HYPERION REPORTING AND ANALYSIS FRAMEWORK

RELEASE 11.1.2.0

DEVELOPER'S GUIDE

ORACLE®

**ENTERPRISE PERFORMANCE
MANAGEMENT SYSTEM**

Reporting and Analysis Framework Developer's Guide , 11.1.2.0

Contents

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Part I

API Documentation

In API Documentation:

- Getting Started with Java APIs
- Batch Driver Sample Program Control File
- EPM Workspace Artifacts
- Sample Java Programs

1

Getting Started with Java APIs

Note: Using APIs to enhance, extend, or customize Oracle Enterprise Performance
Management Workspace, Fusion Edition is considered nonstandard and is not supported
by Oracle Customer Support.

Basic Steps

Using Oracle's Hyperion Reporting and Analysis services from a Java application program
requires only a few basic steps:

1. Import Reporting and Analysis and EPM WorkspaceSoftware Development Kit (SDK)
    package into the program by using an import statement.

2. Obtain a Session interface. This process authorizes the account and provides access to all
    Reporting and Analysis services.

3. Invoke Reporting and Analysis services from a Java program.

Understanding Hyperion Home and Install Home

When multiple Oracle Oracle products are installed on one computer, common internal and
third-party components used by the products are installed to a central location, called *Hyperion
Home*. The Hyperion Home location is defined in the HYPERION_HOME system environment
variable.

The default location for Hyperion Home is `C:\Hyperion` (Windows), or *user_home*/
`Hyperion` (UNIX). (During installation, the installer searches for the HYPERION_HOME
environment variable on the installation computer. If it exists, the installer uses the previously
defined location).

The default installation location (*Install Home*) for Reporting and Analysis is *Hyperion Home*
`\products\Foundation\workspace` (Windows), or *Hyperion Home*/products/
`Foundation/workspace` (UNIX).

Importing the Reporting and Analysis and EPM Workspace SDK Package

Import statements enable the Java compiler to find all class definitions referenced by a program. To access the classes and methods provided by the Reporting and Analysis and EPM Workspace SDK package, include this statement in the source code for your Java program:

```
import com.sqribe.rm.*;
```

This statement defines each of the interfaces and their implementations that your program uses during the course of its execution. All classes and interfaces in this package are bundled in `<EPM_ORACLE_HOME>/common/raframeworkrt/11.1.2.0/lib/rmapi.jar` file. This `.jar` file has dependencies on several other `.jar` files as specified in `<MIDDLEWARE_HOME>/user_projects/epmsystem1/bin/ReportingAnalysis/SDK/ set_sdk.env (.sh)` file. See Appendix A, "How to Use the API Sample Programs.".

Obtaining a Session Interface

The first step in establishing contact with Reporting and Analysis and EPM Workspace services is to obtain a Session interface by using a static `getInstance()` method of the Oracle's Hyperion® Interactive Reporting SDK `SessionFactory` class. This class validates the login information that you pass from your program to EPM Workspace and subsequently passes back to your program the `Session` interface that represents the connection between EPM Workspace and your program. This code fragment demonstrates how to use Reporting and Analysis and EPM Workspace SDK to connect to EPM Workspace:

```
try
{
String user = args[0];
String pwd = args[1];
String host = args[2];
String port = args[3];
Session theSession = SessionFactory.getInstance(user, pwd, host, port);
…
…
}
catch (ReportMartException e)
```

Invoking Reporting and Analysis and EPM Workspace Services

After you establish the connection to EPM Workspace and obtain a Session interface, use the returned *Session* to obtain access to the other interfaces that provide access to Reporting and Analysis services. For example, a highly used interface is the `Repository` interface, which provides various methods for obtaining documents, jobs, reports, links and other entities. Your program can obtain the `Repository` interface as follows:

```
try
{
String user = args[0];
```

```
String pwd = args[1];
String host = args[2];
String port = args[3];
Session theSession = SessionFactory.getInstance(user, pwd, host, port);
Repository theRepository = theSession.getRepository();
}
catch (ReportMartException e)
```

Java Exceptions

Most method calls to classes in the `com.sqribe.rm.*` package throw a
`ReportMartException`, which is an Exception class derived from the standard
`java.lang.Exception` class. Throwing exceptions is the normal mechanism used in the APIs
to communicate a negative result from a method invocation. To avoid providing countless
subclasses of `ReportMartException` to cover all possible failure scenarios, the
`ReportMartException` classes are limited, with most subclasses related to login processing:

- **UnknownReportMartException**: Thrown when the host name cannot be resolved or when
  no Service Agent is running on the host

- **UserValidationException**: Thrown when the user ID or password that is provided is invalid

All calls to API methods must be enclosed in a `try/catch` block, as seen in this example code
that shows how to log on to EPM Workspace:

```
import com.sqribe.rm.*;
public class TestSDK
{
static Session theSession = null;
public static void main(Strings [] args)
{
// assume that args contains username, password and host
try
{
theSession = SessionFactory.getInstance(args[0], args[1], args[2]);
...
... your code goes here
...
}
catch (UnknownReportMartException e1)
{
// unable to connect to specified host
System.out.println(e1.getMessage());
}
catch (UserValidationException e2)
{
// invalid account or password
System.out.println(e2.getMessage());
}
}
}
```

Java-Accessed Functions

Java programs can take advantage of Reporting and Analysis services to access various EPM Workspace functions, depending on the privileges granted by the administrator to the account used to connect to EPM Workspace.

These examples identify the functions that can be accessed by a Java API program:

- Creating or deleting EPM Workspace folders or hierarchies of EPM Workspace folders and copying data files between folders and the local file system

- Importing, or modifying properties for documents or jobs

- Creating, modifying, or deleting resources, such as printers or output directories

2 Batch Driver Sample Program Control File

About the Batch Driver Control File

The batch driver control file contains the data that the batch driver program processes. The batch driver program reads a batch driver control file and processes the data in it. A sample batch driver control file, `batch_driver.cf` is in the directory that contains the sample program source files. The sample file shows the syntax for all artifacts that can be bulk loaded using the batch driver sample program.

This single line control statement is an example from the `batch_driver.cf` file that adds a data artifact. It assumes that the artifacts accessor1 and accessor2 exist in the system.

```
data name=aDocument^path=/batchFolder^file=D:\Doc
\bart.doc^browse=true^autodelete=true^desc=A test document by
batchdriver^expire=2005-06-22 12:00:00^keyword=MS doc^keyword=Design
doc^perm=accessor1^perm=accessor2.
```

Syntax for batch driver control files:

- All control statement parameters must be included on one physical line. Each line in the file is read and executed one at a time.

- Control files can contain zero or more control statements.

- Comments are supported and must be entered on their own lines. Comment lines begin with these symbols: number sign (#), two slashes (//), or an exclamation point (!).

- Blank lines are comments.

- Each attribute `name-value` pair is separated by the caret symbol (^).

- No spaces can occur between the attribute name, the equal sign (=), and the attribute value of the statement parameters.

- A value can contain one or more blanks. Attribute parameters can be specified in any order after the control verb.

Control Statements

Control statements are used to specify artifact attributes that you want to add to EPM Workspace using the batch driver sample program. Control statements include a starting term that denotes control line type and attribute-value assignment statements. The terms and attribute names are not case-sensitive, but the values assigned to attributes are case-sensitive.

These tables list the control statement types that are supported by the BatchDriver program and the available attributes. Although the syntax descriptions in the following topics may show the attribute values on multiple lines, in practice all must be specified on one control line.

Control Statement	Description	Reference
accessor	Define an accessor for access control (permission)	"Accessor Control Statements" on page 19
category	Define category attributes	"Category Control Statements" on page 19
data	Add a data artifact	"Data Control Statements" on page 19
oce	Add an Interactive Reporting database connection	"Interactive Reporting Database Connection Control Statements" on page 20
bqydoc	Add an Interactive Reporting document	"Interactive Reporting Control Statements" on page 21

Note: Each attribute marked by an asterisk (*) can be specified multiple times. In statements where the attribute value is specified as choice1 || choice2, one of the specified values may be provided. Assigned attribute values are interpreted as shown in this table.

Attribute	Description
date	YYYY-MM-DD_HH:MM:SSZZZ where ZZZ is an optional time zone offset from GMT, for example, PST is '-08'
name	Name string uniquely identifying an artifact
int	Integer value
string	String value; for example, e-mail address
path	Folder in the repository to which the artifact is imported
id	Unique non-blank string value used to reference a control statement
dref	Reference to the control statement in which <id> value was used
boolean	Value TRUE or FALSE
datasource	Repository database, Oracle's Hyperion® SQR® Production Reporting combination
ifile	File name on local file system

Accessor Control Statements

Accessor control statements create accessor artifacts in the cache, but accessor artifacts are not stored in the database separately. Accessor artifacts are used in one or more artifacts as the value for the permission (perm) attribute and thus defines access control for other artifacts. Assessor *ids* are used to reference accessor artifacts when defining access control for other artifacts. A user, group, or role must be in the system before it is assigned as an accessor.

Example

```
accessor id=accessor1^type=group^name=world^sysrole=FULL CONTROL
```

Attribute Name	Attribute Type	Accessor Control Statement Description
id	id	Accessor identifier in this program; not the objectID or uuid
type	user | group | brole	Accessor type—user, group, or business role
name	name	Name of the user, group, or business role
sysrole	role name	Name of the system role to be associated

Category Control Statements

Category control statements create categories or update category properties. The value passed to the perm attribute is the name of an accessor artifact that was previously defined in the control file and assigned a name; for example: accessor1, accessor2.

Example

```
category path=/batchFolder^browse=true^autodelete=true^desc=A test Folder by
batchdriver^expire=2005-06-20 12:00:00^perm=accessor2
```

Attribute Name	Attribute Type	Category Control Statement Description
path	path	Category path
browse	boolean	Browsable
autodelete	boolean	Autodelete
desc	string	Description of the category
expire	date	Expiration date
perm	accessor *	Access control—other users, groups, and roles privileged to access this folder

Data Control Statements

Data control statements import files to the repository from specified locations on your desktop. The resulting data artifact is loaded into the specified folder and is assigned the attributes specified by the parameters.

The value passed to the `perm` attribute is the name of an accessor artifact that was previously defined in the control file and assigned a name; for example: accessor1, accessor2.

Example

```
data name=aDocument^path=/batchFolder^file=D:\Doc\bart.doc ^desc=A test document by
batchdriver ^browse=true^autodelete=true^expire=2004-06-22 12:00:00^keyword=MS
doc^keyword=Design doc^perm=accessor2^perm=accessor4
```

Attribute Name	Attribute Type	Data Control Statement Description
data	name	Name in the repository
path	path	Folder in the repository to which the artifact is imported
file	filename	Name of the file on the file system
desc	string	Description of the data name
browse	boolean	Browsable
autodelete	boolean	Autodelete
expire	date	Expiration date
keyword	string *	Keyword strings
perm	accessor *	Access control—other users privileged to access this artifact

Interactive Reporting Database Connection Control Statements

Interactive Reporting database connection control statements load Interactive Reporting database connections into the repository from the specified file source from specified locations on your desktop. The resulting Interactive Reporting database connections are loaded into specified folders and are configured according to the attributes specified by the parameters.

Example

```
oce name=anOCEDocument^path=/batchFolder^file=D:\Foundation\BQYFiles\odbc.oce^desc=A
test Interactive Reporting database connection by batchdriver
^prompt=false^dbuser=guest^dbpass=guest
^metaoce=Cqdb.oce^metaUseThis=false^metauser=guest
^metapass=guest^allowSSO=true^browse=true^autodelete=true ^expire=2004-06-24 12:00:
00^keyword=OCE doc^keyword=Connection file^perm=accessor2^perm=accessor4
```

Attribute Name	Attribute Type	Interactive Reporting Database Connection Control Statement Description
id	string	Interactive Reporting database connection identifier, usually the name (not objectID/uuid); this is optional and may be omitted if the Interactive Reporting database connection is not used in this run of the program
name	name	Name of the Interactive Reporting database connection in the repository

Attribute Name	Attribute Type	Interactive Reporting Database Connection Control Statement Description
path	path	Folder in the repository to which the artifact is imported
file	filename	File name to load
desc	string	Description of the Interactive Reporting database connection
prompt	boolean	Whether to prompt the user
dbuser	string	If prompt is false, the user for database connectivity
dbpass	string	If prompt is false, the password for database connectivity
metaoce	oce name	If there is a meta Interactive Reporting database connection, specify its name
metaUserThis	boolean	Whether to use the Interactive Reporting database connection connectivity information for the meta Interactive Reporting database connection
metauser	string	If metaUseThis is false, the user for meta Interactive Reporting database connection database connectivity
metapass	string	If metaUseThis is false, the password for meta Interactive Reporting database connection database connectivity
browse	boolean	Browsable
autodelete	boolean	Auto delete on expire
keyword	string *	Keyword strings
expire	date	Expiration date
perm	accessor *	Access control—other users privileged to access this artifact

Interactive Reporting Control Statements

Interactive Reporting control statements load Interactive Reporting documents into the repository from specified file sources from specified locations on your desktop. The resulting Interactive Reporting documents are loaded into specified folders and are configured according to the attributes specified by the parameters.

Example

```
bqydoc name=aBQYDocument^path=/batchFolder^file=Simple.bqy^desc=A test Interactive
Reporting document by batchdriver^desc=A test Interactive Reporting document by
batchdriver^ihtml=true^oce=anOCEDocument^oceprompt=2
^dbuser=guest^dbpass=guest^pregen=2^browse=true^autodelete=true^expire=2005-06-24 12:00:
00^keyword=BQY doc^keyword=CQ 22535^perm=accessor1^perm=accessor3
```

Attribute Name	Attribute Type	Interactive Reporting Control Statement Description
name	name	Name of the Interactive Reporting document in the repository

Attribute Name	Attribute Type	Interactive Reporting Control Statement Description
path	path	Folder in the repository to which the artifact is imported
file	filename	File name to load
desc	string	Description of the Interactive Reporting Control Statement
ihtml	boolean	Whether to enable the Interactive Reporting document for use in HTML
oce	idref||oce name	Interactive Reporting database connection ID or name;<oce name> may be used if the Interactive Reporting database connection is imported beforehand For example, not in this run of this program
prompt	boolean	Whether to prompt the user for the Interactive Reporting database connection by using the prompt
dbuser	string	If prompt is false, the user for database connectivity
dbpass	string	If prompt is false, the password for database connectivity
pregen	int	HTML pregeneration options for the Interactive Reporting documents: ● 0: Pregenerate ALL sections ● 1: Pregenerate no sections ● 2: Pregenerate selected sections
browse	boolean	Browsable
autodelete	boolean	Auto delete on expire
keyword	string *	Keyword strings
expire	date	Expiration date
perm	accessor *	Access control—other users privileged to access this artifact

3

EPM Workspace Artifacts

Interfaces

You can use these interfaces to create Java application programs that interact with EPM Workspace:

AbsoluteTimeEvent	CustomCalendar	OCEDocument	Role
Authorization	DataObject	ParameterList	ScheduledTask
BaseObject	ExternallyTriggered Event	PhysicalResource	Scheduler
BQYDocument	Group	Query	Session
BQYJob	InstancePermission	QueryVector	SPFSet
Category	Job	RecurringTimeEvent	SQRJob
Collection	JobOutput	ReportMartEntity	SQRJobOutput
	ObjectID	Repository	User

AbsoluteTimeEvent

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is used to create an event that is triggered at a given time. This event is a nonrecurring, one-time event which occurs only once at the specified point in time. This is an unnamed event type. APIs modify and retrieve various event properties.

