Oracle® Fusion Middleware

Integrator's Guide for Oracle Business Intelligence Enterprise Edition

11*g* Release 1 (11.1.1)

E16364-05

February 2013

Explains how to integrate Oracle Business Intelligence Enterprise Edition with other systems, using web services and other APIs.

ORACLE

Oracle Fusion Middleware Integrator's Guide for Oracle Business Intelligence Enterprise Edition, 11*g* Release 1 (11.1.1)

E16364-05

Primary Author: Nick Fry

Contributing Authors: Stefanie Rhone, Marla Azriel, Christine Jacobs

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

Contents

Preface xiii

Audience xiii
Documentation Accessibility xiii
Related Documentation and Other Resources xiv
System Requirements and Certification xiv
Conventions xiv

New Features for Oracle Business Intelligence Developers xv

New Features for Oracle BI EE 11*g* Release 1 (11.1.1.7) xv
New Features for Oracle BI EE 11*g* Release 1 (11.1.1.6) xv
New Features for Oracle BI EE 11*g* Release 1 (11.1.1.5) xvi
New Features for Oracle BI EE 11*g* Release 1 (11.1.1.3) xvi

1 Introduction to Oracle Business Intelligence Web Services

1.1 Overview of the Oracle Business Intelligence Web Services 1-1
1.2 What are the Oracle Business Intelligence Session-Based Web Services? 1-1
1.3 What are the Oracle Business Intelligence Web Services for SOA? 1-2
1.4 Configuring and Securing the Oracle Business Intelligence Web Services for SOA 1-3
1.5 Enabling SSL for Web Services Communication 1-3
1.6 Invoking Oracle Business Intelligence Web Services Over HTTPS 1-3

2 Description of Structures in Oracle BI EE Web Services

2.1 AccessControlToken Structure 2-2
2.2 Account Structure 2-2
2.3 ACL Structure 2-3
2.4 ArrayofGUIDS Structure 2-3
2.5 AuthResult Structure 2-3
2.6 CatalogItemsFilter Structure 2-4
2.7 CatalogObject Structure 2-4
2.8 ErrorInfo Structure 2-4
2.9 GetSubItemsParams Structure 2-5
2.10 ImportError Structure 2-5
2.11 ItemInfo Structure 2-6
2.12 NameValuePair Structure 2-6
2.13 PathMap Structure 2-7

2.14 Privilege Structure 2-7
2.15 QueryResults Structure 2-7
2.16 ReportHTMLOptions Structure 2-8
2.16.1 ReportHTMLLinksMode Enumeration 2-8
2.17 ReportParams Structure 2-8
2.18 ReportRef Structure 2-9
2.19 SAColumn Structure 2-9
2.19.1 SADataType Values 2-10
2.19.2 AggregationRule Values 2-11
2.20 SASubjectArea Structure 2-11
2.21 SATable Structure 2-11
2.22 SAWLocale Structure 2-12
2.23 SAWSessionParameters Structure 2-12
2.24 SegmentationOptions Structure 2-13
2.25 SessionEnvironment Structure 2-13
2.26 StartPageParams Structure 2-14
2.27 TreeNodePath Structure 2-14
2.28 UpdateACLParams Structure 2-14
2.28.1 UpdateACLMode Enumeration 2-15
2.29 UpdateCatalogItemACLParams Structure 2-15
2.30 Variable Structure 2-15
2.31 XMLQueryExecutionOptions Structure 2-16

3 Description of Services and Methods in Oracle BI EE Web Services

3.1 ConditionService Service 3-1
3.1.1 evaluateCondition() Method 3-2
3.1.1.1 Signature 3-2
3.1.2 evaluateInlineCondition() Method 3-2
3.1.2.1 Signature 3-2
3.1.3 getConditionCustomizableReportElements() Method 3-3
3.1.3.1 Signature 3-3
3.2 HtmlViewService Service 3-3
3.2.1 About HtmlViewService Bridging and Callback URLs 3-4
3.2.2 addReportToPage() Method 3-4
3.2.2.1 Signature 3-4
3.2.3 endPage() Method 3-5
3.2.3.1 Signature 3-5
3.2.4 getCommonBodyHTML() Method 3-5
3.2.4.1 Signature 3-5
3.2.4.2 Returns 3-6
3.2.5 getHeadersHTML() Method 3-6
3.2.5.1 Signature 3-6
3.2.5.2 Returns 3-6
3.2.6 getHtmlforPageWithOneReport() Method 3-6
3.2.6.1 Signature 3-6
3.2.7 getHTMLForReport() Method 3-7
3.2.7.1 Signature 3-7

3.2.7.2 Returns ... 3-7
3.2.8 setBridge() Method ... 3-7
3.2.8.1 Signature ... 3-7
3.2.8.2 Usage ... 3-8
3.2.8.3 How Callback URLs Are Replaced ... 3-8
3.2.9 startPage() Method ... 3-8
3.2.9.1 Signature ... 3-8
3.2.9.2 Returns ... 3-9
3.3 iBotService Service ... 3-9
3.3.1 writeIBot() Method ... 3-9
3.3.1.1 Signature ... 3-9
3.3.2 deleteIBot() Method ... 3-10
3.3.2.1 Signature ... 3-10
3.3.3 executeIBotNow() Method ... 3-10
3.3.3.1 Signature ... 3-10
3.3.4 moveIBot() Method ... 3-10
3.3.4.1 Signature ... 3-10
3.3.5 sendMessage() Method ... 3-11
3.3.5.1 Signature ... 3-11
3.3.6 subscribe() Method ... 3-11
3.3.6.1 Signature ... 3-11
3.3.7 unsubscribe() Method ... 3-12
3.3.7.1 Signature ... 3-12
3.4 JobManagementService Service ... 3-12
3.4.1 cancelJob() Method ... 3-13
3.4.1.1 Signature ... 3-13
3.4.2 deleteResultSet() Method ... 3-13
3.4.2.1 Signature ... 3-13
3.4.3 getCounts() Method ... 3-13
3.4.3.1 Signature ... 3-13
3.4.4 getJobInfo() Method ... 3-14
3.4.4.1 Signature ... 3-14
3.4.5 getPromptedColumns() Method ... 3-14
3.4.5.1 Signature ... 3-14
3.4.6 prepareCache() Method ... 3-14
3.4.6.1 Signature ... 3-14
3.4.7 purgeCache() Method ... 3-15
3.4.7.1 Signature ... 3-15
3.4.8 saveResultSet() Method ... 3-15
3.4.8.1 Signature ... 3-15
3.4.9 writeListFiles() Method ... 3-16
3.4.9.1 Signature ... 3-16
3.5 MetadataService Service ... 3-17
3.5.1 clearQueryCache() Method ... 3-17
3.5.1.1 Signature ... 3-17
3.5.2 describeColumn() Method ... 3-17
3.5.2.1 Signature ... 3-17

3.5.2.2 Returns ... 3-18
3.5.3 describeSubjectArea() Method ... 3-18
3.5.3.1 Signature ... 3-18
3.5.3.2 SASubjectAreaDetails Values ... 3-18
3.5.3.3 Returns ... 3-18
3.5.3.4 Usage ... 3-18
3.5.4 describeTable() Method ... 3-19
3.5.4.1 Signature ... 3-19
3.5.4.2 SATablesDetails Values ... 3-19
3.5.4.3 Returns ... 3-19
3.5.5 getSubjectAreas() Method ... 3-19
3.5.5.1 Signature ... 3-19
3.5.5.2 Returns ... 3-19
3.5.5.3 Usage ... 3-20
3.6 ReplicationService Service ... 3-20
3.6.1 ExportFlags Enumeration ... 3-20
3.6.2 ImportFlags Enumeration ... 3-21
3.6.3 Import() Method ... 3-21
3.6.3.1 Signature ... 3-21
3.6.3.2 Returns ... 3-22
3.6.4 export() Method ... 3-22
3.6.4.1 Signature ... 3-22
3.6.5 markForReplication() Method ... 3-22
3.6.5.1 Signature ... 3-22
3.6.6 purgeLog() Method ... 3-23
3.6.6.1 Signature ... 3-23
3.7 ReportEditingService Service ... 3-23
3.7.1 applyReportDefaults() Method ... 3-23
3.7.1.1 Signature ... 3-23
3.7.1.2 Returns ... 3-24
3.7.2 applyReportParams() Method ... 3-24
3.7.2.1 Signature ... 3-24
3.7.2.2 Returns ... 3-24
3.7.3 generateReportSQL() Method ... 3-24
3.7.3.1 Signature ... 3-24
3.7.3.2 Returns ... 3-25
3.8 SAWSessionService Service ... 3-25
3.8.1 getCurUser() Method ... 3-25
3.8.1.1 Signature ... 3-25
3.8.1.2 Returns ... 3-26
3.8.2 GetSessionEnvironment() Method ... 3-26
3.8.2.1 Signature ... 3-26
3.8.2.2 Returns ... 3-26
3.8.3 getSessionVariable() Method ... 3-26
3.8.3.1 Signature ... 3-26
3.8.3.2 Returns ... 3-26
3.8.4 impersonate() Method ... 3-26

3.8.4.1 Signature ... 3-27
3.8.4.2 Returns ... 3-27
3.8.5 impersonateex() Method ... 3-27
3.8.5.1 Signature ... 3-27
3.8.5.2 Returns ... 3-27
3.8.6 keepAlive() Method ... 3-27
3.8.6.1 Signature ... 3-28
3.8.7 logoff() Method ... 3-28
3.8.7.1 Signature ... 3-28
3.8.8 logon() Method ... 3-28
3.8.8.1 Signature ... 3-28
3.8.8.2 Returns ... 3-28
3.8.9 logonex() Method ... 3-28
3.8.9.1 Signature ... 3-28
3.8.9.2 Returns ... 3-29
3.9 SecurityService Service ... 3-29
3.9.1 forgetAccounts() Method ... 3-29
3.9.1.1 Signature ... 3-29
3.9.2 getAccounts() Method ... 3-30
3.9.2.1 Signature ... 3-30
3.9.3 getGlobalPrivilegeACL() Method ... 3-30
3.9.3.1 Signature ... 3-30
3.9.4 getGlobalPrivileges() Method ... 3-30
3.9.4.1 Signature ... 3-30
3.9.5 getGroups() Method ... 3-31
3.9.5.1 Signature ... 3-31
3.9.6 getMembers() Method ... 3-31
3.9.6.1 Signature ... 3-31
3.9.7 getPermissions() Method ... 3-31
3.9.7.1 Signature ... 3-31
3.9.7.2 Returns ... 3-32
3.9.8 getPrivilegesStatus() Method ... 3-32
3.9.8.1 Signature ... 3-32
3.9.9 isMember() Method ... 3-32
3.9.9.1 Signature ... 3-32
3.9.10 joinGroups() Method ... 3-32
3.9.10.1 Signature ... 3-32
3.9.11 leaveGroups() Method ... 3-33
3.9.11.1 Signature ... 3-33
3.9.12 renameAccounts() Method ... 3-33
3.9.12.1 Signature ... 3-33
3.9.12.2 Returns ... 3-33
3.9.13 updateGlobalPrivilegeACL() Method ... 3-33
3.9.13.1 Signature ... 3-33
3.10 WebCatalogService Service ... 3-34
3.10.1 ErrorDetailsLevel Enumeration ... 3-35
3.10.2 ReadObjectsReturnOptions Enumeration ... 3-35

3.10.3 copyItem() Method 3-35
3.10.3.1 Signature 3-35
3.10.4 copyItem2() Method 3-36
3.10.4.1 Signature 3-36
3.10.5 createFolder() Method 3-36
3.10.5.1 Signature 3-36
3.10.6 createLink() Method 3-36
3.10.6.1 Signature 3-36
3.10.7 deleteItem() Method 3-37
3.10.7.1 Signature 3-37
3.10.8 getItemInfo() Method 3-37
3.10.8.1 Signature 3-37
3.10.8.2 Returns 3-37
3.10.9 getSubItems() Method 3-37
3.10.9.1 Signature 3-38
3.10.9.2 Returns 3-38
3.10.10 maintenanceMode() Method 3-38
3.10.10.1 Signature 3-38
3.10.11 moveItem() Method 3-38
3.10.11.1 Signature 3-38
3.10.12 pasteItem2() Method 3-39
3.10.12.1 Signature 3-39
3.10.13 readObjects() Method 3-39
3.10.13.1 Signature 3-39
3.10.13.2 Returns 3-40
3.10.14 removeFolder() Method 3-40
3.10.14.1 Signature 3-40
3.10.15 setItemAttributes() Method 3-40
3.10.15.1 Signature 3-40
3.10.16 setItemProperty() Method 3-41
3.10.16.1 Signature 3-41
3.10.17 setOwnership() Method 3-41
3.10.17.1 Signature 3-41
3.10.18 updateCatalogItemACL() Method 3-41
3.10.18.1 Signature 3-41
3.10.19 writeObjects() Method 3-42
3.10.19.1 Signature 3-42
3.10.19.2 Returns 3-42
3.11 XMLViewService Service 3-42
3.11.1 XMLQueryOutputFormat Enumeration 3-43
3.11.2 cancelQuery() Method 3-43
3.11.2.1 Signature 3-43
3.11.3 executeSQLQuery() Method 3-43
3.11.3.1 Signature 3-44
3.11.3.2 Returns 3-44
3.11.4 executeXMLQuery() Method 3-44
3.11.4.1 Signature 3-44

3.11.4.2 Returns 3-44
3.11.5 fetchNext() Method 3-45
3.11.5.1 Signature 3-45
3.11.5.2 Returns 3-45
3.11.6 getPromptedFilters() Method 3-45
3.11.6.1 Signature 3-45
3.11.7 upgradeXML() Method 3-45
3.11.7.1 Signature 3-45

4 Using the Oracle Business Intelligence Server Metadata Web Service

4.1 Overview of the Oracle BI Server Metadata Web Service 4-1
4.2 Configuring the Oracle BI Server Metadata Web Service 4-1
4.2.1 Configuring the Oracle BI Server Metadata Web Service Connection to Oracle BI Server 4-2
4.2.1.1 Setting Up an Oracle Business Intelligence JDBC Data Source 4-2
4.2.2 Securing the Oracle BI Server Metadata Web Service 4-2
4.2.2.1 Applying Policies 4-3
4.2.2.2 Configuring WSM 4-3
4.2.2.3 Assigning the manageRepositories Permission 4-3
4.3 Calling the Oracle BI Server Metadata Web Service 4-3
4.3.1 Calling the Oracle BI Server Metadata Web Service Synchronously 4-3
4.3.1.1 callProcedure() Method 4-4
4.3.1.2 callProcedureWithResults() Method 4-4
4.3.1.3 startExtender() Method 4-5
4.3.1.4 Sample Code 4-6
4.3.2 Calling the Oracle BI Server Metadata Web Service Asynchronously 4-7
4.3.2.1 Creating the Callback Service 4-7
4.3.2.2 Configuring the Callback Service 4-7
4.3.2.3 Sample Code 4-8
4.4 Using the Oracle BI Server XML Procedures 4-8
4.4.1 Extract Project Procedure 4-9
4.4.2 Modify Metadata Procedure 4-9
4.4.3 Query Metadata Procedure 4-10
4.4.4 Query Projects Procedure 4-11

5 Using Actions to Integrate Oracle BI EE with External Systems

5.1 What is the Action Framework? 5-1
5.1.1 What Functionality is Provided by the Action Framework? 5-2
5.1.2 Action Types and Action Execution 5-2
5.2 Overview of the Action Framework Configuration 5-3
5.2.1 Configuration Checklist by Action Type 5-3
5.2.2 Overview of Targets 5-4
5.3 Configuring the Action Framework 5-5
5.3.1 Aliases 5-5
5.3.2 Registries 5-6
5.3.2.1 Navigate to EPM Content Action Type Registry Example 5-6

5.3.2.2 Invoke a Java Method Action Type Registry Example 5-7
5.3.2.3 Invoke a Web Service Action Type Registry Example 5-7
5.3.2.4 Registry Elements Descriptions 5-8
5.3.2.5 Valid Values for the Provider-Class Element 5-10
5.3.3 Content Types 5-10
5.3.4 Accounts 5-11
5.3.4.1 Account Elements Descriptions 5-11
5.3.5 Policies 5-12
5.3.5.1 Policy Elements Descriptions 5-12
5.3.5.2 Policy Files 5-12
5.3.6 Proxy 5-13
5.3.6.1 Proxy Elements Descriptions 5-14
5.3.7 ebusinesssuiteconfig 5-14
5.3.8 siebelcrmconfig 5-14
5.4 Overview of Action Security 5-14
5.4.1 Oracle BI EE Credentials 5-15
5.4.2 Oracle BI EE Privileges 5-15
5.4.3 Oracle BI Presentation Catalog Permissions 5-15
5.5 Adding and Maintaining Credentials for Use With the Action Framework 5-16
5.5.1 Adding a Credential Map and Credential Key to the Credential Store 5-16
5.5.1.1 Example of Creating the Credential Map and Credential Key 5-16
5.5.2 Creating a Default Keystore 5-17
5.5.3 Configuring Oracle Web Services Manager 5-18
5.6 Target Functionality for Actions 5-19
5.6.1 Navigate to EPM Content 5-19
5.6.1.1 Prerequisites for This Action Type 5-19
5.6.1.2 What Happens When This Action Type is Invoked? 5-19
5.6.2 Navigate to E-Business Suite 5-21
5.6.2.1 Overview of Passing Context to Oracle E-Business Suite Java Forms 5-21
5.6.3 Navigate to Siebel CRM 5-21
5.6.4 Invoke a Web Service 5-22
5.6.4.1 Prerequisites for This Action Type 5-22
5.6.4.2 Example of a WSIL Document 5-22
5.6.4.3 Troubleshooting Actions to Invoke a Web Service 5-23
5.6.4.4 What Happens When This Action Type is Invoked? 5-23
5.6.5 Supported Functionality for Calling Web Services 5-25
5.6.5.1 Transport 5-25
5.6.5.2 Messaging 5-25
5.6.5.3 SOAP 5-25
5.6.5.4 Response Document 5-26
5.6.5.5 Service Description 5-26
5.6.5.6 Discovery Services 5-26
5.6.5.7 Security 5-26
5.6.5.8 Reliable Messaging and Transactions 5-26
5.6.6 Invoke a Java Method (EJB) 5-26
5.6.6.1 Prerequisites for This Action Type 5-26
5.6.6.2 Parameters for the EJB 5-27

