Oracle® Fusion Middleware

Scheduling Jobs Guide for Oracle Business Intelligence Enterprise Edition

12*c* (12.2.1)

E57388-01

October 2015

Explains how to manage Oracle Business Intelligence Enterprise Edition Scheduler and Job Manager, including how to configure Scheduler, how to use Job Manager and its menus, and how to program scripts and Java Jobs. Includes information about the SA System subject area.

ORACLE

Oracle Fusion Middleware Scheduling Jobs Guide for Oracle Business Intelligence Enterprise Edition, 12*c* (12.2.1)

E57388-01

Primary Author: Guriqpal Gill

Contributing Author: Nick Fry

Contributor: Oracle Business Intelligence development, product management, and quality assurance teams

Contents

4 Programming BI Scheduler VBScript and JScript Jobs

5 Programming BI Scheduler Java Jobs

6 Oracle BI Scheduler Job Manager Menus

Preface

The Oracle Business Intelligence Foundation Suite is a complete, open, and integrated solution for all enterprise business intelligence needs, including reporting, ad hoc queries, OLAP, dashboards, scorecards, and what-if analysis. The Oracle Business Intelligence Foundation Suite includes Oracle Business Intelligence Enterprise Edition.

Oracle Business Intelligence Enterprise Edition (Oracle BI EE) is a comprehensive set of enterprise business intelligence tools and infrastructure, including a scalable and efficient query and analysis server, an ad-hoc query and analysis tool, interactive dashboards, proactive intelligence and alerts, and an enterprise reporting engine.

The components of Oracle BI EE share a common service-oriented architecture, data access services, analytic and calculation infrastructure, metadata management services, semantic business model, security model and user preferences, and administration tools. Oracle BI EE provides scalability and performance with data-source specific optimized request generation, optimized data access, advanced calculation, intelligent caching services, and clustering.

This guide explains how to automate agents and jobs using the Oracle BI Scheduler and Job Manager. For example, you might want to automatically deliver analyses, dashboards, briefing books, or alerts to BI users once per week, or automatically run a script once per day to load Oracle BI Server statistics into a database for analysis.

Audience

This guide is intended for administrators who are responsible for managing Job scheduling in Oracle Business Intelligence.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents and Other Resources

See the Oracle Business Intelligence documentation library for a list of related Oracle Business Intelligence documents.

In addition:

- Go to the Oracle Learning Library for Oracle Business Intelligence-related online training resources.
- Go to the Product Information Center support note (Article ID 1267009.1) on My Oracle Support at https://support.oracle.com.

System Requirements and Certification

Refer to the system requirements and certification documentation for information about hardware and software requirements, platforms, databases, and other information. Both of these documents are available on Oracle Technology Network (OTN).

The system requirements document covers information such as hardware and software requirements, minimum disk space and memory requirements, and required system libraries, packages, or patches:

http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-requirements-100147.html

The certification document covers supported installation types, platforms, operating systems, databases, JDKs, and third-party products:

http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

New Features for Scheduling Jobs for Oracle Business Intelligence

There are no new job scheduling features in Oracle BI EE 12c (12.2.1).

1 Introducing Oracle BI Scheduler

You use Oracle BI Scheduler and Job Manager to automate agents and jobs. For example, you might want to automatically deliver analyses, dashboards, briefing books, or alerts to BI users once per week, or automatically run a script once per day to load Oracle BI Server statistics into a database for analysis.

This chapter contains the following sections:

1.1 What is Oracle BI Scheduler?

Oracle BI Scheduler is a server that manages and schedules jobs. Oracle BI Scheduler supports two kinds of jobs:

- Scripted jobs.

  Scripted jobs are configured and submitted using Job Manager. For example, a scripted job could periodically load the Oracle BI Server usage statistics into a back-end database. In this example Oracle BI Scheduler communicates with the BI Server. However, scripted jobs might not access the BI Server, for example by saving the output of an agent to a shared drive. Scripted jobs can also be configured through agents and actions.

  Oracle BI Scheduler supports jobs that are written in the Java programming language or in the VBScript and JScript scripting languages.

  > **Note:** Scripting for agents and scripts that are defined by the Oracle BI Scheduler Job Manager are supported only under Windows platforms. The Java interfaces support all platforms.

- Agents.

  Agents are configured and submitted for execution using Oracle BI Delivers. Agents deliver content to end users. Content can be analyses, dashboards, briefing books, or alerts. After delivering content, agents can also execute actions. Actions include Java actions, URL actions, Web service actions, and server script actions. Agents can also run other agents, creating chains of agents. Oracle BI Scheduler communicates with Oracle BI Presentation Services for unscripted jobs.

Note: There are thus two different forms of custom Java that can be executed: Java scripted jobs run in their own right and Java actions run as part of an agent.

1.2 About Oracle BI Scheduler Server Components

Oracle BI Scheduler consists of the following components:

- Oracle BI Scheduler Job Manager
- Oracle BI Scheduler Service process: <Oracle_Home>/bin/obish<*n*>
- Mail tab under the Configuration tab in Fusion Middleware Control
- Command line job invocation tool:
  - Windows operating systems: `saschinvoke.exe`
  - UNIX operating systems: `saschinvoke`

1.3 Topics of Interest in Other Guides

Some topics that might be of interest to security administrators are covered in other guides. Table 1–1 lists these topics and indicates where to go for more information.

Table 1–1 Topics Covered in Other Guides

Topic	Where to Go for More Information
Configuration settings affecting agents	*Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*
Configuring data sources	*Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition*
Information about security, including configuring SSO and SSL	*Oracle Fusion Middleware Security Guide for Oracle Business Intelligence Enterprise Edition*
Installing and upgrading	*Oracle Fusion Middleware Installation Guide for Oracle Business Intelligence* *Oracle Fusion Middleware Upgrade Guide for Oracle Business Intelligence Enterprise Edition*

2

Configuration Tasks for Oracle BI Scheduler

This chapter explains that depending on your specific deployment, you must perform the following Oracle BI Scheduler configuration tasks:

- Section 2.1, "Configuring a Database for the Oracle BI Scheduler"
- Section 2.2, "Configuring Oracle BI Scheduler Instances in a Clustered Environment"

Keep the following points in mind:

- If you are not using Oracle BI Scheduler, then you do not need the information in this chapter.
- Configuration that is required for running agents (as opposed to running jobs in general) is described in "Configuring and managing Agents" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.
- If you are migrating an Oracle Business Intelligence environment to a new system, then ensure that you also migrate the Oracle Business Intelligence Server repository file and the Oracle BI Scheduler tables. For information, see "Moving Between Environments" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*. The Oracle BI Scheduler tables are required for agents.

2.1 Configuring a Database for the Oracle BI Scheduler

You can use Fusion Middleware Control to configure common settings that are used by agents with Oracle BI Scheduler.

Note: You can automatically populate or repopulate credential information from the Weblogic data source. If connectivity details change (for example, password expiry or database failure), then you update the Weblogic BI platform data source and run DH/bitools/bin/sync_midtier_db.sh.

The file DH/config/fmwconfig/biconfig/OBISCH/schedulerconfig.xml contains the DSN name, and the credentials are stored in oracle.bi.system map datasource.biplatform key (visible in EM).

The following procedure describes how to configure the back-end database and tables:

- Section 2.1.1, "Configuring a System DSN entry for SQL Server Databases."

2.1.1 Configuring a System DSN entry for SQL Server Databases

For SQL Server databases, the Data Source Name (DSN) that is used in the Fusion Middleware Control Scheduler configuration must match an existing ODBC DSN for the SQL Server S_NQ_SCHED database.

If you do not have a System DSN entry, then create a new one as described in the following procedure.

To configure the SQL Server database DSN entry:

1. From the Windows Start menu, select **Settings**, then **Control Panel**, then **Administrative Tools**, then **Data Sources (ODBC)**.
2. Start the ODBC Data Source Administrator.
3. Select the System DSN tab, and click **Add**.
4. Select the driver SQL Server, and click **Finish**.
5. In the Create a New Data Source to SQL Server wizard, do the following:
   a. Enter a name and description for the data source.
   b. Select the SQL Server from the Server list, and click **Next**.
   c. For server verification of the login ID authenticity, select the appropriate authentication for the S_NQ_SCHED SQL Server database schema. Click **Next**.
6. Select the **Change the default database to** field and select the S_NQ_SCHED database from the list. Click **Next**.
7. Update any language or log file settings if appropriate, and click **Finish**.
8. To verify the connection settings, click the **Test Data Source** button, and click **OK**.
9. Click **OK** to exit ODBC Data Source Administrator.

2.2 Configuring Oracle BI Scheduler Instances in a Clustered Environment

In a clustered environment you can have zero, one, or two scheduler instances. Configure zero if you do not require support for scheduled jobs or agents. Configure one if you do not require high availability.

A two-node cluster environment always consists of two instances of the scheduler which are automatically configured as primary and secondary nodes.

3

Using Oracle BI Scheduler Job Manager

This chapter describes how to manage agents and jobs using Job Manager, and contains the following topics:

3.1 Opening Job Manager

Job Manager is a Windows tool that you use to manage agents and jobs. For example, you can start and stop the Oracle BI Scheduler, add and manage jobs, and manage job instances.

To open Job Manager

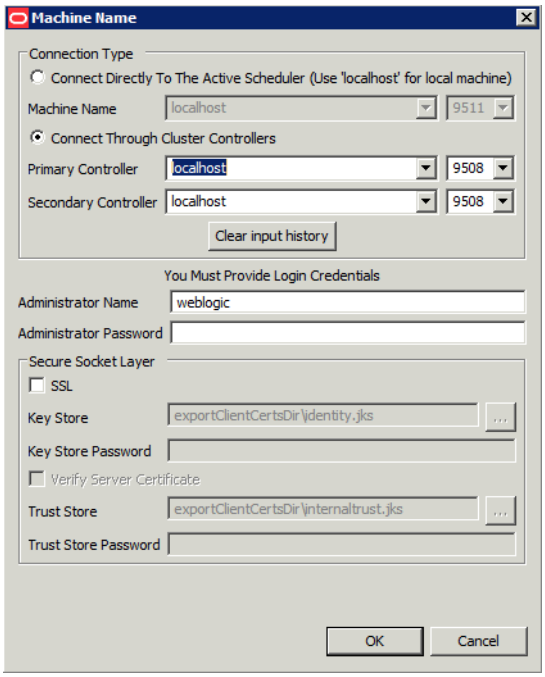
1. In Windows, select Start, then Programs, then Oracle Business Intelligence Enterprise Edition Plus Client, then Job Manager.

   Note: 'Oracle Business Intelligence Enterprise Edition Plus Client' is the default program folder name of the Oracle BI EE Client installation.

2. In Job Manager, from the **File** menu select **Open Scheduler Connection** to display the Machine Name dialog.

3. Select the **Connect Through Cluster Controllers** option (the Oracle-recommended connection method), and select a **Primary Controller** and associated port number.

   Note: The Secondary Controller connection is optional.



4. Specify appropriate login credentials, then click OK.

3.2 Adding Oracle BI Scheduler Jobs in Job Manager

Job Manager enables you to add new scripted jobs. Agents can only be created in Oracle BI Presentation Services.

Add an Oracle BI Scheduler job in Job Manager using the following procedure.

To add an Oracle BI Scheduler job in Job Manager:

1. In Job Manager, from the **Jobs** menu, select **Add New Job**.

   To find out how to open Job Manager, see Section 3.1, "Opening Job Manager".

2. Enter the appropriate information in the dialog.

   See the following sections for field descriptions:

   - Section 6.3, "General Oracle BI Scheduler Job Properties"
   - Section 6.4, "Job Action Properties Available in Job Manager"
   - Section 6.5, "Job Triggers in Job Manager"

3.3 Modifying Oracle BI Scheduler Jobs in Job Manager

You can modify an Oracle BI Scheduler job in Job Manager using the following procedure.

