.

You can create an external connection profile to a data source used by a report. Changes to the profile are automatically picked up by the report. The settings in a connection configuration file override any connection configuration properties in the connection profile. The sample connection configuration file in Listing 3-1 externalizes the file path to the connection profile, C:\Mypath.

Listing 3-1 BIRT connection configuration file example

```
<oda-data-source
   extensionID="org.eclipse.birt.report.data.oda.jdbc" name="JDBC
   Data Source - SQL Server" id="783">
   <property name="odaDriverClass">com.actuate.jdbc.sqlserver.
      SQLServerDriver
   </property>
   <property name="odaURL">jdbc:actuate:sqlserver://DBSRV1-W2K
   </property>
</oda-data-source>
<ConnectOptions Type=".eclipse.birt.report.data.oda.jdbc_ JDBC
   Data Source - SQL Server ">
   <Property PropName="OdaConnProfileStorePath">C:\Mypath
   </Property>
</ConnectOptions>
```

In a BIRT report, the configuration key used to specify a data source is the unique ID of the ODA data source extension and data source name defined in the BIRT report design or library. You must concatenate the string as follows:

```
extensionID + "_" + data source name
```

For example, the key is org.eclipse.birt.report.data.oda.jdbc_SQL Server.

Changing a configuration file

The Factory process reads the configuration file for the configuration key values when the process starts. After changing a configuration file, you must restart Factory processes for changes to take effect. Only Factory processes that start after changes in the configuration file use the new information. To ensure that report executable files use updated configuration file information, confirm that no reports are active and stop Factory processes that are running before you change the configuration file. After you change the file, iHub starts a Factory process for the next report request.

Specifying the location of the connection configuration file

There is no default location for the connection configuration file. To use a connection configuration file, you create the file and then specify its name and location using the ConnConfigFile parameter in Configuration Console.

From Server Templates—Advanced, choose Runtime from the list of property settings. Specify the location of the file using the Configuration file for database connections and search path parameter shown in Figure 3-1.

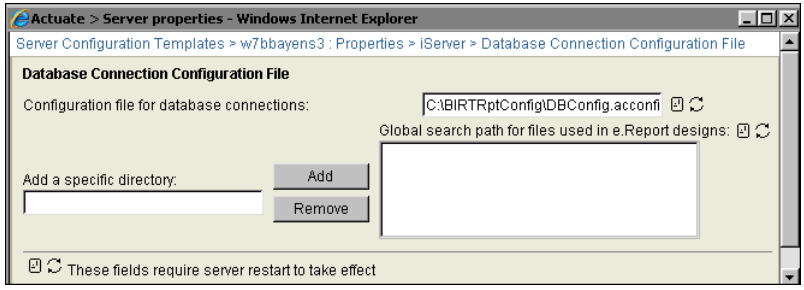

Figure 3-1 Specifying the location of a connection configuration file

On UNIX and Linux, the value of the parameter can be a path and file name only. On Windows, it can be either a path and file name or a URL. For example:

```
\\server1\configs\serverconfig.xml
```

or:

```
http://myserver/configs/testconfig.xml
```

If you do not specify a value for the configuration file parameter, iHub uses the data source connection properties in the report executable file.

Encrypting the connection properties

Actuate BIRT supports the encryption of connection properties in the connection configuration file. The encryption conversion is created in Actuate BIRT Designer Professional, using BIRT's encryption framework. The encryption user interface reads a user-specified configuration file, and writes the encrypted values for a

specified property type to a new output file. The configuration file must have the file name extension .acconfig.

The run-time decryption processing runs in Actuate BIRT Designer Professional, iHub, and Actuate Java Component. You must deploy the encrypted version of a configuration file to the iHub or Actuate Java Component environments, and set up the database configuration for iHub.

For more information about the encryption conversion and the BIRT encryption mechanism, see *Using Actuate BIRT Designer Professional*.

Configuring a cluster to use a connection configuration file

If you have a cluster of iHubs, each node must have access to the connection configuration file. The path can be a local absolute path on each machine and must be specified for each iHub. If you use a single copy of the file for a cluster, put it in a shared location, then specify the path to that shared location for all iHubs in the cluster.

Externalizing the connection profile properties on the iServer

The iServer database connection configuration file is used to externalize data source connection properties for a BIRT report design. As the connection profile store URL is the ODA data source property, OdaConnProfileStorePath, the file path to the connection profile can itself be externalized. When the report is deployed to the iServer and executed, the server reads the connection profile from the file path specified in the iServer's database connection configuration file. The file path specified in the report design is ignored.

Understanding externalization precedence

Data source properties in a report design can be externalized to the connection profile and to the iServer connection configuration file. In addition the Connection Profile Store URL itself can be externalized. The following precedence rules explain how iServer determines the final list of data source properties for report execution.