5.6.6.3 What Happens When This Action Type is Invoked? 5-27
5.6.7 Invoke a Browser Script 5-29
5.6.7.1 JavaScript Functions 5-29
5.6.7.2 UserScript.js 5-29
5.6.7.2.1 JavaScript Example 1 5-29
5.6.7.2.2 JavaScript Example 2 5-30
5.6.7.3 What Happens When This Action Type is Invoked? 5-31
5.6.8 Invoke a Server Script 5-32
5.6.8.1 Prerequisites for This Action Type 5-32
5.6.8.2 What Happens When This Action Type is Invoked? 5-32
5.6.9 Invoke Agent 5-34
5.6.9.1 How Filters Work in Invoke Agent Actions 5-34
5.6.9.2 What Happens When This Action Type is Invoked? 5-34
5.6.10 Java Job 5-36
5.6.10.1 What Happens When This Action Type is Invoked? 5-36

6 Integrating Oracle BI Presentation Services into Corporate Environments Using HTTP and JavaScript

6.1 Incorporating Oracle Business Intelligence Results into External Portals or Applications 6-1
6.1.1 About the Oracle BI Presentation Services Prompted URL 6-1
6.1.2 About the Oracle BI Presentation Services Go URL 6-2
6.1.3 Structure of the Basic Oracle BI Presentation Services Go URL 6-2
6.1.4 Optional Parameters for the Oracle BI Presentation Services Go URL 6-2
6.1.4.1 Displaying All Records in a Table 6-4
6.2 Referencing Dashboard Content in External Portals or Applications 6-4
6.2.1 About the Oracle BI Presentation Services Dashboard URL 6-4
6.2.2 Structure of the Basic Oracle BI Presentation Services Dashboard URL 6-5
6.2.3 Optional Commands and Parameters for the Oracle BI Presentation Services Dashboard URL 6-5
6.3 Using the Oracle BI Presentation Services Go URL to Issue SQL and Pass Filters 6-7
6.3.1 Issuing SQL Commands and Passing Filters 6-7
6.3.2 Passing Filters to the Oracle BI Presentation Services Go URL Through a URL (Navigation) 6-7
6.3.2.1 Navigation Parameters 6-8
6.3.2.2 Navigation Examples 6-9
6.3.2.3 Navigation Using JavaScript 6-9
6.3.2.4 Navigation from HTML Results 6-11
6.4 Example of an Oracle Business Intelligence Third-Party SQL Tool Integration 6-11
6.4.1 Example of integrating a third-party SQL tool 6-12
6.5 Retrieving Links to Dashboard Pages Using Scripts 6-12

7 Oracle Business Intelligence Systems Management API

7.1 What is the Oracle Business Intelligence Systems Management API? 7-1

8 Integrating Other Clients with Oracle Business Intelligence

8.1 Overview of Integrating with Oracle Business Intelligence 8-1
8.2 About Integrating with the Oracle BI Server as a Data Source 8-2
8.2.1 About Routing Requests to the Physical Layer 8-2
8.2.2 About Integrating with the Oracle BI Server Using JDBC 8-3
8.3 ODBC Conformance Level 8-3
8.4 Configuring an ODBC DSN for the Oracle BI Server on Windows 8-4
8.4.1 Configuring the ODBC DSN for Advanced SSL Settings 8-7
8.5 Configuring an ODBC DSN for the Oracle BI Server on Linux or UNIX 8-8

9 Using Discoverer Data in Applications

9.1 Exposing Discoverer Worksheets in Applications 9-1
9.2 Converting Discoverer Metadata to Use in Oracle BI EE 9-1

10 Integrating with Oracle E-Business Suite Security

10.1 Creating a Database Object and Connection Pool for the Oracle E-Business Suite Database 10-1
10.2 Setting Up Authentication 10-2
10.2.1 Setting Up Session Variables for Authentication 10-2
10.2.2 Updating authenticationschemas.xml 10-4
10.2.3 Updating instanceconfig.xml 10-6
10.3 Embedding Links to Oracle Business Intelligence in Oracle E-Business Suite 10-6
10.3.1 Domain Prerequisites 10-7
10.3.2 Creating a Form Function 10-7
10.3.3 Creating a Menu That Invokes the Form Function 10-8
10.3.4 Assigning the Menu to a Responsibility 10-9
10.3.5 Assigning the Responsibility to a User 10-10
10.3.6 Setting Up a Profile 10-11

11 Embedding Oracle BI EE In Oracle's Siebel CRM

11.1 Overview of Embedding Oracle BI EE in Oracle's Siebel CRM 11-1
11.2 Configuring Oracle HTTP Server 11-1
11.3 Configuring the Siebel Application to Find Oracle BI Through HTTP Server 11-3
11.4 Modifying the Siebel URLs to Reference the /analytics Directory 11-3
A.1 Sample Action Framework Configuration File A-1
A.2 Sample Java Script File A-4

Glossary

Index

Preface

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

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

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

This guide contains information about developing Oracle BI EE-based applications and integrations between Oracle BI EE and other Oracle systems.

Audience

This document is intended for application developers, data service providers, and middle tier administrators who want to programatically access and use the Oracle BI EE components and use them to create applications or integrations with other components.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documentation and Other Resources

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

In addition, go to the Oracle Learning Library for Oracle Business Intelligence-related online training resources.

System Requirements and Certification

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

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

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

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

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

Conventions

The following text conventions are used in this document:

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

New Features for Oracle Business Intelligence Developers

This preface describes new features and upgrade considerations in Oracle Business Intelligence 11*g* Release 1 (11.1.1).

This preface contains the following topics:

- "New Features for Oracle BI EE 11g Release 1 (11.1.1.7)"
- "New Features for Oracle BI EE 11g Release 1 (11.1.1.6)"
- "New Features for Oracle BI EE 11g Release 1 (11.1.1.5)"
- "New Features for Oracle BI EE 11g Release 1 (11.1.1.3)"

New Features for Oracle BI EE 11*g* Release 1 (11.1.1.7)

This section describes new features for Oracle BI EE 11*g* Release 1 (11.1.1.7). It contains the following topic:

- "Calling the Oracle Business Intelligence Metadata Web Service Asynchronously"

Calling the Oracle Business Intelligence Metadata Web Service Asynchronously

The Oracle Business Intelligence Metadata Web Service provides asynchronous and synchronous calls to Oracle BI Server stored procedures. You use these procedures to obtain information about the metadata and to modify the metadata. See Section 4.3, "Calling the Oracle BI Server Metadata Web Service".

New Features for Oracle BI EE 11*g* Release 1 (11.1.1.6)

This section describes new features for Oracle BI EE 11*g* Release 1 (11.1.1.6). It contains the following topics:

- "New Features for 11.1.1.6.0"
- "New Features for 11.1.1.6.2"

New Features for 11.1.1.6.0

There are no new features for Oracle BI EE 11*g* Release 1 (11.1.1.6.0).

New Features for 11.1.1.6.2

There are no new features for Oracle BI EE 11*g* Release 1 (11.1.1.6.2).

New Features for Oracle BI EE 11*g* Release 1 (11.1.1.5)

This section describes new features for Oracle BI EE 11*g* Release 1 (11.1.1.5). It contains the following topics:

- "Oracle Business Intelligence Metadata Web Service"
- "Navigate to Siebel CRM Action Type"

Oracle Business Intelligence Metadata Web Service

The Oracle BI Metadata Web Service provides a web services interface to call the Oracle BI Server stored procedures. You use these procedures to obtain information about the metadata and to modify the metadata. See Chapter 4, "Using the Oracle Business Intelligence Server Metadata Web Service".

Navigate to Siebel CRM Action Type

The Navigate to Siebel CRM action type allows the content designer to add an action that navigates to a view (such as an opportunity) in a Siebel CRM application. This action type allows users to navigate from a dashboard that is embedded in a Siebel CRM application to a record in a view in the CRM application. This action type requires you to perform a security integration between Oracle's Siebel CRM and Oracle BI EE. You must also configure the Action Framework before it is available in Oracle BI EE Presentation Services.

See Chapter 5, "Using Actions to Integrate Oracle BI EE with External Systems" and Chapter 11, "Embedding Oracle BI EE In Oracle's Siebel CRM".

New Features for Oracle BI EE 11*g* Release 1 (11.1.1.3)

New features in Oracle BI EE 11*g* Release 1 (11.1.1.3) include:

- "Web Services"
- "Action Framework"

Web Services

Oracle Business Intelligence includes the following new web services:

- ConditionService Service
- JobManagementService Service

New methods and structures were added to the existing web services.

See for more information, see Chapter 3, "Description of Services and Methods in Oracle BI EE Web Services".

Action Framework

The Action Framework provides functionality for creating, managing, and invoking actions. Actions can be included within analyses, dashboards, agents, KPIs, and Scorecard objectives. Actions provide functionality to navigate to related content and invoke operations, functions, or processes in external systems.

The Action Framework must be configured to enable the Actions functionality in Oracle BI Presentation Services. See Chapter 5, "Using Actions to Integrate Oracle BI EE with External Systems" for more information.

1

Introduction to Oracle Business Intelligence Web Services

This chapter describes the Oracle Business Intelligence Session-Based Web Services (SOAP) and the Oracle Business Intelligence Web Services for SOA. Describes configuring and securing the web services for SOA, enabling SSL, and invoking web services over HTTPS.

This chapter includes the following sections:

- Section 1.1, "Overview of the Oracle Business Intelligence Web Services"
- Section 1.2, "What are the Oracle Business Intelligence Session-Based Web Services?"
- Section 1.3, "What are the Oracle Business Intelligence Web Services for SOA?"
- Section 1.4, "Configuring and Securing the Oracle Business Intelligence Web Services for SOA"
- Section 1.5, "Enabling SSL for Web Services Communication"
- Section 1.6, "Invoking Oracle Business Intelligence Web Services Over HTTPS"

1.1 Overview of the Oracle Business Intelligence Web Services

Oracle Business Intelligence provides two types of web services: Oracle Business Intelligence Session-Based Web Services and Oracle Business Intelligence Web Services for SOA.

You can use Oracle Business Intelligence Session-Based Web Service to call Oracle Business Intelligence programmatically to invoke many different business intelligence items. For more detailed information about these types of web services, see "What are the Oracle Business Intelligence Session-Based Web Services?"

The Oracle Business Intelligence Web Services for SOA provide a web service unique to each analysis, condition, or agent saved to the Oracle BI Presentation Catalog. For more detailed information about this types of web service, see "What are the Oracle Business Intelligence Web Services for SOA?"

1.2 What are the Oracle Business Intelligence Session-Based Web Services?

The Oracle Business Intelligence Session-Based Web Services are an application programming interface (API) that implements SOAP. These web services are designed for programmatic use, where you use one web service for invoking many different

business intelligence objects. These web services also provide functionality on a wide range of Presentation Services operations.

The Oracle Business Intelligence Session-Based Web Services allow you to perform three types of functions:

- Extract results from Oracle Business Intelligence Presentation Services and deliver them to external applications and web application environments.
- Perform Oracle Business Intelligence Presentation Services management functions.
- Execute Oracle BI EE alerts (known as Intelligent Agents).

Oracle Business Intelligence Session-Based Web Services allow external applications such as J2EE and .NET to use Oracle Business Intelligence as an analytical calculation and data integration engine. It provides a set of Presentation Services that allow external applications to communicate with Oracle Business Intelligence Presentation Services.

Oracle Business Intelligence Session-Based Web Services require a valid Oracle Business Intelligence session ID to be passed as a parameter. This means that the calling application first needs to make a call to get the session before calling the web service. A final call is made to log out.

The formal definition of services and methods in Oracle BI EE web services can be retrieved in WSDL format. You can use a proxy generation tool to create proxy/stub code to access the services. Depending upon the client version you are using, you can access the WSDL document at one of the following Oracle BI EE web services URLs:

```
http://host:port/analytics-ws/saw.dll/wsdl/v6

http://host:port/analytics-ws/saw.dll/wsdl/v7
```

1.3 What are the Oracle Business Intelligence Web Services for SOA?

The Oracle Business Intelligence Web Services for SOA contains three web services, ExecuteAgent, ExecuteAnalysis, and ExecuteCondition, which are hosted by the middleware J2EE application. These web services enable you to use third-party web services clients (for example, Oracle SOA Suite) to browse for and include business intelligence objects in service oriented architecture components. These web services are used to consume specific values or small sets of values to feed into conditional logic or subsequent steps. This approach allows end users to execute analyses, evaluate conditions, and to invoke Agents in their processes, event routing, and business rules. Note that the Oracle Business Intelligence Web Services for SOA will only return XML strings.

The Oracle Business Intelligence Web Services for SOA support calling agents, analyses, and conditions only. Prompted filters and presentation variables included in the business intelligence objects are supported. Oracle BI EE dynamically creates a dedicated Web Services Description Language (WSDL) document with its own name space for each agent, analysis, and condition that content designers save to the catalog.

The Oracle Business Intelligence Web Services for SOA dynamically provide WSIL documents to allow you to browse for and select agents, analyses, and conditions stored in the catalog. The WSIL lists each available business intelligence object as a WSDL document with a unique name. Since each object has a dedicated WSDL document, the WSDL explicitly lists prompted filters for you to complete. If the your SOA development tool does not support WSIL browsing, you can still access the WSIL from a web browser user interface. You can access the WSIL by accessing the following URL on the Oracle Business Intelligence Presentation Server:

```
https://host:port/biservices/inspection?wsil
```

The WSIL is not available until it has been properly configured. For more information about this required configuration, see "Configuring and Securing the Oracle Business Intelligence Web Services for SOA". For specific instructions about adding the wsil.browsing key to the credential store, see "Adding and Maintaining Credentials for Use With the Action Framework".

1.4 Configuring and Securing the Oracle Business Intelligence Web Services for SOA

During installation, each web service (executeAgent, executeAnalysis, and executeCondition) is assigned the "policy:oracle/wss_username_token_service_policy" security policy. This policy requires the calling SOAP message to include a user name and token (password) in the WS-Security header. The user credentials that are passed to web services through the incoming SOAP message can be any valid business intelligence user who has the proper access to the target business intelligence object being invoked. This method of security means that web services can be called in a single step without first retrieving a session ID. Note that if required, you can change the security policy used by the web services to any security policy available in Oracle WebLogic Server.

Whereas invoking web services uses the credentials passed in the calling SOAP message to invoke the target functionality, browsing the web services using the WSIL uses a single user account. It is not currently possible to invoke the browsing mechanism using the credentials of the user performing the browsing using this mechanism.

To enable browsing for web services, you must go to Fusion Middleware Control, access the oracle.bi.enterprise map, which is located on the bifoundation_domain, and manually add the "wsil.browsing" credential to the credential store. This key holds the user ID and password for the valid user defined in the identity store. For example, if you want to browse for target web services as the user "abell," you will add the credentials of "abell" to the wsil.browsing key in the credential store.

In practice, a special user should be created in the identity store specifically for browsing the catalog for use with this functionality. This user should not have any business intelligence objects in their personal folder (my folders), as other users will not be able to invoke this functionality.

For more information about setting up users and credentials, see *Oracle Fusion Middleware Security Guide for Oracle Business Intelligence Enterprise Edition*.

1.5 Enabling SSL for Web Services Communication

Oracle recommends that you enable HTTPS on the Managed Server that hosts the Analytics and BI middleware J2EE applications. Un-encrypted credentials that are passed to the target web service may be intercepted, and using SSL is a way to mitigate this risk. After you set up SSL, see "Invoking Oracle Business Intelligence Web Services Over HTTPS" for information about certificates.

1.6 Invoking Oracle Business Intelligence Web Services Over HTTPS

To invoke Oracle Business Intelligence Web Services when using HTTPS, the client calling the web service on the server (for example, Oracle BPEL calling Oracle Business Intelligence Web Services for SOA) needs to trust the server certificate. The server may have an authentic certificate provided by a well-known certificate authority, in which

case the client may trust the server certificate without further configuration. However, by default, this is not the case, and the root certificate used by the WebLogic Managed Servers that are hosting the web services should be imported into the appropriate keystore of the web services client that is calling these web services.

Oracle recommends that in a production environment you use a certificate signed by a well-know certificate authority.

Use the following procedure to confirm the location of the root certificate of the Managed Servers that the web services client needs to trust.

To confirm the location of the root certificate:

1. Open the WebLogic console in a browser. By default, the location of the WebLogic console is:

   ```
   http://host:7001/console
   ```

2. From the Oracle WebLogic Server Administration Console, select the SSL tab and go to the Identity area. By default, the Certificate location is from the Demo Identity Keystore. If this is the case, navigate to the Keystores tab and review the location of the Demo Identity Keystore.

   Note that the Demo Identity Keystore's default location is:

   MW_HOME/wlserver_10.3_server/lib/DemoIdentity.jks

3. Use the Oracle Keytool utility to view and export the root certificate.

2

Description of Structures in Oracle BI EE Web Services

This chapter describes the structures used by the Oracle Business Intelligence Session-Based Web Services.

This document uses JavaScript-like syntax to describes structures. The exact syntax and implementation depends on the SOAP code generation tool and the target language used by your application development environment.