Note: If, while adding or modifying a job in Job Manager, you enter a script in the Script field, then the Oracle BI Scheduler creates a file with an SCS extension in the following directory:

\orainst\bifoundation\<OracleBISchedulerComponent>\coreapplication_obisch1\scripts\scheduler

Where orainst is the install location for Oracle Business Intelligence.

Oracle BI Scheduler's job scripts are stored in this location (not in the back-end database), so do not remove scripts from here.

To modify an Oracle BI Scheduler job in Job Manager:

1. In Job Manager, select the job to modify.

   To find out how to open Job Manager, see Section 3.1, "Opening Job Manager".

2. From the **Jobs** menu, select **Modify Job**.

3. In the Modify Job dialog, change the job properties.

   For field descriptions, see Section 6.3, "General Oracle BI Scheduler Job Properties," Section 6.4, "Job Action Properties Available in Job Manager," and Section 6.5, "Job Triggers in Job Manager."

3.4 About Adding Agent Jobs

You cannot add agent jobs using Job Manager. Agents are most commonly added through Oracle BI Presentation Services. For information, see Section 1.1, "What is Oracle BI Scheduler?"

You can, however, modify an agent job using the Modify Job dialog. For information, see Section 3.5, "Modifying Agents in Job Manager."

3.5 Modifying Agents in Job Manager

You can modify an individual agent using the Modify Job dialog in Job Manager.

To modify an agent in Job Manager:

1. In Job Manager, select the agent to modify.

   To find out how to open Job Manager, see Section 3.1, "Opening Job Manager".

2. From the **Jobs** menu, select **Modify Job**.

3. In the Modify Job dialog, change the agent job properties.

   For field descriptions, see Section 6.3, "General Oracle BI Scheduler Job Properties," Section 6.4, "Job Action Properties Available in Job Manager," and Section 6.5, "Job Triggers in Job Manager."

Agent-specific job properties are described in the following table.

Note: A default value in these fields indicates that the value that is specified in "Agent Scheduler Configuration Settings"*Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* is active.

Agent Property	Description
BI Presentation Server	Specifies the Web server that this agent contacts when it runs. Do not change this setting, because the agent might not exist on a different Web server. This feature was added for debugging purposes only.
Debug Log	Determines whether debugging information is written to a log. This overrides the Scheduler's Debug configuration setting for just this one agent. If set to true, then log files named Agent-<*jobID*>-<*InstanceID*>.log are created in the active Scheduler's log directory when the agent is executed.

3.6 Re-Running a Job Instance

You can re-run any type of job with failed job instances to deliver results to recipients which haven't received the agent (for example, agent jobs with status Failed, Warning, Cancelled, Timed Out)). When you re-run a failed agent job, only the failed items are delivered. For example, if there are 1,000 recipients of an agent, but 200 of those recipients belonged to a group that does not have the appropriate request viewing permissions, then the administrator can correct the permission settings and re-run the job. In this example, only the 200 users in the modified group receive the agent. The agent is not re-delivered to the other 800 users who successfully received the agent.

To re-run a job instance:

1. In Job Manager, display the Instance list.

   To find out how to open Job Manager, see Section 3.1, "Opening Job Manager".

2. Locate the job instance to re-run.
3. Right-click the job instance and select **Re-run Instance(s) Now**.

   The **Re-run Instance(s) Now** menu option is not available for successfully completed jobs.

3.7 Managing Oracle BI Scheduler Job Instances

An *instance* in the Oracle BI Scheduler is a record that stores information regarding a specific execution of an Oracle BI Scheduler job.

You can perform the following tasks using Oracle BI Scheduler job instances:

- Section 3.7.1, "Viewing or Refreshing Oracle BI Scheduler Job Instances"
- Section 3.7.2, "Cancelling Oracle BI Scheduler Job Instances"
- Section 3.7.3, "Purging Oracle BI Scheduler Job Instances"

To work with Oracle BI Scheduler job instances:

1. Click the Instances tab in the lower-left corner of the Job Manager window.
2. When instances are present, use the tree in the left pane to locate instances and view information about them.
   - The Instances menu is described in the table in Section 6.2.4, "Instances Menu in Job Manager."
   - Instances properties are described in the table in Section 6.2.5, "Instance Properties in Job Manager."

3.7.1 Viewing or Refreshing Oracle BI Scheduler Job Instances

You can view Oracle BI Scheduler job instance information using the following procedure.

> **Note:** In some environments, if numerous instances have run and instances have not been purged in some time, then this process can take a few seconds.

To view Oracle BI Scheduler job instance information:

1. In Job Manager, go to the Instance list.
2. Select a particular job instance and from the **Instances** menu, select **View Instance**.

   A description of the Instance properties shown in the Instance window is given in Section 6.2.4, "Instances Menu in Job Manager."

To refresh Oracle BI Scheduler job instances:

1. In the Instance List, from the **Instances** menu, select **Refresh Instance List**.

3.7.2 Cancelling Oracle BI Scheduler Job Instances

Registered canceled instances are described in Section 4.8.6, "RegisterCancelCommand Method." The cancel event is issued to the Oracle BI Scheduler and the instance is marked as canceled when its registered cancel methods are called.

To cancel an Oracle BI Scheduler job instance:

1. In Job Manager, display the Instance list.

   To find out how to open Job Manager, see Section 3.1, "Opening Job Manager".
2. Select a particular job instance, and from the **Instances** menu, select **Cancel Instance(s)**.

3.7.3 Purging Oracle BI Scheduler Job Instances

Purging a job instance involves removing it from the back-end database using one of the following methods:

- Section 3.7.3.1, "Using the Job Manager Instances List"
- Section 3.7.3.2, "Using the Job Manager Purge Instances Window"

3.7.3.1 Using the Job Manager Instances List

The following procedure purges Oracle BI Scheduler job instances through the Instances List.

To purge Oracle BI Scheduler job instances through the Instances List:

1. In Job Manager, display the Instance list.

   To find out how to open Job Manager, see Section 3.1, "Opening Job Manager".
2. Select the instances from the Instance List and click **Delete**.

3.7.3.2 Using the Job Manager Purge Instances Window

The following procedure purges job instances through the Purge Instances window.

To purge Oracle BI Scheduler job instances through the Purge Instances window:

1. In Job Manager, display the Instance list.

   To find out how to open Job Manager, see Section 3.1, "Opening Job Manager".

2. Click the Purge Instance(s) icon on the toolbar or from the **Instances** menu, select **Purge Instances** to open the Purge Instances window.

   You can purge instances by JobID, by UserID, or by End Time.

   If you select the End Time method, then all jobs with an End Time less than or equal to the given time are purged.

3. Select the purge method to use.

4. Click **OK** when you have finished to return to the Job Manager window.

4

Programming BI Scheduler VBScript and JScript Jobs

This chapter describes how you can use the Oracle BI Scheduler to schedule general purpose scripts that extend the functionality of Oracle Business Intelligence.

Scripts can either be standalone Script Jobs (in Job Manager), or Script Actions tagged onto the end of agents. (For more information on agents, see *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*). Both types of script use the same script facilities, with a few exceptions. For example Script Actions can access the result set that is being delivered by the agent, while standalone Script Jobs have no access to result sets.

Oracle BI Scheduler includes a Script object that encapsulates a running script.The Script object represents a script and exposes the properties and methods of a script. You can access its methods and properties directly because its name is implied. For example, to access the JobID property, you can specify JobID, not Script.JobID.

> **Note:** Scripting for agents and scripts defined by Oracle BI Scheduler Job Manager are supported only under Windows platforms and are not supported under UNIX.

This chapter describes how to configure custom script properties, includes scripting examples, and provides detailed information about script job properties. It contains the following topics:

4.1 Configuring Custom Script Properties for Oracle BI Scheduler

Use the following procedure to modify the properties of an existing Oracle BI Scheduler script.

To add this job as a standalone job in Job Manager, see Section 3.2, "Adding Oracle BI Scheduler Jobs in Job Manager."

> **Note:** The script has to exist on the Oracle BI Scheduler server computer before you can configure the properties.

To configure custom Oracle BI Scheduler script properties:

1. Set the custom properties according to Section 6.4, "Job Action Properties Available in Job Manager."

   For example, for the purgeSASCache.js script, use the values that are shown in the following table. To view an example of the SASCache.js script, see Section 4.2, "Creating a Custom Script Example - Cache Clearance."

Field	Value or Setting
Script Type	JScript
Script / Script File check boxes	Script File
Script	purgeSASCache.js
Parameter(0): User	Administrator
Parameter(1): Password	*your_password*

2. Click **OK**.

4.2 Creating a Custom Script Example - Cache Clearance

You can use the purgeSASCache.js script to periodically purge all of the cache from the Oracle BI Server. The file must be saved in the following directory:

DOMAIN_HOME/config/fmwconfig/biconfig/OBISCH/schedulerconfig.xml
<DefaultScriptPath/>

```
//////////////////////////////////////////////////////////
//purgeSASCache.js
//
//Purges the cache on SAS.
//Parameter(0) - The user name to pass in to NQCMD.
//Parameter(1) - The password for the aforementioned user.
//////////////////////////////////////////////////////////
//The full path to nqcmd.exe
var nqCmd = "[$INSTALLDIR]\\server\\Bin\\nqcmd.exe";
//The data source name
var dsn = "BI Web";
//The user to execute the queries
var user = Parameter(0);
//The password of the aforementioned user
var pswd = Parameter(1);
//The ODBC procedure call for purging the cache
var sqlStatement = "{call SAPurgeAllCache()};";
//////////////////////////////////////////////////////////
```

```
//Returns a string from the file name
////////////////////////////////////////////////////////
function GetOutput(fso, fileName)
{
    var outStream = fso.OpenTextFile(fileName, 1);
    var output = outStream.ReadAll();
    outStream.Close();
    return output;
}
////////////////////////////////////////////////////////
// Get WshShell object and run nqCmd. Capture the output
// so that we can handle erroneous conditions.
var wshShell = new ActiveXObject("WScript.Shell");
// Create a temp file to input the SQL statement.
var fso = new ActiveXObject("Scripting.FileSystemObject");
var tempFolder = fso.GetSpecialFolder(2);
var tempInFileName = fso.GetTempName();
var tempOutFileName = fso.GetTempName();
tempInFileName = tempFolder + "\\" + tempInFileName;
tempOutFileName = tempFolder + "\\" + tempOutFileName;
var tempInFile = fso.CreateTextFile(tempInFileName, true);
tempInFile.WriteLine(sqlStatement);
tempInFile.Close();
try
{
    // execute
    var dosCmd = nqCmd + " -d \"" + dsn + "\" -u \"" + user
        + "\" -p \"" + pswd + "\" -s \"" + tempInFileName + "\"" +
        " -o \"" + tempOutFileName + "\"";
    wshShell.Run(dosCmd, 0, true);
    var output = GetOutput(fso, tempOutFileName);
    // Remove the temp files
    fso.DeleteFile(tempInFileName);
    if (fso.FileExists(tempOutFileName)) {
        fso.DeleteFile(tempOutFileName);
}
    // Check the output for any errors
    if (output.indexOf("Processed: 1 queries") == -1) {
        ExitCode = -1;
        throw Error(-1, output);
    }
    else if (output.indexOf("Encountered") != -1) {
        ExitCode = -2;
        throw Error(-2, output);
    }
        ExitCode = 0;
} catch (e) {
    if (fso.FileExists(tempInFileName)) {
        fso.DeleteFile(tempInFileName);
    }
    if (fso.FileExists(tempOutFileName)) {
        fso.DeleteFile(tempOutFileName);
    }
    throw e;
}
```

4.3 Configuring Custom Script Properties for Oracle BI Delivers

You set script properties on the Actions tab of an agent in Oracle BI Delivers. See *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition* for details. Refer also to Section 3.5, "Modifying Agents in Job Manager."