Data source properties in the iHub connection configuration file override the data source in the connection profile that overrides the data source connection properties in the report. The ascending order of precedence for iServer looks like this:

- Data source properties in the report design
- Data source properties in the connection profile
- Data source properties in the iHub connection configuration file

The following sample connection configuration file externalizes the file path to the connection profile and shows the required structure:

```
<Config>
<Runtime>
<ConnectOptions Type="org.eclipse.birt.report.data.oda.jdbc_SQL
   Server Data Source">
   <Property PropName="OdaConnProfileStorePath">
      C:\SqlServer.profile
   </Property>
</ConnectOptions>
</Runtime>
</Config>
```

The connection profile referenced by the BIRT report design is read when the report is executed in iServer. The path to the connection profile in the design has to be visible to iServer.

Referencing the external connection profile

The path to the external connection profile is stored in the BIRT report design. The ODA data source property, ConnectionProfileStoreURL, holds this value. The path can be a relative or an absolute file path, or a URL. File paths, whether relative or absolute, must be accessible by the Information Console web application when the report is deployed to Information Console. Similarly, this path must be accessible by the iServer when the report is deployed to the iServer. Actuate does not recommend the use of absolute file paths. Typically, the location of the connection profile in all three environments, Actuate BIRT Designer, Information Console, and iServer, resolves to a different path. Absolute paths have the disadvantage that the absolute path used in the Actuate BIRT Designer environment on Windows will not be available when the report is deployed to Information Console or iServer on UNIX. On UNIX, you can use relative paths with the use of soft links, but these links are not available on Windows.

Using a relative path, you deploy the connection profile to iHub, and this resolves the issue with different environments and not accessible absolute paths.

When the absolute file path to the connection profile is different in the design environment compared to the Information Console and iHub deployment environments, there are some options to avoid having to change the report design file before deployment, as described in the following section.

When specifying network paths in BIRT reports always use the Universal Naming Convention (UNC) to describe the path, instead of a mapped drive letter. Windows XP and later do not allow processes running as services to access network resources through mapped network drives. For this reason, a report that uses a mapped drive letter to access a resource runs in Actuate BIRT Designer Professional. The same report fails when running on iHub or Information

Console, because the iHub or Information Console processes cannot resolve the mapping address.

For example, a BIRT report uses a flat file Production.csv as a data source. The flat file is located on a shared network drive on a machine, named ProductionServer. The UNC network path to the file is \\ProductionServer\e$\Data and it is mapped as X:\ in your system. Using the path X:\ to define the data source HOME folder works only in Actuate BIRT Designer. Using the UNC path \\ProductionServer\e$\Data in the data source definition is the correct way to define network paths.

Chapter

4

Configuring fonts in BIRT iHub

This chapter contains the following topics:

- About configuring fonts
- Understanding font configuration file priorities
- Understanding how BIRT engine locates a font
- Understanding the font configuration file structure

About configuring fonts

iHub supports rendering BIRT reports in different formats such as PDF, Microsoft Word, PostScript, and PowerPoint. The processes that do the conversion use the fonts installed on your system to display the report characters.

BIRT uses a flexible mechanism that supports configuring font usage and substitution. This mechanism uses font configuration files for different purposes that control different parts of the rendering process. The configuration files can configure the fonts used in specific operating systems, for rendering to specific formats, in specific locales only, or combinations of these parameters.

The plug-in folder, org.eclipse.birt.report.engine.fonts, contains the font configuration files. Table 4-1 shows the location of this folder in the supported BIRT environments.

Table 4-1 Locations of the font configuration file plug-in folder

Environment	Font configuration file folder location
BIRT Designer Professional	$Actuate<release>\BRDPro\eclipse\plugins
iHub	$Actuate<release>\iHub\Jar\BIRT\platform \plugins

Understanding font configuration file priorities

BIRT reports use five different types of font configuration files. The font configuration file naming convention includes information about the rendering format, the system platform, and the system locale, as shown in the following general format:

```
fontsConfig_<Format>_<Platform>_<Locale>.xml
```

The platform name is defined by the Java System property, os.name. The current Java Network Launch Protocol (JNLP) specification does not list the values for the os attributes. Instead it states that all values are valid as long as they match the values returned by the system property os.name. Values that only match the beginning of os.name are also valid. If you specify Windows and the os.name is Windows 98, for example, the operating system name is accepted as valid.

The following sample Java class code shows how to check the os.name property for the value on your machine:

```
class WhatOS
{
  public static void main( String args[] )
   {
    System.out.println( System.getProperty("os.name") );
   }
}
```

BIRT supports the following types of font configuration files, with increasing priority:

- For all rendering formats
  These files have no format specifier in their names. These configuration files are divided into three sub-types:
  - The default configuration file:
    `fontsConfig.xml`
  - Configuration files for a specific platform, for example:
    `fontsConfig_Windows_XP.xml`
  - Configuration files for a specific platform and locale, for example:
    `fontsConfig_Windows_XP_zh.xml`
    `fontsConfig_Windows_XP_zh_CN.xml`
- For certain formats only
  These files include the format specifier in their names. These configuration files are divided into two sub-types:
  - The default configuration file for a format, for example:
    `fontsConfig_pdf.xml`
  - Configuration files for a format for a specific platform:
    `fontsConfig_pdf_Windows_XP.xml`

Understanding how BIRT engine locates a font

The PDF layout engine renders the PDF, PostScript, and PowerPoint formats. The engine tries to locate and use the font specified at design-time. The PDF layout engine searches for the font files first in the fonts folder of the plug-in, org.eclipse.birt.report.engine.fonts. If the specified font is not in this folder, the BIRT engine searches for the font in the system-defined font folder. You can change the default load order by using the settings in the font configuration file.