Authorization

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface provides methods used to obtain or modify information about some entities stored in the Authorization System. Methods are provided for accessing roles and for retrieving and listing system or business roles.

BaseObject

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface extends `ReportMartEntity` and provides additional methods used to access additional data artifact attributes stored in EPM Workspace. These additional attributes are infused to obtain or modify information about artifacts stored in EPM Workspace. Methods enable specifying and querying these attributes associated with a `BaseObject`:

- Custom property values that are specified and saved with the artifact in EPM Workspace
- Artifact type or metatype for the artifact
- Artifact ownership
- Automatic artifact deletion when an expiration date is assigned
- Whether the artifact can be browsed

BQYDocument

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface provides methods to get information regarding Interactive Reporting documents. It provides a mechanism to get Interactive Reporting database connection mappings for the Interactive Reporting document Query sections and enables you to set the Interactive Reporting database connections for the Query sections. It also supports methods to check and set the Interactive Reporting document iHTML rendering.

BQYJob

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface extends the `Job` interface. It enables you to set and update actions associated with an Interactive Reporting job. Methods are available for setting cycles and actions associated with those cycles. It is possible to set the Interactive Reporting job to run in the foreground or background and to assign a default calendar to be associated with this job by using the methods included for this interface. New classes (such as `BQYOLAPParameter`, `BQYOLAPSlicerParameter`, and so on) support OLAP and OLAP Slicer parameters for Interactive Reporting jobs. The `BQYParameter class` and the `BQYSectionInfo` interface provide new methods to support variable limit parameters containing complex sections (union subquery and master data model).

Category

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface provides methods that enable your program to copy files and the directory contents to and from the local file system, automatically creating the corresponding *Category* hierarchy in EPM Workspace as needed. Methods are also provided to enable recursive deletion of categories and their contents.

Collection

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface provides utility methods similar to a `Vector` that enables you to access artifact sets. These `Collection` interfaces are commonly associated with the output from job executions.

CustomCalendar

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is used to define a calendar specific to the business, such as fiscal and manufacturing. `CustomCalendar` and `CalendarYear` provide a structure to define a customizable calendar. Different custom calendar types could be defined including internal, external, and default. Only one default calendar can be defined and it is constructed by the system. `CustomCalendar` contains a calendar year list and information about nonworking days. A calendar year contains information about a custom year, such as the period limits and quarters limits. Calendar years are defined only for internal calendars.

DataObject

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface defines additional methods that can be invoked on artifacts that are stored in EPM Workspace, such as HTML documents, Production Reporting documents, and comma-separated value (CSV) data files. The interface enables your program to obtain keyword lists associated with the artifact, to update the `DataObject` content, and to retrieve contents from the local file system.

ExternallyTriggered Event

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface defines an event that is triggered by an external action, which is a two-stage process. When this event is triggered, it is set to run immediately. Externally Triggered Events are always PUBLIC events.

Group

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface provides methods used to access the Group artifact attributes defined in Oracle's Hyperion® Shared Services. Using this interface, your program can invoke methods that enable you to add a member to the group, to discover the roles that are group members, or to delete the group.

InstancePermission

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

The system automatically creates this interface including default access control when EPM Workspace creates an artifact.

The default access control grants full control to the artifact owner. This interface is equivalent to the `Permissions` class in Brio Portal and is deprecated in EPM Workspace. The interface is obtained by invoking the `getInstancePermission()` method on an artifact. After retrieval, it can be modified to enforce new access control on it. It provides necessary methods to grant different levels of access control for different users, groups, and business roles.

Two other interfaces, `RoleAccessor` and `Role`, are very closely related to this interface. For example, here are the steps to grant the MODIFY role to User1 on an artifact doc1:

- Retrieve the instance permission associated with the artifact by invoking the `doc1.getInstancePermission()` method.
- Create a `RoleAccessor` artifact using the `createRoleAccessor()` method in the `Authorization` interface by sending User1 as a parameter.
- Retrieve the role artifact for MODIFY role using the `getRoleByName()` method of the `Authorization` interface and passing MODIFY as the parameter.
- Add this role to the roles accessor created previously by invoking the `addSystemRole()` method on the role accessor artifact.
- Add the role accessor to the instance permission artifact using the `addRoleAccessor()` method on the instance permission artifact.
- Use the `update()` method on the instance permission artifact.

Job

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface executes predefined jobs, submits the jobs to the Job Service, and retrieves the data artifacts that are generated by job execution. By using this interface, your program can alter runtime parameters prior to running the job. It can also query the job for information about databases used, the output artifact life span, and many other job attributes. Methods exist that support job execution asynchronously in the background and get execution status.

JobOutput

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is generated by the job execution by Job Service. After the job executes, your program may extract a `JobOutput` interface from the Job, and using the methods in `JobOutput`, can extract all output data artifacts that were generated during job execution.

ObjectID

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

Each artifact stored in EPM Workspace contains a unique identifier that represents the artifact. Whereas multiple artifacts can be stored in a folder with the same name and metatype, they are distinguished from each other by the universally unique identifier (UUID) assigned to the artifact. This interface represents the UUID when invoking Interactive Reporting SDK methods that require an artifact ID.

OCEDocument

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is the artifact wrapper around the Interactive Reporting database connection that is imported to the repository. It provides methods to retrieve or set properties for the Interactive Reporting database connection, such as the default database user name and password to be used. It also provides methods to set the Interactive Reporting database connection to prompt users for the database user name and password and to enable the single sign-on feature for this Interactive Reporting database connection.

ParameterList

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface encapsulates `JobDef` artifact vectors and other attributes controlling the job execution within the cluster. In addition to the `JobDef` vector, the `ParameterList` interface maintains a hold feature that enables the user to suspend execution of all scheduled tasks associated with this `ParameterList` artifact until the hold mechanism is turned off. There are three `ParameterList` view types: public, personal, and unnamed. The default type is personal.

PhysicalResource

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface defines methods common to printer and output directory resources in EPM Workspace. Two physical resource artifacts types are extended from the `PhysicalResource` interface:

- `PrinterPhysicalResource`

- `OutputDirPhysicalResource`

Physical resource artifacts are created using methods in the `Repository` interface. They are addPrinterResource() and addOutputDirectory() for importing a printer and an output directory, respectively. Two `OutputDirPhysicalResource` artifact types can be created: simple output directory and an FTP output directory. `PrinterPhysicalResource` and `OutputDirPhysicalResource` define methods to modify physical resource artifact properties.

Query

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

Your program obtains this interface using the `Repository` interface when the program wants to search EPM Workspace for data artifact sets that match filter specifications. The `Query` interface defines methods that enable you to obtain lists of groups and users. It also provides other more generic methods that enable searching by artifact name, by keywords, or by a list of other artifact attributes, such as creation and access dates, artifact type, or artifact ownership.

QueryVector

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is returned from search methods supported by the `Query` interface. `QueryVector` provides access functions that enables your program to recover elements returned from the search method invocation. The interface enables the program to discover the number of elements returned by the search, to determine whether the returned element list is empty, to access each element in the `QueryVector` interface by a numeric index value, or to obtain an `Enumeration` interface for more comprehensive manipulation of the artifact sets returned by the search request.

RecurringTimeEvent

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

Use this interface for repetitive event triggering by specifying the date and time in various ways. There are three recurring time event types: public, personal, or unnamed. The default view type of a Recurring Time Event is personal. An unnamed event is a per-schedule event. It is valid only for the schedule for which it is created, and is deleted as soon as the schedule is deleted or the schedule is updated with another event. You can also create events that are a combination of Recurring Time Events and Externally Triggered Events.

ReportMartEntity

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface provides access to artifact attribute information that is common to all artifacts stored in EPM Workspace, except for session-related data artifacts. It is used by an application to obtain or set basic attributes related to identifying an artifact in EPM Workspace, and for finding out certain key attributes such as these:

- The various timestamps associated with an artifact, such as its creation or last modification date
- Information related to artifact identity, such as the name and description

Repository

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface provides access to utility methods in the Reporting and Analysis and EPM Workspace SDK. It is used to obtain most of the major interfaces through which your program accesses the Reporting and Analysis services.

Role

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface provides methods used to access the Role artifact attributes defined in Shared Services. Using this interface, your program can invoke methods that enable you to manipulate roles.

ScheduledTask

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is used to associate a parameter list with an event. In addition to the parameter list and event, this artifact also contains additional properties to control how the parameter list is executed.

Scheduler

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

Use this interface to access Event Service to create, retrieve, list, and delete these artifacts:

- AbsoluteTimeEvent
- RecurringTimeEvent
- ExternallyTriggeredEvent
- ParameterList
- CustomCalendar
- CalendarYear
- ScheduledTask

Session

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is the primary interface through which your program accesses EPM Workspace. It is returned to your program when it successfully logs in to EPM Workspace using a `SessionFactory.getInstance()` methods

SPFSet

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface represents the output from Job Service execution of Production Reporting documents. This interface enables you to access the various output types, such as HTML, postscript, or other output formats that were generated by executing the Production Reporting document.

SQRJob

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is a `Job` interface extension that provides additional functionality supported by executing and processing Production Reporting documents in Job Service. Additional methods are available in this interface to set and retrieve the ask parameters that are processed at the time of job execution, and to obtain the program output data artifacts generated during Production Reporting document execution.

SQRJobOutput

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is returned by the `getProgramOutputs()` method of `SQRJob` interface. This extended interface enables your program to obtain the data artifacts generated by executing a Production Reporting document, and the `SPFSet` interface that enables the program to obtain the various listing data artifacts.

User

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This interface is returned from various SDK method calls to represent a user account defined in Shared Services. This interface enables your program to obtain various account attributes such as group membership, default category, and default permissions. In addition, methods are provided to add the user to new groups, set new default permissions, descriptions, and other attributes, and to access and modify single sign-on properties for a User artifact.

Classes

You can use these classes to create Java application programs that interact with EPM Workspace:

JobParameter	SessionFactory
LoggerHelper	UnknownReportMartException
ObjectType	UserValidationException
ReportMartException	

JobParameter

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

The Interactive Reporting SDK generates this class when you invoke the `setParameters()` method on a `Job` interface. This class provides information about data parameters passed to a job executing at runtime.

LoggerHelper

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This class provides methods to enable clients to log their messages using ODL logging architecture. The client acquires a `LoggerHelper` artifact instance in every source file in which it wants to log messages. The client uses the static `getLogger()` method that takes a class name (String or Class artifact) as a parameter. Usually, this class is the fully qualified source file name.

The installer creates two directories:

- `<MIDDLEWARE_HOME>/user_projects/epmsystem1/diagnostics/logs/ReportingAnalysis/SDK`—Log files generated by the ODL model.

- `<MIDDLEWARE_HOME>/user_projects/epmsystem1/config/ReportingAnalysis/SDK`—XML configuration file for the ODL model. The configuration file name is `logging.xml`. Set the configuration file and the `LoggingConfiguration.class` as JAVA properties:

  ○ `Djava.util.logging.config.class=oracle.core.ojdl.logging.LoggingConfiguration`

  ○ `Doracle.core.ojdl.logging.config.file=<MIDDLEWARE_HOME>/user_projects/epmsystem1/config/ReportingAnalysis/SDK/logging.xml`

The configuration file contains `sdk_file_handler`. in which the following properties can be changed:

- `path`—Full path of the file where log messages are written. The default path is `<MIDDLEWARE_HOME>/user_projects/epmsystem1/diagnostics/logs/ReportingAnalysis/SDK/sdk.log`

- `maxFileSize`—Maximum log file size. If the file exceeds this limit, it is renamed and a new log file is created.

The log level is configured by setting the level attribute of logger element. By default the log level is set to `WARNING:1`.

Table 1   Log Levels

ODL Log Level	ODL Description
INTERNAL_ERROR:1	The program has experienced an error for some internal or unexpected non-recoverable exception.

ODL Log Level	ODL Description
ERROR:1	A problem requiring attention from the system administrator has occurred.
WARNING:1	An action occurred or a condition was discovered that should be reviewed and may require action before an error occurs.
NOTIFICATION:1	A report of a normal action or event. This could be a user operation, such as "login completed" or an automatic operation such as a log file rotation.
NOTIFICATION:16	A configuration-related message or problem.
TRACE:1	A trace or debug message used for debugging or performance monitoring. Typically contains detailed event data.
TRACE:16	A fairly detailed trace or debug message.
TRACE:32	A highly detailed trace or debug message.

The following logging configuration file entities are used to write log files:

- `sdk_file_handler`—Writes log messages in a file located in `<MIDDLEWARE_HOME>/user_projects/epmsystem1/diagnostics/logs/ReportingAnalysis/SDK`. You can change this location in the path property of the handler.

  Declaration:

  ```
  <log_handler name="sdk_file_handler"
  class="oracle.core.ojdl.logging.ODLHandlerFactory">

  ...
  </log_handler>
  ```

- `oracle.EPMSDK.com.sqribe.rm`—Writes log messages to the file specified in the path property of `sdk_file_handler`. Default log level is `WARNING:1`. You can change the default if desired.

  Declaration:

  ```
  <logger name="oracle.EPMSDK.com.sqribe.rm" level="WARNING:1"
  useParentHandlers="false">

  ...
  </logger>
  ```

ObjectType

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

When artifacts are stored in EPM Workspace, they are associated with a metatype or artifact type that enables your program to filter search operations based on artifact type, and by keyword searches or by name. This class provides static methods that enable you to extract predefined `ObjectType class` instance variables that represent all built-in types found in EPM Workspace.

ReportMartException

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This class is an Exception artifact thrown by most method invocations on the Interactive Reporting SDK package. To handle these exceptions, your program must use "try-catch" constructs to handle the exceptions when they are thrown. Each `ReportMartException` artifact provides a message that describes the error condition responsible for generating the exception.