This chapter contains the following sections:

- Section 2.1, "AccessControlToken Structure"
- Section 2.2, "Account Structure"
- Section 2.3, "ACL Structure"
- Section 2.4, "ArrayofGUIDS Structure"
- Section 2.5, "AuthResult Structure"
- Section 2.6, "CatalogItemsFilter Structure"
- Section 2.7, "CatalogObject Structure"
- Section 2.8, "ErrorInfo Structure"
- Section 2.9, "GetSubItemsParams Structure"
- Section 2.10, "ImportError Structure"
- Section 2.11, "ItemInfo Structure"
- Section 2.12, "NameValuePair Structure"
- Section 2.13, "PathMap Structure"
- Section 2.14, "Privilege Structure"
- Section 2.15, "QueryResults Structure"
- Section 2.16, "ReportHTMLOptions Structure"
- Section 2.17, "ReportParams Structure"
- Section 2.18, "ReportRef Structure"
- Section 2.19, "SAColumn Structure"
- Section 2.20, "SASubjectArea Structure"
- Section 2.21, "SATable Structure"
- Section 2.22, "SAWLocale Structure"

- Section 2.23, "SAWSessionParameters Structure"
- Section 2.24, "SegmentationOptions Structure"
- Section 2.25, "SessionEnvironment Structure"
- Section 2.26, "StartPageParams Structure"
- Section 2.27, "TreeNodePath Structure"
- Section 2.28, "UpdateACLParams Structure"
- Section 2.29, "UpdateCatalogItemACLParams Structure"
- Section 2.30, "Variable Structure"
- Section 2.31, "XMLQueryExecutionOptions Structure"

2.1 AccessControlToken Structure

Use this structure to describe permissions granted to a specific account in the access control list. This structure is used in the"SecurityService Service".

Table 2–1 lists the fields in this structure.

Table 2–1 AccessControlToken Structure Fields

Fields	Description
Account account	Specifies a reference to the Account structure.
int permissionMask	Specifies a combination of the following flags: 1 = Permission to read item content 2 = Permission to traverse directory 4 = Permission to change item content 8 = Permission to delete an item 16 = Permission to assign permissions to other accounts 32 = Permission to take ownership of the item 2048 = Permission to run an Oracle BI Publisher report live 4096 = Permission to schedule an Oracle BI Publisher report 8192 = Permission to view output from an Oracle BI Publisher report 65535 = Permission to grant full control of the item.

2.2 Account Structure

Use this structure to hold user names or group names. It has a flag to indicate whether the name is a user or a group. This structure is used in the "SecurityService Service".

Table 2–2 lists the fields in this structure.

Table 2–2 Account Structure Fields

Fields	Description
String accountName	Specifies an account name or group name.

Table 2–2 (Cont.) Account Structure Fields

Fields	Description
int accountType	Specifies whether the account is a user or a group or both (0 = user, 1 = catalog group, 2 = app role, 3 = all). Note the following information for advanced use of this field. If accountType is greater than or equal to 4, the system treats the Name or GUID as a pattern. Namely, accountType may be 4 for fetching users, 5 for fetching cat groups, 6 for fetching app roles, 7 all pattern. Using this field in this way can be expensive, slow, and result in the system returning many records. When receiving an Account, both Name and GUID are set. The accountType will be 0 for user, 1 for catalog group, 4 for application role.
String GUID	Specifies the unique ID which identifies the account.

2.3 ACL Structure

Use this structure to hold the access control list (ACL). This structure is used in the "SecurityService Service".

Table 2–3 list the fields in this structure.

Table 2–3 ACL Structure Fields

Fields	Description
AccessControlToken[] accessControlTokens	Specifies the full list of permissions.
String dummy	For internal purposes.

2.4 ArrayofGUIDS Structure

Use this structure to specify a list of GUIDs representing a saved result set. This structure is used in the "SecurityService Service".

Table 2–4 lists the field in this structure.

Table 2–4 ArrayofGUIDS Structure Fields

Fields	Description
String[] guid	Specifies a list of GUIDs representing the saved result set.

2.5 AuthResult Structure

Use this structure to specify authorization details during an authentication. This structure is used in the "SecurityService Service" (in the "impersonateex() Method" and "logonex() Method").

Table 2–5 lists the fields in this structure.

Table 2–5 AuthResult Structure Fields

Fields	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

Table 2–5 (Cont.) AuthResult Structure Fields

Fields	Description
boolean authCompleted	If set to TRUE, then the authorization is complete. If set to FALSE, then the authorization process is in progress and the logonex or impersonatex process should be called again.

2.6 CatalogItemsFilter Structure

Use this structure to filter catalog items and changes based on the path and timestamp. This structure is used in the "ReplicationService Service".

Table 2–6 lists the fields in this structure.

Table 2–6 CatalogItemsFilter Structure Fields

Fields	Description
String[] items	Specifies the list of folders and their descendants to include in the filter. If this value is null, then all nodes in the catalog are included.
Calendar from	Specifies the time period on which to filter. Only items and changes with timestamps within that period satisfy the filter. Either or both of those fields could be null, in which case corresponding bound is considered not set.
Calendar to	Specifies the time period on which to filter. Only items and changes with timestamps within that period satisfy the filter (from <= timestamp <= to). Either or both of those fields could be null, in which case the corresponding bound is considered not set.

2.7 CatalogObject Structure

Use this structure to retrieve or specify all information for a particular catalog object in a single method. This structure is used in the "WebCatalogService Service".

Table 2–7 lists the fields in this structure.

Table 2–7 CatalogObject Structure Fields

Fields	Description
String catalogObject	Specifies an XML representation of the object.
catalogObjectBytes	Specifies the returned content of the catalog object as string or bytes. What you specify in this field is determined by the readObjects method.
ItemInfo itemInfo	Specifies catalog information about the object, supplied in the ItemInfo common structure. For information about the ItemInfo structure, see Section 2.11, "ItemInfo Structure".
ErrorInfo errorInfo	Specifies the level of error information to be supplied as specified by the ErrorDetails argument in the readObjects method.

2.8 ErrorInfo Structure

Use this structure to retrieve error information during Presentation Catalog Service method invocations. This structure is used in the "WebCatalogService Service".

Table 2–8 lists the fields in this structure.

Table 2–8 ErrorInfo Structure Fields

Fields	Description
String code	Specifies the error code to display.
String context	Specifies the service and method in which the error occurred.
String details	Specifies detailed information about the error.
String message	Specifies a human-readable description of the error.

2.9 GetSubItemsParams Structure

Use this structure to contain optional parameters used in a getSubItems method. This structure is used in the "WebCatalogService Service".

Table 2–9 lists the fields in this structure.

Table 2–9 GetSubItemsParams Structure Fields

Fields	Descriptions
GetSubItemsFilter filter	For internal use only.
boolean includeACL	If set to TRUE, then ACL information is included in the resulting ItemInfo structures.
int withPermission and int withPermissionMask	Specifies that you want to filter the resulting items collection by access level. The only items included in the result are those for which the following expression is true: `(itemPermission & withPermissionMask) = (withPermission & withPermissionMask)` where `itemPermission` is a combination of permission flags for the current catalog item.
int withAttributes and int withAttributesMask	Specifies that you want to filter the resulting items collection by attribute flags. The only items included in the result are those for which the following expression is true: `(itemAttributes & withAttributesMask) = (withAttributes & withAttributesMask)` where `itemAttributes` is a combination of attribute flags for the current catalog item.

2.10 ImportError Structure

Use this structure to describe the cause of a failure during an import. This structure is used in the "ReplicationService Service".

Table 2–10 lists the fields in this structure.

Table 2–10 ImportError Structure Fields

Fields	Description
String item	Specifies the path to the changed item. For example, /users/jchan/analyses/.
String operation	For internal use only.
String catalogError	Specifies an error string, describing the reason for the failure.
String file	Specifies in which file the error occurred.

Table 2–10 (Cont.) ImportError Structure Fields

Fields	Description
String line	Specifies the line in which the error occurred.

2.11 ItemInfo Structure

Use this structure to contain catalog information about an object. This structure is used in the "WebCatalogService Service".

Table 2–11 lists the fields in this structure.

Table 2–11 ItemInfo Structure Fields

Fields	Description
String path	Specifies the path to the object in the catalog. For example, /users/jchan/analyses/.
ItemInfoType type	Specifies a character string that indicates the type. Valid values are: ■ Folder ■ Link ■ Missing ■ NoAccess ■ Object
String caption	Specifies the localized name of the object in the catalog. For example, in French, 'My Folders' is displayed as 'Mes Dossiers'.
int attributes	Specifies a combination of the following flags: 1 = read only 2 = archive 4 = hidden 8 = system
Calendar lastModified	Specifies the date and time that the object was last modified, in Calendar format.
Calendar created	Specifies the date and time that the object was created (saved) in the catalog, in Calendar format.
Calendar accessed	Specifies the data and time that the object was last accessed by a user, in Calendar format.
String signature	Specifies the signature of the catalog object.
NameValuePair[] itemProperties	Specifies an array of object properties.
ACL aclXX	Specifies the Access Control List for this catalog item.
Account owner	Specifies the owner of the object.
String targetPath	If the ItemInfoType field is set to "Link," this field specifies the target path for the object.

2.12 NameValuePair Structure

Use this structure to denote named properties, such as COLOR=RED. This structure is used in the "WebCatalogService Service".

Table 2–12 lists the fields in this structure.

Table 2–12 NameValuePair Structure Fields

Fields	Description
String name	Specifies a character string that contains the name of the property, such as COLOR.
String value	Specifies a character string that contains the value, such as RED.

2.13 PathMap Structure

Use this structure to specify the location to which you want to copy the data included in the export method. This structure is used by the ReplicationService Service.

Table 2–13 lists the fields in this structure.

Table 2–13 PathMap Structure Fields

Fields	Description
PathMapEntry pathMapEntries	Specifies the location to which you want to copy the data included in the export method.

2.14 Privilege Structure

Use this structure to represent global privileges. In Oracle BI EE, you configure these privileges using the Manage Privileges screen. This structure is used in Section 3.9, "SecurityService Service".

Table 2–14 lists the fields in this structure.

Table 2–14 Privilege Structure Fields

Fields	Description
String name	Specifies the name of a privilege.
String description	Specifies the description of a privilege.

2.15 QueryResults Structure

Use this structure to specify query details during query execution.This structure is used in the Section 3.11, "XMLViewService Service" (in the executeXMLQuery method).

Table 2–15 lists the fields in this structure.

Table 2–15 QueryResults Structure Fields

Fields	Description
String rowset	Specifies the rowset XML encoded in the string.
String queryID	Specifies the unique ID of the query, which can be used in fetchNext calls.
boolean finished	If set to TRUE, then there are no more rows to return. If set to FALSE, then another fetchNext call is needed to return more rows.

2.16 ReportHTMLOptions Structure

Use this structure to define options for displaying results on an HTML page. This structure is used in the Section 3.2, "HtmlViewService Service".

Table 2–16 lists the field in this structure.

Table 2–16 ReportHTMLOptions Structure Field

Field	Description
boolean enableDelayLoading	Internal use only. This field is always set to 1, which means that Oracle Business Intelligence web services is never required to provide results immediately, and displays a message indicating that it is waiting for results.
String linkMode	Specifies whether to display drills or links in the current browser window or a new browser window. For more information about valid values, see Section 2.16.1, "ReportHTMLLinksMode Enumeration".

2.16.1 ReportHTMLLinksMode Enumeration

This enumeration specifies a list of valid values for the ReportHTMLLinksMode field in the "ReportHTMLOptions Structure".

Table 2–17 lists the values in this enumeration.

Table 2–17 ReportHTMLLinksMode Enumeration Values

Values	Description
String InPlace	Specifies that drills or links should replace only the content of the current analysis without changing the rest of the page.
String NewPage	Specifies that drills or links should be displayed in a new browser window.
String SamePage	Specifies that drills or links should replace the current browser window.

2.17 ReportParams Structure

Use this structure to replace existing filters and variables in an analysis. This structure is common to all services in Oracle BI EE web services.

Table 2–18 lists the fields in this structure.

Table 2–18 ReportParams Structure Fields

Fields	Description
String[] filterExpressions	Specifies an array of Oracle Business Intelligence web services filter expressions in the form Object[] filter_ expression, filter_expression ...
Variable[] variables	Specifies an array of variable values to be set before method execution. This structure is used in Section 3.11.4, "executeXMLQuery() Method" and Section 3.7.3, "generateReportSQL() Method".
NameValuePair[] nameValues	Should be set to NULL. This field is for internal use only.
TemplateInfo[] templateInfos	Should be set to NULL. This field is for internal use only.
String viewName	Specifies which view to use when generating XML data for the analysis.

Table 2–19 shows how filter expressions are applied to an analysis.

Table 2–19 How Filter Expressions Are Applied to an Analysis in Oracle BI EE Web Services

Step	Internal Processing
1	Obtains XML representations of the analysis and each filter expression.
2	For each expression element, locates the child node of the type sqlExpression (the type is determined by the value of the xsi:type attribute), and references its inner text.
3	In the analysis XML, locates all nodes that also have a child node of type sqlExpression where the inner text matches that located in the preceding step.
4	Replaces all nodes found in Step 3 with the expression from Step 2.

Table 2–20 shows how variables are applied to an analysis.

Table 2–20 How Variables Are Applied to an Analysis in Oracle BI EE Web Services

Step	Internal Processing
1	Obtains XML representations of the analysis.
2	For each variable, locates all nodes in the analysis XML that have a type of variable, attribute scope equal to analysis, and inner text that matches the variable name.
3	Replaces each node located in Step 2 with the new variable value.

2.18 ReportRef Structure

Use this structure to reference an analysis in one of the following ways:

- The location of the analysis in the catalog.
- The ReportDef object that defines the analysis. This field should always be null.
- The XML that defines the analysis.

Note: Only one of the fields in ReportRef should be populated.

The ReportRef structure is common to all services in Oracle BI EE web services.

Table 2–21 lists the fields in this structure.

Table 2–21 ReportRef Structure Fields

Fields	Description
String reportPath	Specifies a string value that provides the path to the analysis in the catalog. For example, /users/jchan/analyses/.
String reportXML	Specifies a string value that contains the XML that defines the analysis.

2.19 SAColumn Structure

Use this structure to represent the logical column in the Subject Area. This structure is used in Section 3.5, "MetadataService Service".

Table 2–22 lists the fields in this structure.

Table 2–22 SAColumn Structure Fields

Fields	Description
String name	Specifies a column name used in SQL statements.
String displayName	Specifies a localized name, used in Oracle Business Intelligence Answers.
String description	Specifies a string to contain the description of the column name.
boolean nullable	If set to TRUE, then the column can be null.
String dataType	Specifies the type of data that a column contains. For more information, see Section 2.19.1, "SADataType Values".
boolean aggregateable	If set to TRUE, then the column can be aggregated.
String aggrRule	If the column contains aggregated data, this value specifies the type of aggregation used. For more information, see Section 2.19.2, "AggregationRule Values".

2.19.1 SADataType Values

The SADataType indicates the type of data that a column contains. The following list shows the data types available:

- BigInt
- Binary
- Bit
- Char
- Coordinate
- Date
- Decimal
- Double
- Float
- Integer
- Invalid
- LongVarBinary
- LongVarChar
- Numeric
- Real
- SmallInt
- Time
- TimeStamp
- TinyInt
- Unknown
- VarBinary
- VarChar

2.19.2 AggregationRule Values

The SADataType specifies the default aggregation rule for the column. The following list shows the aggregation functions available:

- Avg
- BottomN
- Complex
- Count
- CountDistinct
- CountStar
- DimensionAggr
- First
- Last
- Max
- Min
- None
- Percentile
- Rank
- ServerDefault
- SubTotal
- Sum
- TopN

2.20 SASubjectArea Structure

Use this structure to represent Subject Area attributes. This structure is used in Section 3.5, "MetadataService Service".

Table 2–23 lists the fields in this structure.

Table 2–23 SASubjectArea Structure Fields

Fields	Description
String name	Specifies the table name that is used in SQL statements.
String displayName	Specifies the localized name, used in Oracle Business Intelligence Answers.
String description	Specifies the description of the subject area.
SATable[] tables	Specifies a collection of tables for this subject area. For information about the SATable structure, see Section 2.21, "SATable Structure".

2.21 SATable Structure

Use this structure to represent the logical table in the Subject Area. This structure is used in the Section 3.5, "MetadataService Service".

Table 2–24 lists the fields in this structure.

Table 2–24 SATable Structure Fields

Fields	Description
String name	Specifies the table name that is used in SQL statements.
String displayName	Specifies the localized name, used in Oracle Business Intelligence Answers.
String description	Specifies the description of the table name.
SAColumn[] columns	Specifies an array of the table's columns. For information about the SAColumn structure, see Section 2.19, "SAColumn Structure".

2.22 SAWLocale Structure

Use this structure to define the locale for the current session. This structure is used in Section 3.8, "SAWSessionService Service".

Table 2–25 lists the fields in this structure.

Table 2–25 SAWLocale Structure Fields

Fields	Description
String language	Specifies the language code. Values for language should conform to the ones used in Java, in the java.util.Locale class (ISO-639, ISO-3166).
String country	Specifies the country code. Values for country should conform to the ones used in Java, in the java.util.Locale class (ISO-639, ISO-3166).

2.23 SAWSessionParameters Structure

Use this structure to define optional parameters for the current session. This structure is used in Section 3.8, "SAWSessionService Service".

Table 2–26 lists the fields in this structure.

Table 2–26 SAWSessionParameters Structure Fields

Fields	Description
SAWLocale locale	Specifies the locale to be used, supplied in the SAWLocale structure. For information about the SAWLocale structure, see Section 3.8, "SAWSessionService Service".
String userAgent	Specifies whether the HTMLView service will be used with current session. It specifies the userAgent string of the browser, where Oracle Business Intelligence Presentation Services HTML content is displayed. Oracle Business Intelligence Presentation Services uses this information to produce browser-specific HTML.
String syndicate	Internal use only.
LogonParameter logonParams	Specifies the parameters used for authentication.
boolean asyncLogon	If set to TRUE, then asynchronous login is enabled. If set to FALSE (default), then asynchronous login is not enabled.
String sessionID	Specifies the unique ID of the session. This field is used in Section 3.8.9, "logonex() Method" and Section 3.8.5, "impersonateex() Method".

2.24 SegmentationOptions Structure

Use this structure to define the segment or segment tree to override the defaults specified in the Oracle Marketing Analytics user interface. This structure is used in the Section 3.5, "MetadataService Service".

Table 2–27 lists the fields in this structure.