When the required font for a character is not available in the search path or is incorrectly installed, the engine uses the fonts defined in the UNICODE block for

that character. If the UNICODE definition also fails, the engine replaces the character with a question mark (?) to denote a missing character. The font used for the ? character is the default font, Times-Roman.

The engine maps the generic family fonts to a PDF embedded Type1 font, as shown in the following list:

- cursive font styles to Times-Roman
- fantasy font styles to Times-Roman
- monospace font styles to Courier
- sans-serif font styles to Helvetica
- serif font styles to Times-Roman

Understanding the font configuration file structure

The font configuration file, fontsConfig.xml, consists of three major sections, <font-aliases>, <composite-font>, and <font-paths> sections.

<font-aliases> section

In the <font-aliases> section, you can:

- Define a mapping from a generic family to a font family. For example the following code defines a mapping from the generic type "serif" to a Type1 font family Times-Roman:

  ```
  <mapping name="serif" font-family="Times-Roman"/>
  ```

- Define a mapping from a font family to another font family. This definition is useful if you want to use a font for PDF rendering which differs from the font used at design time. For example, the following code shows how to replace simsun with Arial Unicode MS:

  ```
  <mapping name="simsun" font-family="Arial Unicode MS"/>
  ```

Earlier versions of BIRT Designer Professional use the XML element <font-mapping> instead of <font-aliases>. In the current release, a <font-mapping> element works in the same way as the <font-aliases> element. When a font configuration file uses both <font-mapping> and <font-aliases>, the engine merges the different mappings from the two sections. If the same entries exist in both sections, the settings in <font-aliases> override those in <font-mapping>.

<composite-font> section

The <composite-font> section is used to define a composite font, which is a font consisting of many physical fonts used for different characters. For example, to define a new font for currency symbols, you change font-family in the following <block> entry to the Times Roman font-family:

```
<composite-font>
…
<block name="Currency Symbols" range-start="20a0"
   range-end="20cf" index="58" font-family="Times Roman" />
…
</composite-font>
```

The composite fonts are defined by <block> entries. Each <block> entry defines a mapping from a UNICODE range to a font family name, which means the font family is applied for the UNICODE characters in that range. You cannot change the block name or range or index as it is defined by the UNICODE standard. The only item you can change in the block element is the font-family name.

You can find information about all the possible blocks at http://www.unicode.org/charts/index.html.

In cases when the Times Roman font does not support all the currency symbols, you can define the substitution character by character using the <character> tag, as shown in the following example:

```
<composite-font>
    …
      <character value="?" font-family="Angsana New"/>
      <character value="\u0068" font-family="Times Roman"/>
    …
</composite-font>
```

Note that characters are represented by the attribute, value, which can be presented two ways, the character itself or its UNICODE code.

You can find information about all the currency symbols from http://www.unicode.org/charts/symbols.html.

A composite font named all-fonts is applied as a default font. When a character is not defined in the desired font, the font defined in all-fonts is used.

<font-paths> section

If the section <font-paths> is set in fontsConfig.xml, the engine ignores the system-defined font folder, and loads the font files specified in the section, <font-paths>. You can add a single font path or multiple paths, ranging from one font path to a whole font folder, as shown in the following example:

```
<path path="c:/windows/fonts"/>
<path path="/usr/X11R6/lib/X11/fonts/TTF/arial.ttf"/>
```

If this section is set, the PDF layout engine only loads the font files in these paths and ignores the system-defined font folder. If you want to use the system font folder as well, you must include it in this section.

On some systems, the PDF layout engine does not recognize the system-defined font folder. If you encounter this issue, add the font path to the <font-paths> section.

Chapter

5

Working with BIRT encryption

This chapter contains the following topics:

- About BIRT encryption
- About the BIRT default encryption plug-in
- Deploying encryption plug-ins to iHub

About BIRT encryption

BIRT provides an extension framework to support users registering their own encryption strategy with BIRT. The model implements the JCE (Java™ Cryptography Extension). The Java encryption extension framework provides multiple popular encryption algorithms, so the user can just specify the algorithm and key to have a high security level encryption. The default encryption extension plug-in supports customizing the encryption implementation by copying the BIRT default plug-in, and giving it different key and algorithm settings.

JCE provides a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. Support for encryption includes symmetric, asymmetric, block, and stream ciphers. The software also supports secure streams and sealed objects.

A conventional encryption scheme has the following five major parts:

- Plaintext, the text to which an algorithm is applied.
- Encryption algorithm, the mathematical operations to conduct substitutions on and transformations to the plaintext. A block cipher is an algorithm that operates on plaintext in groups of bits, called blocks.
- Secret key, the input for the algorithm that dictates the encrypted outcome.
- Ciphertext, the encrypted or scrambled content produced by applying the algorithm to the plaintext using the secret key.
- Decryption algorithm, the encryption algorithm in reverse, using the ciphertext and the secret key to derive the plaintext content.