SessionFactory

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This class provides static `getInstance()` methods that enable your program to obtain the initial `Session` interface it requires to do useful work in EPM Workspace. Variations of the `getInstance()` methods are available that enable you to specify combinations of account, password, hostname, and TCP port number that are required for logging in to EPM Workspace.

UnknownReportMartException

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This Exception class can potentially be thrown during the initial invocation of a `SessionFactory#getInstance()` method by your program when attempting to establish contact with EPM Workspace. If your program handles this exception, one of these events can occur:

- The hostname you passed cannot be resolved to a TCP/IP address

- The server itself is not accessible by the network

- Contact cannot be established with the Service Broker running on that host

In the last case, check to make sure that the port number specified corresponds to the port on which the Service Broker is listening.

UserValidationException

See the Javadocs in `<EPM_ORACLE_HOME>/common/docs/ReportingAnalysis/SDK/javadoc`.

This Exception class is thrown if a validation error occurs when your program is connecting to EPM Workspace. If the try-catch logic traps this error, then the user account or password provided to the `SessionFactory.getInstance()` method is invalid.

4

Sample Java Programs

Prerequisites for Running the Sample Programs

The sample Java programs, sample property files, and sample control files are in `<EPM_ORACLE_HOME>/products/biplus/SDK/samples/java`.

For detail information on how to use the sample programs see Appendix A, "How to Use the API Sample Programs."

Items needed to use the sample programs:

- Reporting and Analysis and EPM Workspace SDK installed on your computer
- Java compiler and runtime environment installed on your computer
- GSM host name and port number
- Access to Core services using a URL (host name and port number), for example, `http://localhost:45000/workspace`
- Valid username and password with administrator access control

Running the Sample Programs

The `execapi.bat` file in `<MIDDLEWARE_HOME>/user_projects/epmsystem1/bin/ReportingAnalysis/SDK` is generated by the installer and can be used to configure the environment and run the sample programs.

➤ To run a sample program:

1 **Compile the program.**

   For compilation information, see "Compiling the Sample Programs" on page 92.

2 **Check that EPM Workspace is up and running.**

a. In a Web browser, enter a URL of the form `http://host:port/path`. For example:

```
http://localhost:45000/workspace
```

b. Enter a valid username and password.

3 At a command line, change to *Install Home*\SDK\bin and run `execapi.bat,` passing the name of the program and the arguments for the program. For example:

```
execapi BatchDriver administrator administrator qastar 6800 batch_driver.cf
```

4 Log in to EPM Workspace to validate your changes.

API Samples

Topics that provide detailed descriptions of the API samples:

AddBQYDocument.java

Use this sample to import an Interactive Reporting document to EPM Workspace and to set essential attributes. The Interactive Reporting Service, in addition to Core services, must be running to import Interactive Reporting documents using this sample. A properties file is used to get information needed for importing an Interactive Reporting document. A sample properties file, `inputBQYDoc.txt,` is included in the directory with the source code.

A sample attribute in the property file is the `BQY.iHTMLView flag` which determines if the Interactive Reporting document is available when using Interactive Reporting.

Input parameters:

`user pwd GSM_host GSM_port target_folder BQY_file property_file`

Output:

Interactive Reporting document is imported to the repository in the target folder.

AddBQYJob.java

Use this sample to import an Interactive Reporting job and to set some of its properties. Interactive Reporting Service, in addition to the common services, must be running to import Interactive Reporting jobs using this sample. A properties file is used to get information needed for importing an Interactive Reporting document. A sample properties file, `inputBQYJob.txt,` is included in the directory with the source code.

A sample property in the property file sets the `BQY.iHTMLView flag` to true, which makes the Interactive Reporting document available when using Interactive Reporting.

Input parameters:

`user pwd GSM_host GSM_port target_folder BQY_file property_file`

Output:

Interactive Reporting job is imported to the repository in the target folder.

AddCategory.java

The `Addcategory.java` sample demonstrates the creation of a folder and is installed with the Reporting and Analysis and EPM Workspace SDK in `Install Home\SDK\Samples\java.` All folders in the path that do not exist in the system are created in the process.

Input parameters:

`user pwd host port catpath`

For example:

`java AddCategory username password saturn 6800 /Sales`

Output:

Folders are created in EPM Workspace.

AddDocument.java

Use this sample to import a document to a target folder and to set the properties of the document.

Input parameters:

`user pwd GSM_host GSM_port document_name target_folder`

Output:

Document is imported to the target folder.

AddExternalLink.java

Use this sample to create an external link (URL) from EPM Workspace to another Web site. Link properties and the target folder are passed as arguments.

Input parameters:

`user pwd GSM_host GSM_port target_folder link_name description URL`

Output:

An external link (*link_name*) is created in the target folder.

AddFavorites.java

Use this sample to add folders and favorite items. The user needs administrator privileges to create the artifacts.

Input parameters:

`user pwd GSM_host GSM_port gif1 gif2`

Output:

Folders are created and favorite items are added to a user's favorite item list in EPM Workspace.

Example of syntax:

```
java -classpath %CLASSPATH% AddFavorites administrator administrator localhost 6800 D:
\Test1.gif D:\Test2.gif
```

AddGroup.java

Use this sample to add groups and to add child groups to a parent group. This sample specifically adds Group1, Group2, Group3, and Group4 to the parent group, Master. If the Master group does not exist, it is also added.

Input parameters:

user pwd GSM_host GSM_port

Output:

New groups are created in Shared Services.

AddLink.java

Use this sample to add a link into the repository. The sample imports a file to the specified folder and creates a link to the file in the root folder.

Input parameters:

user pwd host port path document

Output:

Links are created in EPM Workspace.

AddObjectType.java

Use this sample to show how to add a metatype to EPM Workspace. The user needs administrator privileges to create the artifacts.

Input parameters:

user pwd host port path mimetype filenameextension iconpath

Output:

Metatypes are created in EPM Workspace.

AddOCEDocument.java

Use this sample to import an Interactive Reporting database connection (OCE file) and to set some of its properties. The database user ID and password are optional. An Interactive Reporting database connection is used by an Interactive Reporting document or job for making a connection to a data source.

Input parameters:

user pwd GSM_host GSM_port target_folder OCE_name [isPrompt dbUsername dbPassword]

Output:

Interactive Reporting database connection is added to the target folder in EPM Workspace.

AddRole.java

Use this sample to add a role definition.

Input parameters:

user pwd GSM_host GSM_port role role_description

Output:

New role is included in Shared Services.

AddSPF.java

Use this sample to import an Production Reporting output collection, which includes all output produced by an Production Reporting program (job). An output collection includes SPF files, which are a Oracle proprietary output, and other formats such as PDF and HTML.

Input parameters:

<user> <pwd> <GSM host> <GSM port> <target Folder> <SPF File>*user pwd GSM_host GSM_port target_folder SPF_file*

Output:

A group of Production Reporting files (output collection) is imported to the target folder in EPM Workspace.

AddSubscription.java

Use this sample to add subscriptions for a user to a document category. A subscribed user receives e-mail notifications on a document whenever it changes. The user needs administrator privileges to create the artifacts.

Input parameters:

user pwd host port subscribed_user path [subscribed_user_email]

Output:

A subscribed user receives a notification e-mail whenever the document changes. Log on to the servlet as the user who ran this sample program. Navigate to the artifact and look at its subscriptions. The corresponding check box should be selected.

Examples of the syntax:

```
java -classpath %CLASSPATH% AddSubscription administrator administrator localhost 6800
TESTSUBS1 D:\Test1.txt email@some_address.com
```

or

```
java -classpath %CLASSPATH% AddSubscription administrator administrator localhost 6800
TESTSUBS1 /TestDir1/TestObj email@some_address.com
```

AddUser.java

Use this sample to add a user to Oracle's Hyperion® Shared Services. In addition it is possible
to specify if the user being created has the administrator role or has the ability to execute jobs.

Input parameters:

user pwd GSM_host GSM_port username password allow_job_run-boolean *administrator_role*-
boolean

Output:

A user is created in EPM Workspace with or without the administrator role.

AddVersions.java

Use this sample to demonstrate version capability in EPM Workspace for documents. The
document to be imported must exist in the system.

Input parameters:

user pwd GSM_host GSM_port workspace_document (for example, /Finance/Sales.doc)
local_document (for example, c:\Sales2.doc)

Output:

A new version of a document is imported.

AutoZip.java

Use this sample to show the use of autozip functionality. This program imports a document to
EPM Workspace along with a zipped version of the document.

Input parameters:

user pwd GSM_host GSM_port target_folder document

Output:

A document and a zipped version of the document are imported to EPM Workspace.

BatchDriver.java

Use this sample to bulk load artifacts into EPM Workspace and to modify artifact assessors. You can bulk load these artifacts using this sample program:

- Folders
- Documents
- Interactive Reporting database connections
- Interactive Reporting documents

`BatchDriver.java` reads a control file that contains control statements describing artifacts that are created in the repository. For more information on the control file and the control statements used for this sample, see "Batch Driver Sample Program Control File" on page 17. A sample control file, `batch_driver.cf` is in `<EPM_ORACLE_HOME>/products/biplus/SDK/samples/java`.

Input parameters:

`user pwd host port control_file` (for example, batch_driver.cf)

Output:

A batch load of one or more of these artifacts: accessor, category, data, Interactive Reporting database connection, Interactive Reporting document.

An example of the syntax when you compile the program:

```
java -cp .;%classpath% BatchDriver administrator administrator venice 6800 TEST1 C:
\Hyperion\Brio\batch_driver.cf
```

An example of the syntax when using the installed compiled version of the program:

```
java -cp .;%classpath% com.sqribe.rm.BatchDriver administrator administrator venice 6800
TEST1 C:\Hyperion\Brio\batch_driver.cf
```

CategoryDelete.java

Use this sample to delete an empty folder in EPM Workspace. If the folder to be deleted does not exist, a message is displayed.

Input parameters:

`user pwd host port Folder` (for example, /Sales/West)>

Output:

Named empty folder is deleted.

ExecuteBQYJob.java

Use this sample to execute (or schedule) an Interactive Reporting job. It is derived from the `SampleBase` class and makes use of the `getSymbol()` method implemented in that class. Refer to `SampleBase.java` for information about how to pass attribute values to this and other classes derived from `SampleBase`.

Input parameters:

user pwd host post bqypath=/folder/bqyjob (for example, /SampleSDK/TestBQYjob)

Output:

Job output from the Interactive Reporting job. This varies based on job run.

ExecuteSQRJob.java

Use this sample to load an Production Reporting program into a target folder. After the program is loaded, it is executed once to create an output collection.

Input parameters:

user pwd host post target_foler Production_Reporting_program_file (for example, Sales.sqr)>

Output:

Imported Production Reporting program and the corresponding output collection for one run.

ExpirationDate.java

Use this sample class to load and set an auto-deletion date for a EPM Workspace file. The file to be loaded for this sample should not be an Interactive Reporting document or an Production Reporting program.

Input parameters:

user pwd host port folder file_to_be_loaded

Output:

Imported file set to auto-deletion after thirty days from the system date on which the program is run.

FetchCategory.java

Use this sample to obtain information about folder properties.

Input parameters:

```
user pwd host port folder (for example, /Finance)
```

Output:

Output with folder name goes to the console.

FetchDocument.java

Use this sample to obtain a file, for example, a DOC or TXT file from the repository. If multiple versions exist, all of them are extracted and stored on the local computer. Numeric extensions (0, 1,…,.n) are added to the file name after the extraction.

Input parameters:

```
user pwd host port folder docname
```

Output:

One or more (if multiple versions exist) document artifacts.

ListEvents.java

Use this sample to list events defined in EPM Workspace.

Input parameters:

```
user pwd host port
```

Output:

Console listing of events in the system.

ListGroups.java

Use this sample to list all groups defined in EPM Workspace.

Input parameters:

```
user pwd host port
```

Output:

Console listing of groups in EPM Workspace.

ListUsers.java

Use this sample to list all users defined in EPM Workspace.

Input parameters:

user pwd host port

Output:

Console listing of users in the system.

Login.java

Use this sample to establish and close a EPM Workspace connection. It is recommended that you run this sample before running others. Running the sample ensures that the basic connection to EPM Workspace is established without problem.

Input parameters:

user pwd host port

Output:

Messages `Connection Established` and `Connection closed`.

ObjectByPath.java

This sample shows how to fetch a BaseObject from the repository using the full path artifact name. The artifact name passed as an argument must include the full path.

Input parameters:

user pwd host port object_path

Output:

Some of the artifact properties are listed on the console

ObjectById.java

Use this sample to demonstrate fetching a BaseObject from the repository using the artifact ID passed as an argument. The artifact ID is unique to every artifact in the system and is used internally by EPM Workspace to access documents or folders in the system.

Input parameters:

user pwd host port objectID

Output:

Some of the artifact properties are listed on the console

PublishEvent.java

Use this sample to create or update an event in EPM Workspace. The sample control file, `rte.cf,` is included in the folder with the source program.

Input parameters:

user pwd host port control_file

Output:

Event named in the control file is created in EPM Workspace.

PublishOutputDirectory.java

Use this sample to define an output directory in EPM Workspace. You can run this sample in two ways. The input parameters section shows the two available options. The sample control file, `outdir.cf,` is included in the folder with the source program.

Input parameters:

Option I: *user pwd host port* name=OutputDir1 dirname="C:\BPS\OutDir"

Option II: *user pwd host port control_file*

Output:

Named directory is created in EPM Workspace.

PublishPrinterResource.java

Use this sample to define a printer resource in EPM Workspace. You can use this sample in two ways, as shown in the input parameters section. The sample control file, `printer.cf,` is included in the folder with the source program.

Input parameters:

Option I: *user pwd host port* name=myPrinter1 pname="HP LaserJet"

Option II: *user pwd host port control_file*

Output:

Named event is created in EPM Workspace.

QueryGroup.java

This sample shows how to query group members of a group in EPM Workspace.

Input parameters:

user pwd host port group

Output:

Shows members of the group being queried.

QueryUser.java

Use this sample to query user properties in EPM Workspace.

Input parameters:

user pwd host port user

Output:

Shows user properties on the console.

ReplaceObject.java

Use this sample to replace the contents of the specified artifact in EPM Workspace.

Input parameters:

user pwd host port user folder document

Output:

Replaces the original document with a copy.

SQRParms.java

Use this sample to obtain the parameters and their type, with the specified Production Reporting job.

Input parameters:

user pwd host port user Production_Reporting_job_UUID

Output:

Shows Production Reporting job parameters and their properties on the console.

TriggerExternalEvent.java

Use this sample to trigger an external event. When this event is triggered, it is set to run immediately. Externally Triggered Events are always PUBLIC events.

Input parameters:

acct pwd host port Ext Event Name

Output:

External event named in the control file is triggered.