> **Note:** The script has to exist on the Oracle BI Scheduler server computer before you can create the agent. Create the custom script, and then create the agent to call the script. See Section 4.4, "Creating a Custom Script Example - Copy Results to the File System."

To configure custom script properties for agents:

1. On the Home page in Oracle BI EE, click the New menu and select the **Agent** option.
2. Display the Conditions tab and select the **Use a condition** box.
3. Click **Create** and **Browse** to select an analysis.
4. Click **OK**.
5. Click **OK**.
6. Display the Actions tab.
7. Click the **Add New Action** icon and select the **Invoke Server Script** menu option.

   The New Action - Invoke Server Script dialog is displayed.
8. Click the **Add Document Parameter** icon.
9. Select the first row of parameters and click the **Delete** button.
10. Enter properties for the parameter displayed.

    For example, for the script that is shown in Section 4.4, "Creating a Custom Script Example - Copy Results to the File System," you use the values that are described in the following table.

Field	Value or Setting
Language	JavaScript
Script Path	createResultfile.js
Name	Enter a name for the parameter.
Prompt	Enter a prompt. For example "Result Set:"
Value	Select Value from the list. This maps a fixed value, for example, PDF. Select Condition Analysis
Optional	Clear the check box.

11. Use the plus icon to display a new row for a second parameter.
12. Enter Result Set Extension in the **Prompt** field.
13. Enter .pdf into the **Value** field.
14. Click **OK**.
15. Save the agent.

This script runs after the Conditional Request of the agent.

4.4 Creating a Custom Script Example - Copy Results to the File System

This example configures a script for the Oracle BI Scheduler that copies the results of an agent to another directory. The script copies the temporary file that contains the results of the Conditional Request to the agent log directory. The JobID, InstanceID, and UserID are used in the file name to guarantee that the result sets do not overwrite each other with each execution of the agent, for each user, or for other agents that share this script.

To add this job in an agent, see Section 4.3, "Configuring Custom Script Properties for Oracle BI Delivers."

The example script uses the following values:

- The agent log directory on the Oracle BI Scheduler computer is DOMAIN_HOME/servers/obisch1/logs.
- The agent is run as Administrator.
- The Custom Script properties are set according to the table in Section 4.3, "Configuring Custom Script Properties for Oracle BI Delivers."
- If the job ID is 101 and instance ID is 1208, then you see a file called 101-1208-Administrator-AgentScript1.pdf.

The output of this example, after the agent is run, is a file on the Oracle BI Scheduler computer called DOMAIN_HOME/servers/obisch1/logs/101-1208-Administrator-AgentScript1.PDF

This file contains the results of the Conditional Request in PDF format.

For all script jobs from chained agents, the full path name to the temporary file is specified in Parameter(0).

```
////////////////////////////////////////////////////////////
//
// createResultFile.js
//
// Copies the results in the temporary file to a new file name
//
// Parameter(0) = Agent Result File Path
// Parameter(1) = Last Part of Output File Name (no path)
//
////////////////////////////////////////////////////////////
var FSO = new ActiveXObject("Scripting.FileSystemObject");
var fileName = GetConfigurationValue("Log Dir", "Agents") +
    "\\" + JobID + "-" + InstanceID + "-" + UserID + "-" +
    Parameter(1);
var fooFile = FSO.CopyFile(Parameter(0), fileName, true);
```

4.5 Oracle BI Scheduler Read-Only Script Object Properties

The Oracle BI Scheduler supports the read-only script object properties that are described in Table 4–1.

Table 4–1 Oracle BI Scheduler Read-Only Script Object Properties

Object Property	Description	Return Value	Syntax
JobID	Returns the job identification number that is associated with this instance.	long	NA
InstanceID	Returns the instance identification number that is associated with this instance.	long	NA
ParameterCount	Returns the number of job parameters that is associated with the job script.	long	NA
Parameter (index)	Returns a specific parameter that is associated with the script. Parameter (index) returns an error if the given index is less than zero or greater than ParameterCountminus 1.	string	Parameter(index) Index is the zero-based index of the parameter.
Script	Returns the Script object that represents the current script. This object implements the COM IDispatch interface and can be passed as arguments to methods of other objects that exist on the system. Implementing the COM IDispatch is particularly useful when handling cancel events to a running instance. See Section 4.8.6, "RegisterCancelCommand Method".	script object	NA
UserID	Returns the user identification number that is associated with the instance.	string	NA

4.6 Oracle BI Scheduler Read/Write Script Object Properties

The Oracle BI Scheduler supports the read/write script object properties that are shown in Table 4–2.

Table 4–2 Oracle BI Scheduler Read/Write Script Object Properties

Object Property	Description	Return Value
Message	Sets or returns the Message property of the running instance. The Message property can convey meaningful error information. Setting this value changes the Message field of a Job Instance without stopping execution of the current Job Script. If the JScript throw() method is called and this property has been set, then the value is appended to the message description in the JScript or VBScript Error object. COM objects that implement the IDispatch interface can be accessed from within Job Scripts. If any method fails and properly provides error information through the SetErrorInfo() method, then that information is contained in the Message field of the Job Instance. If the Message property is set before the COM object error is generated, then that string value is appended to the COM object error information.	string
Severity	Sets the instance status. You can set it to any of the Severity Constants, as described in Section 4.7.1, "Severity Constants.". By default, it is set to nqSeverityInformation.	string
ExitCode	Sets or returns the Exit Code property that is associated with the instance. The default is 0 (zero). See the description of ExitCode instance properties in Section 6.2.5, "Instance Properties in Job Manager."	long

4.7 Oracle BI Scheduler Script-Defined Constants

The Oracle BI Scheduler supports the following script-defined constants. These constants are used by the methods to schedule new jobs.

4.7.1 Severity Constants

This topic is part of Section 4.7, "Oracle BI Scheduler Script-Defined Constants."

Severity constants are used in the severity property of a Message (error message) returned by a script to determine the status of a job instance. Table 4–3 describes Severity values.

Table 4–3 Severity Constant Values

Value	Description
nqSeverityInformation	Set the Severity property to *Information* if the Message contains only information for the job instance; that is, no error condition is reflected. The status of the instance is set to Completed. This is the default if Severity is not set.
nqSeverityWarning	Set the Severity property to *Warning* if the Message contains text that describes a non-critical failure. The instance status is set to Warning.
nqSeverityError	Set the Severity property to *Error* if the Message contains text that indicates a critical failure. The instance status is set to Failed.

4.7.2 DayEnum Constants

This topic is part of Section 4.7, "Oracle BI Scheduler Script-Defined Constants."

The DayEnum values are used with the scheduling functions to identify days in a month, from Day 1 to Day 31. Table 4–4 describes DayEnum values.

Table 4–4 DayEnum Constant Values

Value	Description
nqDay1	Day 1
nqDay2	Day 2
nqDay3	Day 3
...	...
nqDay31	Day 31

4.7.3 DayOfWeekEnum Constants

This topic is part of Section 4.7, "Oracle BI Scheduler Script-Defined Constants."

The DayOfWeekEnum values are used with the scheduling functions to identify days in a week. Table 4–5 describes DayOfWeekEnum values.

Table 4–5 DayOfWeekEnum Constant Values

Value	Description
nqSunday	Sunday
nqMonday	Monday
nqTuesday	Tuesday
nqWednesday	Wednesday
nqThursday	Thursday
nqFriday	Friday
nqSaturday	Saturday

4.7.4 JobFlagsEnum Constants

This topic is part of Section 4.7, "Oracle BI Scheduler Script-Defined Constants."

The JobFlagsEnum values are used with the scheduling methods of the Script object to control how a job behaves. Table 4–6 describes JobFlagsEnum values.

Table 4–6 JobFlagsEnum Constant Values

Value	Description
nqJobNoFlags	This flag indicates that the job has no special behavior.
nqJobDeleteWhenDone	This flag indicates that the job is deleted when there are no more scheduled run times.
nqJobDisabled	This flag indicates that the job is disabled. This is useful for preventing a job from running at the scheduled time or times.
nqJobHasEndDate	This flag indicates that the job has a valid end date.
nqJobExecuteWhenMissed	If for some reason the Oracle BI Scheduler is down when the job is supposed to start, then this flag indicates that the job should run when the Oracle BI Scheduler starts again.
nqJobDeleteScriptWhenDone	When a job is removed and this flag is set, the script that is associated with the job is deleted. This is useful only with the nqJobScriptContainsPath flag.
nqJobScriptContainsPath	This flag indicates that the script that is associated with the job contains a path to a file that contains the actual script code.
nqJobStartNow	When this flag is set, the begin date and start time are ignored. Instead, these fields are set to the current time of the Oracle BI Scheduler.

4.7.5 MonthEnum Constants

This topic is part of Section 4.7, "Oracle BI Scheduler Script-Defined Constants."

The MonthEnum values are used with the scheduling functions to identify months. Table 4–7 describes MonthEnum values.

Table 4–7 MonthEnum Constant Values

Value	Description
nqJanuary	January
nqFebruary	February
nqMarch	March
nqApril	April
nqMay	May
nqJune	June
nqJuly	July
nqAugust	August
nqSeptember	September
nqOctober	October
nqNovember	November
nqDecember	December

4.7.6 OccurrenceEnum Constants

This topic is part of Section 4.7, "Oracle BI Scheduler Script-Defined Constants."

The OccurrenceEnum values are used with the scheduling functions to identify the occurrence of a given day. Table 4–8 describes OccurrenceEnum values.

Table 4–8 OccurrenceEnum Constant Values

Value	Description
nqFirst	First occurrence
nqSecond	Second occurrence
nqThird	Third occurrence
nqFourth	Fourth occurrence
nqLast	Last occurrence

4.8 Oracle BI Scheduler Script Object Methods and Events

You use script object methods and events for the Oracle BI Scheduler when writing programs, as described in Chapter 4, "Programming BI Scheduler VBScript and JScript Jobs." The following sections describe methods and events:

- Section 4.8.1, "CreateArray Method"
- Section 4.8.2, "DeregisterCancelCommand Method"
- Section 4.8.3, "GetConfigurationValue Method"
- Section 4.8.4, "GetTempFileName Method"
- Section 4.8.5, "LaunchProcess Method"
- Section 4.8.6, "RegisterCancelCommand Method"
- Section 4.8.7, "ScheduleJobDaily Method"
- Section 4.8.8, "ScheduleJobMonthlyDate Method"

- Section 4.8.9, "ScheduleJobMonthlyDOW Method"
- Section 4.8.10, "ScheduleJobNow Method"
- Section 4.8.11, "ScheduleJobOnce Method"
- Section 4.8.12, "ScheduleJobWeekly Method"
- Section 4.8.13, "OnError Event"

4.8.1 CreateArray Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Creates an Array object.

Usage: This method is provided only for JScript because local JScript Array objects cannot be passed directly to the Script methods. This method is called to create an array object and to pass the array object to Script methods that accept an array as an argument.

Syntax 1: Set array = CreateArray ()

Syntax 2: Set array = CreateArray (size)

Syntax 3: Set array = CreateArray (element 0, element 1, ..., element *n*)

The different syntax versions create arrays as follows:

- Syntax 1 creates an array of size 0 (zero).
- Syntax 2 creates an array with the specified size.
- Syntax 3 creates an array filled with the specified elements.