Table 2–27 SegmentationOptions Structure Fields

Fields	Description
OverrideType cacheOverride	Specifies how you want to override the Oracle Marketing Analytics' "Cache the block for future update counts requests" user interface option. If set to Default, then the cache override is not specified in the structure or the structure is not specified. The Default value specifies to use what is defined in the user interface option for each criteria block. If set to None, the system overrides the user interface-defined values and sets all criteria blocks to disable the "Cache the block for future update counts requests" user interface option. If set to All, the system overrides the user interface-defined values and sets all criteria blocks to enable the "Cache the block for future update counts requests" user interface option.
OverrideType countOverride	Specifies if the system should use the getCounts method to generate the count numbers. If set to Default, then the count override is not specified in the structure or the structure is not specified. If set to All, the system executes the getCounts method. When set to All, the system calculates count numbers for all criteria blocks.
NameValuePair govRules	Specifies a value to enforce the corresponding contract planning rules for the segment or segment tree.
NameValuePair prompts	Specifies the prompt values to apply to the columns in the segment or segment tree. This process filters data when generating counts. If you do not provide a value in this field, then the system does not apply filter criteria to columns in segments.
Boolean removeCacheHits	Specifies that you want to clear cache entries that contain count information. If set to True, the system queries against the most current data. To do this, the system removes all existing cache entries that contain count information for the target segment or segment tree. The system then repopulates the cache with new count number entries calculated by the getCounts method.
BigDecimal samplingFactor	Specifies the size of the data set for calculating counts. The getCounts method calculates the count number of all criteria blocks against a subset of the data determined by this value. The default value is 100. The default value determines that the count number is calculated against the whole data set.

2.25 SessionEnvironment Structure

Use this structure to return environment information for the current session. This structure is used in Section 3.8, "SAWSessionService Service".

Table 2–28 lists the fields in this structure.

Table 2–28 SessionEnvironment Structure Fields

Fields	Description
String userName	Specifies the name of the current user.
ItemInfo homeDirectory	Specifies the full path to the user's home directory in the catalog. For example, /users/<*user login ID*>.
ItemInfo[] SharedDirectories	Specifies the full paths to shared directories to which the current user has at least read access. **Note**: By default, only administrators are allowed to list direct descendents of the "/shared" directory. Retrieving the SessionEnvironment object is the only way to enable users to navigate its shared area.

2.26 StartPageParams Structure

Use this structure to define options in startPage method invocations. This structure is used in Section 3.2, "HtmlViewService Service".

Table 2–29 lists the fields in this structure.

Table 2–29 StartPageParams Structure Fields

Fields	Description
String idsPrefix	Specifies a prefix to be used with IDs and names of all HTML elements to avoid name conflicts on an HTML page.
boolean dontUseHttpCookies	If set to TRUE, thenOracle Business Intelligence Presentation Services cannot rely on cookies for passing the sessionID. Instead, the sessionID is included as a parameter in callback URLs.

2.27 TreeNodePath Structure

Use this structure to specify a segment tree path and branch Id number for a branch in the segment tree. This structure is used in the Section 3.4, "JobManagementService Service".

Table 2–30 lists the fields in this structure.

Table 2–30 TreeNodePath Structure Fields

Fields	Description
String treeNode	Specifies the segment tree's branch Id number that contains the members to include in the list.
String treePath	Specifies the path to the segment tree.

2.28 UpdateACLParams Structure

Use this structure to set options in updateACL method invocations. This structure is used in Section 3.9, "SecurityService Service".

Table 2–31 lists the fields in this structure.

Table 2–31 UpdateACLParams Structure Fields

Fields	Description
UpdateACLMode updateFlag	Specifies how to update the ACL mode. For more information, see Section 2.28.1, "UpdateACLMode Enumeration".

2.28.1 UpdateACLMode Enumeration

This enumeration specifies a list of valid values for the update flag in the UpdateACLParams Structure.

Table 2–32, " UpdateACLMode Enumeration Values" lists the values in this enumeration.

Table 2–32 UpdateACLMode Enumeration Values

Values	Description
String ReplaceACL	Specifies the ACL value to update.
String ReplaceForSpecifiedAccounts	Specifies a list of accounts to update in the ACL.
String DeleteAccountsFromACL	Specifies a list of accounts to remove from the ACL.

2.29 UpdateCatalogItemACLParams Structure

Use this structure to provide additional parameters in Section 3.10.18, "updateCatalogItemACL() Method". This structure is used inSection 3.10, "WebCatalogService Service".

Table 2–33 lists the fields in this structure.

Table 2–33 UpdateCatalogItemACLParams Structure Fields

Fields	Description
UpdateACLMode updateFlag	Specifies how to update the ACL mode. For more information, see Section 2.28.1, "UpdateACLMode Enumeration".
boolean recursive	If set to TRUE, then the method is applied to the catalog item and all descendents, which are identified by the path. If set to FALSE, then the method is only applied to the catalog item.

2.30 Variable Structure

Use this structure to reference a variable in the analysis and replace it with another variable. This structure is common to all services in Oracle BI EE web services.

Table 2–34 lists the fields in this structure.

Table 2–34 Variable Structure Fields

Fields	Description
String name	Specifies a character string that contains the name of the variable to replace.
Object value	Specifies the value of the variable.

2.31 XMLQueryExecutionOptions Structure

Use this structure to specify optional parameters during a query. This structure is used in Section 3.11, "XMLViewService Service" (in the executeXMLQuery method).

Table 2–35 lists the fields in this structure.

Table 2–35 XMLQueryExecutionOptions Structure Fields

Fields	Description
boolean async	If set to TRUE, then asynchronous query execution is enabled. If set to FALSE, then asynchronous query execution is disabled.
int maxRowsPerPage	Specifies the maximum number of rows to be returned by a executeXMLQuery or fetchNext method.
boolean refresh	If set to TRUE, then the server re-submits the query to refresh the data. If set to FALSE, then the Oracle Business Intelligence Server uses data in the cache.
boolean presentationInfo	If set to TRUE, then store localized presentation information in the metadata section of the record set XML. Presentation information consists of the following: ■ Column heading information (stored in the columnHeading field). ■ Table heading information (stored in the tableHeading field).
String type	Specifies the query ID, which can be used in logs to diagnose errors.

3

Description of Services and Methods in Oracle BI EE Web Services

This chapter describes the services and methods used by the Oracle Business Intelligence Session-Based Web Services.

This document uses JavaScript-like syntax to describes structures. The exact syntax and implementation depends on the SOAP code generation tool and the target language used by your application development environment

This chapter includes the following sections:

- Section 3.1, "ConditionService Service"
- Section 3.2, "HtmlViewService Service"
- Section 3.3, "iBotService Service"
- Section 3.4, "JobManagementService Service"
- Section 3.5, "MetadataService Service"
- Section 3.6, "ReplicationService Service"
- Section 3.7, "ReportEditingService Service"
- Section 3.8, "SAWSessionService Service"
- Section 3.9, "SecurityService Service"
- Section 3.10, "WebCatalogService Service"
- Section 3.11, "XMLViewService Service"

3.1 ConditionService Service

Use this service to evaluate Oracle BI EE conditions programatically. This service also allows users to obtain the customizable filters available in a condition.

Table 3–1 shows the supported methods.

Table 3–1 ConditionService Methods

Method Name	Description
Section 3.1.1, "evaluateCondition() Method"	Evaluates a condition saved to the catalog.
Section 3.1.2, "evaluateInlineCondition() Method"	Evaluate a condition supplied as a parameter.

Table 3–1 (Cont.) ConditionService Methods

Method Name	Description
Section 3.1.3, "getConditionCustomizableReportElements() Method"	Obtains the customizable filters of a condition saved to the catalog.

3.1.1 evaluateCondition() Method

Use this method to evaluate a Condition that is stored in the catalog. This method returns an XML string containing the result of the condition (true or false).

3.1.1.1 Signature

boolean evaluateCondition(String path, String[] reportCustomizationParameters, String sessionID);

Arguments	Description
String path	Specifies the full path and name of the Condition in the catalog. For example, /users/jchan/Conditions/IsRegionUnderBudget.
String [] reportCustomizationParameters	Specifies the customization parameters XML, which is only used if the Condition has customizable filters. This XML is validated against the customization schema available in orahome/bifoundation/web/schemas/analysis_customization.xsd.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.1.2 evaluateInlineCondition() Method

Use this method to evaluate a condition defined outside of Oracle BI EE Presentation Services. The Condition XML is supplied in the conditionXML parameter. This method returns an XML string with the result of the condition evaluation, true or false.

3.1.2.1 Signature

boolean evaluateInlineCondition(String conditionXML, String[] reportCustomizationParameters, String sessionID);

Arguments	Description
String conditionXML	Specifies the Condition XML. This XML is validated against the condition schema available in orahome/bifoundation/web/schemas/condition.xsd.
String[] ListreportCustomizationParameters	Specifies the customization parameters XML, which is only used if the Condition has customizable filters. This XML is validated against the customization schema available in orahome/bifoundation/web/schemas/analysis_customization.xsd.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.1.3 getConditionCustomizableReportElements() Method

Use this method to determine the customizable filters available in a condition that is stored in the catalog. This method returns an XML string containing the definition of the customizable filters available in the condition. The XML is in the format defined in the customization schema available in orahome/bifoundation/web/schemas/analysis_customization.xsd.

3.1.3.1 Signature

String[] getConditionCustomizableReportElements(String path, String sessionID);

Arguments	Description
String path	Specifies the full path and name of the condition in the catalog. For example, /users/jchan/Conditions/IsRegionUnderBudget.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2 HtmlViewService Service

Use this service to embed Oracle BI EE HTML results in third-party dynamic web pages, such as Active Server Pages (ASP) or JavaServer Pages (JSP), and portal frameworks. The embed process merges Oracle BI EE web services content with the content of third-party web pages.

Table 3–2 shows the supported methods.

Table 3–2 HtmlViewService Methods

Method Name	Description
Section 3.2.2, "addReportToPage() Method"	Adds results to an HTML page.
Section 3.2.3, "endPage() Method"	Destroys a server page object and all data associated with it.
Section 3.2.4, "getCommonBodyHTML() Method"	Retrieves HTML to include in the <BODY> section.
Section 3.2.5, "getHeadersHTML() Method"	Retrieves HTML to include in the <HEAD> section.
Section 3.2.6, "getHtmlforPageWithOneReport() Method"	Retrieves HTML for a page that contains only one analysis.
Section 3.2.7, "getHTMLForReport() Method"	Retrieves HTML to display a particular set of results.
Section 3.2.8, "setBridge() Method"	Specifies a bridge URL to receive communications. Can be useful when the Oracle Business Intelligence web services server and the Presentation Services that the user is accessing reside on different machines or when you want to modify the results in your application development environment.
Section 3.2.9, "startPage() Method"	Creates a new page object and returns its ID.

The methods in the HTMLViewService service extract fragments of HTML code that can be inserted in third-party web pages. Table 3–3 describes the HTML code excerpts and desired page locations.

Table 3–3 HTML Code Fragments and Page Locations for the HtmlViewService Service

HTML Code Fragment	Desired Page Location
Header	Should be inserted in the <HEAD> section of an HTML page. The code contains links to common JavaScript files and style sheets.
Report Objects	Can be inserted anywhere in the <BODY> section.
Common Body	Should be inserted in the <BODY> tag after all analysis links. The code contains hidden HTML elements that are used to implement drilldown links.

For each returned analysis object, the HTML code fragment contains a callback link that is followed automatically when the web page is loaded by the browser. The code fragment does not contain the full user interface definition of the analysis. While the analysis is being constructed by Oracle BI EE Presentation Services, the interface displays the Oracle BI EE web services "Searching..." image embedded on the third-party web page.

For smooth analysis transitioning, Oracle BI EE Presentation Services tracks the Oracle Business Intelligence analyses that have been added to third-party web pages by maintaining information in an internal logical page object during the construction of the third-party web page. The HtmlViewService service methods explicitly refer to the internal logical page by its ID.

3.2.1 About HtmlViewService Bridging and Callback URLs

To embed an analysis with active drilldown links, the HtmlViewService service allows the web browser to issue callback requests from embedded analyses to the Oracle BI EE Presentation Services server. Although it is possible to route requests directly to the Oracle BI EE Presentation Services server, in many cases it is preferable to route requests through the Oracle BI EE instance that originally serviced the third-party page. Also, in situations where Oracle BI EE Presentation Services and the third-party web server do not belong to the same Domain Name Service (DNS) domain, users may get JavaScript errors related to browser security constraints for cross-domain scripting.

To avoid these issues, use the setBridge() method to modify callback URLs to point to the third-party web server. Be aware that a web component executed by the third-party web server to re-route requests to Oracle BI EE Presentation Services is not provided. This function would need to be fulfilled by the third-party application. For more information about the setBridge() method, see Section 3.2.8, "setBridge() Method".

3.2.2 addReportToPage() Method

Use this method to add results to an HTML page.

3.2.2.1 Signature

void addReportToPage(String pageID, String reportID, ReportRef report, String reportViewName, ReportParams reportParams, ReportHTMLOptions options, String sessionID);

Arguments	Description
String pageID	Specifies a character string page ID returned by the startPage() method. For information about the startPage () method, see Section 3.2.9, "startPage() Method".
String reportID	Specifies a character string that identifies the analysis containing the results to add to the page. It should be used to reference this analysis in subsequent method invocations; for example, corresponding user interface elements generated by the Oracle Business Intelligence Presentation Services server would reference the same ID.
ReportRef report	Specifies the analysis definition, supplied in the ReportRef structure.
String reportViewName	Specifies the view to display. If this parameter is null, the analysis' default view is used. The view name should match the one used to identify the view in the analysis XML definition.
ReportParams reportParams	Optional. Specifies the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure. For more information, see Section 3, "Description of Services and Methods in Oracle BI EE Web Services".
ReportHTMLOptions options	Optional. Specifies the display options to apply to the analysis after execution, supplied in the ReportHTMLOptions structure. For more information, see Section 2.15, "QueryResults Structure".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2.3 endPage() Method

Use this method to destroy the Oracle BI EE Presentation Services server page object and all data associated with it.

3.2.3.1 Signature

void endpage(String pageID, String sessionID);

Arguments	Description
String pageID	Specifies the ID of the page object, which is returned by the startPage() method (for more information, see Section 3.2.9, "startPage() Method").
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2.4 getCommonBodyHTML() Method

Use this method to retrieve HTML to include in the <BODY> section.

3.2.4.1 Signature

String getCommonBodyHTML(String pageID, String sessionID);

Arguments	Description
String pageID	Specifies the ID of the page object, which is returned by the startPage() method (for more information, see Section 3.2.9, "startPage() Method").
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2.4.2 Returns

Returns a string containing the HTML to include in the <BODY> section.

3.2.5 getHeadersHTML() Method

Use this method to retrieve HTML to include in the <HEAD> section.

3.2.5.1 Signature

String getHeadersHTML(String pageID, String sessionID);

Arguments	Description
String pageID	Specifies the ID of the page object, which is returned by the startPage() method (for more information, see Section 3.2.9, "startPage() Method").
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2.5.2 Returns

Returns a string containing the HTML to include in the <HEAD> section.

3.2.6 getHtmlforPageWithOneReport() Method

Use this method to retrieve the HTML for a page that contains only one analysis. A page that contains only one analysis does not have <BODY> and <HEAD> sections.

3.2.6.1 Signature

String getHtmlForPageWithOneReport(String reportID, ReportRef report, String reportViewName, ReportParams reportParams, ReportHTMLOptions reportOptions, StartPageParams pageParams, String sessionID);

Arguments	Description
String pageReportID	Specifies the analysis ID returned by the getHtmlForPageWithOneReport() method. For information about the addReportToPage method, see Section 3.2.2, "addReportToPage() Method".
ReportRef report	Specifies the analysis definition, supplied in the ReportRef structure.
String reportViewName	Specifies the view to display. If this parameter is null, the analysis' default view is used. The view name should match the one used to identify the view in the analysis XML definition.

Arguments	Description
ReportParams reportParams	Optional. Specifies the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure. For more information, see Section 3, "Description of Services and Methods in Oracle BI EE Web Services".
ReportHTMLOptions reportOptions	Optional. Specifies the display options to apply to the analysis after execution, supplied in the ReportHTMLOptions structure. For more information, see Section 2.15, "QueryResults Structure".
StartPageParams pageParams	Specifies the options to use when starting the page, supplied in the StartPageParams structure. For information about the StartPageParams structure, see Section 2.26, "StartPageParams Structure".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2.7 getHTMLForReport() Method

Use this method to retrieve an HTML excerpt to display the results for a particular analysis. Before invoking this method, use the addReportToPage method to add the results to an HTML page.

3.2.7.1 Signature

String getHTMLForReport(String pageID, String pageReportID, String sessionID);

Arguments	Description
String pageID	Specifies the ID of the page object, which is returned by the startPage() method (for more information, see Section 3.2.9, "startPage() Method").
String pageReportID	Specifies the analysis ID returned by the addReportToPage() method. For information about the addReportToPage method, see Section 3.2.2, "addReportToPage() Method".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2.7.2 Returns

Returns a string containing the HTML excerpt that displays the specified analysis.

3.2.8 setBridge() Method

Use this method to specify a bridge URL to receive communications. Specifying a bridge URL can be useful when the Oracle BI EE web services server and the user's web server reside on different machines, or when you want to modify the results in your application development environment.

After the setBridge() method is called, all requests from the client browser to the Oracle BI EE Presentation Services server are sent to the bridge URL, which then forwards requests to the Oracle BI EE Presentation Services server.

3.2.8.1 Signature

setBridge(String bridge, String sessionID);

Arguments	Description
String bridge	Specifies the bridge URL. For example, `http://myserver/myapplication/sawbridge`
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2.8.2 Usage

You must make sure that the client browser provides a handler to the bridge URL in the form of a Java servlet, an Active Server Pages (ASP) page, a Common Gateway Interface (CGI), an Internet Server application programming interface (ISAPI), or an equivalent application.

You must also perform the following tasks:

- Decode the path of the requested Oracle BI EE web services resource in the RedirectURL argument of the request character string. For information about the RedirectURL argument, see Section 3.2.8.3, "How Callback URLs Are Replaced".
- Forward all other request arguments, together with all headers and the request body, to the bridge URL.
- Copy the response from the Oracle BI EE Presentation Services server to the response stream.

3.2.8.3 How Callback URLs Are Replaced

The new callback URL is based on the bridge URL, with the addition of a RedirectURL argument. The value of the RedirectURL argument should be the original value of the URL, encoded using standard URL encoding rules.