Attributes and Supporting Classes

Most of the sample API programs that are provided include these common attributes:

- Command line arguments

- Opening and processing data lines read from a data file

- Look up of EPM Workspace folder, account, and group artifacts

These example programs take advantage of artifact-oriented programming techniques by extending the base class that implements a set of common functions. These are the supporting classes:

- `AddBQYBase.java`

- `SampleBase.java`

- `SampleUtilities.java`

P a r t   I I
Customization

In Customization:

- Java Server Pages
- Customizing E-mail Notifications
- SmartCuts
- Extended Services

5

Java Server Pages

Note: Customizing JSPs should only be done by developers with JSP and Java programming experience. Custom JSPs are not supported by Oracle Customer Support.

JSP Directory

EPM Workspace uses about one thousand JSPs, which are in */deployment*/jsp.

For all application servers except WebLogic 9.x, deployed Web applications and their associated files are placed in *HyperionHome*/deployments/*AppServNameAndVersion*. For WebLogic 8.x, deployed Web applications and their associated files are placed in *Install Home*/AppServer/InstalledApps/*AppServName*/*version*

/deployment/jsp includes these subdirectories:

- administrator—JSPs used in the Administer module, which is used to configure many system properties, administrator preferences, and usage tracking properties
- browser—JSPs used in Explore (the browse servlet), which is used for browsing, viewing, and importing content, and for running jobs; subdirectories contain JSPs for preferences (which allows users to customize Explore), running Interactive Reporting jobs, and running generic jobs
- com—JSPs used by various Hyperion tools
- dataaccess—JSPs used by Oracle's Hyperion® Interactive Reporting Web Client
- iHTMLServlet—JSPs used by Interactive Reporting
- irPortlet—JSP used by the Interactive Reporting portlet
- personalpage—JSPs used by Personal Pages
- scheduler—JSPs that make up the Schedule module, which is used for managing scheduled jobs and for creating and managing events

- shared— JSPs used by multiple modules; includes JSPs used by widgets, wizards, and top and bottom frames, for example:
  - ❍ `hiddenWidget.jsp`
  - ❍ `tooltipsHandler.jsp`
  - ❍ `statusMessage.jsp`
  - ❍ `widgetBeginBorder.jsp`

JSPs Identified by User Tasks

This section identifies the JSPs for the user interfaces of the three most common user tasks:
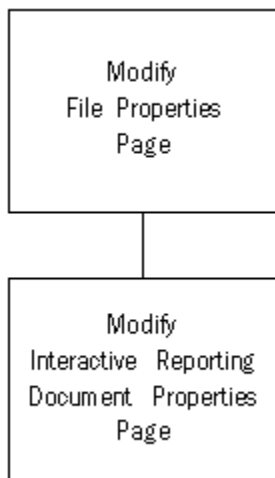
- "Modifying File Properties" on page 52
- "Creating Events" on page 54
- "Scheduling Jobs" on page 56

This information is provided for each user task:

- **Flow chart**—Shows the user interface pages used for the most common way to complete this task.
- **Figure**—A complete graphic of each page from the flow chart showing which JSPs control the page.
- **Table**—List of the main JSPs and their paths for the graphic shown.

Modifying File Properties

Modifying file properties involves logging in to EPM Workspace, using Explore to locate the item with properties to be modified, and modifying the properties:

The Modify File Properties page is composed of these widgets:

- "General Widget" on page 53

- "Advanced Options Widget" on page 53

General Widget

JSP details for the General widget:

1. Primary JSP – `/jsp/browser/fileGeneralWidget.jsp`

2. Primary JSP – Associated JSPs and Include Files

3. Primary JSP – `/jsp/browser/setName.jsp`

4. Primary JSP – `/jsp/browser/setDescription.jsp`

5. Primary JSP – `/jsp/browser/owner.jsp`

6. Primary JSP – `/jsp/browser/size.jsp`

7. Primary JSP – `/jsp/browser/smartcut.jsp`

Advanced Options Widget

JSP details for the Advanced Options widget:

1. Primary JSP – `/jsp/browser/fileAdvOptionsWidget.jsp`

2. Primary JSP – Associated JSPs and Include Files

3. Primary JSP – `/jsp/browser/mimeTypes.jsp`

4. Primary JSP – `/jsp/browser/filePriorityjsp`

5. Primary JSP – `/jsp/browser/charEncodingjsp`

6. Primary JSP – `/jsp/browser/hiddenItemjsp`

7. Primary JSP – `/jsp/browser/autoDeletejsp`

8. Primary JSP – `/jsp/browser/allowExceptionsjsp`

9. Primary JSP – `/jsp/browser/setAutoKeywordsjsp`

10. Primary JSP – `/jsp/browser/assignKeywordsjsp`

File priority is displayed on this widget when turned on by the administrator.

Interactive Reporting Database Connection Widget

JSP details for Interactive Reporting Database Connection widget:

Ref #	Primary JSPs	Ref #	Associated JSPs and Include Files
1	`/jsp/browser/fileOCEOptionsWidgetjsp`	2	`/jsp/browser/passThroughjsp`
3	`/jsp/browser/fileMetaOCELinkWidgetjsp`		

Creating Events

Creating an event involves navigating to the manage events page in the Schedule module and going through the recurring events wizard or the triggered events wizard to create an event:
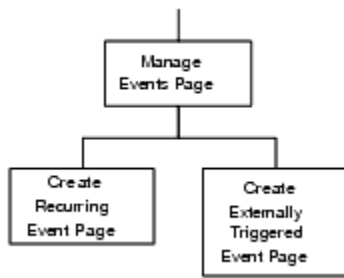
The pages are described in these topics:

- "Create Recurring Event Page" on page 55
- "Create Externally Triggered Event Page" on page 56

Create Recurring Event Page

JSP details for the Create Recurring Event page:

1. `/jsp/scheduler/events/createPublicRTEjsp`
2. Associated JSPs and Include Files
3. `/jsp/scheduler/events/editEventGeneralWidgetjsp`
4. `/jsp/scheduler/events/editDaysToRunWidgetjsp`
5. `/jsp/scheduler/events/editTimeToRunjsp`
6. `/jsp/scheduler/events/editDatesWidgetjsp`

Days To Run

The Days to Run widget also uses these JSPs in `/jsp/scheduler/events/runsched`:

- By Day—`day.jsp`
- By Week—`week.jsp`
- By Period—`period.jsp`
- By Quarter—`quarter.jsp`
- By Year—`year.jsp`
- Every—`every.jsp`
- Advanced Days of Periods—`advancedDaysOfPeriod.jsp`
- Advanced Days of Week—`advancedDaysOfWeek.jsp`

Time To Run

The Time to Run widget uses these JSPs in `/jsp/scheduler/events/timetorun`:

- Once Per Day—`once.jsp`
- More Than Once Per Day—`every.jsp`
- After External Event—`external.jsp`

Create Externally Triggered Event Page

JSP details for the Externally Triggered Event page:

1. Primary JSP – `/jsp/scheduler/events/editXTEjsp`
2. Associated JSP – `/jsp/scheduler/events/editEventGeneralWidgetjsp`

Scheduling Jobs

Scheduling jobs involves navigating to the Schedule module and creating a schedule by using the scheduling wizard. All pages listed in this task flow are in the scheduling wizard except the

first page, the Scheduling Information page. Interactive Reporting jobs, Production Reporting jobs, and generic jobs require different pages for parameters (set values):
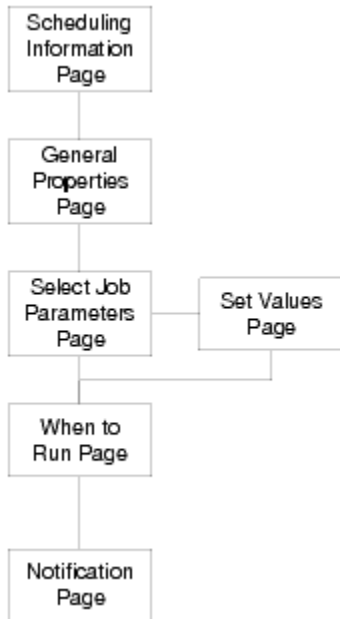
The pages are described in these topics:

- "Scheduling Information Page" on page 57
- "Schedule General Properties Page" on page 58
- "Select Job Parameters" on page 58 (use Set Values page when using new parameters)
- "Set Values Page" on page 58
- "When to Run Page" on page 60
- "Notification Page" on page 61

Scheduling Information Page

JSP details for the Scheduling Information page:

1. Primary JSP – `/jsp/scheduler/listJobDatajsp`

2. Associated JSPs and Include Files – `/jsp/scheduler/listJobScheduleDatajsp`

3. Associated JSPs and Include Files – `/jsp/scheduler/listJobParameterDatajsp`

4.   Associated JSPs and Include Files – `/jsp/shared/wizardHeadingjsp`

Schedule General Properties Page

JSP details for the Schedule General Properties page, which includes the database connect string widget:

1.   Primary JSP – `/jsp/scheduler/schedJobGeneraljsp`

2.   Associated JSPs and Include Files – `/jsp/scheduler/schedJobGeneralWidgetjsp`

3.   Associated JSPs and Include Files – `/jsp/scheduler/schedBQYDatabaseWidgetjsp`

4.   Associated JSPs and Include Files – `/jsp/scheduler/schedSQRDatabaseWidgetjsp`

5.   Associated JSPs and Include Files – `/jsp/shared/wizardHeadingjsp`

Select Job Parameters

JSP details for the Select Job Parameters page:

Primary JSP
`/jsp/scheduler/schedJobSelParamjsp`
Associated JSPs and Include Files
`/jsp/scheduler/schedJobSelParamWidgetjsp`
`/jsp/shared/wizardHeadingjspp`

Set Values Page

The user interface to set values for Interactive Reporting jobs, Production Reporting jobs, and generic jobs is specific to the job type. For Interactive Reporting jobs, see "Interactive Reporting Jobs" on page 59, for Production Reporting jobs, see "Production Reporting Jobs" on page 60, and for generic jobs, see "Generic Jobs" on page 60.

Interactive Reporting Jobs

The Interactive Reporting Job Parameter wizard contains these widgets:

Cycles

JSP details for the Cycles widget:

Primary JSP	Associated JSPs
/jsp/scheduler/BQYJobParameterGeneraljsp	/jsp/scheduler/BQYMultiCycleJobParameterGeneraljsp
	/jsp/scheduler/BQYSingleCycleJobParameterGeneraljsp

Multiple cycle and single cycle Interactive Reporting jobs use different JSPs.

Define Cycle Widget

JSP details for the Define Cycles widget:

Primary JSP	Associated JSPs and Include Files
/jsp/scheduler/editBQYCycleGeneraljsp	/jsp/shared/BQYDefineCycleWidgetjsp
	/jsp/shared/bqylimits/jobParamsjsp
	/jsp/shared/BQYProcessOptionsWidget
	/jsp/shared/BQYActionWidgetjsp
	/jsp/shared/accessControlWidgetjsp

Set Values

JSP details for the Set Values widget:

Primary JSP	Associated JSPs
/jsp/shared/bqylimits/jobParamsjsp	jsp/shared/bqylimits/relationalJobParamsjsp
	/jsp/shared/bqylimits/mddJobParamsjsp

Actions

JSP details for the Action widget:

Primary JSP	Associated JSPs
/jsp/scheduler/editBQYActionGeneraljsp	/jsp/shared/editBQYActionWidgetjsp

These JSPs are called in `editBQYActionWidget.jsp` and are used by one of the five action pages: export, save document, print document, e-mail section, and e-mail document:

- `/jsp/shared/editBQYActionExportWidget.jsp`
- `/jsp/shared/editBQYActionSaveWidget.jsp`
- `/jsp/shared/editBQYActionPrintWidget.jsp`
- `/jsp/shared/editBQYActionEmailWidget.jsp`
- `/jsp/shared/editBQYActionEmailSectionWidget.jsp`

Production Reporting Jobs

JSP details for the Production Reporting Job Parameters wizard.

Primary JSP	Associated JSPs
/jsp/shared/SQRJobParameterGeneraljsp	/jsp/shared/SQRJobParametersWidgetjsp
	/jsp/shared/saveJobParamjsp
	/jsp/shared/wizardHeadingjsp

Generic Jobs

JSP details for the Generic Job Parameters wizard.

Primary JSP	Associated JSPs
/jsp/shared/GenericJobParameterGeneraljsp	/jsp/shared/SQRJobParametersWidgetjsp
	/jsp/shared/saveJobParamjsp
	/jsp/shared/wizardHeadingjsp

When to Run Page

JSP details for the When to Run widget:

Primary JSPs	Reference No	Associated JSPs
/jsp/scheduler/schedJobSelEvent.jsp	1	/jsp/scheduler/schedJobSelEventWidget.jsp
/jsp/scheduler/schedJobNewEvent.jsp	2	/jsp/scheduler/schedJobNewEventWidget.jsp
		/jsp/shared/wizardHeading.jsp

For the pages following the When To Run page, see .

Notification Page

JSP details for the Notification page:

1. Primary JSP – `/jsp/scheduler/schedJobNotify.jsp` Associated JSPs – `/jsp/scheduler/schedJobNotifyBrioOneWidget.jsp`

2. Associated JSPs – `/jsp/scheduler/schedJobNotifyEmailWidget.jsp`

3. Associated JSPs – `/jsp/scheduler/schedSQRNotifyStoreWidget.jsp`

4. Associated JSPs – `/jsp/shared/wizardHeading.jsp`

6

Customizing E-mail Notifications

Configuring E-mail Notifications

Note: You need the administrator role to configure properties in the Administer module.

You can configure these e-mail properties using the Administer module:

- The outgoing e-mail server

- The *from* e-mail account

- Whether e-mail attachments are enabled; and if so, the maximum number of bytes allowed in an attachment

- The duration to keep scheduled jobs and background jobs in the notification logs

Template files define the formats of e-mail notification messages. Each notification type has its own template file. Therefore, you can customize the e-mail notifications for each notification type.

During installation, default template files are installed. If you are satisfied with the default files and do not want to customize them, you can skip this section.

Notification Types

EPM Workspace supports these types of e-mail notifications:

- **Item Notifications**—End users use Explore to subscribe to a file (or artifact) in the repository. Explore allows you to specify the e-mail address that is sent an e-mail notification when the item is modified.

  Item e-mail notifications are sent when the item in the repository is modified. For example, if a user stores a new version of a file in the repository, an e-mail notification is sent to all users who subscribed to the file.

- **Folder Notifications**—End users use Explore to subscribe to repository folders and optionally subfolders. Explore enables users to specify the e-mail address to which an e-mail notification is sent when an item is added to or modified in a given folder or subfolder.

- **Job Output Notifications**—End users use Explore to subscribe to a program's output. When the program is executed by the Job Service, an e-mail notification is sent to all users who subscribed to the program's output.