Example 4–1 Example

```
var i;
var array1= CreateArray(2);
for (i = 0; i < array1.Size; i++)
{
   array1(i) = i;
}

   array1.Resize(4);
for (i = 2; i < array1.Size; i++)
{
   array1(i) = i;
}

var array2 = CreateArray(0, 1, 2,3);
for (i = 0; i < array2.Size; i++)
{
   if (array1(i) != array2(i))
   break;
}
```

Arguments: See Table 4–9 for CreateArray method arguments.

Return Value: Returns an Array object.

Table 4–9 CreateArray Method Arguments

Argument	Description
size	A long value that specifies the initial size of the array.
element0 … elementn	The values to place in the array. This creates an array with the lower and upper bounds of 0 (zero) and n, respectively.

4.8.2 DeregisterCancelCommand Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Deregisters a previously registered cancel method.

Usage: Call this method to deregister the most recently registered cancel method after a long operation has completed successfully. You need not call this method if the script was canceled.

Syntax: `DeregisterCancelCommand`

4.8.3 GetConfigurationValue Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Returns the value in the configuration relative to the root registry entry of the Oracle BI Scheduler.

Usage: Returns the string value for a registry setting relative to the Oracle BI Scheduler. The `configKey` and `subkeyPath` strings must be identical to those in the registry.

Syntax: value = `GetConfigurationValue(`configKey `[,` subkeyPath`])`

Arguments: See Table 4–10 for GetConfigurationValue method arguments.

Return Value: Returns a string value.

Table 4–10 GetConfigurationValue Method Arguments

Argument	Description
configKey	A string that specifies the registry key name to return.
subkeyPath	(Optional) A string value that specifies the registry path below the Oracle BI Scheduler's root path.

4.8.4 GetTempFileName Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Returns a temporary file name.

Usage: GetTempFileName() does not create a file. It only provides a temporary file name for use in creating a file. Files that are created in job scripts are not deleted automatically when the script terminates.

Syntax: `tfname = GetTempFileName()`

Return Value: Returns a string value.

4.8.5 LaunchProcess Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Executes a command line in a new process.

Usage: Call this method to execute a command line in a new process. If the wait argument is set to True, then this method returns the exit code that is returned by the process.

Syntax: `exitcode = LaunchProcess ( commandLine [, wait, terminateOnCancel] )`

Arguments: See Table 4–11 for LaunchProcess method arguments.

Return Value: Returns a long value.

Table 4–11 LaunchProcess Method Arguments

Argument	Description
commandLine	A string that specifies the command line to execute.
wait	(Optional) A Boolean value that specifies whether the method should wait for the process to terminate. The default is True.
terminateOnCancel	(Optional) A Boolean value that specifies whether the method should terminate the process when the script is canceled. The default is True.

4.8.6 RegisterCancelCommand Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Registers a method to be called when the script is canceled.

Usage: Occasionally, an object's method takes a long time to complete. If the job is canceled before the call returns, then the script engine still must wait until the call returns. This could potentially take hours and limit resources. This method solves the problem by registering a method that is asynchronously called by the script engine if the script gets canceled.

Cancel methods should be registered before calling the method that executes a long operation. When the method returns, the cancel method should be deregistered by calling DeregisterCancelCommand().

A good practice is to hide implementation details of a COM object from the caller, having the COM object itself handle all registration and deregistration of cancel commands. Pass an instance of the Script object to the COM object, then call the RegisterCancelCommand() and DeregisterCancelCommand() methods because the Script object implements the IDispatch interface.

Syntax: `RegisterCancelCommand source, methodName [, arguments]...`

Arguments: See Table 4–12 for RegisterCancelCommand method arguments.

Table 4–12 RegisterCancelCommand Method Arguments

Argument	Description
source	An object whose method is being registered.
methodName	A string that specifies the method name.

Table 4–12 (Cont.) RegisterCancelCommand Method Arguments

Argument	Description
arguments	Optional arguments to be passed into the method.

4.8.7 ScheduleJobDaily Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Schedules a new job with a Daily trigger.

Syntax: `ScheduleJobDaily name, description, scriptType, script, startDate, startTime, endTime, minutesInterval, daysInterval [, parameters, flags, maxRunTimeMS, maxConcurrentInstances, endDate]`

Arguments: See Table 4–13 for ScheduleJobDaily method arguments.

Table 4–13 ScheduleJobDaily Method Arguments

Argument	Description
name	A string that specifies the name of the job.
description	A string that specifies the description of the job.
scriptType	A string that specifies the script type that is associated with the job (either VBScript or JScript).
script	A string that specifies the script code or path (if the nqJobScriptContainsPath flag is set) that is associated with the job.
startDate	A date value that specifies the date that the job is activated.
startTime	A date value that specifies the time that the job is activated.
endTime	A date value that specifies the time that the job is deactivated.
minutesInterval	A long value that specifies the number of minutes between consecutive job executions.
daysInterval	An integer value that specifies the number of days between job invocations.
parameters	(Optional) A string array of parameter values that is passed to the script. The default is an empty array.
flags	(Optional) A long value that specifies the flags that are associated with the job. For valid settings, see Section 4.7.4, "JobFlagsEnum Constants". The default is nqJobNoFlags.
maxRunTimeMS	(Optional) A long value that specifies the maximum time in milliseconds that a job runs before it is terminated. The default is 0 (zero), which means the job can run indefinitely.
maxConcurrentInstances	(Optional) A long value that specifies the maximum number of concurrent running instances of this job. The default is 0 (zero), which means no limit.
endDate	(Optional) A date value that specifies the time that the job is deactivated.

4.8.8 ScheduleJobMonthlyDate Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Schedules a new job with a Monthly by Date trigger.

Syntax: `ScheduleJobMonthlyDate name, description, scriptType, script, startDate, startTime, endTime, minutesInterval, whichDays, whichMonths [, parameters, flags, maxRunTimeMS, maxConcurrentInstances, endDate]`

Arguments: See Table 4–14 for ScheduleJobMonthlyDate method arguments.

Table 4–14 ScheduleJobMonthlyDate Method Arguments

Argument	Description
name	A string that specifies the name of the job.
description	A string that specifies the description of the job.
scriptType	A string that specifies the script type that is associated with the job (either VBScript or JScript).
script	A string that specifies the script code or path (if the nqJobScriptContainsPath flag is set) that is associated with the job.
startDate	A date value that specifies the date that the job is activated.
startTime	A date value that specifies the time that the job is activated.
endTime	A date value that specifies the time that the job is deactivated.
minutesInterval	A long value that specifies the number of minutes between consecutive job executions.
whichDays	An long value that specifies the days of the month on which the job runs. For valid settings, see Section 4.7.2, "DayEnum Constants."
whichMonths	An integer value that specifies the months in which the job runs. For valid settings, see Section 4.7.5, "MonthEnum Constants."
parameters	(Optional) A string array of parameter values that is passed to the script. The default is an empty array.
flags	(Optional) A long value that specifies the flags that are associated with the job. For valid settings, see Section 4.7.4, "JobFlagsEnum Constants." The default is nqJobNoFlags.
maxRunTimeMS	(Optional) A long value that specifies the maximum time in milliseconds that a job runs before it is terminated. The default is 0 (zero), which means the job can run indefinitely.
maxConcurrentInstances	(Optional) A long value that specifies the maximum number of concurrent running instances of this job. The default is 0 (zero), which means no limit.
endDate	(Optional) A date value that specifies the time that the job is deactivated.

4.8.9 ScheduleJobMonthlyDOW Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Schedules a new job with a monthly by day of the week (DOW) trigger.

Syntax: `ScheduleJobMonthlyDOW name, description, scriptType, script, startDate, startTime, endTime, minutesInterval, whichOccurrences, whichDays, whichMonths [, parameters, flags, maxRunTimeMS, maxConcurrentInstances, endDate]`

Arguments: See Table 4–15 for ScheduleJobMonthlyDOW method arguments.

Table 4–15 ScheduleJobMonthlyDOW Method Arguments

Argument	Description
name	A string that specifies the name of the job.
description	A string that specifies the description of the job.
scriptType	A string that specifies the script type that is associated with the job (either VBScript or JScript).
script	A string that specifies the script code or path (if the nqJobScriptContainsPath flag is set) that is associated with the job.
startDate	A date value that specifies the date that the job is activated.
startTime	A date value that specifies the time that the job is activated.
endTime	A date value that specifies the time that the job is deactivated.
minutesInterval	A long value that specifies the number of minutes between consecutive job executions.
whichOccurrences	An integer value that specifies the occurrences of days of the week on which the job runs. For valid settings, see Section 4.7.2, "DayEnum Constants".
whichDays	An integer value that specifies the days of the week on which the job runs. For valid settings, see Section 4.7.3, "DayOfWeekEnum Constants".
whichMonths	An integer value that specifies the months in which the job runs. For valid settings, see Section 4.7.5, "MonthEnum Constants".
parameters	(Optional) A string array of parameter values that is passed to the script. The default is an empty array.
flags	(Optional) A long value that specifies the flags that are associated with the job. For valid settings, see Section 4.7.4, "JobFlagsEnum Constants". The default is nqJobNoFlags.
maxRunTimeMS	(Optional) A long value that specifies the maximum time in milliseconds that a job runs before it is terminated. The default is 0 (zero), which means the job can run indefinitely.
maxConcurrentInstances	(Optional) A long value that specifies the maximum number of concurrent running instances of this job. The default is 0 (zero), which means no limit.
endDate	(Optional) A date value that specifies the time that the job is deactivated.

4.8.10 ScheduleJobNow Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Schedules a new job with a Run Now trigger.

Syntax: `ScheduleJobNow name, description, scriptType, script [,` parameters, flags, `maxRunTimeMS]`

Arguments: See Table 4–16 for ScheduleJobNow method arguments.

Table 4–16 ScheduleJobNow Method Arguments

Argument	Description
name	A string that specifies the name of the job.

Table 4–16 (Cont.) ScheduleJobNow Method Arguments

Argument	Description
description	A string that specifies the description of the job.
scriptType	A string that specifies the script type that is associated with the job (either VBScript or JScript).
script	A string that specifies the script code or path (if the nqJobScriptContainsPath flag is set) that is associated with the job.
parameters	(Optional) A string array of parameter values that is passed to the script. The default is an empty array.
flags	(Optional) A long value that specifies the flags that are associated with the job. For valid settings, see Section 4.7.4, "JobFlagsEnum Constants". The default is nqJobNoFlags.
maxRunTimeMS	(Optional) A long value that specifies the maximum time in milliseconds that a job runs before it is terminated. The default is 0 (zero), which means the job can run indefinitely.

4.8.11 ScheduleJobOnce Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Schedules a new job with a Run Once trigger.

Syntax: `ScheduleJobOnce name, description, scriptType, script, startDate, startTime [, parameters, flags, maxRunTimeMS]`

Arguments: See Table 4–17 for ScheduleJobOnce method arguments.

Table 4–17 ScheduleJobOnce Method Arguments

Argument	Description
name	A string that specifies the name of the job.
description	A string that specifies the description of the job.
scriptType	A string that specifies the script type that is associated with the job (either VBScript or JScript).
script	A string that specifies the script code or path (if the nqJobScriptContainsPath flag is set) that is associated with the job.
startDate	A date value that specifies the date that the job is activated.
startTime	A date value that specifies the time that the job is activated.
parameters	(Optional) A string array of parameter values that is passed to the script. The default is an empty array.
flags	(Optional) A long value that specifies the flags that are associated with the job. For valid settings, see Section 4.7.4, "JobFlagsEnum Constants". The default is nqJobNoFlag.
maxRunTimeMS	(Optional) A long value that specifies the maximum time in milliseconds that a job runs before it is terminated. The default is 0 (zero), which means the job can run indefinitely.