Internally, Oracle BI EE web services usually uses relative URLs for callback links. For example, if the original callback link is saw.dll?Go and the bridge URL is

```
http://myserver/myapplication/sawbridge
```

then the new callback URL is

```
http://myserver/myapplication/sawbridge?RedirectURL=saw.dll%3fGo
```

3.2.9 startPage() Method

Use this method to create a page object and returns its ID.

3.2.9.1 Signature

String startPage(StartPageParams options, String sessionID);

Arguments	Description
StartPageParams options	Specifies the options to use when starting the page, supplied in the StartPageParams structure. For information about the StartPageParams structure, see Section 2.26, "StartPageParams Structure".

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.2.9.2 Returns

Returns a string containing the Oracle Business Intelligence Presentation Services page ID.

3.3 iBotService Service

Use this service to save, edit, delete, subscribe, unsubscribe, customize, and execute Oracle BI EE agents. Note that as of the Oracle Business Intelligence 11*g* (11.1.1) release, "iBots" have been renamed to "agents."

Table 3–4 shows the supported methods.

Table 3–4 iBotService Methods

Method Names	Description
Section 3.3.1, "writeIBot() Method"	Writes a new agent into the catalog and registers it with Oracle BI Scheduler.
Section 3.3.2, "deleteIBot() Method"	Deletes an agent from the catalog and deregisters it from the Oracle BI Scheduler.
Section 3.3.3, "executeIBotNow() Method"	Executes an agent saved in the catalog.
Section 3.3.4, "moveIBot() Method"	Moves an agent from one catalog folder to another.
Section 3.3.5, "sendMessage() Method"	Sends a message to an Oracle BI EE user, group, or user and group.
Section 3.3.6, "subscribe() Method"	Subscribes to a published agent. Also customizes your subscription.
Section 3.3.7, "unsubscribe() Method"	Unsubscribes from an agent.

3.3.1 writeIBot() Method

Use this method to write a new agent to the catalog and to register it with Oracle BI Scheduler. Note that this method is different from the WebCatalogService service's "writeObjects() Method", which only writes to the catalog.

3.3.1.1 Signature

int writeIBot (CatalogObject obj, String path, boolean resolveLinks, boolean allowOverwrite, String sessionID);

Arguments	Description
CatalogObject obj	Specifies the object to be written to the catalog. The object's XML is validated against analysis_ibot.xsd, which is located in orahome/bifoundation/web/schemas directory.
String path	Specifies the full path and name of the agent in the catalog. For example, /users/jchan/iBots/BrandDollars.
boolean resolveLinks	If set to TRUE and the path in the catalog refers to a link, then the object is written to the location pointed to by the link.

Arguments	Description
boolean allowOverwrite	Specifies whether to overwrite an existing object. Set to TRUE to overwrite any object already present in the location specified by "path."
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.3.2 deleteIBot() Method

Use this method to delete a saved agent. Deleting an agent not only removes it (the object) from the catalog, but it also deregisters the agent from the Oracle BI Scheduler. Note that this method is different from the "WebCatalogService Service" "deleteitem" method because the "deleteitem" method does not deregister the agent from the Oracle BI Scheduler.

3.3.2.1 Signature

void deleteIBot (String path, String sessionID);

Arguments	Description
String path	Specifies the full path and name of the agent in the catalog. For example, /users/jchan/iBots/BrandDollars.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.3.3 executeIBotNow() Method

Use this method to execute an agent that is stored in the catalog. Note that this method does not change the agent's original schedule.

3.3.3.1 Signature

void executeIBotNow(String path, String sessionID);

Arguments	Description
String path	Specifies the full path and name of the agent in the catalog. For example, /users/jchan/iBots/BrandDollars.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.3.4 moveIBot() Method

Use this method to move an agent from one catalog folder to another. Note that this method is different from the "WebCatalogService Service" "moveItem" method because the "moveItem" method moves the catalog object **and** informs the Oracle BI Scheduler that the object was moved.

3.3.4.1 Signature

void moveIBot(String fromPath, String toPath, boolean resolveLinks, boolean allowOverwrite, String sessionID);

Arguments	Description
String fromPath	Specifies the full catalog path of the agent to be moved.
String toPath	Specifies the full catalog path where the agent will be moved to.
boolean resolveLinks	Specifies if you want to move the child objects. If this argument is set to TRUE and the path specified in the "fromPath" argument is a link, then the child object pointed to by that link will be moved.
boolean allowOverwrite	Specifies if you want to overwrite an existing object. If this argument is set to TRUE and another catalog object existed in the path specified by "toPath", it will be overwritten.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.3.5 sendMessage() Method

Use this method to send a message to a Oracle BI EE user, group, or user and group. The message is delivered according to the corresponding recipient's delivery profile, which was set up in the "My Account dialog" in Oracle BI Presentation Services.

3.3.5.1 Signature

String sendMessage(String[] recipient, String[] group, String subject, String body, String priority, String sessionID);

Arguments	Description
String[] recipient	Specifies the GUID of the Oracle BI EE user to whom you want to send the message. You can include more than one user in this argument.
String[] group	Specifies the GUID of the Oracle BI EE group to whom you want to send the message. You can include more than one group in this argument.
String subject	Specifies the subject line of the message.
String body	Specifies the text to be included in the body of the message.
String priority	Specifies the message's priority. You can specify "High," "Normal," or "Low."
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.3.6 subscribe() Method

Use this method to subscribe to a published agent. If the agent allows customization, then you can also specify the customization XML.

3.3.6.1 Signature

void subscribe(String path, String customizationXml, String sessionID);

Arguments	Description
String path	Specifies the full path and name of the agent in the catalog. For example, /users/jchan/iBots/BrandDollars.

Arguments	Description
String customizationXml	Specifies the customization XML (only if the agent allows customizations). This XML is validated against the customization schema available in orahome/bifoundation/web/schemas/analysis_customization.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.3.7 unsubscribe() Method

Use this method to unsubscribe from an agent. This method also deletes any user customizations.

3.3.7.1 Signature

void unsubscribe(String path, String sessionID);

Arguments	Description
String path	Specifies the full path and name of the agent in the catalog. For example, /users/jchan/iBots/BrandDollars.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4 JobManagementService Service

Use this service to execute marketing segmentation and list generation functionality.

Table 3–5 shows the supported methods.

Table 3–5 JobManagementService Methods

Method Name	Description
Section 3.4.1, "cancelJob() Method"	Cancels a running marketing job.
Section 3.4.2, "deleteResultSet() Method"	Deletes a saved result set.
Section 3.4.3, "getCounts() Method"	Generates the count numbers for a segment or segment tree.
Section 3.4.4, "getJobInfo() Method"	Gets status and detailed information about a marketing job.
Section 3.4.5, "getPromptedColumns() Method"	Returns prompted columns for a segment or segment tree.
Section 3.4.6, "prepareCache() Method"	Caches a segment or segment tree for list export.
Section 3.4.7, "purgeCache() Method"	Purges the whole cache, entries for a segment, or entries for the segment tree.
Section 3.4.8, "saveResultSet() Method"	Saves a list of members based on the most recent updated counts.

Table 3–5 (Cont.) JobManagementService Methods

Method Name	Description
Section 3.4.9, "writeListFiles() Method"	Generates a list for list export, segment campaign load, or segment tree campaign load.

3.4.1 cancelJob() Method

Use this method to cancel a running marketing job.

3.4.1.1 Signature

JobInfo cancelJob(BigInteger jobID, String sessionID);

Arguments	Description
BiInteger jobID	Specifies the marketing job ID.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4.2 deleteResultSet() Method

Use this method to delete one or more saved results sets of target-level objects identified by a list of GUID values belonging to the specified segment.

3.4.2.1 Signature

JobInfo deleteResultSet(String targetLevel, ArrayOfGUIDs guiDs, String segmentPath, String sessionID);

Arguments	Description
String targetLevel	Specifies the target level of the saved results set.
ArrayOfGUIDs guiDs	Specified a list of GUIDs representing the saved result set. For information on the ArrayOfGUIDs structure, see Section 2–4, " ArrayofGUIDS Structure Fields."
String segmentPath	Specifies the path to the segment.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4.3 getCounts() Method

Use this method to generate the count numbers for either a segment or a segment tree.

3.4.3.1 Signature

JobInfo getCounts(String segmentPath, String treePath, SegmentationOptions segmentationOptions, String sessionID);

Arguments	Description
String segmentPath	Specifies the path to the segment. Used when getCounts is performed on a segment.
String treePath	Specifies the path to the segment tree. Used when getCounts is performed on a segment tree.

Arguments	Description
Segmentation Options	Specifies the segment and segment tree override options supplied in the SegmentationOptions structure. These options are used instead of the segment or segment tree defaults as specified in the Oracle Marketing Analytics user interface. For more information about the SegmentationOptions structure, see Section 2.24, "SegmentationOptions Structure".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4.4 getJobInfo() Method

Use this method to get the status and detailed job information about a marketing job.

3.4.4.1 Signature

JobInfo getJobInfo(BigInteger jobID, String sessionID);

Arguments	Description
BigInteger jobID	Specifies the marketing job ID.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4.5 getPromptedColumns() Method

Use this method to return prompted columns for a segment or segment tree.

3.4.5.1 Signature

PromptedColumnInfo getPromptedColumns(String segmentPath, String treePath, String sessionID);

Arguments	Description
String segmentPath	Specifies the path to the segment. Used when getPromptedColumns is performed on a segment.
String treePath	Specifies the path to the segment tree. Used when gePromptedColumns is performed on a segment tree.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4.6 prepareCache() Method

Use this method to cache a segment or segment tree for list export.

3.4.6.1 Signature

JobInfo prepareCache(String segmentPath, String treePath, Boolean refresh, String sessionID);

Arguments	Description
String segmentPath	Specifies the path to the segment. Used when prepareCache is performed on a segment.
String treePath	Specifies the path to the segment tree. Used when prepareCache is performed on a segment tree.
Boolean refresh	If set to TRUE, then the system populates cache entries on cache hits and populates cache entries on cache misses.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4.7 purgeCache() Method

Use this method to purge the whole cache, purge entries for a segment from the cache, or purge entries for the segment tree from the cache.

3.4.7.1 Signature

JobInfo purgeCache(String segmentPath, String treePath, Boolean ignoreCacheRef, String sessionID);

Arguments	Description
String segmentPath	Specifies the path to the segment.Used when purgeCache is performed on a segment.
String treePath	Specifies the path to the segment tree. Used when purgeCache is performed on a segment tree.
Boolean ignoreCacheRef	If set to TRUE, the system purges the cache entries that are currently used and referenced by segments, segment trees, expired cache entries, and non-existent cache-entries.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4.8 saveResultSet() Method

Use this method to save a list of members that qualify for the segment or segment tree based on the most recent updated counts. The system extracts and saves the members to the segment path that you specify.

3.4.8.1 Signature

JobInfo saveResultSet(String segmentPath, TreeNodePath treeNodePath, String savedSegmentPath, SegmentationOptions segmentationOptions, String srCustomLabel, Boolean appendStaticSegment, String sessionID);

Arguments	Description
String segmentPath	Specifies the path to the segment.
TreeNodePath treeNodePath	Specifies the segment tree path and branch ID number.
String savedSegmentPath	Specifies the path and name of the new segment where the system extracts and saves the resulting members.

Arguments	Description
Segmentation Options segmentation Options	Specifies the segment and segment tree override options supplied in the SegmentationOptions structure. These options are used instead of the segment or segment tree defaults as specified in the Oracle Marketing Analytics user interface. For more information about the SegmentationOptions structure, see Section 2.24, "SegmentationOptions Structure".
String srCustomLabel	Specifies the custom label for creating a static segment.
Boolean appendStaticSegment	Specifies a flag to create a static segment entry under the same saved segment path.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.4.9 writeListFiles() Method

Use this method to generate lists for list export, segment campaign load, or segment tree campaign load.

3.4.9.1 Signature

JobInfo writeListFiles(ReportRef report, ReportParams reportParams, String segmentPath, TreeNodePath treeNodePath, SegmentationOptions segmentationOptions, String filesystem, BigInteger timeout, String sessionID);

Arguments	Description
ReportRef report	Specifies the analysis reference definition supplied in the ReportRef structure. For more information about the ReportRef structure, see Section 2.18, "ReportRef Structure".
ReportParams reportParams	Specifies the filters or variables to apply to the analysis before execution. This information is supplied in the ReportParams common structure. For more information about the ReportParams structure, see Section 2.17, "ReportParams Structure".
String segmentPath	Specifies the path to the segment.
TreeNodePath treeNodePath	Specifies the segment tree path and branch ID.
SegmentationOptions segmentationOptions	Specifies the segment and segment tree override options supplied in the SegmentationOptions structure. These options are used instead of the segment or segment tree defaults as specified in the Oracle Marketing Analytics user interface. For more information about the SegmentationOptions structure, see Section 2.24, "SegmentationOptions Structure".
String filesystem	Specifies the path to the shared directory that contains list files.
BigInteger timeout	Specifies the time out value. This is always set to 0.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.5 MetadataService Service

Use this service to retrieve descriptions of Oracle Business Intelligence Presentation Services schema objects, such as columns, tables, and subject areas.

Table 3–6 shows the supported methods.

Table 3–6 MetadataService Methods

Method Names	Description
Section 3.5.1, "clearQueryCache() Method"	Clears the query cache.
Section 3.5.2, "describeColumn() Method"	Retrieves column information for a specified column in a specified subject area and table.
Section 3.5.3, "describeSubjectArea() Method"	Retrieves subject area information for a specified subject area.
Section 3.5.4, "describeTable() Method"	Retrieves table information for a specified table in a specified subject area.
Section 3.5.5, "getSubjectAreas() Method"	Retrieves the list of subject areas available.

3.5.1 clearQueryCache() Method

Use this method to clear the query cache.

3.5.1.1 Signature

boolean clearQueryCache(String sessionID);

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.5.2 describeColumn() Method

Use this method to retrieve column information for a specified column in a specified subject area and table.

3.5.2.1 Signature

SAColumn describeColumn(String subjectAreaName, String tableName, String columnName, String sessionID);

Arguments	Description
String subjectAreaName	Specifies the subject area to be queried.
String tableName	Specifies the table to be queried.
String columnName	Specifies the name of the column to be queried.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.5.2.2 Returns

Returns an SAColumn Object. For information on the SAColumn structure, see Section 2.19, "SAColumn Structure".

3.5.3 describeSubjectArea() Method

Use this method to retrieve subject area information about the specified subject area.

3.5.3.1 Signature

SASubjectArea describeSubjectArea (String subjectAreaName, SASubjectAreaDetails detailsLevel, String sessionID);

Arguments	Description
String subjectAreaName	Specifies the subject area to be queried.
SASubjectAreaDetails detailsLevel	Specifies the information to be retrieved about the subject area. For information on the SASubjectAreaDetails structure, see Section 3.5.3.2, "SASubjectAreaDetails Values".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.5.3.2 SASubjectAreaDetails Values

Use this method to specify what information should be retrieved about the subject area. Table 3–7 lists the available values.

Table 3–7 SASubjectAreaDetails Values

Values	Description
IncludeTables	Include table list with minimum information about each table.
IncludeTablesAndColumns	Include full table and column information.
Minimum	Do not include table and column information.

3.5.3.3 Returns

Returns an SASubjectArea Object (for more information, see Section 2.20, "SASubjectArea Structure").

3.5.3.4 Usage

Depending on the value of the detailsLevel parameter, the returned object contains the information specified in Table 3–8.

Table 3–8 detailsLevel Values

Value of detailsLevel	Description
IncludeTables	Specifies that the tables field is not null and contains the collection of tables for this subject area. Each table object has the columns field set to null.
InludeTablesAndColumns	Specifies that the tables field is not null and contains the collection of tables for this subject area. For each table object the columns field contains the corresponding collection of columns.
Minimum	Specifies that the table list is not available. The tables field in the resulting subject area object is null.

3.5.4 describeTable() Method

Use this method to retrieve table information for a specified table in a specified subject area.

3.5.4.1 Signature

SATable describeTable (String subjectAreaName, String tableName, SATableDetails detailsLevel, String sessionID);

Arguments	Description
String subjectAreaName	Specifies the subject area to be queried.
String tableName	Specifies the table to be queried.
SATableDetails detailsLevel	Specifies the information to retrieve about the table. For information on the SATableDetails structure, see Section 3.5.4.2, "SATablesDetails Values".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.5.4.2 SATablesDetails Values

Used to specify the information to retrieve about the table. Table 3–9 lists the available values.

Table 3–9 SATableDetails Values

Values	Description
IncludeColumns	Populate the columns field in the SATable Object.
Minimum	Do not include column information. The columns field in the SATable Object is set to null.

3.5.4.3 Returns

Returns an SATable Object. For information on the SATable structure, see Section 2.21, "SATable Structure".

3.5.5 getSubjectAreas() Method

Use this method to retrieve the list of subject areas that are available.

3.5.5.1 Signature

List[]getSubjectAreas(String sessionID);

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.5.5.2 Returns

Returns an array of SASubjectArea objects. For information on the SASubjectArea structure, see Section 2.22, "SAWLocale Structure".

3.5.5.3 Usage

SASubjectArea objects returned by this method do not have table information available. The tables field is null. The approach to querying at all levels is to use getSubjectAreas() to retrieve the list of subject areas and then use describeSubjectArea() to retrieve the list of tables. Next, use describeTable() to retrieve the list of columns in a specified table, and finally, use describeColumn() to retrieve information on a specified column.

3.6 ReplicationService Service

Use this service to provide catalog replication methods.

Table 3–10 shows the supported methods.

Table 3–10 ReplicationService Methods

Method Names	Description
Section 3.6.3, "Import() Method"	Import changes from the log file.
Section 3.6.4, "export() Method"	Exports catalog changes to a specified log file.
Section 3.6.5, "markForReplication() Method"	Change the "replicable" flag on a specified folder and its descendants.
Section 3.6.6, "purgeLog() Method"	Cleans the specified logs.

3.6.1 ExportFlags Enumeration

This enumeration specifies the changes to export during export methods. Table 3–11 lists the values in this enumeration.