- **Scheduled Job Notifications**—End users use Explore to subscribe to scheduled job notifications, which may announce:

  - Scheduled job completed successfully
  - Scheduled job failed
  - Scheduled job is being retried

    You can turn off scheduled job retry notifications for all users by setting the *sched_retry_notifications* Java property to *false* on the Event Service.

  - FTP delivery of scheduled job's output succeeded
  - FTP delivery of scheduled job's output failed

Template File Directory

The template files are in *Install Home*`/lib/notification` on each server host.

Note: For information about Install Home, see "Understanding Hyperion Home and Install Home" on page 13.

The templates must be changed on the host of the Event Service. If you modify templates for job notifications, copy the modified templates to that directory (*Install Home*`/lib/ notification`) on every computer hosting a Job Service.

Choosing HTML or Text Format

The `notification.properties` file (in *Install Home*`/lib/msgs`) contains properties that specify the template file to use for each notification type. For example, here are the lines from `notification.properties` that specify the template files to be used for item notifications:

```
itemEmailFileHTML.string=item_email.html
itemEmailFileText.string=item_email.txt
```

The setting of the `-Ddisable_htmlemail` property chooses between the HTML and text versions of the template. If `-Ddisable_htmlemail` is set to *false,* the Event Service uses `item_email.html` when it sends an item notification. This property is set in `startCommonServices.bat`. For information on setting this property, see the *Oracle Enterprise Performance Management Workspace Administrator's Guide.*

Template File Replacement Tokens

Template files define the format and content of e-mail notification messages. The template files can contain tokens that are replaced with runtime values, and other data. For example, HTML tags are typically placed in the HTML template files.

Replacement tokens are enclosed in less than (<) and greater than (>) signs and start with the string `BRIO_TAG`. Their values are dynamically replaced in e-mail notifications.

Table 2   Replacement Tokens

Token	Description
<BRIO_TAG_USER>	Name of the user who triggered the e-mail notification Applies to item and folder e-mail notifications
	The `notificationOriginator` property in the `notificationproperties` file provides a parameterized message that you can use to add text around the user name in the e-mail notification message
<BRIO_TAG_DESC>	Description that users enter when they create an item in the repository. Users can enter descriptions when they add a version of a file to the repository When this occurs, this tag is replaced by the user's description
	The `notificationDescription` property in the *notificationproperties* file provides a parameterized message that you can use to add text around the users description in the e-mail notification message
<BRIO_OBJECT_ LINK>	SmartCut to the item that is the target of the e-mail notification
	For an item subscription, it would be the SmartCut to the file that was the target (for example, the file that had a new version created) For a program output subscription, it specifies the SmartCut to the program output
<BRIO_OBJECT_ FULLPATH>	Full path of the item that triggered the e-mail notification
	This is useful within a template HTML file as a target rather than the SmartCut
<BRIO_TAG_ MODIFY_LINK>	SmartCut to the link to modify the subscription to the item that triggered the e-mail
	This is useful to allow a user to discontinue or modify the subscription
<BRIO_TAG_ MODIFY_MSG>	Text associated with changing a subscription The *notificationproperties* file specifies properties that are substituted for this replacement token:
	• **cancelItemSubscriptionSmartcut—**Specifies text for changing an item subscription
	• **cancelCategorySubscriptionSmartcut**—Specifies text for changing a folder subscription
	• **cancelProgramOutputSubscriptionSmartcut**—Specifies text for changing a program output subscription
<BRIO_TAG_ EXCEPTION_TITLE>	String to use as the title for an exception message
	This replacement token is replaced with the *exceptionMessage* value in the `notificationproperties` file
<BRIO_TAG_ EXCEPTION_MSG>	This replacement token is replaced with the exception messages that are the result of a job's execution in the Job Service

Token	Description
<BRIO_TAG_ RETRY_MSG>	Reason why a scheduled job is being retried
	The *notificationproperties* file specifies properties that are substituted for this replacement token:
	● **scheduledJobRetry**—An error occurred and the job is being retried
<BRIO_TAG_USER_ DEFINED>	Body component for e-mail notifications about the scheduled job
<BRIO_TAG_NO_ ATTACHMENT>	Reason why no attachment was attached to the e-mail notification Typically, this is not put into the e-mail notification
	If an attachment was requested in the subscription and yet no attachment is attached, then this replacement token is used to let the user know why no attachment was applied
	The notificationproperties file specifies properties that are substituted for this replacement token:
	● **attachmentErrorUnknown**—An unknown error occurred
	● **attachmentTooBig**—The size of the e-mail attachment is larger than the maximum set by the administrator
	● **attachmentNotAllowed—**There was a permission violation and the attachment could not be attached
	● **attachmentsDisabled**—The administrator disabled e-mail attachments

Properties in the notification.properties File

When Event Service starts, it reads the contents of the `notification.properties` file. This file contains messages that go into e-mail notifications and other properties (see Table 3 on page 67). Never delete a record from this file. The `notification.properties` file is in:

Install Home\lib\msgs

Each line in the `notification.properties` file is formatted as:

name=value

The name ends in *.string* or *.text*. If the property name ends in *.string*, then the value is constant. Conversely, if the value contains a variable, then the property name ends with *.text*. Variables start with a left curly brace ({) and end with a right curly brace (}). Variables contain a number that starts with zero.

When the Event Service creates e-mail notification messages, it frequently replaces a replacement token (see Table 2, "Replacement Tokens ," on page 65) with the property value from the `notification.properties` file. Consider this variable property value assignment in the `notification.properties` file:

NotificationOriginator.text={0} triggered email notification

With this definition, the `<BRIO_TAG_USER>` replacement token in an e-mail notification template file is replaced with the name of the user who triggered the e-mail notification, followed by the text *triggered e-mail notification*. If a user named *bob* triggers the e-mail notification, then the string reads: *bob triggered e-mail notification*.

Table 3 Notification Properties

Property	Description
attachmentsDisabled	Text to insert into the e-mail notification when the attachment cannot be added because attachments are disabled for the domain Editing the system properties using Administer module can enable e-mail attachments
attachmentErrorUnknown	Text to insert into the e-mail notification when adding an attachment to the e-mail fails for some unknown reason
attachmentNotAllowed	Text to insert into the e-mail notification when the attachment cannot be added due to an access violation If you receive an access violation, check the item permissions
attachmentTooBig	Text to insert the e-mail notification when the attachment cannot be added because it is too big Editing the system properties in the Administer module can set the maximum size of the e-mail attachment
cancelCategorySubscriptionSmartcut	Text to insert in the e-mail notification when specifying a SmartCut to the folder subscription The location of the {0} identifies when the SmartCut to the subscription is placed
CancelItemSubscriptionSmartcut	Text to insert in the e-mail notification when specifying a SmartCut to the item subscription The location of the {0} identifies when the SmartCut to the subscription is placed
cancelProgramOutputSubscriptionSmartcut	Text to insert in the e-mail notification when specifying a SmartCut to a job output subscription The location of the {0} identifies when the SmartCut to the subscription is placed
categorySubscriptionSubject E	e-mail subject field when a folder subscription notification e-mail is sent The string "{0}" is replaced with the item name
	Folder subscription e-mails are sent to users who subscribed to a folder when a file or other interesting item is added to the folder
	To include a single quotation mark in the subject, enter two single quotation marks For example, to produce the subject *Information You've Subscribed To*, enter as the value of this field: `Information You''ve Subscribed To`
charset	Character set used to encode the subject field for e-mail notifications
exceptionMessage	Heading for exception information reported in the e-mail notification This property is only used when sending e-mail notifications for items with associated exception information
htmlEmail	Whether e-mail notifications are in HTML or text format If this property's value is *true*, then e-mail notifications are in HTML
itemSubscriptionSubject	E-mail subject field when an item subscription notification e-mail is sent The string "{0}" is replaced with the item name
	Item subscription e-mails are sent to users who subscribed to a EPM Workspace item when a new version of the file is imported or its properties are changed
	To include a single quotation mark in the subject, enter two single quotation marks For example, to produce the subject *Information You've Subscribed To*, enter as the value of this field: `Information You''ve Subscribed To`
notificationDescription	Item subscription e-mail notifications contain the description that end users provided when they created a new version of a file The string "{0}" is replaced with the file description associated with the new version

Property	Description
notificationOriginator	User who triggered the e-mail notification The string "{0}" is replaced with the user name
	Item and folder subscription e-mail notifications contain the user who triggered the e-mail notification in the e-mail body The *notificationOriginator* property allows customized formatting For example, the customer could change this value to:
	`notificationOriginatortext={0} triggered this email notification`
	With the above definition, the e-mail notification body contains:
	"bob triggered this e-mail notification"
	if the EPM Workspace user "bob" triggered the notification
jobOutputSubscriptionSubject	E-mail subject field when a job output subscription notification e-mail is sent The string "{0}" is replaced with the job name
	Job output subscription e-mails are sent to users who subscribed to a job, when new output of that job is imported
	To include a single quotation mark in the subject, enter two single quotation marks For example, to produce the subject *Information You've Subscribed To*, enter as the value of this field: `Information You''ve Subscribed To`
scheduledJobFailed	Text to be placed in the e-mail notification when a scheduled job fails to run The string {0} is replaced by the reason for the failure The reason can be the text defined in *scheduledJobFailedJobKilled* or *scheduledJobFailedUnknown* properties, or it can be the text returned from the external program that was unsuccessfully executed
scheduledJobFailedJobKilled	When a scheduled job is killed and the user requested e-mail notification, this text is used as the
scheduledJobFailedUnknown	Reason why the job failed when a scheduled job fails for unknown reasons and e-mail notification was requested
scheduledJobRetry	Text of the e-mail message sent by the scheduler when it couldn't run the job for some reason and is attempting to retry the job again This is sent when notifications were requested
scheduledJobSubject	E-mail subject field when a scheduled job e-mail notification is sent The string {0} is replaced with the schedule name and the string {1} is replaced with the job name
	This type of e-mail notification is sent when a scheduled job is executed and the person who scheduled the job checked the box for e-mail notification
	To include a single quotation mark in the subject, enter two single quotation marks For example, to produce the subject *Information You've Subscribed To*, enter as the value of this field: `Information You''ve Subscribed To`
scheduledJobUnableToSched	Text of the e-mail message that is sent by the scheduler when it couldn't run the job for some reason This is sent if notifications were requested and all retries are exhausted
smartcut	Allows additional text to be added with the SmartCut to the item The {0} must be present in the string and identifies the location where the SmartCut is placed
unknownSmartcut	String used when the SmartCut cannot be determined
viewScheduleSmartcut	Text to be placed in e-mail notifications when a scheduled job is run The user must request e-mail notification when scheduling the job to get this information The location of the {0} identifies where the SmartCut to the schedule is placed

Images in HTML-Formatted E-mail Notifications

You can place images, such as GIF or JPEG files, into HTML-formatted e-mail messages.

➤ To place an image into an HTML-based e-mail message:

1 **Add an HTML IMG tag to the HTML template file.**

2 **Put the image file into the image directory that corresponds to the given e-mail notification type.**

The image files are placed in a subdirectory of:

Install Home/lib/notification/images

Each notification type has its own subdirectory under `images`.

As an example, assume you want to place an image named `logo.gif` into item e-mail notifications. To do this you would add the string:

`<IMG SRC=logo.gif>`

to the file:

Install Home/lib/notification/item_email.html

and then store `logo.gif` in:

Install Home/lib/notification/images/item

7

SmartCuts

About SmartCuts

SmartCuts enable you to integrate EPM Workspace with your intranet and other Web-based applications. This section provides information on the SmartCut commands and a few SmartCut examples.

By embedding SmartCuts in your applications, you enable users to view EPM Workspace content or run jobs from those applications:

● Used within EPM Workspace items, SmartCuts enable *report surfing*, or hyperlinking directly between content of any type. For example, a white paper can link to a glossary document, or a sample code listing can link to a job that runs the code and returns output.

● SmartCuts enable more complex output structures, such as *drill-down* reports, where users can choose down through levels of detail.

● SmartCuts enable a user to send live links to documents to other users. When the recipient selects the URL, the SmartCut takes the user directly to the document, the job run form, or the report output. To use a URL this way, write it with the path to the document, form, or job and with any parameters the job requires.

● SmartCuts can automatically log users into the Web modules. The optional fields **user**, **pass,** and **server** are used to pass the login information. For more information about this and other ways users can be automatically logged in, see the *Oracle Enterprise Performance Management Workspace Administrator's Guide.*

Available SmartCuts include **get** and **run.**

Review these sections for information on SmartCuts. SmartCuts for Interactive Reporting documents and jobs are supported with additional optional variables. For more information on these variables see "SmartCut Variables for Interactive Reporting Documents and Jobs" on page 76.

SmartCut Considerations

Keep in mind these items when working with SmartCuts:

- **Workspace sessions**—You can close a EPM Workspace session opened with a SmartCut (without closing the browser) by appending `/login` to the end of the browse servlet URL:

  ```
  http://database/workspace/browse
  ```

  For example, if this is the SmartCut in the address line:

  ```
  http://venice/workspace/browse/withnav_get/sales/Hyperion
  ```

  Replace everything after `browse`, with `login`:

  ```
  http://venice/workspace/browse/login
  ```

  This URL ends your current session and opens the EPM Workspace login screen.

- **SmartCut URL case-sensitivity**—When writing SmartCuts, keep in mind that EPM Workspace URLs are case-sensitive and do not accept space characters. In place of a space, insert **%20**. Some example SmartCuts in this section illustrate this with the report name *Sales Charts* referenced as `Sales%20Charts`.

get Command

Use the **get** command to retrieve and view a document, report, or form, or to retrieve job output, and to retrieve one job output artifact.

These uses of the **get** command are discussed in these topics:

Getting and Viewing Documents, Reports, or Forms

To view a report, a document, or a form stored in the repository, use the **get** SmartCut, which has this syntax:

```
http://servletsHost:servletsPort/workspace/browse/get/folderPath/item[?
version=versionNumber][?mimetype=mimeType][?latest=true]
```

where:

servletsHost is the name of the host computer on which the servlets run

servletsPort is the port number of the Web server on which the servlets run (you can omit :*servletsPort* if port number is 80)

`browse` is the default Web application deployment name. If you named this directory differently in your Web server software, substitute the correct name.

folderPath can be one folder name, or it can be a folder path. (For example: *folder/subfolder/subfolder/….*) The folder name should match the folder names stored in EPM Workspace.

item is the name of the item you want to view. Word documents, spreadsheets, HTML forms, and reports are examples of the types of items you can specify. The *item* name must match the name given to the item when it was imported to EPM Workspace.

versionNumber, part of the optional `?version` argument, is the version number of the item. If you do not supply the version number, the SmartCut retrieves the latest version.

mimeType is the MIME type of the item.