4.8.12 ScheduleJobWeekly Method

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Schedules a new job with a Weekly trigger.

Syntax: `ScheduleJobWeekly name, description, scriptType, script, startDate, startTime, endTime, minutesInterval, weeksInterval, whichDays [, parameters, flags, maxRunTimeMS, maxConcurrentInstances, endDate]`

Argument: See Table 4–18 for ScheduleJobWeekly method arguments.

Table 4–18 ScheduleJobWeekly Method Arguments

Argument	Description
name	A string that specifies the name of the job.
description	A string that specifies the description of the job.
scriptType	A string that specifies the script type that is associated with the job (either VBScript or JScript).
script	A string that specifies the script code or path (if the nqJobScriptContainsPath flag is set) that is associated with the job.
startDate	A date value that specifies the date that the job is activated.
startTime	A date value that specifies the time that the job is activated.
endTime	A date value that specifies the time that the job is deactivated.
minutesInterval	A long value that specifies the number of minutes between consecutive job executions.
weeksInterval	An integer value that specifies the number of weeks between job invocations.
whichDays	An integer value that specifies the days of the week on which the job runs. See Section 4.7.3, "DayOfWeekEnum Constants" for valid settings.
parameters	(Optional) A string array of parameter values that is passed to the script. The default is an empty array.
flags	(Optional) A long value that specifies the flags that are associated with the job. For valid settings, see Section 4.7.4, "JobFlagsEnum Constants". The default is nqJobNoFlags.
maxRunTimeMS	(Optional) A long value that specifies the maximum time in milliseconds that a job runs before it is terminated. The default is 0 (zero), which means the job can run indefinitely.
maxConcurrentInstances	(Optional) A long value that specifies the maximum number of concurrent running instances of this job. The default is 0 (zero), which means no limit.
endDate	(Optional) A date value that specifies the time that the job is deactivated.

4.8.13 OnError Event

This topic is part of Section 4.8, "Oracle BI Scheduler Script Object Methods and Events."

Occurs when the script engine encounters a run-time error while executing the script. This is intended for cleanup purposes, but the creative use of try/catch blocks in JScript and appropriate Error Handling in VBScript are often superior alternatives to using this event.

Usage: The script engine calls this procedure when it encounters a run-time error while executing the script. Define this procedure in your script to perform cleanup

activities before the script terminates, such as deleting temporary files and releasing resources.

Syntax: `OnError`

Example 4–2 Using VBScript:

```
Public Sub OnError()
   LogFile.WriteLine "Encountered a runtime error in the script."
LogFile.Close
End Sub
```

Example 4–3 Using JScript:

```
function OnError()
{
   LogFile.WriteLine("Encountered a runtime error in the
   script.");
LogFile.Close();
```

4.9 Troubleshooting JScript and VBScript Job Failures

If a JScript or VBScript job fails with the error "nQSError: 66001] Failed to create the ActiveX scripting engine.", then the required script engine (VBScript or JScript) is not available or is broken on this computer.

You can reregister the script DLL files in Windows by running the command "`regsvr32 vbscript.dll`" for a VBScript failure or "`regsvr32 jscript.dll`" for a JScript failure. If these return the message "failed module could not be found", then you must repair the Windows installation to re-instate the missing DLL files. You can achieve this by rolling back to a restore point, or by carrying out a repair using the original operating system installation discs.

5

Programming BI Scheduler Java Jobs

This chapter explains how to use the Oracle BI Scheduler to schedule Java programs, called Java jobs, that extend the functionality of Oracle Business Intelligence. Java jobs can be either standalone Scheduler Jobs in Job Manager (Windows), or existing Java Actions in Oracle BI Delivers that are added to the end of agents and upgraded from Release 11*g* (Windows or UNIX).

Note that existing Java actions that are upgraded from Release 11*g* can be executed, but not created in Release 12*c* (12.2.1). Instead, you should use the EJB-based Java actions that are available in this release. For more information on actions, see *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

This chapter describes programming Java jobs for the Oracle BI Scheduler, and contains the following topics:

5.1 Using Oracle BI Scheduler Java Jobs

Java jobs are Java programs that are executed by the JavaHost process on behalf of the Oracle BI Scheduler. Java jobs are different than Java EJB actions. A Java job is deployed in a JAR file, with the entry point defined by one class. That class must implement the SchedulerJavaExtension interface. The job's context is provided by the input SchedulerJobInfo parameter.

This section contains the following information:

5.1.1 Adding and Configuring Custom Java Jobs in Job Manager

You add and configure custom Java jobs in the Modify Job and Add Jobs dialogs in Job Manager. Refer to Section 3.2, "Adding Oracle BI Scheduler Jobs in Job Manager", and Section 3.3, "Modifying Oracle BI Scheduler Jobs in Job Manager".

> **Note:** The Java program must exist on the Oracle BI Scheduler server computer before you can create the job in Job Manager. Create the Java program, and then create the job to call the Java program.

To configure and add custom Java jobs in Job Manager:

1. Set the custom properties according to the descriptions in Section 6.4, "Job Action Properties Available in Job Manager".
2. In the Add New Job window, enter the properties.

   For example, for the Java program filecopy.jar, use the values that are shown in the following table. To view an example of the filecopy.jar program, see Section 5.1.2, "Example: Creating a Java Program for Agents".

Field	Value or Setting
Script Type	Java
Class Name	sched.sched The Java class that you created in Section 5.1.2, "Example: Creating a Java Program for Agents."
Class Path	filecopy.jar The JAR file that contains the Java class.
Parameters	c:\tmp\report.pdf **Note:** The owner of the JavaHost process must have write permissions on this directory point.

3. Click **OK**.

   The Java program is run after the Conditional Request of the agent is run.

   You cannot add a new Java job action to an agent in Oracle BI Delivers. You can use only existing ones that have been upgraded to this release. However, you can add a new Java job to a job in Job Manager. For information, see Section 3.2, "Adding Oracle BI Scheduler Jobs in Job Manager."

5.1.2 Example: Creating a Java Program for Agents

This example creates a Java program that copies the results of an agent to another directory. The example creates a Java class that contains filecopy logic.

To create a Java program to be used with agents:

1. Create a Java program using a Java editor.
   a. Create a new Java class called 'sched'.
   b. Paste the following code into the Java editor:

```
package sched;
    import java.io.*;
    import java.lang.Thread;
```

```
    import
com.siebel.analytics.scheduler.javahostrpccalls.SchedulerJavaExtension;
    import
com.siebel.analytics.scheduler.javahostrpccalls.SchedulerJobException;
    import
com.siebel.analytics.scheduler.javahostrpccalls.SchedulerJobInfo;

    public class sched implements SchedulerJavaExtension{
    public void run(SchedulerJobInfo jobInfo) throws SchedulerJobException
    {
      System.out.println("JobID is:" + jobInfo.jobID());
      System.out.println("Instance ID is:" + jobInfo.instanceID());
      System.out.println("JobInfo to string is:" + jobInfo.toString());
      try
      {
        // File outputFile = new File("D:\\JavaJob.txt");
        File attachFile = jobInfo.getResultSetFile();

        InputStream in = new FileInputStream(attachFile.getAbsolutePath());
       OutputStream out = new FileOutputStream(jobInfo.parameter(0));
        byte[] buf = new byte[1024];
        int len;
        while ((len = in.read(buf)) > 0)
        {
          out.write(buf, 0, len);
        }
        in.close();
        out.close();

      }
    catch(Exception ex)
    {
      throw new SchedulerJobException(1, 1, ex.getMessage());
    }
    }
    public void cancel()
    {
    }
    }
```

c. Add the schedulerrpccalls.jar file from the ORACLE_HOME/bi/bifoundation/javahost/lib/scheduler directory into your classpath.

d. Compile the Java Class without errors.

e. Jar the compiled output to a file. For example, filecopy.jar.

f. Note the location of the file and ensure that there are no errors.

5.1.3 Example: Configuring a Java Program

This example configures the Java program that you created in Section 5.1.2, "Example: Creating a Java Program for Agents" to enable the Java job to work with agents.

To configure a Java program to be used with agents:

1. Copy the filecopy.jar file that you created in Section 5.1.2, "Example: Creating a Java Program for Agents" to the following directory:

   `ORACLE_HOME/bi/bifoundation/javahost/lib`

2. Make the following changes to the JavaHost configuration file, which is called config.xml:

```
<Scheduler>
  <Enabled>True</Enabled>
<DefaultUserJarFilePath>D:\<ORACLE_
HOME>\bi\bifoundation\javahost\lib</DefaultUserJarFilePath>
</Scheduler>
```

   See "Using the Javahost Service for Oracle BI Presentation Services" *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for information on working with the JavaHost configuration file.

   If the JavaHost file is not configured correctly, then the agent log file can stop getting written to, although the agent and the Scheduler are still running. In this situation, you stop the Scheduler using the Windows Task Manager.

3. Restart the JavaHost service.

5.2 Oracle BI Scheduler Java Jobs

The Oracle BI Scheduler integrates with the JavaHost Service to support a custom Java program. The Oracle BI Scheduler provides two Java interfaces (SchedulerJavaExtension and SchedulerJobInfo) and one Java class (SchedulerJobException). You provide a class that implements the SchedulerJavaExtension interface.

Note: For more information about the JavaHost service, see, "Using the Javahost Service for Oracle BI Presentation Services" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

5.3 Adding Java Jobs for Oracle BI Scheduler

Use the following procedure to add a Java job for the Oracle BI Scheduler.

Note: The compiled Java class file has to exist on the JavaHost computer before you can configure the properties.

To add a Java Job for Oracle BI Scheduler:

1. In Job Manager, select the Jobs menu, then select **Add New Job** to display the Add New job window.

   To find out how to open Job Manager, see Section 3.1, "Opening Job Manager".

2. In the Script Type field, select Java.

3. Specify the custom properties. For information about setting these values, see Section 6.4, "Job Action Properties Available in Job Manager".

   Example values and settings for a Java job with the class name "sample.Test", file path "Sample", and no additional paths and parameters are included below.

Field	Value or Setting
Script Type	Java

Field	Value or Setting
Class Name	sample.Test
Class File (Jar File)	Sample

4. Click **OK**.

5.4 Oracle BI Scheduler Custom Java Program Package

The public interfaces and class for Oracle BI Scheduler Custom Java Program are packaged as *com.siebel.analytics.scheduler.javahostrpccalls*. There are two interfaces and one class, which are described in following topics:

- Section 5.5, "SchedulerJavaExtension Interface"
- Section 5.6, "SchedulerJobInfo Interface"
- Section 5.7, "SchedulerJobException Class"

5.5 SchedulerJavaExtension Interface

Your custom code must implement the following interface:

```
package com.siebel.analytics.scheduler.javahostrpccalls;
public interface SchedulerJavaExtension {
public void run(SchedulerJobInfo jobInfo) throws SchedulerJobException;
public void cancel();
}
```

This interface has two methods: run and cancel. The following table describes the methods:

Method	Description
run	This method is invoked by the JavaHost. It provides one SchedulerJobInfo object (described below), which contains instance-related properties such as user ID, Job ID, and Instance ID and parameters. The method is declared to throw SchedulerJobException, which is also described below.
cancel	This method is invoked if the Job instance is still running while Scheduler wants to cancel it. The cancel method is called concurrently by a different thread. Your implementation must therefore protect any data that is shared by the run and cancel methods by synchronization blocks. A typical implementation would be to set a 'cancelCalled' boolean in the cancel method implementation, and check this in any long running loops in the run implementation.