About the BIRT default encryption plug-in

BIRT's default encryption algorithm is implemented as a plug-in named:

`com.actuate.birt.model.defaultsecurity_<Release>`

Table 5-1 shows the location of this plug-in folder in the supported BIRT environments.

Table 5-1 Locations of the default encryption plug-in folder

Environment	Default encryption plug-in folder location
BIRT Designer Professional	$Actuate<Release>\BRDPro\eclipse\plugins
iServer	$Actuate<Release>\iHub\Jar\BIRT\platform \plugins

More information about how to implement encryption in BIRT reports is available in *Using Actuate BIRT Designer Professional*.

The BIRT default encryption plug-in consists of the following main modules:

- acdefaultsecurity.jar
- encryption.properties
- META-INF/MANIFEST.MF
- plugin.xml

About acdefaultsecurity.jar

This JAR file contains the encryption classes. The default encryption plug-in also provides key generator classes that can be used to create different encryption keys.

About encryption.properties

This file specifies the encryption settings. BIRT loads the encryption type, encryption algorithm, and encryption keys from the encryption.properties file to do the encryption. The file contains pre-generated default keys for each of the supported algorithms.

About META-INF/MANIFEST.MF

META-INF/MANIFEST.MF is a text file that is included inside a JAR to specify metadata about the file. Java's default ClassLoader reads the attributes defined in MANIFEST.MF and appends the specified dependencies to its internal classpath. The encryption plug-in ID is the value of the Bundle-SymbolicName property in the manifest file for the encryption plug-in. You need to change this property when you deploy multiple instances of the default encryption plug-in, as described later in this chapter.

About plugin.xml

plugin.xml is the plug-in descriptor file. This file describes the plug-in to the Eclipse platform. The platform reads this file and uses the information to populate and update, as necessary, the registry of information that configures the whole platform.

Deploying encryption plug-ins to iHub

If you deploy report designs to iServer, you also need to deploy the encryption plug-in to iHub. iHub loads all encryption plug-ins at start up. During report

execution the server reads the encryptionID property from the report design file and uses the corresponding encryption plug-in to decrypt the encrypted property. Every time you create reports using an encryption plug-in, make sure you deploy the plug-in to iServer, otherwise report execution on the server fails.

How to deploy an encryption plug-in instance to iServer

1. Copy:

   ```
   $ACTUATE_HOME\BRDPro\eclipse\plugins
      \com.actuate.birt.model.defaultsecurity_<Release>_rsa
   ```

   to:

   ```
   \Actuate\iHub\Jar\BIRT\platform\plugins
   ```

2. Publish the report design to iServer.
3. Restart iServer to load the encryption plug-in.
4. Log in to iServer using Information Console and run the report. iServer uses the encryption plug-in to decrypt the password.

Chapter

6

Using custom emitters

This chapter contains the following topics:

- About custom emitters
- Deploying custom emitters to iHub
- Rendering in custom formats
- Configuring the default export options for a BIRT report

About custom emitters

In Actuate BIRT Designer Professional or Interactive Viewer you can choose to render BIRT reports in several different formats, as shown in Figure 6-1.

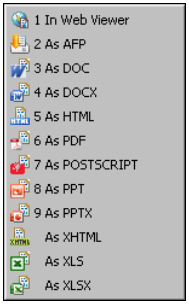

Figure 6-1 Rendering formats

Actuate provides report rendering for the following file formats:

- AFP - Advanced Function Printing document format
- DOC - Microsoft Word document
- DOCX - Microsoft Word document, introduced in Windows 7
- HTML - HyperText Markup Language document, a standard format used for creating and publishing documents on the World Wide Web
- PDF - Created by Adobe, a portable file format intended to facilitate document exchange
- POSTSCRIPT - A page description language document for medium- to high-resolution printing devices
- PPT - Microsoft PowerPoint document
- PPTX - Microsoft PowerPoint document for Windows 7
- XHTML - Extensible HyperText Markup Language document, the next generation of HTML, compliant with XML standards
- XLS/XLSX - MS-Excel document

If you need to export your document to a format not supported by Actuate, for example CSV or XML, you must develop a custom emitter. Actuate supports using custom emitters to export reports to custom formats. After a system administrator places custom emitters in the designated folder in iHub, users are able to use them as output formats when scheduling BIRT report jobs or exporting BIRT reports. Custom emitters are also supported as e-mail attachment formats.

The iHub uses the custom emitter format type as a file extension for the output file when doing the conversion. When you develop custom emitters always use the same name for a format type and an output file extension type. Management Console and Actuate Information Console display the options of each emitter for the user to choose when exporting a report.

Integrating and Extending BIRT, published by Addison-Wesley, provides detailed information about how to develop custom emitters in BIRT.

Deploying custom emitters to iHub

The custom emitters in BIRT are implemented as plug-ins and packaged as JAR files. To make them available to the Actuate products that support them, copy the emitters to the following folder:

```
Actuate<release>/MyClasses/eclipse/plugins
```

To deploy custom emitter to iHub copy the plug-ins to:

```
Actuate<release>/iHub/MyClasses/eclipse/plugins
```

The MyClasses folder appears at different levels on different platforms but it is always available in the product's installation folder.