The SmartCut examples that follow invoke EPM Workspace and retrieve the target document, report, or form stored in the system. If the user is not logged into a EPM Workspace servlet, the login page displays prior to processing the URL and retrieving the item.

This example shows a SmartCut that retrieves an e-mail address list stored in a folder called *Sales*, which is a subfolder of *SampleContent*. This document is an HTML file, so the document is returned to the browser for viewing.

```
http://apollo/workspace/browse/get/SampleContent/Sales/Email_Address_list
```

The SmartCut in this example retrieves a report that was generated from a previously run Production Reporting job. You can also run the report instantly. See for more information.

```
http://apollo/workspace/browse/get/SampleContent/Sales/sales.htm
```

Note: `sales.htm` is the name associated with the report when it was imported to EPM Workspace after the Production Reporting job ran.

Getting Report Output

The SmartCut **get** can be used to return job output, like other EPM Workspace items. It works differently for Production Reporting jobs or non-Production Reporting jobs as shown by these examples.

Production Reporting Jobs

For an Production Reporting program, specify the SQRProgramOutput item, as follows:

```
http://hostname:port/workspace/browse/get/folderPath/Job-Name?jobOutput=true
```

This SmartCut returns the latest output for the job. Specifically, for Production Reporting programs, it returns the HTML output file generated by the most recent run of the Production Reporting program.
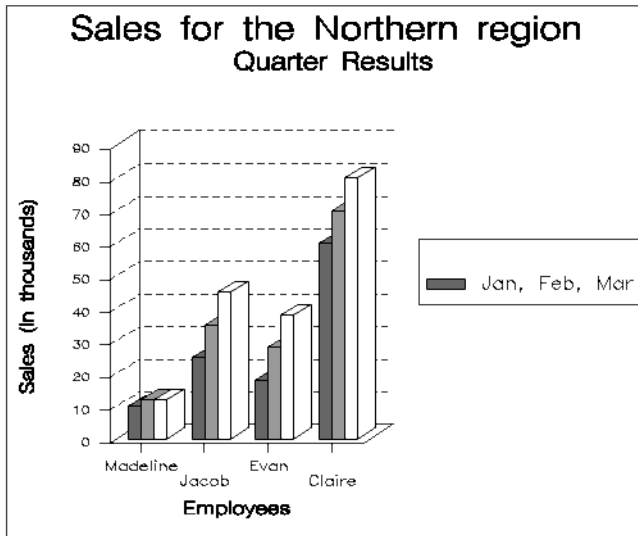
For the Sales Charts job, which is in the Sales folder under SampleContent, the SmartCut would be:

```
http://apollo/workspace/browse/get/SampleContent/Sales/Output%20from%20Sales%20Charts
```

Note: Remember to use %20 in place of a space character. Keep in mind that SmartCuts are case-sensitive.

This example returns this report output:

Non-Production Reporting Jobs

For a non-Production Reporting report, the SmartCut **get** should specify the OutputCollection item, as follows:

```
http://hostname:port/workspace/browse/get/folderPath/JobName?mimetype=mimeType
```

This SmartCut returns the latest output for the job.

To look up the exact MIME type, in Explore, select an item of the type you want to get, right-click and select Properties from the shortcut menu, then click Advanced.

For a report named Expenses in the Finance folder that is run by an application called ReportGenerator, the SmartCut to get the output would be:

```
http://apollo/workspace/browse/get/Finance/Expenses?mimetype=ReportGenerator%20Job
%20Output
```

Note that the `?mimetype=type` parameter is necessary, because the job itself is called Expenses. Without the `mimetype` parameter, the SmartCut would run the job. By specifying the MIME type, you can return only the output artifacts.

Getting One Job Output Artifact

If a job runs multiple times and you want to get only the latest output, use the `latest=true` query parameter. This parameter distinguishes between output artifacts of the same name with

different dates. If the `latest=true` parameter is present, only the most recent output is returned.

Suppose an Production Reporting program generates reports in various formats, including PDF. To retrieve the PDF output file from the most recent run of the job, you would use a **get** SmartCut resembling this:

`http://`*hostname:port*`/workspace/browse/get/`*folderPath/JobName.pdf?latest=true*

Without the `latest=true` parameter, if the job had been run more than once, the SmartCut returns a listing of all PDF output files.

The `latest=true` parameter is used where there are multiple items of the same name. If there are multiple *versions* of a versioned item, the SmartCut **get** returns only the latest version by default. If you want a another version, specify it by number in the `?version=`*versionNumber* parameter.

run Command

You can direct the Browse servlet to run a job stored in EPM Workspace by simply invoking a URL and passing the input parameters as part of the SmartCut **run**.

Note: Passing parameters in the SmartCut works for Oracle's Hyperion® SQR® Production Reporting jobs and generic jobs, but not for Interactive Reporting jobs. For Interactive Reporting jobs, see .

To run the report or job, use the **run** SmartCut, which has this syntax:

`http://`*hostname:portNumber*`/workspace/browse/run/`*folderPath/JobName*`[?`
`dbuser=`*dbUser*`&dbpass=`*dbPassword*`][&Param_1=`*param_1*`&Param_2=`*param_2*`]`
`[&AskParam_1=`*askparam1*`]...`

Optionally, you can include these variables in the SmartCut **run**:

- **dbuser, dbpass**

  These variables are used to establish the database connection for running a report. They are optional in the SmartCut because they can alternatively be configured when importing a job.

- **Param_1, Param_2, ... (or ASKparam_1, ... for ASK parameters)**

  Note the initial capital letters.

  These parameters are used to execute the report or job. If a report does not require parameters, the SmartCut need not contain parameters. If the report or job requires parameters, and some or all required parameters are omitted in the SmartCut or POST operation, the parameter form displays.

This example is a SmartCut that specifies a report to be run but includes no parameters. Because no parameters are specified, the input parameter form is retrieved and presented for the user to provide the necessary input parameters prior to running the job.

```
http://apollo/workspace/browse/run/SampleContent/Sales/Sales%20Charts
```

This example shows a URL that specifies a report to be run. Input parameters are specified as part of the SmartCut **run**. This example directly invokes the job and return the report output.

```
http://apollo/workspace/browse/run/SampleContent/Production/Inventory_Information/
Supplier_Exception_Re-port?Param_1=ALL
```

This example shows a SmartCut that fills in all required parameters. Using this URL results in Explore running the report using the parameters entered. Explore then displays the report output.

```
http://apollo/workspace/browse/run/SampleContent/Sales/Sales%20Charts?
user=sanderson&pass=sa357sf607&server=mercury&Param_1=SouthEastern&Param_2=4&Param_3=LIN
E&Param_4=YES
```

Note: To determine what input parameters are needed for the report, view the job properties in Explore.

SmartCut Variables for Interactive Reporting Documents and Jobs

The SmartCut variables for Interactive Reporting documents, jobs, and job output allow fuller access to Interactive Reporting documents through other applications. You can use these variables to produce URLs with access to EPM Workspace applications, documents, jobs, and job output.

Some variables can be used with the get or run command, and some variables work with Interactive Reporting (*bqtype*=ihtml), while other variables work with Interactive Reporting jobs only. For more information on each variable see the section on that variable.

Variable	Get	Run	Interactive Reporting Only	Interactive Reporting Jobs Only
bqtype	X	X		
mimetype or filename	X	X		
dest	X	X		
version	X			
jobOutput	X			X
SectionName	X	X	X	
Toolbar	X	X	X	
BoundRect	X	X	X	
ShowForm		X		X
Limit and LimitValue		X		X

Note: All variables are case-sensitive.

bqtype

Use *bqtype* to specify which application to use to display the Interactive Reporting document or job output. If *bqtype* is specified, the `filename` and `mimetype` variables are not necessary and are ignored.

If *bqtype* equals `plugin`, the system checks that the user has permission to use the Interactive Reporting Web Client. If the user has permission, the Interactive Reporting document or job output is opened using the Interactive Reporting Web Client. If the user does not have permission, an error message is displayed.

If the *bqtype* equals `ihtml`, the system checks that the user has permission to use EPM Workspace. If the user has permission, the Interactive Reporting document or job output is opened using EPM Workspace. If the user does not have permission, an error message is displayed.

If *bqtype* is not set, the system checks if the user has permission to use Interactive Reporting first. If the user does not have permission, the system checks to see if the user has permission to use the Oracle's Hyperion® Interactive Reporting Web Client. If the user does not have permission to use either application, an error message is displayed.

***bqtype* syntax:**

`http://`*hostname*`:`*portNumber*`/workspace/browse/`**get**`/`*folderPath*`?bqtype=<plugin/ihtml>`

mimetype or filename

Use *mimetype* to specify the Interactive Reporting job output type to open. If multiple artifacts of a MIME type exist, a job listing is displayed. Use *filename* to specify a file. If *filename* is specified, the `mimetype` is not necessary and is ignored. The default `mimetype` is BQY.

***mimetype* syntax:**

`http://`*hostname:portNumber*`/workspace/browse/get/`*folderPath*`?mimetype=text/html&jobOutput=true`

`http://`*hostname:portNumber*`/workspace/browse/run/`*folderPath*`?mimetype=text/html&ShowForm=false`

***filename* syntax:**

`http://`*hostname:portNumber*`/workspace/browse/get/`*folderPath*`?filename=`*filename*`&jobOutput=true`

`http://`*hostname:portNumber*`/workspace/browse/run/`*folderPath*`?`
`filename=`*filename*`&ShowForm=false`

Tip: The file name can be accessed by mousing over the job output icon or Interactive Reporting icons. The MIME type can be accessed by mousing over the link of the job output or the Interactive Reporting document.

dest

Use the *dest* variable instead of *folderpath* to specify the UUID of an item when there are multiple documents with the same name.

Instead of this syntax:

`http://`*hostname*`:`*portNumber*`/workspace/browse/`**get**`/`*folderPath*`/`

Use this syntax:

`http://`*hostname*`:`*portNumber*`/workspace/browse/`**get**`?dest=UUID`

version

Use *version* to specify the version number. If this is not specified the latest version is used.

version **syntax:**

`http://`*hostname*`:`*portNumber*`/workspace/browse/`**get**`/`*folderPath*`?bqtype=<plugin/ihtml>&version=`*versionNumber*

SectionName

Use to reference a section from an Interactive Reporting document. If *SectionName* is not specified, the default section specified in the Oracle's Hyperion® Interactive Reporting document is displayed.

SectionName **syntax:**

`http://`*hostname:portNumber*`/workspace/browse/`**get**`/`*folderPath*`?bqtype=ihtml&SectionName=<name>`

Toolbar

Use to specify whether the toolbar is displayed. If a toolbar setting is not specified, the toolbar is not displayed.

If *Toolbar* equals `Standard`, the full toolbar is displayed which includes access to help. If *Toolbar* equals `Navigation`, only the tool bar for paging is displayed.

Toolbar **syntax:**

`http://`*hostname:portNumber*`/workspace/browse/`**get**`/`*folderPath*`?bqtype=ihtml&SectionName=<name>&Toolbar=Standard`

BoundRect

Use to provide the Web browser height and width.

BoundRect syntax:

```
http://hostname:portNumber/workspace/browse/get/folderPath?
bqtype=ihtml&SectionName=<name>&ToolBar="Navigation"&BoundRect=<w,h>
```

Note: *w* is width, *h* is height, in pixels.

jobOutput

Use this variable to display job output after a job is run.

jobOutput syntax:

```
http://hostname:portNumber/workspace/browse/get/folderPath?bqtype=ihtml&jobOutput=true
```

ShowForm

Use this variable to give the user access to set or change limits. The default is true. *ShowForm* must be false when passing parameters through the URL.

ShowForm syntax:

```
http://hostname:portNumber/workspace/browse/run/folderPath?bqtype=plugin&ShowForm=true
```

```
http://hostname:portNumber/workspace/browse/run/folderPath?
bqtype=plugin&ShowForm=false&Limit1=q1\state&LimitValue1=CA&LimitValue1=MI
```

Limit and LimitValue

Multiple variable limits can be passed using the *Limit* and *LimitValue* parameters when *ShowForm* equals false. Each *Limit* can have multiple *LimitValue*s.

Limit and **LimitValue** syntax:

```
http://hostname:portNumber/workspace/browse/run/folderPath?
bqtype=ihtml&ShowForm=false&Limit1=q1\state&LimitValue1=CA&LimitValue1=MI&Limit2=q2\Prod
uct&LimitValue2=A001
```

To pass date limits you must define the day, month, year, time, and time zone as this syntax shows:

```
http://hostname:portNumber/workspace/browse/run/folderPath?
bqtype=ihtml&jobOutput=true&ShowForm=false&Limit1=q1/
SaleDate&LimitValue1=dd=12,mm=12,yyyy=2004,time=13:30:00,tz=America/Los_Angeles
```

Date Syntax	Description
dd	day
mm	month
yyyy	year
hh	hour
mm	minute
ss	second
tz	time zone

The default hour, minute, second is midnight. The default time zone is the time zone of the server's location. This field uses recognized Java time zones.

You can use constants for dates:

Name	Value
First of Month	FirstOfMonth
First of Previous Month	FirstOfPreviousMonth
First of Quarter	FirstOfQuarter
First of Week	FirstOfWeek
First of Year	FirstOfYear
Friday	Friday
Last of Month	LastOfMonth
Last of Previous Month	LastOfPreviousMonth
Last of Quarter	LastOfQuarter
Last of Week	LastOfWeek
Last of Year	LastOfYear
Monday	Monday
Previous Business Day	PreviousBusinessDay
Today	Today
Yesterday	Yesterday

The syntax when using a date constant is:

```
http://hostname:portNumber/workspace/browse/run/folderPath?
bqtype=ihtml&ShowForm=false&Limit1=q1\SaleDate&LimitValue1=FirstOfMonth,tz=America/
Los_Angeles
```

Note: The Ignore and Don't Prompt flag defaults apply when a job is run. When limit values are passed using a SmartCut and the Ignore flag is set to true, the new values are ignored. To change these flags, modify the properties of the job through Explore before running the job.

SmartCut Examples

Two examples for using SmartCuts are provided in these topics:

- "Example: Accessing EPM Workspace Content from Web Applications" on page 81
- "Example: Using SmartCuts in HTML Forms" on page 83

Example: Accessing EPM Workspace Content from Web Applications

Using SmartCuts, you can integrate EPM Workspace into your intranet, your home page, or Web-based applications, as the example in this section illustrates. One click from your intranet (or home page) takes you to EPM Workspace information.