Note: Only one of the fields in ExportFlags should be populated.

Table 3–11 ExportFlags Enumeration Values

Method Names	Description
String processAllChanges	Specifies that you want to export flags that were changed in a given directory on the LOCAL computer and on REMOTE computers, after first referring to the replication logs to find out what changes were made (combines processLocalChanges and processRemoteChanges).
String processAll_ForMerge	Specifies how you want to manage conflicts in items that already exist. If the item exists and the export item's last modified time stamp for the export item is *greater than* the existing item's time stamp, then the existing item is replaced. If the item exists and the export item's last modified time stamp for the export item is *less than* the existing item's time stamp, then the existing item is *not* replaced.
String processAll_ForReplace	Specifies that you want to use the export items regardless of both the export and existing items' timestamps.

Table 3–11 (Cont.) ExportFlags Enumeration Values

Method Names	Description
String processAll_ForWriteIfNotExists	Specifies that if any items included in the export exist, they will not be replaced.
String processLocalChanges	Specifies that you want to export flags that were changed in a given directory on the LOCAL computer, after first referring to the replication logs to find out what changes were made.
String processRemoteChanges	Specifies that you want to export flags that were changed in a given directory on REMOTE computers, after first referring to the replication logs to find out what changes were made.

3.6.2 ImportFlags Enumeration

This enumeration specifies the changes to import during import methods. Table 3–12 lists the values in this enumeration.

Table 3–12 ImportFlags Enumeration Values

Method Names	Description
String processAllChanges	Specifies that you want to import flags that were changed in a given directory on the LOCAL computer and on REMOTE computers, after first referring to the replication logs to find out what changes were made (combines processLocalChanges and processRemoteChanges).
String processLocalChanges	Specifies that you want to import flags that were changed in a given directory on the LOCAL computer, after first referring to the replication logs to find out what changes were made.
String processRemoteChanges	Specifies that you want to import flags that were changed in a given directory on REMOTE computers, after first referring to the replication logs to find out what changes were made.

3.6.3 Import() Method

Use this method to import changes from the log file.

Note: In a Java environment, you must specify the import method as _ import(), which avoids conflict with the reserved word 'import'.

3.6.3.1 Signature

List[] (String filename, ImportFlags flag, Calendar lastPurgedLog, boolean updateReplicationLog, boolean returnErrors, CatalogItemsFilter filter, PathMap pathMap, String sessionID);

Argument	Description
String filename	Specifies the name of the log file.
ImportFlags flag	Specifies the subset of changes (for example, local or remote) to import. For more information, see "ImportFlags Enumeration".

Argument	Description
Calendar lastPurgedLog	Specifies the date and time that the log was last cleaned up. If the change in the export file was made after that time, then import uses local logs to determine if it should be replayed, otherwise it uses the last access time.
boolean updateReplicationLog	If set to TRUE, then the replication log is updated. If set to FALSE, then the replication log is not updated.
boolean returnErrors	If set to TRUE, then the function returns an array of ImportError objects which describes cases when changes recorded in the import file that satisfy filter conditions were not replayed.
CatalogItemsFilter filter	Specifies that you should filter changes made within a particular time period, and to catalog items in specified folders. Can be null.
PathMap pathMap	Specifies to which directory the imported changes are copied. Use this argument to copy items in one directory to another directory.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.6.3.2 Returns

Returns an ImportError structure containing the list of errors encountered. For more information, see Section 2.10, "ImportError Structure".

3.6.4 export() Method

Use this method to export catalog changes to a specified log file.

3.6.4.1 Signature

void export (String filename, CatalogItemsFilter filter, ExportFlags flag, boolean exportSecurity, String sessionID);

Argument	Description
String filename	Specifies the name of the log file.
CatalogItemsFilter filter	Specifies the subset of changes to be exported. The filter.items field cannot be null.
ExportImportFlags flag	Specifies the changes (for example, local or remote) to export. For more information, see "ExportFlags Enumeration".
exportSecurity	Specifies the security information for the corresponding object.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.6.5 markForReplication() Method

Use this method to change the "replicable" flag on a specified folder and its descendants.

3.6.5.1 Signature

void markForReplication (String item, boolean replicate, String sessionID);

Argument	Description
String item	Specifies the path of the folder.
boolean replicate	If set to TRUE, then mark the folder as replicable. If set to FALSE, then remove the replicable flag.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.6.6 purgeLog() Method

Cleans the specified logs.

3.6.6.1 Signature

void purgeLog(List[] items, Calendar timestamp, String sessionID);

Argument	Description
List[] items	List of folder paths to clean.
Calendar timestamp	Cleans only those log items where the last modified time is earlier than the timestamp.
String sessionID	A string value that contains the session ID to log off from the SOAP session. The session ID is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.7 ReportEditingService Service

Use this service to merge arguments and Oracle Business Intelligence Presentation Services data to create and return the results.

Table 3–13 shows the supported methods.

Table 3–13 ReportEditingService Methods

Method Names	Description
Section 3.7.1, "applyReportDefaults() Method"	Applies analysis default arguments to the analysis and returns the results.
Section 3.7.2, "applyReportParams() Method"	Applies report arguments to the analysis object and returns the results.
Section 3.7.3, "generateReportSQL() Method"	Retrieves the SQL query for a given analysis.

3.7.1 applyReportDefaults() Method

Use this method to apply analysis default arguments to the analysis and returns the results.

3.7.1.1 Signature

String applyReportDefaults(ReportRef reportRefs, String sessionID);

Arguments	Description
ReportRef object	Specifies the path to the analysis definition, supplied in the ReportRef common structure.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.7.1.2 Returns

Returns the result of applying the default analysis arguments to the specified analysis object.

3.7.2 applyReportParams() Method

Use this method to apply analysis arguments to the analysis and return the results.

3.7.2.1 Signature

Object applyReportParams(ReportRef reportRef, ReportParams reportParams, boolean encodeInString, String sessionID);

Arguments	Description
ReportRef reportRef	Specifies the path to the analysis definition, supplied in the ReportRef common structure.
ReportParams reportParams	Optional. Specifies the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure. For more information, see Section 3, "Description of Services and Methods in Oracle BI EE Web Services".
boolean encodeInString	If set to TRUE, then the returned analysis object is encoded as a character string.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.7.2.2 Returns

Returns the result of applying analysis arguments to the specified analysis object. If you set encodeInString to true, then the result is encoded as a character string.

3.7.3 generateReportSQL() Method

Use this method to retrieve the logical SQL query for a given analysis.

3.7.3.1 Signature

String generateReportSQL(ReportRef reportRef, ReportParams reportParams, String sessionID);

Arguments	Description
ReportRef reportRef	Specifies the path to the analysis definition supplied in the ReportRef common structure.

Arguments	Description
ReportParams reportParams	Optional. Specifies the path to the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure. For more information, see Section 3, "Description of Services and Methods in Oracle BI EE Web Services".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.7.3.2 Returns

A string containing the SQL query for the specified analysis.

3.8 SAWSessionService Service

Use this service to provide authentication methods such as logon and logoff, and other session-related methods.

Table 3–14 shows the supported methods.

Table 3–14 SAWSessionService Methods

Method Name	Description
Section 3.8.1, "getCurUser() Method"	Retrieves the current user ID for the session.
Section 3.8.2, "GetSessionEnvironment() Method"	Retrieves the environment object for the current session.
Section 3.8.3, "getSessionVariable() Method"	Retrieves a list of session variables.
Section 3.8.4, "impersonate() Method"	Logs on and then impersonates the user.
Section 3.8.5, "impersonateex() Method"	Logs on and then impersonates the user. Similar to the impersonate method, but impersonateex can specify optional session parameters.
Section 3.8.6, "keepAlive() Method"	Instructs Oracle Business Intelligence Presentation Services not to end particular sessions due to inactivity.
Section 3.8.7, "logoff() Method"	Logs the user off Oracle Business Intelligence Presentation Services.
Section 3.8.8, "logon() Method"	Authenticates the user.
Section 3.8.9, "logonex() Method"	Authenticates the user. Similar to the logon method, but logonex can specify optional session parameters.

3.8.1 getCurUser() Method

Use this method to retrieve the current user name for the session.

3.8.1.1 Signature

String getCurUser(String sessionID);

Argument	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.8.1.2 Returns

Returns a string indicating the current user name for the session.

3.8.2 GetSessionEnvironment() Method

Use this method to retrieve the environment object for the current session.

3.8.2.1 Signature

SessionEnvironment getSessionEnvironment (String sessionID);

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.8.2.2 Returns

This method returns a session environment object (for more information, see Section 2.25, "SessionEnvironment Structure").

3.8.3 getSessionVariable() Method

Use this method to retrieve a list of session variables.

3.8.3.1 Signature

List[] getSessionVariables(List[] names, String sessionID);

Arguments	Description
List[] names	Specifies the names of the session variables.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.8.3.2 Returns

This method returns values of the Oracle BI EE variables associated with the current session.

3.8.4 impersonate() Method

Use this method to log on and impersonate the user during the SAWSessionService service. This method is useful when you need to create sessions for multiple users and have only the administrator's name and password. You do not need to use the (logon) method if you use the impersonate() method.

If user authentication or impersonation fails, an exception is thrown.

3.8.4.1 Signature

String impersonate(String name, String password, String impersonateID);

Arguments	Description
String name	Specifies the user name to log on and authenticate.
String password	Specifies the password for the user. If there is no password for the user, leave this field empty (void).
String impersonateID	Specifies the user name to impersonate the authenticated user.

3.8.4.2 Returns

This method returns the session ID and sets an HTTP session cookie. The session ID is used in other methods to identify the Oracle Business Intelligence web services session.

3.8.5 impersonateex() Method

Use this method to log on and impersonate the user in the SAWSessionService service. Similar to the impersonate method, but impersonateex can specify optional session parameters. This method is useful when you need to create sessions for multiple users and have only the administrator's name and password. You do not need to use the (logon) method if you use the impersonateex() method.

If user authentication or impersonation fails, then an exception is thrown.

3.8.5.1 Signature

AuthResults impersonateex(String name, String password, String impersonateID, SAWSessionParameters sessionparams);

Arguments	Description
String name	Specifies the user name to log on and authenticate.
String password	Specifies the password for the user. If there is no password for the user, leave this field empty (void).
String impersonateID	Specifies the user name to impersonate the authenticated user.
SAWSessionParameters sessionparams	Optional. Specifies the session parameters to use, supplied in the SAWSessionParameters structure. For information about the SAWSessionParameters structure, see Section 2.23, "SAWSessionParameters Structure".

3.8.5.2 Returns

This method returns the AuthResult structure containing the session ID, and also sets an HTTP session cookie. The session ID is used in other methods to identify the Oracle Business Intelligence Presentation Services session. For more information, seeSection 2.5, "AuthResult Structure".

3.8.6 keepAlive() Method

Use this method to instruct Oracle BI EE Presentation Services not to end particular web user sessions due to inactivity. The effect of this method on session lifetime is the same as if those users performed an activity in the browser such as clicking an analysis, or invoking a method.

3.8.6.1 Signature

void keepAlive(String[] sessionID);

Argument	Description
String[] sessionID	Specifies the session IDs to remain logged on.

3.8.7 logoff() Method

Use this method to log off the user from Oracle BI EE Presentation Services.

3.8.7.1 Signature

void logoff(String sessionID);

Argument	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.8.8 logon() Method

Use this method to authenticate the user. If authentication fails, an exception is thrown.

3.8.8.1 Signature

String logon(String name, String password);

Arguments	Description
String name	Specifies the user name to authenticate.
String password	Specifies the password for the user. If there is no password, leave this field empty (void).

3.8.8.2 Returns

This method returns the session ID and sets an HTTP session cookie. The session ID is used in other methods to identify the Oracle BI EE Presentation Services session.

3.8.9 logonex() Method

Use this method to authenticate the user. Logonex() to the logon method, but logonex can specify optional session parameters. If authentication fails, an exception is thrown.

3.8.9.1 Signature

AuthResult logonex(String name, String password,
SAWSessionParameters sessionparams);

Arguments	Description
String name	Specifies the user name to authenticate.
String password	Specifies the password for the user. If there is no password, leave this field empty (void).

Arguments	Description
SAWSessionParameters sessionparams	Optional. Specifies the sessionparams to use, supplied in the SAWSessionParameters structure. For information about the SAWSessionParameters structure, see Section 2.23, "SAWSessionParameters Structure".

3.8.9.2 Returns

This method returns the AuthResult structure containing the session ID, and also sets an HTTP session cookie. The session ID is used in other methods to identify the Oracle Business Intelligence Presentation Services session.

3.9 SecurityService Service

Use this service to provide methods for identifying accounts and privileges. Table 3–15 shows the supported methods.

Table 3–15 SecurityService Methods

Method Names	Description
Section 3.9.1, "forgetAccounts() Method"	Removes an Oracle BI EE Presentation Services internal ID to account name mapping.
Section 3.9.2, "getAccounts() Method"	Searches for Oracle BI EE user accounts
Section 3.9.3, "getGlobalPrivilegeACL() Method"	Gets the Access Control List for global privileges.
Section 3.9.4, "getGlobalPrivileges() Method"	Gets the list of all global privileges.
Section 3.9.5, "getGroups() Method"	Gets a list of catalog groups that are members of the account (for example, user or group
Section 3.9.6, "getMembers() Method"	Gets direct members of the catalog group.
Section 3.9.7, "getPermissions() Method"	Get the list of permissions for the specified user.
Section 3.9.8, "getPrivilegesStatus() Method"	Lists all privileges and their statuses.
Section 3.9.9, "isMember() Method"	Confirms if a catalog group is a member of the user or group.
Section 3.9.10, "joinGroups() Method"	Adds a user to a catalog group as a member.
Section 3.9.11, "leaveGroups() Method"	Removes a member from a group.
Section 3.9.12, "renameAccounts() Method"	Change the name of an user account.
Section 3.9.13, "updateGlobalPrivilegeACL() Method"	Update the Access Control List for global privileges.

3.9.1 forgetAccounts() Method

Use this method to remove an Oracle BI EE Presentation Services internal ID to account name mapping. This action is useful when an account mapping was created by mistake, for example, as a side effect of an updateGlobalSAWPrivilegeACL method with a misspelled account name.

3.9.1.1 Signature

void forgetAccounts(List[], int cleanuplevel, String sessionID);

Argument	Description
List[]	Specifies the accounts to forget, supplied in the Account structure. For information about the Account structure, see Section 2.3, "ACL Structure".
int cleanuplevel	Specifies the amount of mapping information to remove. Set to 0 to remove the mapping from an internal account ID and a user or group name. Set to 1 to remove the user directory if accounts refer to a user.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.2 getAccounts() Method

Use this method to search for Oracle BI EE user accounts (for example, LDAP users, catalog groups, or application roles).

3.9.2.1 Signature

List[] getAccounts(List[], String sessionID);

Argument	Description
List[]	Specifies user names, catalog group names, and application role names. A flag indicates if the name is a user, group, or application role.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.3 getGlobalPrivilegeACL() Method

Use this method to retrieve the Access Control List for global privileges.

3.9.3.1 Signature

ACL getGlobalPrivilegeACL(String privilegeName, String sessionID);

Argument	Description
String privilegeName	Specifies the name of the privilege to retrieve.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.4 getGlobalPrivileges() Method

Use this method to retrieve the list of global privileges.

3.9.4.1 Signature

List[] getGlobalPrivileges(String sessionID);

Argument	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.5 getGroups() Method

Use this method to get a list of catalog groups that are members of the account (for example, user or group).

3.9.5.1 Signature

List[] getGroups(List[], Boolean expandGroups, String sessionID);

Argument	Description
List[]	Specifies user or group.
Boolean expandGroups	Specifies to expand the groups to which the members belong.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.6 getMembers() Method

Use this method to get direct members of the catalog group.

3.9.6.1 Signature

List[] getMembers(List[] group, Boolean expandGroups, String sessionID);

Argument	Description
List[]	Specifies the catalog group for which you want a list of members.
Boolean expandGroups	Specifies to expand the groups to which the members belong.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.7 getPermissions() Method

Use this method to retrieve a list of permissions for the specified user, based on the specified access control list.

Note: This method also returns any permissions that are inherited by a user's security group, even if the access control list does not specify the group's permissions.

3.9.7.1 Signature

List[] getPermissions(List[], Account account, String sessionID);

Argument	Description
List[]	Specifies the access control list for the user specified by Account account.
Account account	Specifies the name of the user for whom to find permission for the ACLs. Can be the user's name or a GUID.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.7.2 Returns

Returns permissions information in the permissionMask field in the AccessControlToken structure (for more information, see Section 2.1, "AccessControlToken Structure").

3.9.8 getPrivilegesStatus() Method

Use this method to list all privileges and their statuses.

3.9.8.1 Signature

List[] getPrivilegesStatus(List[] privileges, String sessionID);

Argument	Description
List[] privileges	Specifies a list of privileges.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.9 isMember() Method

Use this method to confirm if a catalog group is a member of the user or group.

3.9.9.1 Signature

boolean isMember(List[] group, List[] member, Boolean expandGroups, String sessionID);

Argument	Description
List[] group	Specifies the username, catalog group, or application role name.
List[] member	Specifies the name of the member to verify. Consider the example isMember(BIAdministrator, Administrator, false). This example asks if the user Administrator is a member of the BIAdministrator application role.
Boolean expandGroups	Specifies ....
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.10 joinGroups() Method

Use this method to join a catalog group as a member.

3.9.10.1 Signature

void joinGroups(List[] group, List[] member, String sessionID);

Argument	Description
List[] group	Specifies the name of the group to join or become a member. Consider the following example: join(Marketing, UserA). This example illustrates that UserA will join the Marketing catalog group.
List[] member	Specifies the name of the underlying member. For more information, see the example included in the previous argument.

Argument	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.11 leaveGroups() Method

Use this method to remove a member from a group.

3.9.11.1 Signature

void leaveGroups(List[] group, List[] member, String sessionID);

Argument	Description
List[] group	Specifies the group from which to remove a member.
List[] member	Specifies the member that you want to remove from the group.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.12 renameAccounts() Method

Use this method to change the name of a user account in the catalog.