Plug-ins depend on other plug-ins to function properly. It is a good practice to verify all required plug-ins are installed in the system. To get the list of all required plug-ins open MANIFEST.MF file of your custom plug-in, as shown in Listing 6-1. Depending on the way the plug-ins are developed, Import-Package or Require-Bundle entries declare plug-in dependencies on a package or bundle level.

Listing 6-1 MANIFEST.MF

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Mycsv
Bundle-SymbolicName: org.eclipse.birt.report.engine.emitter.mycsv;
singleton:=true
Bundle-Version: 2.6.2
Bundle-Activator:
   org.eclipse.birt.report.engine.emitter.mycsv.Activator
Require-Bundle: org.eclipse.birt.chart.engine;
   bundle-version="2.6.2",org.eclipse.birt.report.engine
Bundle-ActivationPolicy: lazy
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
```

Every time you deploy a custom emitter you need to restart the product. This ensures the emitter JAR is added to the classpath and the product can discover the new rendering format.

The following tools support custom emitters:

- Actuate BIRT Designer
- BIRT Interactive Viewer
- Information Console
- Management Console

Rendering in custom formats

After deploying the custom emitter you can see the new rendering formats displayed along with built-in emitters in the following places:

- Preview report in Web Viewer in Actuate BIRT Designer
- Output page of schedule job in Management Console and Information Console
- Attachment Notification page of schedule job in Management Console and Information Console
- Export Content in Actuate BIRT Viewer and Actuate BIRT Interactive Viewer

The following examples show the deployment and usage of a custom CSV emitter. The CSV emitter renders a report as a comma separated values file. The JAR file name is org.eclipse.birt.report.engine.emitter.csv.jar. The custom format type is MyCSV.

To test the emitter functionality with Management Console or Information Console, you schedule any BIRT report design or report document from the examples in the Public folder. The examples that follow use the report from the sample Encyclopedia volume for an iHub:

```
Public/BIRT and BIRT Studio Examples/CustomerList.rptdesign
```

How to deploy a custom emitter

The assumption in this example is that the Actuate products are installed in Actuate<Release> folder on Windows.

1. Copy org.eclipse.birt.report.engine.emitter.csv.jar to:

   ```
   Actuate<Release>\iHub\MyClasses\eclipse\plugins
   ```

2. Register the emitter with iHub.
   1. Open the following file:

      ```
      Actuate<Release>\iHub\etc\jfctsrvrconfig.xml
      ```

      JREM uses this configuration file at startup to load the registered emitters.

2 Navigate to the end of the file to find the following entry:

```
<node name="BIRTReportRenderOption">
```

The entry contains a list of emitter descriptions separated by a semicolon. The emitter description must have the format type and the emitter id separated by a colon. For example, the PDF emitter is described as:

```
pdf:org.eclipse.birt.report.engine.emitter.pdf;
```

3 Add your emitter description to the beginning of the <entry name="RenderFormatEmitterIdMapping"> tag:

```
MyCSV:org.eclipse.birt.report.engine.emitter.mycsv;
```

The whole tag would look like this:

```
<node name="BIRTReportRenderOption">
<!-- The value is "render_format:emitter_ID" separated by ";",
   for example, pdf:org.eclipse.birt.report.engine.emitter.pdf;
   xml:org.eclipse.birt.report.engine.emitter.xml -->
<entry name="RenderFormatEmitterIdMapping">
   MyCSV:org.eclipse.birt.report.engine.emitter.csv;
   html:org.eclipse.birt.report.engine.emitter.html;
   xhtml:com.actuate.birt.report.engine.emitter.xhtml;
   pdf:org.eclipse.birt.report.engine.emitter.pdf;
   postscript:org.eclipse.birt.report.engine.emitter.postscript;
   xls:com.actuate.birt.report.engine.emitter.xls;
   ppt:org.eclipse.birt.report.engine.emitter.ppt;
   pptx:com.actuate.birt.report.engine.emitter.pptx;
   doc:org.eclipse.birt.report.engine.emitter.word;
   docx:com.actuate.birt.report.engine.emitter.docx
</entry>
</node>
```

3 Restart the iHub to make it load the new plug-in in its classpath:

- Restart Actuate iHub <Release> from Start➤Settings➤Control Panel➤Administrative Tools➤Services, as shown in Figure 6-2.
- If you use a separately deployed Information Console, you must also restart Apache Tomcat for Actuate Information Console <Release>.

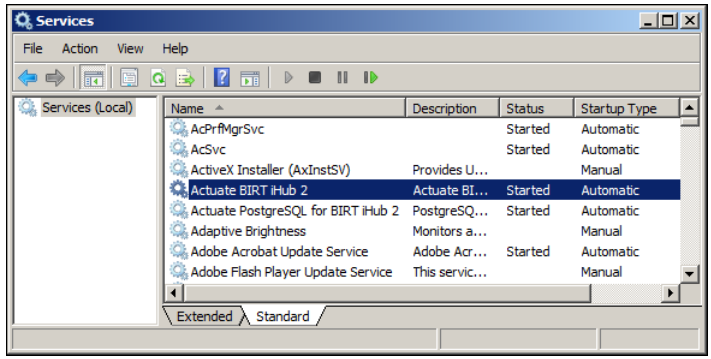

Figure 6-2 Services

The following procedures show how to export a BIRT report to a custom format in different products. The procedures use an example format, MyCSV.

How to deploy and use a custom emitter in Actuate BIRT Designer

This example assumes that iHub is installed in the Actuate<Release> folder on Windows.

1. Copy the emitter to:

   `Actuate<Release>\MyClasses\eclipse\plugins`

2. Reopen the designer.
3. Preview the report in Web Viewer.

   The new MYCSV format appears in the list of formats, as shown in Figure 6-3.

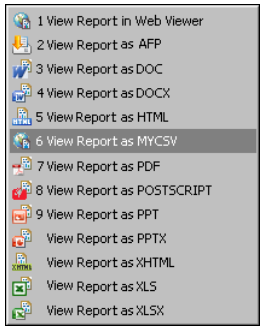

Figure 6-3 List of available formats in Web Viewer

How to export a BIRT report in Management Console

1. Open Management Console.
2. Navigate to the Public/BIRT and BIRT Studio Examples folder.

3 Click the blue arrow next to CustomerList.rptdesign and select the Schedule option from the menu.

4 On the Schedule page select the Output tab.