This example creates a "dashboard" or "kiosk" with several buttons that access Web and EPM Workspace content. The example shows how you can create a central starting point for accessing all one's electronically available information. The buttons of the example "kiosk" page go to:

- Your company's intranet
- Your company's Web site
- Your partners' extranets
- EPM Workspace content
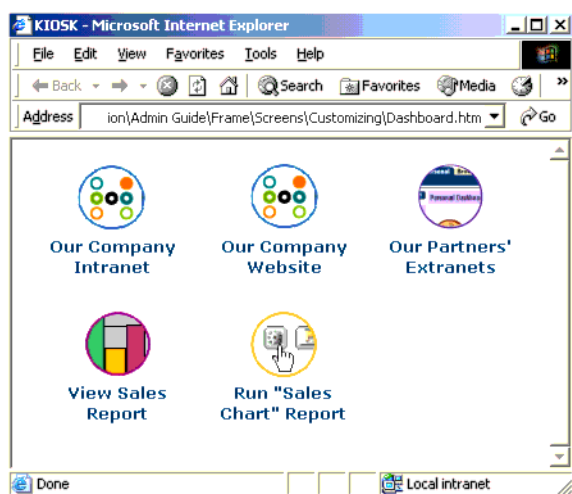
The two lower buttons access EPM Workspace content using SmartCuts that differ:

- Displaying report output using the SmartCut **get**.
- Running a report using the SmartCut **run**.

➤ To create the example, follow these steps:

1 **Find or create the graphics you want to use for the kiosk buttons, and save the graphics files in the** `wsmedia/personalize` **subdirectory of your Web application deployment directory (**`/workspace` **by default).**

2 **In Notepad (or any editor that does not change the code), enter the code below.**

If you are viewing this documentation in Adobe Acrobat Reader 4.0 or above, you can copy the text using the Text Select Tool, and paste it into your editor.

The code shown is `kiosk0.htm`, with the SmartCut names shown in boldface. Text you should replace (in step 3) is shown in bold italics:

```
<HTML>
<HEAD>
<TITLE>KIOSK</TITLE>
</HEAD>
<BODY>
<TABLE>
<TR>
<TD> <A HREF="http://intranet"><IMG border=0 src="Install Home/workspace/wsmedia/
personalize/graphic1"></A></TD>
<TD>&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp</TD>
<TD><A HREF="http://website"><IMG border=0 src="Install Home/workspace/wsmedia/
personalize/graphic2"></A></TD>
<TD>&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp&nbsp</TD>
<TD><A HREF="http://extranetsList.htm"><IMG border=0 src="Install Home/workspace/
wsmedia/personalize/graphic3"></A></TD>
<TD>&nbsp</TD>
</TR>
<TR VALIGN="TOP">
<TD ALIGN="CENTER"> <FONT FACE="Arial" COLOR="saddlebrown" SIZE=+0><B>Our
Company<BR>Intranet </B></FONT></TD>
<TD>&nbsp</TD>
<TD ALIGN="CENTER"> <FONT FACE="Arial" COLOR="saddlebrown" SIZE=+0><B>Our
Company<BR>Website</B></FONT></TD>
<TD>&nbsp</TD>
<TD ALIGN="center"> <FONT FACE="Arial" COLOR="saddlebrown" SIZE=+0>
<B>Our Partners'<BR>Extranets</B></FONT></TD>
</TR>
<TR>
<TD>&nbsp<BR>&nbsp</TD>
<TD>&nbsp</TD>
<TD>&nbsp</TD>
</TR>
<TR>
<TD>&nbsp</TD>
<TD> <A HREF="http://servletsHost:servletsPort/root/browse/get/SampleContent/
Production/Inventory_Information/Central_Facility_Report?
user=username&pass=password&server=GSMhost:GSMport"><IMG border=0 src="Install Home/
workspace/wsmedia/personalize/graphic5"></A></TD>
<TD>&nbsp</TD>
<TD> <A HREF="http://servletsHost:servletsPort/root/browse/run/SampleContent/Sales/
Sales%20Charts?user=username&pass=password&server=GSMhost:GSMport"> <IMG border=0
src="Install Home/workspace/wsmedia/personalize/graphic6"></A></TD>
</TR>
<TR VALIGN="TOP">
```

```
<TD>&nbsp</TD>
<TD ALIGN="center"> <FONT FACE="Arial" COLOR="saddlebrown" SIZE=+0><B>View<BR>Sales
Report</B></FONT></TD>
<TD>&nbsp</TD>
<TD ALIGN="center"> <FONT FACE="Arial" COLOR="saddlebrown" SIZE=+0><B>Run "Sales
<BR>Chart" Report</B></FONT></TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

3 **Make these substitutions in the code:**

- Replace *intranet* with the URL for your company's intranet.

- Replace *Install Home* with the name of your installation directory, which is the directory containing the servlets directory.

- Replace `root/browse` with the Web application deployment directory name or alias, if it is set differently in your Web server software.

- Replace *graphic1, graphic2*, and so on. with file names of your button images.

- Replace *website* with the URL for your company's public Web site.

- Replace *extranetsList.htm* with the URL for a page containing links to your partners' extranets.

- Replace *servletsHost* with the name of the host on which the servlets are installed.

- Replace *servletsPort* with the port number for the Browse servlet.

- Replace the path of the example "Central Facility Report" with that of your report output artifact.

- Replace the path of the example "Sales Charts" with that of your report program.

- Replace *username* with user name.

- Replace *password* with password.

    Note: Supplying the user name and password in SmartCuts is not necessary if you implemented transparent login. See the *Oracle Enterprise Performance Management Workspace Administrator's Guide* for more information.

- Replace *GSMhost* with the name of the host running the Global Service Manager.

- Replace *GSMport* with the port number for the Global Service Manager (default is 6800).

4 **Save the file as** `kiosk.htm`**.**

5 **Incorporate** `kiosk.htm` **into your Web-based application or make it the home page in your browser.**

Example: Using SmartCuts in HTML Forms

SmartCut variables can be used in HTML forms. This example shows the code for a simple form produced using HTML and SmartCut variables:
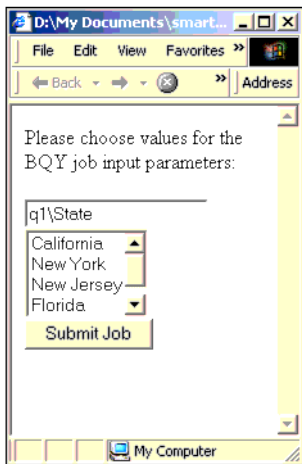
Note: HTML form METHOD=POST is UTF-8 compatible for double byte character encoding.

```
<HTML>
<BODY>
Please choose values for the Interactive Reporting job input parameters:
<FORM name=myJobForm METHOD=POST ACTION="http://servletsHost:servletsPort/workspace/
browse/run/myFolder/myBQYJob" >
<input type=text name=Limit1 value="q1\State"></input>
<select name=LimitValue1 multiple>
<option value="CA">California</option>
<option value="NY">New York</option>
<option value="NJ">New Jersey</option>
<option value="FL">Florida</option>
<option value="HI">Hawaii</option>
</select>
<input type=hidden name=ShowForm value=false>
<input type=hidden name=user value=username>
<input type=hidden name=pass value=password>
<input type=hidden name=server value=servletsHost:servletsPort>
<input type=submit name=submitbutton value="Submit Job">
</FORM>
</BODY>
</HTML>
```

8

Extended Services

Integrating Extended Services

You can integrate services beyond those provided by EPM Workspace that are developed by your organization or a third party with EPM Workspace servlets. *Extended services* may augment the functionality of an existing EPM Workspace feature; for example, extending notifications into a workflow feature. After integration, end users have seamless access to these extended services through the servlets in the context of a regular session.

Note: For an extended service to be integrated with EPM Workspace servlets, the resources (files or programs) specified by the extended service must exist in the same Web server context as the EPM Workspace servlets.

Topics that provide information about using extended services:

- "Configuration File for Extended Services" on page 85
- "URL to Access Extended Services" on page 86
- "Examples Entries and URLs for Service Properties" on page 87
- "Aggregation of Query Parameters" on page 88
- "Relative Links in Extended Service HTML Output" on page 89
- "Local Resource Management" on page 89

Configuration File for Extended Services

Configuration information for extended services is stored in the `services.properties` file, located in `deployment/WEB-INF/config`.

The default `deployment` location depends on your servlet engine. For example, for a WebLogic installation on Windows, the default deployment location is:

`Install Home\AppServer\InstalledApps\WebLogic\9.x\Workspace\applications \workspace`

To help avoid name collisions with internal EPM Workspace commands or commands of other services, every extended service has a URL prefix; however, you do not need to specify a prefix in a `services.properties` entry. If an entry lacks a prefix, the default prefix *ext* is assigned to it.

Each extended service must have two entries in `services.properties`: one for each of the two properties, resource and resource type. The *resource* is the file or program that handles requests for the extended service. The resource may be, for example, an HTML or JSP file or a servlet. The *resource type* indicates whether the resource is represented by a path (`$PATH$`) or a name (`$NAMED$`).

If the resource is identified by a file or program path, the *resource value* should be the path. A path is typically an absolute path from the Web server context root, starting with the "/" (forward slash) character. If the resource is identified by a named resource such as a servlet, the resource value should be the name of the servlet (as known to the Web server).

You can add query parameters to the resource value if the resource is of the type `$PATH$`. This lets you pass parameters with static values to the resource.

Entries in `services.properties` should use this format:

```
[urlPrefix.]serviceCommandName.resource=PathOrName[?param1=value1&param2=value2]
[urlPrefix.]serviceCommandName.type=$PATH$|$NAMED$
```

Examples of `services.properties` entries are shown in .

Note: Command names are case-sensitive.

URL to Access Extended Services

You can access extended services using EPM Workspace servlets by specifying a URL in this format:

protocol://host:port/Hyperion/servlet/servicePrefix/ serviceCommand[?queryParameters]

where:

protocol is HTTP or HTTPS

host is the name or IP address of the host machine on which theEPM Workspace servlets are running

port is the port number of the Web server on which the servlets are running

servlet is the name of a EPM Workspace servlet; it is one of these case-sensitive literal strings: `browse`, `personalpages`, `viewmanager`, or `administration`

servicePrefix is the name of the prefix for the extended service. Prefixes help avoid name collisions with internal EPM Workspace commands or commands of other services. The default service prefix is *ext*.

serviceCommand is the required command name to invoke the extended service

queryParameters are one or more optional parameters that the extended service takes as input

For example:

```
http://apollo:11122/Hyperion/jobmanager/ext/myExtendedService
```

Examples Entries and URLs for Service Properties

These examples of `service.properties` entries and URLs illustrate service configurations that differ. The examples involve a dinner menu service (command name is *dinnerMenu*) for late-working employees.

Note: In this section, the *protocol://host:port* portion of each example URL is omitted for readability and to emphasize the portion specific to extended services. In actual usage, the full URL is necessary to access the extended service.

Example 1

For a named resource where the dinnerMenu service is handled by a servlet known to the Web server as *DinnerMenuServlet*, and where no prefix is specified, the service.properties entries are as follows:

```
dinnerMenu.resource=DinnerMenuServlet
dinnerMenu.type=$NAMED$
```

The type is specified as $NAMED$ because this is a named resource rather than a file path. Because no URL prefix was specified, the default prefix is used. The URL to access this extended service is:

```
/Hyperion/servlet/ext/dinnerMenu
```

If the service takes query parameters at runtime, the parameters are specified at the end of the URL. Assuming that the dinnerMenu service takes two parameters, *dayOfWeek* and *month*, the URL is:

```
/Hyperion/servlet/ext/dinnerMenu?dayOfWeek=Tuesday&month=September
```

Example 2

Same as Example 1, except that the URL prefix is specified as *hyperion*. The `service.properties` entries are:

```
hyperion.dinnerMenu.resource=DinnerMenuServlet
hyperion.dinnerMenu.type=$NAMED$
```

The URL to access this extended service is:

```
/Hyperion/servlet/hyperion/dinnerMenu
```

Example 3

For a resource expressed as a path, where the dinner menu service is handled by a servlet, the `service.properties` entries are:

```
dinnerMenu.resource=/servlet/DinnerMenuServlet dinnerMenu.type=$PATH$
```

Because no URL prefix was specified, the default prefix is used. The URL to access this extended service is:

```
/Hyperion/servlet/ext/dinnerMenu
```

Example 4

Same as Example 3, but passing static query arguments using the *resource* entry. The query arguments are named *startTime* and *location*, with values *19:00* and *CA*, respectively. The `service.properties` entries are as follows. (Differences from Example 3 are in bold.)

```
dinnerMenu.resource=/servlet/DinnerMenuServlet?startTime= 19:00&location=CA
dinnerMenu.type=$PATH$
```

The URL to access this extended service is the same as that in Example 3:

```
/Hyperion/servlet/ext/dinnerMenu
```

Example 5

For a file resource, where requests for the dinnerMenu service are handled by a JSP with the path `/scripts/dinnerMenu.jsp`, the `service.properties` entries are:

```
dinnerMenu.resource=/scripts/dinnerMenu.jsp
dinnerMenu.type=$PATH$
```

Because no URL prefix was specified, the default prefix is used. The URL to access this extended service is:

```
 /Hyperion/servlet/ext/dinnerMenu
```

Example 6

Same as Example 5, except that requests for the dinnerMenu service are handled by an HTML file, whose path is `/wsmedia/docs/dinnerMenu.html`. The `service.properties` entries are:

```
dinnerMenu.resource=/wsmedia/docs/dinnerMenu.html dinnerMenu.type=$PATH$
```

Aggregation of Query Parameters

If query parameters are specified both in the URL and in the definition of a `$PATH$-type` resource in `services.properties`, the query string contains the aggregate of the two sets of query parameters.

Suppose `services.properties` contains these entries:

```
dinnerMenu.resource=/servlet/DinnerMenuServlet?param1=1&param2=2
dinnerMenu.type = $PATH$
```

and the URL to access this service is:

```
/servlet/WebClient/ext/dinnerMenu?param3=3&param4=4
```

The query string contains `?param1=1&param2=2&param3=3&param4=4`

Relative Links in Extended Service HTML Output

If there are relative links in HTML content returned by an extended service, these links are relative to the URL. To specify another path for relative links, precede the content with an HTML `<BASE>` tag that specifies the desired base URL.

Local Resource Management

Because requests for extended services come by way of EPM Workspace servlets, EPM Workspace performs user authentication and session creation and management for an extended service. The extended service does not need to re-authenticate the user; however, the extended service may require information about the user's session to manage local resources associated with the session.

When forwarding a request to an extended service, a EPM Workspace servlet passes a serialized session token artifact containing secure information about the user's session. The string from the session token may be retrieved from the `com.brio.one.client.SessionToken` attribute of the `javax.servlet.http.HttpSession` artifact. After the session token is obtained in its string format, it can be used to join the session through the EPM Workspace API.

If you want to use a serialized session token artifact to get an API session, you must include *Install Home*/SDK/lib/rmapi.jar in the Web application or Application server classpath.

Displaying Extended Service Content on Personal Pages

You can display content from an extended service on a Personal Page by importing an external link to the extended service.

In cases where the system has multiple installations of the servlets, and it is not known in advance which one will be used to display the extended-service content, you should omit the protocol, host, and port information from the URL for the external link. The imported URL has this format:

/workspace/servlet/serviceURLPrefix/serviceCommandName

A

How to Use the API Sample Programs

Preparing to Use the Sample Java Programs

The sample Java programs are in `<EPM_ORACLE_HOME>/products/biplus/SDK/samples/java`. Place custom Java programs in `<MIDDLEWARE_HOME>/user_projects/epmsystem1/ReportingAnalysis/SDK/src`.

To run these programs, compile them `using jc.bat (.sh)` and then use `execapi.bat (.sh)` from a command line to pass the necessary arguments and run them. See "Compiling the Sample Programs" on page 92.

The Batch Driver sample program can be run without compiling. A compiled version of the program is included in `<EPM_ORACLE_HOME>/common/raframeworkrt/11.1.2.0/lib/rmapi.jar` with the other API class files.

These topics describe the setup procedures you must perform before using the samples:

- "Validating EPM Workspace Connections" on page 91
- "Compiling the Sample Programs" on page 92

Note: The Reporting and Analysis and EPM Workspace SDK Installer are automatically installed when you install any Oracle's Hyperion Reporting and Analysis product.

Validating EPM Workspace Connections

Test that EPM Workspace is running on the server you are using.

➤ To validate the connection to EPM Workspace:

1 In a Web browser, enter a URL of the form `http://`*host*`:`*port*`/workspace`. For example:

`http://localhost:45000/workspace`

2 Enter a valid username and password.

Compiling the Sample Programs

All sample programs, except the batch driver sample program, must be compiled using a Java compiler before you can run them. Sample programs from `<EPM_ORACLE_HOME>/products/biplus/SDK/samples` and custom programs from `<EPM_ORACLE_HOME>/ReportingAnalysis/SDK/src` are complied and placed in `<MIDDLEWARE_HOME>/user_projects/epmsystem1/ReportingAnalysis/SDK/classes`. Configure the environment with the Java compiler location.

The `jc.bat` file, which is generated by the installer, contains the environment information needed to compile the sample programs.

These topics describe steps and information for compiling the sample programs:

About the Java Compiler

Use a Java compiler to compile all sample programs. A Java compiler is not needed if you are using the compiled batch driver class, `com.squibe.rm.BatchDriver`, provided by Oracle that is in `<EPM_ORACLE_HOME>/common/raframeworkrt/11.1.2.0/lib/rmapi.jar`.

About execapi.bat

The `execapi.bat` file calls `set_sdk_env.bat` to set up the environment and run the sample files. It is generated by the installer and is in `<MIDDLEWARE_HOME>/user_projects/epmsystem1/bin/ReportingAnalysis/SDK`.

Sample execapi.bat File

```
@echo OFF
@rem -------------------------------------------------------------
@rem Copyright @ 2009, Oracle Coproration. All Rights Reserved
@rem -------------------------------------------------------------
setlocal

set SET_SDK_ENV="set_sdk_env.bat"
if not exist %SET_SDK_ENV% (
    echo ERROR: Set SDK environment script %SET_SDK_ENV% file does not exist.
    goto end
)
call %SET_SDK_ENV%


set SYSPROPS=-Ddirectory="%EPM_ORACLE_HOME%\products\biplus\SDK" -
Djava.io.tmpdir="%EPM_ORACLE_INSTANCE%\diagnostics\logs\ReportingAnalysis\SDK\temp" -
Djava.util.logging.config.class=oracle.core.ojdl.logging.LoggingConfiguration -
Doracle.core.ojdl.logging.config.file=%EPM_ORACLE_INSTANCE%\config\ReportingAnalysis\SDK
```

```
\logging.xml

set LOGPROP=-Dlogging.folder=%EPM_ORACLE_INSTANCE%\diagnostics\logs\ReportingAnalysis


pushd %EPM_ORACLE_INSTANCE%\ReportingAnalysis\SDK\classes

"%JAVA_HOME%\bin\java" %SYSPROPS% %LOGPROP% -classpath "%SDK_CLASSPATH%" %*

popd

:end
Endlocal
```

Using jc.bat to Compile the Sample Programs

Similar to `execapi.bat`, the `jc.bat` file sets up the environment and compiles the sample files. It is generated by the installer and is in `<MIDDLEWARE_HOME>/user_projects/epmsystem1/bin/ReportingAnalysis/SDK`. Update `jc.bat` with the location of an appropriate Java compiler to compile the sample programs.

➤ To update `jc.bat`, replace this line:

```
set JAVAC_EXE="%JAVA_HOME\bin\javac.exe"
```

with the full path to the compiler. For example:

```
set JAVA_EXE="c:\jdk150-04\bin\javac.exe"
```

Sample jc.bat File

```
@echo OFF
@rem --------------------------------------------------------------
@rem Copyright @ 2009, Oracle Coproration. All Rights Reserved
@rem --------------------------------------------------------------
setlocal

if "%1" == "" (
    echo ERROR: Required parameter is missing. Usage: jc ^
    goto end
)

@rem OS can already have JAVA_HOME environment variable
if not "%JAVA_HOME%" == "" (
    set OS_JAVAC_EXE=%JAVA_HOME%\bin\javac.exe
)

set SET_SDK_ENV="set_sdk_env.bat"
if not exist %SET_SDK_ENV% (
    echo ERROR: Set SDK environment script %SET_SDK_ENV% file does not exist.
    goto end
)
call %SET_SDK_ENV%


@rem --------------------------------------------------------------
```

```
@rem Please, specify full path to javac.exe here
@rem                       vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
@rem ----------------------------------------------------------------
if "%JAVAC_EXE%" == "" (
    set JAVAC_EXE=%JAVA_HOME%\bin\javac.exe
)
@rem ----------------------------------------------------------------
@rem                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
@rem ----------------------------------------------------------------


if not exist "%JAVAC_EXE%" (
    if not exist "%OS_JAVAC_EXE%" (
        echo ERROR: Cannot find Java Compiler '%JAVAC_EXE%'
        goto end
    )

    echo WARNING: Cannot find Java Compiler '%JAVAC_EXE%'
    echo Trying to use '%OS_JAVAC_EXE%' instead
    set JAVAC_EXE=%OS_JAVAC_EXE%
)


if exist "%EPM_ORACLE_INSTANCE%\ReportingAnalysis\SDK\src\%1" (
    set FILE_PATH=%EPM_ORACLE_INSTANCE%\ReportingAnalysis\SDK\src\
    goto compile
)

if exist "%EPM_ORACLE_HOME%\products\biplus\SDK\samples\java\%1" (
    set FILE_PATH=%EPM_ORACLE_HOME%\products\biplus\SDK\samples\java\
    goto compile
)

echo ERROR: Class %1 is not found.
goto end


:compile
"%JAVAC_EXE%" -sourcepath "%EPM_ORACLE_HOME%\products\biplus\SDK\samples
\java";"%EPM_ORACLE_INSTANCE%\ReportingAnalysis\SDK\src" -classpath "%SDK_CLASSPATH%"
"%FILE_PATH%%*" -d "%EPM_ORACLE_INSTANCE%\ReportingAnalysis\SDK\classes"

popd

:end
Endlocal
```

Sample set_sdk_env.bat File

```
@echo OFF
@rem ----------------------------------------------------------------
@rem Copyright @ 2009, Oracle Coproration. All Rights Reserved
@rem ----------------------------------------------------------------

call ..\..\setEnv.bat

if "%EPM_ORACLE_HOME%" == "" (
```

```
    echo ERROR: The EPM_ORACLE_HOME environment variable is not defined correctly.
    goto end
)

if "%EPM_ORACLE_INSTANCE%" == "" (
    echo ERROR: The EPM_ORACLE_INSTANCE environment variable is not defined correctly.
    goto end
)

set JAVA_HOME=%EPM_ORACLE_HOME%\common\JRE\Sun\1.6.0

set CIScssJar=%EPM_ORACLE_HOME%\common\CSS\11.1.2.0\lib\css.jar;%EPM_ORACLE_HOME%\common
\CSS\11.1.2.0\lib\ldapbp.jar;%EPM_ORACLE_HOME%\common\SharedServices\11.1.2.0\lib
\commons-httpclient-3.0.jar
set CISojdlJar=%EPM_ORACLE_HOME%\common\loggers\ODL\1.0.0\lib\ojdl.jar
set CISdmsJar=%EPM_ORACLE_HOME%\common\loggers\ODL\1.0.0\lib\dms.jar
set CISjaxpJar=%EPM_ORACLE_HOME%\common\XML\JAXP\1.2.2\xercesImpl.jar;%EPM_ORACLE_HOME%
\common\XML\JAXP\1.2.2\sax.jar;%EPM_ORACLE_HOME%\common\XML\JAXP\1.2.2\jaxp-api.jar;
%EPM_ORACLE_HOME%\common\XML\JAXP\1.2.2\xsltc.jar;%EPM_ORACLE_HOME%\common\XML\JDOM\0.9.
0\jdom-b9.jar

set INSTALLLIB=%EPM_ORACLE_HOME%\products\biplus\lib
set RAFRAMEWORKRT_LIB=%EPM_ORACLE_HOME%\common\raframeworkrt\11.1.2.0\lib

set SDK_CLASSPATH=.;%installhome%\SDK;%RAFRAMEWORKRT_LIB%\foundation.jar;
%EPM_ORACLE_HOME%\common\SharedServices\11.1.2.0\lib\xmlrpc-2.0.1.jar;%EPM_ORACLE_HOME%
\common\SharedServices\11.1.2.0\lib\audit-client.jar;%EPM_ORACLE_HOME%\common
\JakartaCommons\commons-pool-1.3.jar;%EPM_ORACLE_HOME%\common\JakartaCommons\commons-
codec-1.3.jar;%RAFRAMEWORKRT_LIB%\rmapi.jar;%INSTALLLIB%;%CISojdlJar%;%CISdmsJar%;
%RAFRAMEWORKRT_LIB%\iona63.jar;%RAFRAMEWORKRT_LIB%\comutil1_01.jar;%RAFRAMEWORKRT_LIB%
\logi.crypto1.1.2.jar;%CISjaxpJar%;%EPM_ORACLE_HOME%\products\biplus\common\SQR\lib
\xmlParserAPIs.jar;%EPM_ORACLE_HOME%\common\config\11.1.2.0\lib\registry-api.jar;
%EPM_ORACLE_HOME%\products\biplus\common\SQR\lib\spf.jar;%INSTALLLIB%
\commons_collections.jar;%RAFRAMEWORKRT_LIB%\commons-logging.jar;%EPM_ORACLE_HOME%
\common\JakartaCommons\commons-lang-2.1.jar;%installhome%\lib\bqservice.jar;
%EPM_ORACLE_HOME%\common\JDBC\DataDirect\3.7\lib\hyjdbc.jar;%CIScssJar%

:end
```

Compiling Sample Programs

You compile all of the sample programs in the same way. This procedure uses the `Login` sample program.

➤ To compile the `Login` sample program:

1 **At a command line, change to the directory that contains** `jc.bat`.

   For example, `<MIDDLEWARE_HOME>/user_projects/epmsystem1/bin/ReportingAnalysis/SDK`.

2 **Enter** `jc Login.java` **and press** **Enter** **to compile the program.**

3 **Check that** `Login.class` **is in** `<MIDDLEWARE_HOME>/user_projects/epmsystem1/ReportingAnalysis/SDK/classes`..

Running the Login Sample Program

You should run the `Login` sample program before you run the other sample programs to check connection to EPM Workspace.

➤ To run the `Login` sample program:

1 **At a command line, change to the directory that contains** `execapi.bat`.

For example, `<MIDDLEWARE_HOME>/user_projects/epmsystem1/bin/ReportingAnalysis/SDK`.

2 **Enter this syntax:**

`execapi Login username password host port`

For example:

`execapi Login administrator administrator qastar 6800`

If the login is successful, these messages are displayed:

```
connection established
connection closed
```

Batch Driver Tutorial

The batch driver sample program uses these files:

● `execapi.bat`—Sets the environment, passes arguments to the program, and runs the program including using the control file, `batch_driver.cf`. This file is in `<MIDDLEWARE_HOME>/user_projects/epmsystem1/bin/ReportingAnalysis/SDK`.

● `batch_driver.cf`—Control statements that describe artifacts created in the database by the batch driver program. It is in `<EPM_ORACLE_HOME>/products/biplus/SDK/samples/java`. See Chapter 2, "Batch Driver Sample Program Control File."

In this tutorial, the `batch_driver.cf` file contains this single line control statement that adds a data artifact. It assumes that the artifacts accessor1 and accessor2 exist in the system.

```
data name=aDocument^path=/batchFolder^file=D:\Doc
\bart.doc^browse=true^autodelete=true^desc=A test document by
batchdriver^expire=2005-06-22 12:00:00^keyword=MS doc^keyword=Design
doc^perm=accessor1^perm=accessor2.
```

➤ To run the batch driver sample:

1 **At a command line, change to the directory that contains** `execapi.bat`.

For example, `bin/ReportingAnalysis/SDK`.

2 **Run the batch driver sample program.**

● If you are using the installed `BatchDriver.class`, use this syntax:

```
execapi com.sqribe.rm.BatchDriver username password host port
control_filename
```

For example:

```
execapi com.sqribe.rm.BatchDriver administrator administrator
venice 6800 batch_driver.cf
```

- If you compiled the batch driver program, use this syntax:

```
execapi BatchDriver username password host port control_filename
```

For example:

```
execapi BatchDriver administrator administrator venice 6800
batch_driver.cf
```

3 **Open Oracle Enterprise Performance Management Workspace, Fusion Edition and check that the artifact
in the** `batch_driver.cf` **is listed.**

You should see *A test document by batchdriver* listed.

4 **Create a batch file to run this class again by copying and saving the following line from the command
line to a batch file,** `batch_driver.bat`, **in** `<EPM_ORACLE_HOME>/products/biplus/
SDK/samples/java`:

```
execapi com.sqribe.rm.BatchDriver administrator administrator venice 6800
batch_driver.cf
```

or

```
execapi BatchDriver administrator administrator venice 6800 batch_driver.cf
```

Now you can run the batch driver sample program by simply clicking the batch file. Change
the `batch_driver.cf` file to add other artifacts. For more information on the batch driver
control file, see Chapter 2, "Batch Driver Sample Program Control File."

Index