5.6 SchedulerJobInfo Interface

The SchedulerJobInfo interface provides information about the currently running job instance to the custom code:

```
package com.siebel.analytics.scheduler.javahostrpccalls;
import java.io.*;
public interface SchedulerJobInfo {
public final int kJavaJobInformation = 0;
public final int kJavaJobWarning = 1;
public final int kJavaJobError = 2;
int jobID();
int instanceID();
```

```
int parameterCount();
String parameter(int index);
boolean hasResultSet();
File getResultSetFile();
String userID();
int getExitCode();
void setExitCode(int exitCode);
int getStatus();
void setStatus(int status);
String getMessage();
void setMessage(String message);
void appendMessage(String message);
}
```

Three public final integers, *kJavaJobInformation*, *kJavaJobWarning*, and *kJavaJobError* are the suggested values that are used to set the status depending upon the circumstances. The following table describes the circumstances:

Members	Description
public final int kJavaJobInformation = 0	Contains an informational message.
public final int kJavaJobWarning = 1	Contains a warning message.
public final int kJavaJobError = 2	Contains an error message.

The following table describes all the methods that are declared in the interface:

Method	Description
int jobID()	Returns the job ID that is associated with the agent.
int instanceID()	Returns the instance ID that is associated with the agent.
int parameterCount()	Returns how many parameters are associated with the agent.
String parameter(int index)	Returns the indexed parameter for the agent.(1).
boolean hasResultSet()	Specifies if there is a result set for this agent.
File getResultSetFile()	Returns a file of result set for this agent (2).
String userID()	Returns the ID of the user who is running the agent.
int getExitCode()	Returns the exit code for the agent.
void setExitCode(int exitCode)	User can set the exit code for the agent.
int getStatus()	Returns the status code for the agent.
void setStatus(int status)	User can set the status code for the agent.
String getMessage()	Returns the message that is associated with the agent.
void setMessage(String message)	User can set the message that is associated with the agent. It replaces the existing message.
void appendMessage(String message)	User can append an additional message to the agent.

5.7 SchedulerJobException Class

If your custom code cannot complete successfully, then throw an instance of this exception class.

```
package com.siebel.analytics.scheduler.javahostrpccalls;
public final class SchedulerJobException extends Exception {
public SchedulerJobException(int exitCode, int status, String message) {
m_exitCode = exitCode;
m_status = status;
m_message = message;
}
public int getExitCode() {
return m_exitCode;
}
public int getStatus() {
return m_status;
}
public String getMessage() {
return m_message;
}
private int m_exitCode;
private int m_status;
private String m_message;
}
```

The *run* method of the SchedulerJavaExtension interface is declared to throw SchedulerJobException. The following table describes the three members:

Members	Description
int m_exitCode	The framework assigns this exit code to the agent.
int m_status	The framework assigns this status code to the agent.
String m_message	The framework assigns this message to the agent.

5.8 Oracle BI Scheduler Java Extension Example

The following example illustrates how to use the previously described interfaces and class to create a custom Java action. For more information, see Section 5.1, "Using Oracle BI Scheduler Java Jobs."

This example does not contain any long running code, so it is acceptable to do nothing in the cancel method.

When the compiled class runs, it collects the ID of the user who ran the agent, the job ID of the agent, the instance ID of the agent, and all possible parameters into an output file.

```
package sample;
import java.io.*;
import java.lang.Thread;
import com.siebel.analytics.scheduler.javahostrpccalls.SchedulerJavaExtension;
import com.siebel.analytics.scheduler.javahostrpccalls.SchedulerJobException;
import com.siebel.analytics.scheduler.javahostrpccalls.SchedulerJobInfo;
/**
 *
 * @author
 */public class SimpleTest implements SchedulerJavaExtension
{
public void run(SchedulerJobInfo jobInfo) throws SchedulerJobException
{
System.out.println("JobID is:" + jobInfo.jobID());
System.out.println("Instance ID is:" + jobInfo.instanceID());
System.out.println("JobInfo to string is:" + jobInfo.toString());
try
```

```
{
File outputFile = new File("D:\\temp\\JavaJob.txt");
FileWriter out = new FileWriter(outputFile);
out.write("User ID:\t\t" + jobInfo.userID() + "\r\n");
out.write("Job ID:\t\t" + jobInfo.jobID() + "\r\n");
out.write("Instance ID:\t\t" + jobInfo.instanceID() + "\r\n");
out.write("Parameter Count:\t\t" + jobInfo.parameterCount() + "\r\n");
for(int i = 0; i < jobInfo.parameterCount(); ++i)
{
out.write("\tParameter ");
out.write(new Integer(i).toString());
out.write(":\t" + jobInfo.parameter(i) + "\r\n");
}
out.close();
}
catch(Exception ex)
{
throw new SchedulerJobException(1, 1, ex.getMessage());
}
}
public void cancel()
{
}
}
```

6

Oracle BI Scheduler Job Manager Menus

This chapter describes the menus and options that are available in Job Manager, and contains the following topics:

6.1 About Job Manager

Use Job Manager to add, remove, modify, or cancel Oracle BI Scheduler jobs. For example, you can perform the following tasks:

- Set options for an execution schedule, such as a start time, a start date, an interval between executions, and an optional end time and date.
- Add or modify jobs using the Add Job and Modify Job dialogs. These dialogs contain three types of information:
  - General job properties
  - A script area where you can specify the actions to perform
  - A trigger area where you can specify the job trigger

    The trigger defines when the job is run.

6.2 Toolbar Menus in Job Manager

The Job Manager toolbar contains four menus, as described in the following topics:

6.2.1 File Menu in Job Manager

Table 6–1 describes the File menu options.

Table 6–1 Job Manager File Menu Options

Command	Description
Open Scheduler Connection	Opens the Machine Name dialog, which provides alternative connection mechanisms. If you run a single, non-clustered Scheduler, connect using the option Connect directly to the Active Scheduler. If the active scheduler is running on this machine (and is not configured to listen only on a particular network interface), then you might use `localhost` for the machine name and specify the port number (usually 9704). If the Scheduler is clustered, then use the option Connect Through Cluster Controllers. This option ensures that you can successfully connect irrespective of which Scheduler is currently the active Scheduler. If the system has been secured with SSL, then you must select the **SSL** check box. If the default SSL configuration is used, then you can leave all other SSL fields empty.
Close Scheduler Connection	Closes the Job Manager connection to Oracle BI Scheduler.
Exit	Shuts down Job Manager and returns you to the Oracle BI Administration Tool. If you exit Job Manager while a connection to Oracle BI Scheduler is still open, then the connection closes.

6.2.2 Service Management Menu in Job Manager

Table 6–2 describes the Service Management menu options.

Table 6–2 Job Manager Service Management Menu

Command	Description
Pause Scheduling	Stops all jobs from executing until scheduling is continued. Pause Scheduling is sometimes required for maintenance purposes. It allows an administrator to intervene and resolve out of control jobs. A custom job that uses excessive resources and that is being scheduled very frequently, might make any other processes on that computer ineffective, including Job Manager. Pausing scheduling offers a chance to remove or modify the job. Sometimes you might need to Pause Scheduling while Oracle BI Scheduler is stopped. In this case, scheduling continues when Oracle BI Scheduler is restarted, unless you also set the option Pause When Service Starts. For more information about the JavaHost service, see, "General Scheduler Configuration Settings That Affect Agents" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.
Continue Scheduling	Resumes Oracle BI Scheduler's regular execution.
Stop Service	Stops the Oracle BI Scheduler service.

6.2.3 Jobs Menu in Job Manager

Table 6–3 describes the Jobs menu options.

Table 6–3 Job Manager Jobs Menu

Command	Description
Add New Job	Opens the Add New Job window, where you specify the properties for a new job.
Remove Job(s)	Removes the selected job or jobs from Oracle BI Scheduler. When a job is removed, all instances for that job are also removed.

Table 6–3 (Cont.) Job Manager Jobs Menu

Command	Description
Modify Job	Opens the Modify Job window where you can modify the properties for an existing job.
View Job	Opens the View Job window where you can view the properties for a job.
Run Job(s) Now	Immediately runs the scheduled job that you selected.
Refresh Job List	Refreshes the job information that is displayed in the Job List in the right pane. To view the instances that are associated with one or more jobs, first highlight the jobs in the job view. Then press the refresh icon in the instances view below.

6.2.4 Instances Menu in Job Manager

An Oracle BI Scheduler instance records information regarding a specific execution of a job in the Oracle BI Scheduler. Table 6–4 describes the Instances menu options.

Table 6–4 Job Manager Instances Menu

Command	Description
Cancel Instance(s)	Cancels the running job. When a job has been successfully canceled, the job's status is displayed as "Canceled."
Purge Instance	Opens the Purge Instances dialog where you can specify the delete instance method to use. You can delete the instance by Job ID, User ID, and Before a particular time.
View Instance	Displays information about the selected instance.
Re-Run Job Instance(s) Now	Re-run a job instance. When you re-run a failed agent job, only the failed items are delivered.
Refresh Instance List	Refreshes the instance information that is displayed in the Instance List in the Job Instance View pane.

6.2.5 Instance Properties in Job Manager

Table 6–5 describes the properties of Job Manager instances.

Table 6–5 Job Manager Instance Properties

Field	Description
JobID	ID of the job that is associated with this instance.
Status: Running	This is the same for agent jobs and Script jobs. If the instance is running, then the status is running.
Status: Completed	For agent: The agent instance is set to complete if deliveries are successful to all delivery devices of the agent. For Script: This is set according to the Severity property in the script. See Table 4–2, " Oracle BI Scheduler Read/Write Script Object Properties".
Status: Failed	For agent: The agent instance is set to failed if deliveries are unsuccessful to any of the delivery devices of the agent. For Script: This is set according to the Severity property in the script. See Table 4–2, " Oracle BI Scheduler Read/Write Script Object Properties".

Table 6–5 (Cont.) Job Manager Instance Properties

Field	Description
Status: Canceled	Canceling any instance from Job Manager sets the status to canceled.
Status: Timed Out	If the job has a maximum run time and the running time of the instance exceeds this time, then the status of the instance is set to timed out.
Status: Warning	For agent: The agent instance is set to Warning if deliveries are successful to some delivery devices but not all. For Script: This is set according to the Severity property in the script. See Table 4–2, " Oracle BI Scheduler Read/Write Script Object Properties".
Status: ReRunning	When there is a cluster of schedulers, and the active scheduler has failed during job execution, the failed parts of the job are rescheduled.
Status: Try Again	If the job has completed with one or more recoverable errors, the failed parts are rescheduled, and the status set to Try Again, for example, if calls to BI Presentation Server timed out. If the job status remains at status Try Again, it cannot be automatically rescheduled, and you should reschedule the job manually.
InstanceID	Unique ID of this specific instance of the job.
Begin Time	The day and time that the Scheduler initiated the job instance.
End Time	The day and time that the job scheduler completed the job instance.
ExitCode: Agent	The ExitCode of an instance is set to the number of successful deliveries. The count corresponds to the number of successful deliveries to devices, and multiple devices might exist for each recipient of an agent.
ExitCode: Script	The ExitCode of an instance is set according to the ExitCode property in the script. The default is 0 (zero). See Table 4–2, " Oracle BI Scheduler Read/Write Script Object Properties"
Message	Text message that contain any error information of the instance, warnings, or general messages about the instance execution.

6.3 General Oracle BI Scheduler Job Properties

In the Add Job or Modify Job dialog, use the fields to configure or modify the general properties for a job. Table 6–6 describes the general job properties.

In addition, Table 6–7 describes the job action properties, and Table 6–8 describes the recurrent job triggers.

Table 6–6 General Oracle BI Scheduler Job Properties

Field	Description
Name	Short, descriptive name for the job. This field is also displayed in the Job List display in the right pane of the Job Manager window.
Description	Brief description of the job that describes its actions to end users. This field is also displayed in the Job List display in the right pane of the Job Manager window.