3.9.12.1 Signature

void renameAccounts(List[] from, List[] to, String sessionID);

Argument	Description
List[] from	Specifies the old name of the account.
List[] to	Specifies a new name for the account.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.9.12.2 Returns

Returns an array of accounts. If accountsFilter returns a null value, this method returns all accounts cached by the current Presentation Services instance.

3.9.13 updateGlobalPrivilegeACL() Method

Use this method to update the Access Control List for global privileges.

3.9.13.1 Signature

void updateGlobalPrivilegeACL(String privilegeName, ACL acl, UpdateACLParams updateACLParams, String sessionID);

Arguments	Description
String privilegeName	Specifies the name of privilege to update.

Arguments	Description
ACL acl	Specifies the Access Control List to update, supplied in the ACL structure. For information about the ACL structure, see Section 2.3, "ACL Structure".
UpdateACLParams updateACLParams	Specifies the Access Control List parameters to update, supplied in the UpdateACLParams structure. For information about the UpdateACLParams structure, see Section 2.28, "UpdateACLParams Structure".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10 WebCatalogService Service

Use this service to provide methods for navigating and managing the catalog, and to read and write catalog objects in XML format. Table 3–16 shows the supported methods.

Table 3–16 WebCatalogService Methods

Method Names	Description
Section 3.10.3, "copyItem() Method"	Copies an object from one location to another in the catalog.
Section 3.10.4, "copyItem2() Method"	Generates an archive file from the catalog.
Section 3.10.5, "createFolder() Method"	Creates a new folder in the catalog.
Section 3.10.6, "createLink() Method"	Creates a link to the catalog.
Section 3.10.7, "deleteItem() Method"	Deletes an object from the catalog.
Section 3.10.8, "getItemInfo() Method"	Retrieves catalog information for an object.
Section 3.10.9, "getSubItems() Method"	Retrieves the collection of child subitems for an object in the catalog.
Section 3.10.10, "maintenanceMode() Method"	Locks the catalog during maintenance.
Section 3.10.11, "moveItem() Method"	Moves an object in the catalog to a different location in the catalog.
Section 3.10.12, "pasteItem2() Method"	Pastes the copied items.
Section 3.10.13, "readObjects() Method"	Reads an object from the catalog.
Section 3.10.14, "removeFolder() Method"	Deletes a folder from the catalog.
Section 3.10.15, "setItemAttributes() Method"	Sets attribute flags for the specified catalog item.
Section 3.10.16, "setItemProperty() Method"	Sets a property for an object in the catalog.
Section 3.10.17, "setOwnership() Method"	Take ownership of the specified item.
Section 3.10.18, "updateCatalogItemACL() Method"	Update the Access Control List for an item in the catalog.
Section 3.10.19, "writeObjects() Method"	Writes a list of objects to the catalog.

3.10.1 ErrorDetailsLevel Enumeration

This enumeration specifies a list of valid values for methods in the "WebCatalogService Service". Table 3–17 lists the values in this enumeration.

Note: Only one of the values in ErrorDetailsLevel should be selected.

Table 3–17 ErrorDetailsLevel Enumeration Values

Values	Description
String ErrorCode	Specifies that the ErrorInfo.errorCode field is populated.
String ErrorCodeAndText	Specifies that the ErrorInfo.errorCode and ErrorInfo.message fields are populated.
String FullDetails	Specifies that all ErrorInfo fields are populated.

3.10.2 ReadObjectsReturnOptions Enumeration

This enumeration specifies a list of valid values for methods in the "WebCatalogService Service". Table 3–18 lists the values in this enumeration.

Table 3–18 ReadObjectsReturnOptions Enumeration Values

Values	Description
String NoObject	Specifies that the catalogObject and catalogObjectBytes fields are not populated.
String ObjectAsString	Specifies that the catalogObject field is populated and the catalogObjectBytes fields is not populated.
String ObjectAsBinary	Specifies that the catalogObject field is not populated and the catalogObjectBytes fields is populated.
String ObjectAsBinaryUse Mtom	Specifies that the catalogObject field is not populated and the catalogObjectBytes fields is populated and using MTOM to encode the content returned by the SOAP message.

3.10.3 copyItem() Method

Use this method to copy an object from one location in the catalog to another location in the catalog.

3.10.3.1 Signature

void copyItem(String pathSrc, String pathDest, int flagACL, String sessionID);

Arguments	Description
String pathSrc	Specifies the current path to the object in the catalog.
String pathDest	Specifies the location in the catalog where the object should be copied.
int flagACL	Specified whether the item is copied with security. 0 indicates that the item is copied without security. 1 indicates that the item is copied with security.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.4 copyItem2() Method

Use this method to generate an archive file from the catalog.

3.10.4.1 Signature

DataHandler copyItem2(List[] path, boolean recursive, boolean permissions, boolean timestamps, boolean useMtom, String sessionID);

Arguments	Description
List[] path	Specifies the location in the catalog from which the archive was created.
boolean recursive	Specifies whether the child-level folders were included in the archive.
boolean permissions	Specified whether the items are copied with security.
boolean timestamps	Specifies whether to preserve the items' time stamps were preserved.
boolean useMtom	Specifies whether MTOM was used to encode the content returned by the SOAP message.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.5 createFolder() Method

Use this method to create a folder in the catalog.

3.10.5.1 Signature

void createFolder(String path, boolean createIfNotExists, boolean createIntermediateDirs, String sessionID);

Arguments	Description
String path	Specifies the location in the catalog where the folder should be created, including the name of the new folder.
boolean createIfNotExists	If set to TRUE, then the folder object is created in the catalog if it does not already exist. If set to FALSE, then the folder object is not recreated if it already exists.
boolean createIntermediateDirs	If set to TRUE, then an intermediate directory is created. If set to FALSE, the an intermediate directory is not created.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.6 createLink() Method

Use this method to create a link to the catalog.

3.10.6.1 Signature

void createLink(String path, String pathTarget, boolean overwriteIfExists, String sessionID);

Arguments	Description
String Path	Specifies the path to the parent object in the catalog.
String TargetPath	Specifies the location in the catalog to which the link being created should refer.
boolean overwriteIfExists	If set to TRUE, then the link is overwritten if it already exists in the catalog. If set to FALSE, then the link is not overwritten if it already exists in the catalog.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.7 deleteItem() Method

Use this method to delete an object from the catalog. To delete a folder, read Section 3.10.14, "removeFolder() Method".

3.10.7.1 Signature

void deleteItem(String path, String sessionID);

Arguments	Description
String path	Specifies the path to the object in the catalog.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.8 getItemInfo() Method

Use this method to retrieve catalog information for an object.

3.10.8.1 Signature

ItemInfo getItemInfo(String path, boolean resolveLinks, String sessionID);

Arguments	Description
String path	Specifies the path to the object in the catalog.
boolean resolveLinks	If set to TRUE and the path in the catalog refers to a link, then Oracle Business Intelligence retrieves information for the object pointed to by the link.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.8.2 Returns

Returns catalog information for an object in an ItemInfo structure. For more information, see Section 2.11, "ItemInfo Structure".

3.10.9 getSubItems() Method

Use this method to retrieve the collection of child sub-items for an object in the catalog.

3.10.9.1 Signature

List[] getSubItems(String path, String mask, boolean resolveLinks, GetSubItemsParams options, String sessionID);

Arguments	Description
String path	Specifies the path to the parent object in the catalog.
String mask	Specifies a mask that indicates the child subitems to retrieve. The mask character is an asterisk (*). To retrieve all child subitems, use a single asterisk.
boolean resolveLinks	If set to TRUE and the path in the catalog refers to a link, then information is retrieved for the child subitems of the object pointed to by the link.
GetSubItemsParams options	Optional. Specifies parameters to supply to the GetSubItemsParams structure. For information about the GetSubItemsParams structure, see Section 2.9, "GetSubItemsParams Structure".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.9.2 Returns

Returns a collection of child subitems in an ItemInfo structure. For more information, see Section 2.11, "ItemInfo Structure".

3.10.10 maintenanceMode() Method

Use this method to lock the catalog during maintenance.

3.10.10.1 Signature

void maintenanceMode(boolean flag, String sessionID);

Arguments	Description
boolean flag	Set to TRUE if the catalog is locked.
String sessionID)	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.11 moveItem() Method

Use this method to move an object in the catalog to a different location in the catalog.

3.10.11.1 Signature

void moveItem(String pathSrc, String pathDest, int flagACL, String sessionID);

Arguments	Description
String pathSrc	Specifies the current path to the object in the catalog.
String pathDest	Specifies the location in the catalog where the object should be moved.

Arguments	Description
int flagACL	Specified whether the item is moved with security. 0 indicates that the item is moved without security. 1 indicates that the item is moved with security.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.12 pasteItem2() Method

Use this method to paste the copied items.

3.10.12.1 Signature

void pasteItem2(Binary stream DataHandler archive, String replacePath, int flagACL, int flagOverwrite, String sessionID);

Arguments	Description
Binary stream DataHandler archive	Specifies the returned content of the item as string or bytes. What you specify in this field is determined by the readObjects method.
String replacePath	Specifies the location to paste the copied item.
int flagACL	Specified whether the item is pasted with security. 0 indicates that the item is pasted without security. 1 indicates that the item is pasted without security.
int flagOverwrite	Specifies whether the pasted item overwrites existing item. 0 indicates replace all, 1 indicates replace old, 2 indicates replace none, and 3 indicates force replace.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.13 readObjects() Method

Use this method to read an object from the catalog and return a CatalogObject structure.

3.10.13.1 Signature

List[] readObjects(List[] paths, boolean resolveLinks, ErrorDetailsLevel errorMode, ReadObjectsReturnOptions returnOptions, String sessionID);

Arguments	Description
List[] paths	Specifies the location of the object in the catalog.
boolean resolveLinks	If set to TRUE and the path in the catalog refers to a link, then the object is written to the location pointed to by the link.
ErrorDetailsLevel errorMode	Specifies the amount of error information in the errorInfo field in the CatalogObjects structure. For more information, see Section 2.7, "CatalogObject Structure".
ReadObjectsReturnOptions returnOptions	Specifies a list of valid values. For more information, see "ReadObjectsReturnOptions Enumeration".

Arguments	Description
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.13.2 Returns

Returns an array of CatalogObjects.

> **Note:** If a read operation fails for a catalog object (for example, due to an invalid path or insufficient privileges), the errorInfo field for that object contains a description of the error.

3.10.14 removeFolder() Method

Use this method to delete a folder and its contents from the catalog. To delete an object other than a folder and its contents, see Section 3.10.7, "deleteItem() Method".

3.10.14.1 Signature

void removeFolder(String path, boolean recursive, String sessionID);

Arguments	Description
String path	Specifies the path to the folder in the catalog.
boolean recursive	If set to TRUE, then remove the specified folder and its contents. If set to FALSE, then only remove the specified folder if it is empty, otherwise display an exception message.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.15 setItemAttributes() Method

Use this method to set attribute flags for a specified catalog item.

3.10.15.1 Signature

void setItemAttributes (List[] path, int value, int valueOff, boolean recursive, String sessionID);

Arguments	Description
List[] path	Specifies the path to the folder in the catalog.
int value	Specifies which attributes will be added. Specifies a combination of the following flags: 1 = read only 2 = archive 4 = hidden 8 = system
int valueOff	Specifies which attributes will be removed. See the above int value cell for flags.
boolean recursive	Specifies whether to set the properties of items in sub-directories.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.16 setItemProperty() Method

Use this method to set a property for an object in the catalog.

3.10.16.1 Signature

void setItemProperty(List[] path, List[] name, List[] value, boolean recursive, String sessionID);

Arguments	Description
List[] path	Specifies the path to the object in the catalog.
List[] name	Specifies the name of the property to set.
List[] value	Specifies the new setting for the property.
boolean recursive	Specifies whether to set the properties of items in sub-directories.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.17 setOwnership() Method

Use this method to take ownership of the specified item.

3.10.17.1 Signature

void setOwnership(List[]path, Account owner, boolean recursive, String sessionID);

Arguments	Description
List[] path	Specifies the location in the catalog of the object to take ownership.
Account owner	Specifies the account to assign as owner.
boolean recursive	If set to TRUE, then apply this action to the specified folder and its contents. If set to FALSE, then only apply this action to the specified folder if it is empty, otherwise display an exception message.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.18 updateCatalogItemACL() Method

Use this method to update the Access Control List for an item in the catalog.

3.10.18.1 Signature

void updateCatalogItemACL(List[] path, ACL acl, UpdateCatalogItemACLParams options, String sessionID);

Fields	Description
List[] path	Specifies the path to the object in the catalog.
ACL acl	Specifies the Access Control List. For more information, see Section 2.3, "ACL Structure".

Fields	Description
UpdateCatalogItemACLParams options	Specifies additional parameters. For more information, see Section 2.29, "UpdateCatalogItemACLParams Structure".
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.19 writeObjects() Method

Use this method to write an array of objects to the catalog.

3.10.19.1 Signature

List[] writeObjects(List[] catalogObjects, boolean allowOverwrite, boolean createIntermediateDirs, ErrorDetailsLevel errorMode, String sessionID);

Argument	Description
List [] catalogObjects	Specifies the objects to write to the catalog, supplied in the CatalogObject structure. For information about the CatalogObject structure, see Section 2.7, "CatalogObject Structure". All fields of object.itemInfo are ignored, except for the array of item properties, which are applied to the object. The signature of the resulting document is always COXmlDocument1.
boolean allowOverwrite	If set to TRUE, then if the object already exists in the catalog, it is overwritten. If set to FALSE, then if the object already exists in the catalog, it is not overwritten.
boolean createIntermediateDirs	If set to TRUE and the path in the catalog refers to a link, then the object is written to the location pointed to by the link.
ErrorDetailsLevel errorMode	Specifies the amount of error information in the errorInfo field in the CatalogObjects structure.
String sessionID	Specifies the session ID, which is usually returned by the logon method. If the SOAP client engine can handle HTTP cookies, you can omit the session ID or set it to null.

3.10.19.2 Returns

An array of ErrorInfo objects.

3.11 XMLViewService Service

Use this service to retrieve results from Oracle BI EE Presentation Services in XML format. Table 3–19 shows the supported methods.

Table 3–19 XMLView Service Methods

Method Name	Description
Section 3.11.2, "cancelQuery() Method"	Cancels the current query.
Section 3.11.3, "executeSQLQuery() Method"	Runs a SQL query.
Section 3.11.4, "executeXMLQuery() Method"	Runs an XML query.

Table 3–19 (Cont.) XMLView Service Methods

Method Name	Description
Section 3.11.5, "fetchNext() Method"	Returns the next page of data rows.
Section 3.11.6, "getPromptedFilters() Method"	Returns a filter XML structure containing only the analysis' columns with a prompted filter.
Section 3.11.7, "upgradeXML() Method"	Returns an upgraded version of the requested object.

3.11.1 XMLQueryOutputFormat Enumeration

This enumeration specifies a list of valid values for the executeSQLQuery() Method and executeXMLQuery() Method. For example, you might want to return data rows and metadata, or data rows only. Table 3–20 lists the values in this enumeration.

Note: Only one of the values in XMLQueryOutputFormat can be selected.

Table 3–20 XMLQueryOutputFormat Enumeration Values

Values	Description
String SAWRowsetData	Specifies that the query returns only data rows.
String SAWRowsetSchema	Specifies that the query returns only metadata.
String SAWRowsetSchemaAndData	Specifies that the query returns both metadata and data rows.

3.11.2 cancelQuery() Method

Use this method to cancel a query and clean up resources associated with the query. This method should only be used if the query row set is not scrolled to the last row in the data set returned.

Note: If you use this method when the query row set is scrolled to the last row in the data set returned, query data is cleaned up during the last fetchNext method invocation.

3.11.2.1 Signature

void cancelQuery(String queryID, String sessionID);

Argument	Description
String queryID	Specifies the unique ID of the query.
String sessionID	Specifies the unique ID of the session.

3.11.3 executeSQLQuery() Method

Use this method to execute a SQL query and return the results of the query.

Note: If the results returned exceed one page, you need to use the fetchNext() Method to return the next page of rows.

3.11.3.1 Signature

QueryResults executeSQLQuery(String sql, XMLQueryOutputFormat outputFormat, XMLQueryExecutionOptions executionOptions, String sessionID);

Argument	Description
String sql	Specifies the string of SQL code to execute.
XMLQueryOutputFormat outputFormat	Specifies the output format (for more information, see Section 2.31, "XMLQueryExecutionOptions Structure").
XMLQueryExecutionOptions executionOptions	Specifies the query execution options (for more information, see Section 2.31, "XMLQueryExecutionOptions Structure").
String sessionID	Specifies the unique ID of the session.

3.11.3.2 Returns

Returns the results of the query as one or more rows of data in a QueryResults structure (for more information, see Section 2.15, "QueryResults Structure").

3.11.4 executeXMLQuery() Method

Use this method to execute an XML query and return the results of the query.

> **Note:** If the results returned exceed one page, you need to use the fetchNext() Method to return the next page of rows.

3.11.4.1 Signature

QueryResults executeXMLQuery(ReportRef report, XMLQueryOutputFormat outputFormat, XMLQueryExecutionOptions executionOptions, ReportParams reportParams, String sessionID);

Argument	Description
ReportRef reportRef	Specifies the analysis definition, supplied in the ReportRef common structure.
XMLQueryOutputFormat outputFormat	Specifies the output format (for more information, see Section 2.31, "XMLQueryExecutionOptions Structure").
XMLQueryExecutionOptions execution Options	Specifies the query execution options (for more information, see Section 2.31, "XMLQueryExecutionOptions Structure").
ReportParams reportParams	Optional. Specifies the filters or variables to apply to the analysis before execution, supplied in the ReportParams common structure. For information about the ReportParams structure, see Section 2.17, "ReportParams Structure".
String sessionID	Specifies the unique ID of the session.

3.11.4.2 Returns

Returns the results of the query as one or more rows of data in a QueryResults structure (for more information, see Section 2.15, "QueryResults Structure").

3.11.5 fetchNext() Method

Use this method to return the next page of rows retrieved by a query.

> **Note:** The page returned might contain zero rows. If the finished flag is not set, the remaining rows might not be available immediately.