The new MYCSV format appears in the list of the available formats, as shown in Figure 6-4.

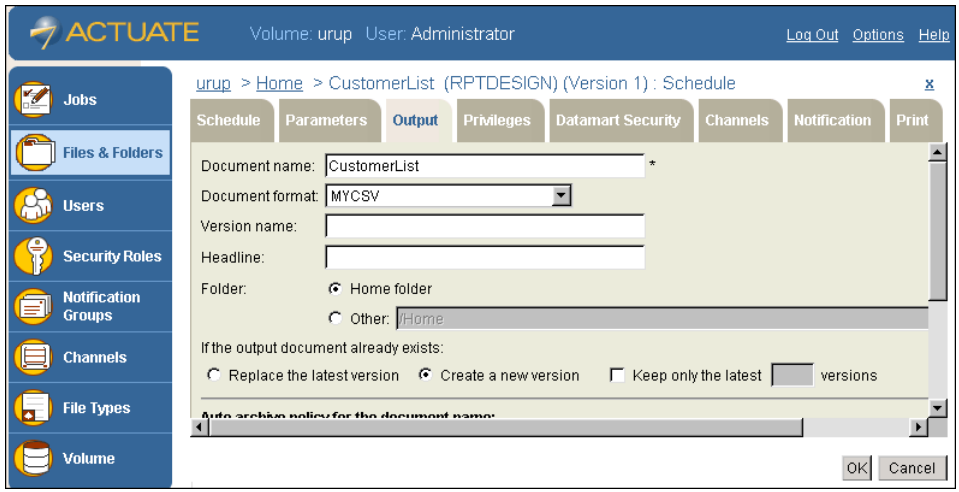

Figure 6-4 Output format in Management Console

5 Choose the Notification tab in the Schedule Job page. Select MYCSV format from the Format for the attached report's drop-down list, as shown in Figure 6-5.

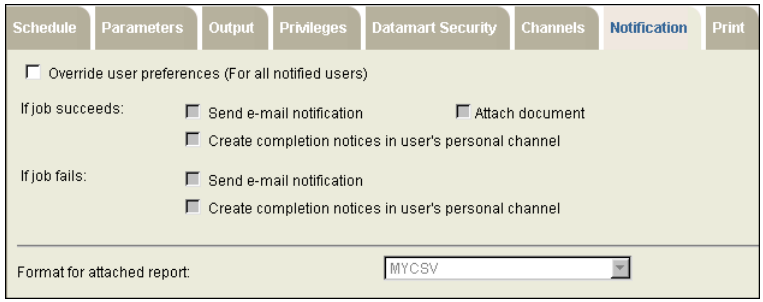

Figure 6-5 Notification tab in the Schedule Job page

6 Choose OK. The generated report is saved as CustomerList.MYCSV in the Encyclopedia volume. The report is also attached to the e-mail notification.

How to export a BIRT report from Information Console

Schedule a BIRT report to run by choosing Save As on the schedule page. The new MYCSV format appears in the Document Format list. You can also select to attach the output report to an e-mail notification, as shown in Figure 6-6.

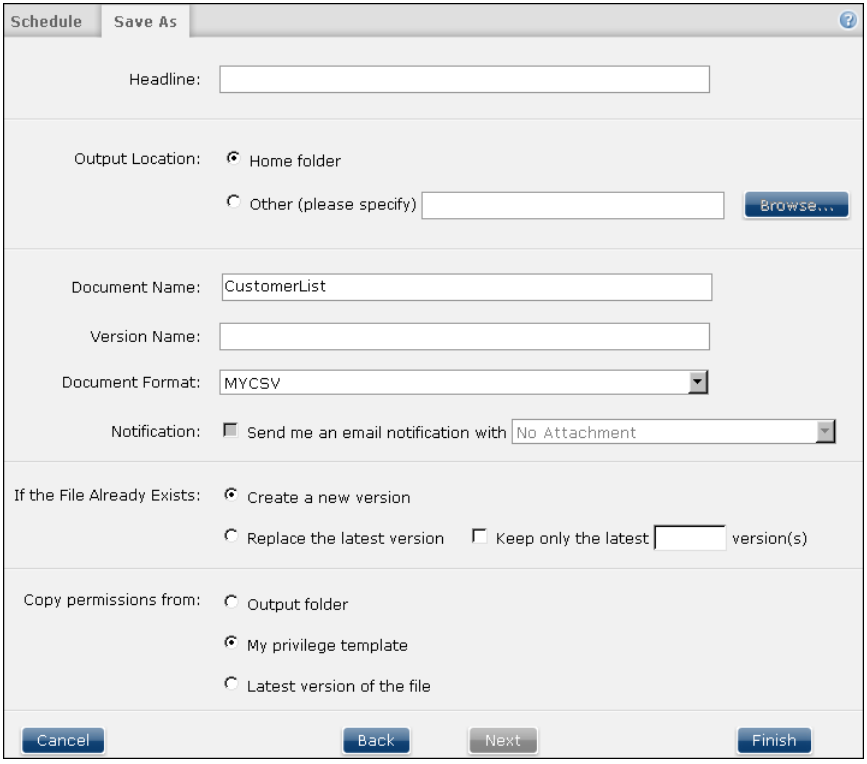

Figure 6-6 Save As tab in the Schedule Jobs page in Information Console

How to export a BIRT report from Actuate BIRT Viewer or Actuate BIRT Interactive Viewer

1. Open a BIRT report in Actuate BIRT Viewer or Interactive Viewer.
2. Select Export Content from the viewer menu. The MyCSV format appears in Export Formats, as shown in Figure 6-7.

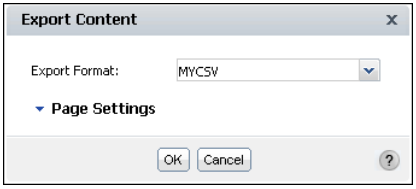

Figure 6-7 Export Content in Actuate BIRT Viewers

3 Choose OK. A File Download window appears, as shown in Figure 6-8. You can choose to open or save the file. The suggested file name is CustomerList.mycsv.

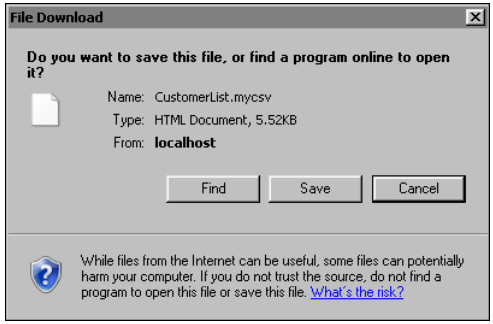

Figure 6-8 File Download

Configuring the default export options for a BIRT report

You can export a BIRT report to various formats from the web viewer. These formats include docx, pptx, xls, xlsx, pdf, ps, doc, and ppt. You can configure the default export options by creating a RenderDefaults.cfg file that contains name-value pairs for the appropriate options. You must create a separate RenderDefaults.cfg file for each format. For example, when you export a BIRT report to XLSX, RenderDefaults.cfg can set Enable pivot table to false by default. Place RenderDefaults.cfg in the following locations:

- On iHub, place RenderDefaults.cfg in:

```
<iHUB-HOME>\Jar\BIRT\platform\plugins\
   com.actuate.birt.report.engine.emitter.config
   .<EMITTER_TYPE>_<RELEASE_NUMBER>.jar
```

  When you create or modify RenderDefaults.cfg, you must restart iHub.

- On the desktop, place RenderDefaults.cfg in:

```
<BDPRO_HOME>\eclipse\plugins\
   com.actuate.birt.report.engine.emitter.config
   .<EMITTER_TYPE>_<RELEASE_NUMBER>.jar
```

The configuration file format should follow these rules:

- All settings should be entered in key = value format.
- Key names are case sensitive.
- String and Boolean values are not case sensitive

- The values true and false are acceptable for Boolean properties.
- Any other value, different than true and false is interpreted as false.

Listing 6-2 shows an example of RenderDefaults.cfg of a print stream emitter.

Listing 6-2 RenderDefaults.cfg for print stream emitter

```
# This file contains the default values to use when emitting print
# stream content.
# All settings are entered in 'key = value' format.
# Key names are case sensitive, but string and boolean values are
# not.
# The values 'true' and 'false' are acceptable for booleans.
# Any other value specified will be interpreted as false.
# Omitting any setting from this file causes the emitter to fall
# back to internal defaults.

# The plex mode for the print job.
# Valid values are Simplex, Duplex, and Tumble.

# Default: Simplex
xifRenderOption.plexMode = Simplex

# The DPI to use for rendered pages. Valid values are 240,300,600,
# and 1440.
# Default: 240
afpRenderOption.pageDPI = 240

# Option to allow black and white images.
# Default: True
afpRenderOption.allowBlackAndWhiteImg = true

# Option to allow single color images.
# Default: True
afpRenderOption.allowSingleColorImg = true

# Option to allow grayscale images.
# Default: True
afpRenderOption.allowGrayscaleImg = true

# Option to allow full color RBG images.
# Default: True
afpRenderOption.allowFullColorRGBImg = true

# Option to allow full color CMYK images.
# Default: True
afpRenderOption.allowFullColorCMYKImg = true
```

For information about creating a RenderDefaults.cfg file, see *Working with Actuate BIRT Viewers.*

Part Three

Deploying information objects

Chapter

7

Deploying information objects

This chapter contains the following topic:

- Deploying information objects

Deploying information objects

Information object files must reside in a project's Shared Resources folder. By default, a project's Shared Resources folder is the project folder. If the Shared Resources folder is not the project folder, you must copy information object files to the Shared Resources folder before publishing. To check the location of the Shared Resources folder, choose Window➤Preferences➤Report Design➤Resource.

When you publish information object files to an Encyclopedia volume, the files are published to the IO Designs folder in the Encyclopedia volume's resource folder. The resource folder's default location is /Resources. You must have write privilege on the resource folder.

How to copy information object files to the shared resources folder

1. In Navigator, select the appropriate project.
2. Choose File➤Copy to Resources➤Copy Information Objects to Shared Resources Folder.
3. In Share Information Objects, shown in Figure 7-1, select the appropriate files and folders. Choose Finish.

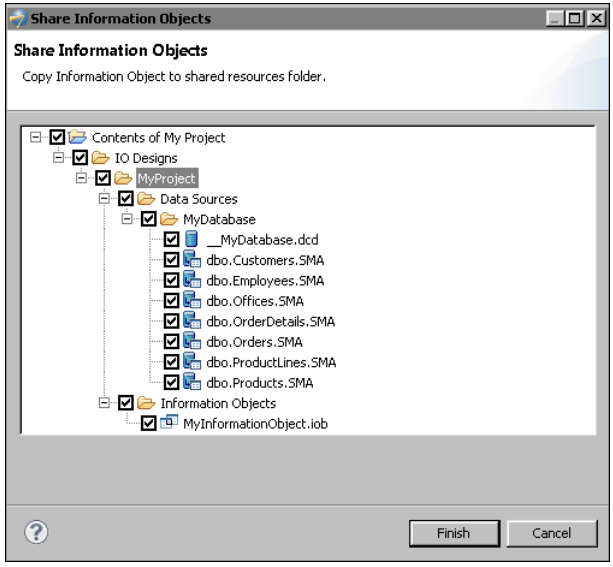

Figure 7-1 Copying information object files and folders to Shared Resources

How to publish information object files as resources

1. Choose File➤Publish➤Publish to iHub.
2. In Publish to iHub, in iHub profile, choose an iHub profile from the drop-down list.
3. In Project, select the appropriate project from the drop-down list.
4. Select Publish Resources.
5. Select the appropriate files and folders, as shown in Figure 7-2.

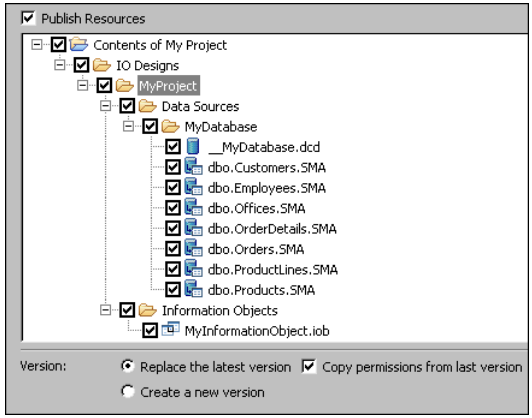

Figure 7-2 Publishing information object files as resources

6. In Version:
   1. Select Replace the latest version to replace the latest version of each file, or Create a new version to create a new version of each file.
   2. To copy permissions from the last version of each file, select Copy permissions from last version. If you do not select Copy permissions from last version, you must set the permissions for each file using Management Console.
7. Choose Publish Files.

   A confirmation dialog, shown in Figure 7-3, appears.

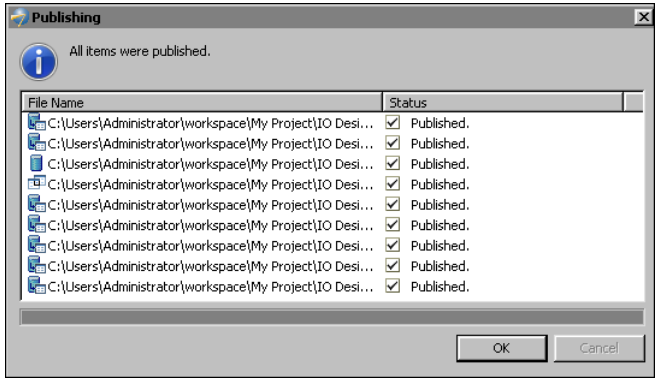

Figure 7-3 Publishing confirmation dialog

8 In Publishing, choose OK.

9 In Publish to iHub, choose Close.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q