Table 6–6 (Cont.) General Oracle BI Scheduler Job Properties

Field	Description
UserID	Required for all jobs. For jobs that communicate with the Oracle BI Server or with Oracle BI Presentation Services, the UserID must be a valid Oracle Business Intelligence user ID. This field is also displayed in the Job List display in the right pane of the Job Manager window. When this job runs, Oracle BI Scheduler executes it on behalf of the user ID that is specified in this field.
Maximum Run Time MS	Specifies the maximum number of milliseconds that this job should run before it is canceled forcibly. If a job exceeds its run time, then it fails with a time-out reason code. To prevent the job from timing out, set this field to 0 (zero). **Note**: One second equals 1,000 milliseconds.
Last Run Time	Display-only field that shows the last time that this job began execution. This field is also displayed in the Job List display in the right pane of the Job Manager window. If the date and time displayed here are for time zones where daylight savings applies, the time zone reflects the daylight savings time. For example, if (GMT) Greenwich Mean Time: Dublin, Edinburgh, Lisbon, London, is set during the summer months, then this means BST (British Summer Time).
Next Run Time	Display-only field that shows recurrent jobs and the next time this job executes. The trigger is used to determine this value. If the date and time displayed here is for time zones where daylight savings applies, the time zone reflects the daylight savings time. For example, if (GMT) Greenwich Mean Time: Dublin, Edinburgh, Lisbon, London, is set during the summer months, then this means BST (British Summer Time).
Running Instance Count	Display-only field that shows the number of currently running instances of this job.
Delete Job When Done	When you select this option, Oracle BI Scheduler deletes the job after its last scheduled execution as defined by its trigger. When there is no next run time, the job is done. When a job is deleted, all instances are deleted as well. For most jobs, you should not select this option, because you can delete a job manually through Job Manager.
Disabled	When you select this option, the job script does not execute when its trigger expires. However, the next run time is still updated according to the trigger settings. The Disabled field is useful when testing or debugging a new job because an administrator can quickly disable a job without losing all information.
Execute When Missed	If you select this option while Oracle BI Scheduler is stopped (either all scheduling pauses or the Oracle BI Scheduler application stops), and if the job's next run time was missed, then the job runs after Oracle BI Scheduler restarts. If you do not select this option, then the job executes at the its next run time, as defined by its trigger.
Delete Script When Job is Removed	If you select this option, then when a job is removed, its associated job script is also removed. If many jobs reference the same job script, then this option should not be set.

6.4 Job Action Properties Available in Job Manager

Table 6–7 describes the job action properties available in the Add Job and Modify Job dialogs. Use the fields in the Script area of the Add Job or Modify Job dialog to define the actions a job performs when it executes.

Table 6–7 Job Manager Job Action Fields

Field	Description
Script Type	Oracle BI Scheduler supports VBScript, JScript, Java, and NQCmd. Set this field according to the type of script that is referenced by the Script field. The fields that are displayed depend upon the type of script that you specify.
Load Script from File	(VBScript and JScript only) In Job Manager, you can enter either a file name or the actual contents of a script in the Script Path field.
Script	(VBScript and JScript only) This value is either a reference to a job script file or the contents of a job script itself. If it is a reference, then enter a file name in this field, such as TestConnect.js. If no path is given, then Oracle BI Scheduler examines the directory that is referred to in the Default Script Path configuration value in For more information about the JavaHost service, see, "General Scheduler Configuration Settings That Affect Agents" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*. If a path is given, then the path must be accessible by the Oracle BI Scheduler application.

Parameters	(VBScript, JScript, and Java only) Field values are passed to the job script through the Parameters array. Enter one parameter per line. For example:
	`c:\oracleBI\data\scheduler`
	`cli_snowflake`
	`SELECT Lastname FROM Employee`

Class Name	(Script Type = Java) The fully qualified implementation class for the Java program action.
Class Path (jar file)	(Script Type = Java) The name of the JAR file that contains the Java program.
Additional Class Path(s)	(Script Type = Java) Other JAR files that contain the utility classes and libraries that the Java program must run properly. Include a comma-delimited list of JAR files. After this parameter, you must append a command line parameter as it is in Job Manager.
DSN (Data Source Name)	(Script Type = NQCmd) The data source name that Oracle BI Scheduler uses to connect to the BI Server. **Note**: Impersonation is used with NQCmd. The value in the User ID is the user that Oracle BI Scheduler tries to impersonate when connecting to the BI Server.
SQL Input File	(Script Type = NQCmd) The fully qualified path to the SQL file that NQCmd executes. Type the full path or click the ... button to browse to the file's location. This field is typically used for files that are generated by the aggregate persistence feature.

Additional Command Line Parameters	(Script Type = NQCmd) Parameters that are passed to NQCmd. Enter one parameter per line. For example:
	`-o`
	`D:\foo\bar.txt`

6.5 Job Triggers in Job Manager

A job trigger determines when and how often a job executes. Use the fields in the Trigger area of the Add Job or Modify Job dialog to define the actions that a job performs when it executes.

There are two types of Oracle BI Scheduler Job triggers—*single-run* triggers and *recurrent* triggers.

6.5.1 Single-Run Triggers

Use the Trigger Type list to select the trigger type. Single-run triggers perform the action once. There are two single-run triggers:

- **Run Now.** This trigger specifies that the job runs immediately. It executes only one time.
- **Run Once.** Jobs of this trigger type execute at the date and time that is specified in the Begin Date and Start Time fields, which become active when you selected **Run Once**. An error occurs if the given time is in the past. If you select the **Set Start Time To Now** option, then this trigger is equivalent to the **Run Now** trigger.

6.5.2 Recurrent Triggers

All recurrent triggers specify that the job execute over a period of time at given intervals. Fields used by recurrent triggers are described in Table 6–8. Recurrent Trigger Types are described in Table 6–9.

Table 6–8 Job Manager Recurrent Trigger Fields

Field	Description
Timezone	Specifies the time zone that is used to execute the job. Displays the default time zone as specified in the instanceconfig.xml file. For information, see For more information about the JavaHost service, see, "Description of Time Zone Settings" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.
Begin Date	Specifies the date when the first recurrent interval runs. The recurrent interval is defined as the time between Start Time and End Time. This field is hidden if you select the Set Start Time to Now option.
End Date	Specifies the date when the last recurrent interval is run. Becomes active when the Has End Date option is set. If no end date is set, then the job runs forever.
Start Time	Specifies the lower bounds of the recurrent interval. The job's first execution for a day occurs at the time that is specified in this value.
End Time	Specifies the upper bounds of the recurrent interval. The job's last execution for a given day occurs at or before the time that is specified in this value. If this value is less than the Start Time value, then the interval spans midnight of the given day. For example, a trigger with a start time of 11:00 P.M. and an End Time of 2:00 A.M. starts its execution on the date that is specified in Begin Date at 11:00 P.M. and continues until 2:00 A.M. on the following day.

Table 6–8 (Cont.) Job Manager Recurrent Trigger Fields

Field	Description
Has End Date	If you select this option, then specify an End Date. If you do not select this option, then the job stays scheduled. **Note**: The schedule is perpetual; the job instance is not. If you restart Oracle BI Scheduler, then the next run time is set as dictated by the job schedule. If an instance is running while you stop Oracle BI Scheduler, then it is canceled.
Set Start Time To Now	If you select this option, then the Begin Date and Start Time fields are ignored and their values are populated with Oracle BI Scheduler's current date and time.
Interval in Minutes	Specifies the number of minutes between subsequent executions of a job during the recurrent interval. A job starts execution promptly at its Start Time and executes again every *n* minutes, where *n* is the value of this field.
Maximum Concurrent Instances	If a job executes every *n* minutes (from the Interval in Minutes field), then a long-running job might have overlapping executions. Use this field to set the number of concurrent running instances. For an unlimited number of concurrent instances, set this value to zero.

6.5.2.1 Recurrent Trigger Types

The recurrent trigger types that are available from the Trigger Type list are described in Table 6–9. The fields pertain to all recurrent triggers. Depending on the trigger type that you select, additional options become active. The examples in the table illustrate how these additional options can be used.

Table 6–9 Job Manager Recurrent Trigger Types

Trigger Type	Description and Example
Daily	Runs a job every day or every few days. The Days Interval field specifies the number of days between each subsequent recurrent interval. For example: To run a job every hour between 8:00 A.M. and 5:00 P.M. starting on January 1, 2010 and ending on January 15, 2010, set the Begin Date to 1/1/10, the Start Time to 8:00 A.M., and the End Time to 5:00 P.M. Set the Has End Date flag, the End Date to 1/15/10, the Interval in Minutes to 60, and the Days Interval to 1. To run a job every five minutes forever, set the Begin Date to the desired date, the Start Time to 12:00 P.M., the end time to 11:59 A.M., the Interval In Minutes to 5, and the Days Interval to 1.
Weekly	Runs a job on specified days of the week. The Weeks Interval specifies the number of weeks between each execution. The Days of the Week field specifies on which days the execution occurs. For example: To run a job at noon every other week on Mondays, Wednesdays, and Fridays, set the Begin Date to the desired date, the Start Time and End Time to 12:00 P.M., the Interval in Minutes to 1, the Weeks Interval to 2, and the Days of the Week to Monday, Wednesday, and Friday.

Table 6–9 (Cont.) Job Manager Recurrent Trigger Types

Trigger Type	Description and Example
Monthly by Date	Runs a job on specific days of the month. The Months field specifies in which months this job executes. The Days field specifies which days of those months. If the given day does not exist for a given month, then that day is ignored. For example: To run a job at 5:00 P.M. on the 1st and 15th of January, February, and March, set the Begin Date to January 1, the Start Time and End Time to 5:00 P.M., the Interval in Minutes to 1, the Months to January, February, and March, and the Days to 1 and 15. To run a job at 2:00 A.M. on every leap day (February 29th), set the Begin Date to January 1, the Start Time and End Time to 2:00 A.M., the Interval in Minutes to 1, the Months to February, and the Days to 29.
Monthly by DOW (Day of Week)	Runs a job on specific occurrences of specified days of the week during given months. The Months field specifies which months this job executes. The Days of the Week field specifies which days of the week the job executes during those months. The Occurrence field specifies which of those days to execute. The occurrence can be any or all of First, Second, Third, Fourth, and Last. The Last value specifies that either the fourth or fifth occurrence of a given day is used, depending on whether there are four or five occurrences during that month. For example: To run a job on the first and third Fridays of December every hour between the hours of 4:00 A.M. and 8:00 P.M., set the Begin Date to the desired date, the Start Time to 4:00 A.M., the End Time to 8:00 P.M., the Interval in Minutes to 60, the Months to December, the Days of the Week to Friday, and the Occurrence to the First and the Third. To run a job at 3:00 A.M. every time that Daylight Saving Time switches over to Standard Time, set the Begin Date to the desired date, the Start Time and End Time to 3:00 A.M., the Months to October, the Days of the Week to Sunday, and the Occurrence to Last.

A

Setting Up the SA System Subject Area

This appendix describes setting up the SA System subject area to retrieve user information from a database and populate user profile information. Oracle BI Presentation Services delivers alerts from Oracle BI Delivers to specified e-mail addresses, phone numbers, and so on. These delivery destinations are stored in the Oracle Business Intelligence profile for each user. In some cases, you might want to automatically populate the phone numbers or e-mail addresses in user profiles.

If you must automate only the population of e-mail addresses in user profiles, then you should populate the e-mail address field for users in your LDAP server or other authentication provider, if possible. These values are used to populate the e-mail address in Oracle Business Intelligence user profiles, enabling users to receive content from Delivers, even if they have not signed into Oracle Business Intelligence. This feature works for any LDAP server that has a mail attribute for its users.

In some cases, however, you might want to automatically populate additional user profile options, in addition to e-mail addresses. For example, you might want to automatically populate a cell phone number as part of the user profile information, if you want Delivers to deliver a format suitable for a cell phone (like text) using an e-mail gateway.

In this situation, you can configure a special subject area in the repository called SA System that retrieves user information from a database and populates the user profile information. This appendix explains how to configure and use the SA System subject area to accomplish this task.

If you choose to use the SA System subject area, then you should discourage users from configuring delivery profiles on their own. By default, values that are specified in delivery profiles take precedence over values that are shown in the SA System subject area.

This appendix contains the following topics:

A.1 About the SA System Subject Area

In previous releases of Oracle Business Intelligence, **SA System** was a subject area that exposed group membership to Delivers and enabled contact information, such as e-mail addresses to be retrieved from a database and used as delivery devices in Delivers. The SA System subject area feature automatically populated delivery devices

and profiles for users instead of requiring users to update their My Account screen in Delivers. The SA System subject area provided the users associated with each group and external e-mail addresses to Delivers.

In this release of Oracle Business Intelligence, Delivers still must determine group and role membership so that it can appropriately deliver alerts. Typically, however, your LDAP identity store is now the source of group and role membership. If SA System is defined and enabled, then membership of application roles and catalog groups is derived from the SA System subject area in Delivers. The names of the applications roles and catalog groups that are selected in an agent are used to determine group membership in the SA System subject area.

Note that you do not need SA System if you are using an LDAP server and you must populate only user profile e-mail addresses. The recommended best practice for populating e-mail addresses in user profiles is to use the mail attribute in your LDAP server. Because most portable devices can read e-mail directly, specific text or SMS formats are often not required for agent delivery, and populating e-mail addresses from LDAP is usually sufficient.

Also note that you do not need to use SA system to use the feature **Get Recipients from the Analysis Used in the Agent Condition**. Instead, this feature is used when the recipients can be determined from the query results and the data to be delivered is specific to those users.

Note that it is possible to configure initialization block-based user authentication using the tables in SA System as a source for the user population. Using the SA System data in this way is separate from using SA System to populate delivery profiles. Rather, these are independent functions that happen to be based on the same user source data.

A.1.1 About Group and Application Role Resolution

In this release of Oracle Business Intelligence, application roles are used to define security policies rather than groups. When you create an agent, you can choose whether it should be delivered to a user, an application role, or a Catalog group.

However, to maintain backward compatibility with previous releases, SA System still uses the Group Name column in the SA System source table to determine the e-mail addresses for the application roles and catalog groups that are specified for agents. Because of this, the SA System subject area functions the same as it did in previous releases, even though the Oracle Business Intelligence security model has changed significantly in the current release.

Because the group membership in SA System is used to determine the list of recipients rather than the membership of either application roles or Catalog groups, users should not add members to Catalog groups. Alternatively, administrators can synchronize the application role and Catalog group memberships with SA System whenever the memberships are updated.

A.2 Setting Up the Data Source for the SA System Subject Area

In your external data source, create a table called SA_SYSTEM_USERS that contains columns that correspond to the various delivery options. In addition, you must ensure that every user and group is present in the data.

Table A–1 shows the columns that are required for the SA_SYSTEM_USERS table. You must create the columns that are listed in the order shown. Any external schema that has the information in this table can be mapped to the SA System subject area.

Table A–1 Columns in the SA_SYSTEM_USERS Table

Column	Data Type	Description
LOGON	VARCHAR (50)	The unique user ID of the user that logs on to the system. This cannot be null.
DISPLAY_NAME	VARCHAR2(100)	The full name of the user. This can be null.
GROUP_NAME	VARCHAR2 (20)	The name of the group to which this user belongs. If a user belongs to multiple groups, then there should be one row for each group in the SA System table. This should not be null if any data access security is based on group membership.
TIMEZONE	VARCHAR2 (100)	This column is currently not used and exists for future use. This should be null.
LANGUAGE	VARCHAR2 (20)	This column is currently not used and exists for future use. This should be null.
LOCALE	VARCHAR2 (20)	This column is currently not used and exists for future use. This should be null.
EMAIL	VARCHAR2 (100)	The primary e-mail address for the user. This is a complete SMTP address such as joe.perez@example.com. This can be null.
EMAIL_PRIORITY	VARCHAR2 (10)	This determines when an alert is delivered to this device. The value can be any combination of the three priorities of an agent: H for high priority, N for normal priority, or L for low priority. For example, if high, normal, and low priority alerts are to be delivered to this device, then the field should be HNL. If only high and normal priority alerts are to be delivered, then the field should be HN. This field should not be null if the Email column is specified. This can be null if Email is null.
EMAIL_TYPE	VARCHAR2 (50)	This field can be one of two text strings, 'html' or 'text'. Because most primary e-mail clients can read rich MIME content (HTML with embedded images), the value 'html' is usually the best choice. Choose the value 'text' to support legacy e-mail clients that can read only plain text e-mail. This field should not be null if the Email column is specified. This can be null if Email is null.
CELL_PHONE	VARCHAR2 (40)	This field is the complete SMTP address for the cell phone device that receives text message alerts. For example, 1015551234@cellphoneprovider.com. Only text messages are sent to this device. This can be null.

Table A–1 (Cont.) Columns in the SA_SYSTEM_USERS Table

Column	Data Type	Description
CELL_PHONE_ PRIORITY	VARCHAR2 (20)	This determines when an alert is delivered to this device. The value can be any combination of the three priorities of an agent: H for high priority, N for normal priority, and L for low priority. This field should not be null if the Cell Phone column is specified. This can be null if Cell Phone is null.
PAGER	VARCHAR2 (20)	This field is the complete SMTP address for the pager device that receives text message alerts. For example, 1015555678@pagerprovider.com. Only text messages are sent to this device. This can be null.
PAGER_PRIORITY	VARCHAR2 (30)	This determines when an alert is delivered to this device. The value can be any combination of the three priorities of an agent: H for high priority, N for normal priority, and L for low priority. This field should not be null if the Pager column is specified. This can be null if Pager is null.
HANDHELD	VARCHAR2 (20)	This field is the complete SMTP address for the handheld device that receives text message alerts. For example, joe.perez@handheldprovider.com. Only text messages are sent to this device. This can be null.
HAND_HELD_ PRIORITY	VARCHAR2 (30)	This determines when an alert is delivered to this device. The value can be any combination of the three priorities of an agent: H for high priority, N for normal priority, and L for low priority. This field should not be null if the Handheld column is specified. This can be null if Handheld is null.

You can use the following sample SQL statement to create the SA_SYSTEM_USERS table shown in Table A–1:

```
CREATE TABLE SA_SYSTEM_USERS
(
LOGON VARCHAR2(50) NOT NULL,
DISPLAY_NAME VARCHAR2(100),
EMAIL VARCHAR2(100),
EMAIL_PRIORITY VARCHAR2(10) DEFAULT 'HNL',
EMAIL_TYPE VARCHAR2(50) DEFAULT 'html',
CELL_PHONE VARCHAR2(40),
CELL_PHONE_PRIORITY VARCHAR2(20),
PAGER VARCHAR2(20),
PAGER_PRIORITY VARCHAR2(30),
HANDHELD VARCHAR2(20),
HANDHELD_PRIORITY VARCHAR2(30),
TIMEZONE VARCHAR2(100),
GROUP_NAME VARCHAR2(20),
LOCALE VARCHAR2(20) DEFAULT 'en',
LANGUAGE VARCHAR2(20) DEFAULT 'en',
PRIMARY KEY (LOGON)
);
```

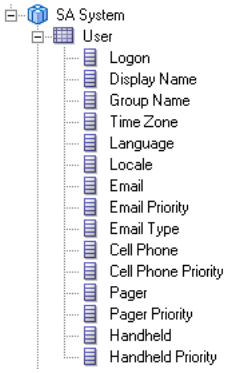
A.3 Importing SA System Data Into the Repository

After you configure the external data source, you must create and build the subject area in the Oracle BI repository. To do this, you first import the SA_SYSTEM_USERS table from the data source into the Physical layer. Then, map the SA_SYSTEM_USERS table and columns from the Physical layer to the Business Model and Mapping layer. Finally, map the SA_SYSTEM_USERS table and columns from the Business Model and Mapping layer to the Presentation layer. The name for the subject area must always be SA System.

See *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition* for information about importing data into the repository and modeling information in the Business Model and Mapping layer and Presentation layer.

Figure A–1 shows the table and columns in the Presentation layer of the repository.

Figure A–1 SA System Subject Area in the Presentation Layer



A.4 Setting Configuration Options for the SA System Subject Area

You can control the availability of the delivery options that are configured in the SA System subject area and the user-defined delivery options by including certain elements in the Oracle BI Presentation Server version of instanceconfig.xml file. These elements take effect only if the SA System subject area is being used.

Before you begin this procedure, ensure that you are familiar with the information in "Using a Text Editor to Update Oracle Business Intelligence Configuration Settings" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

To change configuration options for the SA System subject area:

1. Open the Oracle BI Presentation Server version of instanceconfig.xml file for editing, as described in "Where are Configuration Files Located?" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*
2. Locate the sections in which you must add the following elements:
   - SystemSubjectArea: Specifies whether to recognize the delivery devices and deliver profiles that are configured in the SA System subject area:
     - **true**. Recognizes the delivery devices and delivery profiles that are configured in the SA System subject area and displays them on the My Account page. (Default)

  - **false**. Ignores the delivery devices and delivery profiles that are configured in the SA System subject area and does not display them on the My Account page.

  Include this element within the SubjectAreaMetadata element, which is itself included in the ServerInstance element. The values true and false are case sensitive.

- IgnoreWebcatDeliveryProfiles: Specifies whether to ignore user-defined delivery devices and deliver profiles:
  - **true**. Ignores the user-defined delivery devices and delivery profiles and does not display them on the My Account page. (This means that users cannot create new delivery devices and delivery profiles.)
  - **false**. Recognizes the user-defined delivery devices and delivery profiles and displays them on the My Account page. (Default)

  Include this element within the Alerts element, which is itself included in the ServerInstance element. The values true and false are case sensitive.

- UpperCaseRecipientNames: Specifies that only users whose user names are uppercase can have agents delivered to them.

  For example, suppose that you have users with names of user_lowercase and USER_UPPERCASE. If you set the UpperCaseRecipientNames element to true, then agents are sent only to USER_UPPERCASE.

  Include this element within the Alerts element, which is itself included in the ServerInstance element.

3. Include the elements and their ancestor elements as appropriate, as shown in the following example.

```
<ServerInstance>
  <SubjectAreaMetadata>
    <SystemSubjectArea>true</SystemSubjectArea>
  </SubjectAreaMetadata>
   <Alerts>
     <IgnoreWebcatDeliveryProfiles>false</IgnoreWebcatDeliveryProfiles>
     <UpperCaseRecipientNames>true</UpperCaseRecipientNames>
   </Alerts>
</ServerInstance>
```

4. Save your changes and close the file.
5. Restart Oracle Business Intelligence.

A.4.1 Managing the Case of Login Names for the SA System Subject Area

When the SA System subject area is used, login names are compared to the LOGON column in the SA System subject area. By default, this comparison is case-insensitive. This means, for example that an LDAP username of "Fred" will match an SA System subject area entry where LOGON is "fred". The only exception to this is when you set UpperCaseRecipientNames element in the instanceconfig.xml file to 'true' to ensure delivery only to users whose names are uppercase. If needed, set the LOGON value in the SA_SYSTEM_USERS table to uppercase by setting it within the Upper() function.