3.11.5.1 Signature

QueryResults fetchNext(String queryID, String sessionID);

Argument	Description
String queryID	Specifies the unique ID of the query, which is returned in the QueryResults object.
String sessionID	Specifies the unique ID of the session.

3.11.5.2 Returns

Returns the next page of query results as one or more rows of data in a QueryResults structure (for more information, seeSection 2.15, "QueryResults Structure").

3.11.6 getPromptedFilters() Method

Use this method to retrieve a saved analysis' prompted columns or the prompted columns from an analysis' xml definition. Note that to create an analysis with a prompted column, you must assign the isPrompted operator to it.

3.11.6.1 Signature

List[] getPromptedFilters(ReportRef report, String sessionID);

Argument	Description
ReportRef report	Specifies the analysis' reportPath or a reportXml (report definition).
String sessionID	Specifies the unique ID of the session.

3.11.7 upgradeXML() Method

Use this method to upgrade Oracle BI EE objects that reside outside of the catalog. This method upgrades object from Oracle BI EE 10*g* to Oracle BI EE 11*g*. Use this method primarily for Oracle BI Add-in for Microsoft Office to upgrade the analyses that were saved within Oracle BI Office and not to the catalog. You should only use this method if you want to upgrade the objects in your Oracle BI Office catalog.

Note that if you use methods like executeXmlQuery and use the ReportRef argument to pass an Oracle BI EE 10*g* report definition, that report will be upgraded before the query is run.

3.11.7.1 Signature

String upgradeXML(String xml, String sessionID);

Argument	Description
String xml	Specifies the catalog object's xml.

Argument	Description
String sessionID	Specifies the unique ID of the session.

4

Using the Oracle Business Intelligence Server Metadata Web Service

This chapter introduces the Oracle Business Intelligence Server Metadata Web Service, and describes configuring the web service connection to Oracle BI Server, the methods used to call the web service, securing the web service, and using Oracle BI Server XML procedures.

This chapter includes the following sections:

- Section 4.1, "Overview of the Oracle BI Server Metadata Web Service"
- Section 4.2, "Configuring the Oracle BI Server Metadata Web Service"
- Section 4.3, "Calling the Oracle BI Server Metadata Web Service"
- Section 4.4, "Using the Oracle BI Server XML Procedures"

4.1 Overview of the Oracle BI Server Metadata Web Service

The Oracle BI Server uses a semantic model to store and control behavior related to its operations. The semantic layer, called the Common Enterprise Information Model, is captured inside of metadata that is managed by the Oracle BI Administration Tool.

In previous versions of Oracle Business Intelligence, you would use the Administration Tool to access the information stored in the metadata repository. However, in this version of Oracle Business Intelligence, the Oracle BI Server Metadata Web Service allows you to use a web service interface to call Oracle BI Server stored procedures. These procedures provide the ability to obtain information about the metadata and to modify the metadata.

For example, a developer who wants to update the server repository with new metadata, doesn't have to open Administration Tool or restart the server with a modified repository. The developer can use a stored procedure to directly apply the patch XML created from two different versions of the repository.

There are two techniques of calling the Oracle BI Server Metadata Web Service, synchronous and asynchronous. In cases where the response returns immediately, the synchronous technique might be adequate. However, because request processing can be delayed, it is often useful to utilize the asynchronous technique of calling the Web Service so the client application can continue its work and handle the response later on.

4.2 Configuring the Oracle BI Server Metadata Web Service

This topic contains the following sections:

- Section 4.2.1, "Configuring the Oracle BI Server Metadata Web Service Connection to Oracle BI Server"
- Section 4.2.2, "Securing the Oracle BI Server Metadata Web Service"

4.2.1 Configuring the Oracle BI Server Metadata Web Service Connection to Oracle BI Server

Use this topic if you are not using the default installation of BI Server. By default the Oracle BI Server Metadata Web Service attempts to connect to an instance of BI Server running at the following URL:

```
jdbc:oraclebi://localhost:9703/
```

However, if the BI Server is deployed on a different machine, uses a different port, or if you want to configure an SSL connection between the web service and the server, you should use a Oracle Business Intelligence JDBC data source that you create in Oracle WebLogic Server Administration Console. The Oracle BI Server Metadata Web Service will look for a data source with the following name, and if found, will use it to connect to the Oracle BI Server.

```
jdbc/bi/server
```

4.2.1.1 Setting Up an Oracle Business Intelligence JDBC Data Source

For instructions about how to create an Oracle Business Intelligence JDBC data source, see "How to Create a BI JBDC Data Source in Oracle WebLogic" in *Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Enterprise Edition*.

However, note the following requirements for setting up an Oracle Business Intelligence JDBC data source for use with the Oracle BI Server Metadata Web Service. After you have finished setting up this data source, be sure to stop and start the web service so that your preferences are applied.

- In the **URL** field located on the Connection Properties page of the New JDBC Data Source wizard, specify the port that you configured for Oracle BI Server. The existing instructions state that you should specify "9703," but it is likely that "9703" will not be the correct port number.
- After you created and saved the BI JDBC data source, you must modify the data source's settings. Access the Settings for the data source and click the Connections Pool tab. Modify the settings as follows:
  - In the **Statement Cache Size** field, enter 0. Specifying this value disables the statement cache.
  - In the Advanced tab, go to the **Connection Creation Retry Frequency** field and enter 10 (or any number larger than 0).

4.2.2 Securing the Oracle BI Server Metadata Web Service

You secure the Oracle BI Server Metadata Web Service using Oracle Web Services Manager (WSM). To call the Oracle BI Server Metadata Web Service, you must configure your client code with the correct client side policy, and provide security credentials for a user with permission to call the stored procedures.

This topic contains the following sections:

- Section 4.2.2.1, "Applying Policies"
- Section 4.2.2.2, "Configuring WSM"

- Section 4.2.2.3, "Assigning the manageRepositories Permission"

4.2.2.1 Applying Policies

You apply a security policy by choosing one from a predefined list that is appropriate for the security guidelines of your organization, then assign it to the web service.

An example of a web service security policy is:

```
oracle/wss11_saml_token_with_message_protection_service_policy
```

To successfully call the Oracle BI Server Metadata Web Service, the client code must set up a matching client side policy, and supply the required information. For example, the client side policy that matches the service policy above is:

```
oracle/wss11_saml_token_with_message_protection_client_policy
```

In this example, the client code would have to add a valid SAML token for a user with permission to call the stored procedures to the SOAP message headers. The security policy on the web service ensures that any SOAP messages without a valid token are rejected. If the token is valid, the web service connects to Oracle BI Server using the System user credentials and impersonates the user specified in the token.

For more information about WSM and how to apply and configure predefined policies, see *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

4.2.2.2 Configuring WSM

You must configure WSM before you can successfully use the Oracle BI Server Metadata Web Service. The configuration for the Oracle BI Server Metadata Web Service and the Action Framework are identical.

For information about configuring WSM to work with the Oracle BI Server Metadata Web Service, see "Configuring Oracle Web Services Manager".

4.2.2.3 Assigning the manageRepositories Permission

You must provide the user who will run the Oracle BI Server Metadata Web Service with the manageRepositories permission. For information about how to set up security and apply this permission, see *Oracle Fusion Middleware Security Guide for Oracle Business Intelligence Enterprise Edition*.

4.3 Calling the Oracle BI Server Metadata Web Service

This section contains information about calling the Oracle BI Server Metadata Web Service:

- Section 4.3.1, "Calling the Oracle BI Server Metadata Web Service Synchronously"
- Section 4.3.2, "Calling the Oracle BI Server Metadata Web Service Asynchronously"

4.3.1 Calling the Oracle BI Server Metadata Web Service Synchronously

The Oracle BI Server Metadata Web Service is normally deployed at the following location, for synchronous calls:

```
http://server:port/AdminService/AdminService
```

The WSDL can be found at the following location:

```
http://server:port/AdminService/AdminService?WSDL
```

Use the following methods to synchronously call any stored procedure on the Oracle BI Server:

- Section 4.3.1.1, "callProcedure() Method"
- Section 4.3.1.2, "callProcedureWithResults() Method"
- Section 4.3.1.3, "startExtender() Method"
- Section 4.3.1.4, "Sample Code"

4.3.1.1 callProcedure() Method

Use this method to call procedures that are not expected to return a result set. Table 4–1 contains information about this method's arguments.

Table 4–1 callProcedure Method Arguments

Argument	Data Type	Description
procedureName	String	Name of the procedure to call.
parameters	List of parameters	A sequence of parameter objects. Each parameter defines a type and value corresponding to the stored procedure arguments. Refer to the WSDL for the supported parameter types.

4.3.1.2 callProcedureWithResults() Method

Use this method to call procedures that are expected to return a result set.

This method returns an object that mimics the result set as a sequence of ResultsRow objects. Each ResultsRow contains one or more columns, and each column defines a type and value. If an error occurs, most stored procedures return at least one row with a single string column. Therefore, it is important to carefully check the returned results.

Procedures that return a result set are NQSModifyMetadata, NQSQueryMetadata, NQSExtractMetadataProject (returns record sets of binary data), NQSQueryProjects. These are all query type procedures where the results are interesting, except for NQSModifyMetadata that simply returns the error messages (if any), that occur during the execution of the procedure. You can call NQSModifyMetadata synchronously without callProcedureWithResults, you just wouldn't know what the error messages are.

Table 4–2 contains information about this method's arguments.

Table 4–2 callProcedure Method Arguments

Argument	Data Type	Description
procedureName	String	Name of the procedure to call.
parameters	List of parameters	A sequence of parameter objects. Each parameter defines a type and value corresponding to the stored procedure arguments. Refer to the WSDL for the supported parameter types.

4.3.1.3 startExtender() Method

The method is for internal use only, and calls the BI Server Extender utility that imports flex object changes from ADF sources and maps them to the Business Model and Mapping layer, and Presentation layer.

This method returns a string indicating that the startExtender command was started.

For more information, see "Automatically Mapping Flex Object Changes Using the biserverextender Utility" in *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition*.

The WSDL signature for the startExtender () method is:

```
<xsd:sequence>
  <xsd:element name="adminBeanServerUrl" type="xsd:string"/>
  <xsd:element name="biAdminUser" type="xsd:string"/>
  <xsd:element name="biAdminPassword" type="xsd:string"/>
  <xsd:element name="jobID" type="xsd:string"/>
  <xsd:element name="connectionDetails" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="startExtender" type="tns:startExtender"/>
<xsd:complexType name="startExtenderResponse">
  <xsd:sequence>
    <xsd:element name="return" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

For example, a startExtender() call might be something like this:

```
startExtender(localhost:7001", "wlsuser", "wlspassword", "
   <ConnectionDetails>
     <ConnectionPool>
      <ConnectionPoolName>"oracle.apps.fscm.model.analytics.
        applicationModule.FscmTopModelAM_FscmTopModelAMLocal"."Connection Pool"
      </ConnectionPoolName>
     </ConnectionPool>
     <NewEssbaseConnection>
       <DatabaseName>computer.example.com:10215</DatabaseName>
       <UserName>FUSION_APPS_GL_ESSBASE_APPID</UserName>
       <Password>D7EDED84BC624A917F5B462A4DCA05
            CDCE256EEEEEDC97D5575BA27BADBC24F0E7675
             6D92C558D91CA53D63E2D7450E3FAEA57FA6B914D4D</Password>
       <CubeNamesList>
         <CubeName>BI_VF_USA_Accounting_Flexfie</CubeName>
       </CubeNamesList>
     </NewEssbaseConnection>
   </ConnectionDetails>");
```

Table 4–3 contains information about this method's arguments.

Table 4–3 startExtender() Method Arguments

Argument	Data Type	Description
procedureName	String	Name of the procedure to call.

Table 4–3 (Cont.) startExtender() Method Arguments

Argument	Data Type	Description
parameters	List of parameters	A sequence of parameter objects. Each parameter defines a type and value corresponding to the stored procedure arguments. Refer to the WSDL for the supported parameter types.

4.3.1.4 Sample Code

Use the following code to set up client-side security. You can use this code to set up a client-side instance of the Admin Service configured to pass authentication information in a SAML token. This code must reside in an authenticated web application. The authentication information in the session will be passed automatically to the Oracle BI Server Metadata Web Service. You will have to generate client-side proxy classes to use code such as the example code.

In this example, the proxies have been generated in the package adminwebservice.client.

```
import adminwebservice.client.*;

   /**
    * Looks up an instance of the Admin Service and sets the correct
       authentication information on it.
    * @return Admin Proxy to the admin service.
    */
   protected Admin getAdminService() throws Exception {

     String SERVICE_NAMESPACE = "http://ws.admin.obiee.oracle/";
     String SERVICE_NAME = "AdminService";
     String PORT_NAME = "AdminPort";
     String WSS_POLICY_CONFIG_FILE_SAML = "wss-policy-saml.xml";

     URL wsdlURL = new URL
        ("http://localhost:7001/AdminService/AdminService?WSDL");
     QName serviceQname = new QName(SERVICE_NAMESPACE, SERVICE_NAME);
     QName portQname = new QName(SERVICE_NAMESPACE, PORT_NAME);

     ServiceDelegateImpl serviceDelegate = new ServiceDelegateImpl
        (wsdlURL, serviceQname, OracleService.class);
      Admin admin = serviceDelegate.getPort( portQname, Admin.class);

      ((BindingProvider)admin).getRequestContext().put(ClientConstants.CLIENT_
          CONFIG,toElement(this.getClass().getResourceAsStream(WSS_POLICY_CONFIG_
           FILE_SAML)));

      ((BindingProvider)admin).getRequestContext().put(BindingProvider.
           ENDPOINT_ADDRESS_PROPERTY, getServiceUrl());
       return admin;
   }
```

WSS_POLICY_CONFIG_FILE_SAML

WSS_POLICY_CONFIG_FILE_SAML is the resource name of a file with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<oracle-webservice-clients>
```

```
   <webservice-client>
      <port-info>
         <policy-references>
            <policy-reference uri="oracle/wss11_saml_token_with_
                  message_protection_client_
                   policy"category="security"/>
         </policy-references>
      </port-info>
   </webservice-client>
</oracle-webservice-clients>
```

4.3.2 Calling the Oracle BI Server Metadata Web Service Asynchronously

The Oracle BI Server Metadata Web Service is normally deployed at the following location for asynchronous calls:

```
http://server:port/AsyncAdminService/service
```

The WSDL can be found at the following locations for asynchronous web services:

```
http://server:port/AsyncAdminService/service?wsdl
```

From the client perspective, the asynchronous method call consists of two one-way message exchanges. Before initiating the asynchronous call, the client must deploy a callback service to listen for the response from the asynchronous web service.

This section contains the following topics:

- Section 4.3.2.1, "Creating the Callback Service"
- Section 4.3.2.2, "Configuring the Callback Service"
- Section 4.3.2.3, "Sample Code"

4.3.2.1 Creating the Callback Service

Use the Create Web Service Proxy wizard in JDeveloper, select **Generate As Async** to generate an asynchronous proxy. For more information about creating web service clients using the wizard, see "Creating Web Service Proxies" in the JDeveloper Online Help. You can modify the generated callback service code to process the response. You must then deploy the callback service as a web service. Once deployed, add the URL of the callback service to the client code as the **replyTo** field.

For more information about creating the callback service, see "Defining Asynchronous Web Service Clients" in *Oracle Fusion Middleware Developer's Guide for Oracle Infrastructure Web Services*.

4.3.2.2 Configuring the Callback Service

You can attach policies to the following asynchronous components:

- Client calling asynchronous web service.
- Asynchronous web service.
- Asynchronous callback client.
- Asynchronous callback service.

To attach policies to the components at runtime you can use Fusion Middleware Control. The asynchronous web service and client policies must comply with one another. Similarly, the asynchronous callback client and callback service policies must comply with one another.

For more information about configuring security for the callback service, see "Attaching Policies to Asynchronous Web Services and Clients" in *Oracle Fusion Middleware Developer's Guide for Oracle Infrastructure Web Services*.

4.3.2.3 Sample Code

The following code can be used to create and initialize client-side instance of the asynchronous metadata web service. The code includes an example of configuring the client-side instance to pass authentication information required for the oracle/wss_username_token_client_policy policy. You will have to generate client-side proxy classes to use code such as the example code.

```
   /**
    * Looks up an instance of the Asynchronous Admin Service and
    * sets the correct authentication information on it.
    * @return Asynchronous Admin Proxy to the admin service.
    */
   protected AsyncAdmin getAsyncAdminService() throws Exception {
      String SERVICE_NAMESPACE = "http://ws.admin.obiee.oracle/";
      String SERVICE_NAME = "AsyncAdminService";
      String SECURITY_POLICY = "oracle/wss_username_token_client_policy";

      URL wsdlURL = new
URL("http://localhost:7001/AsyncAdminService/service?wsdl");
      QName serviceQname = new QName(SERVICE_NAMESPACE, SERVICE_NAME);

      AsyncAdminService service = new AsyncAdminService(wsdlURL, serviceQname);
      SecurityPoliciesFeature securityFeature =
         new SecurityPoliciesFeature(new String[] { SECURITY_POLICY });
      AsyncAdmin admin = service.getAsyncAdminPort(securityFeature);

      Map<String, Object> requestContext = ((BindingProvider)
admin).getRequestContext();
      requestContext.put(BindingProvider.USERNAME_PROPERTY, "weblogic");
      requestContext.put(BindingProvider.PASSWORD_PROPERTY, "welcome1");

      return admin;
   }
```

The following code shows how to specify callback information before calling methods of the asynchronous metadata web service:

```
String CALLBACK_URL =
"http://localhost:7001/ClienCallback/AsyncAdminResponseImplPort";
AddressingVersion WS_ADDR_VER = AddressingVersion.W3C;

WSEndpointReference replyTo = new WSEndpointReference(CALLBACK_URL, WS_ADDR_VER);
Header replyToHeader = replyTo.createHeader(WS_ADDR_VER.replyToTag);

String messageID = UUID.randomUUID().toString();
WSBindingProvider wsbp = (WSBindingProvider) getAsyncAdminService();
wsbp.setOutboundHeaders(new StringHeader(WS_ADDR_VER.messageIDTag, messageID),
replyToHeader);
```

4.4 Using the Oracle BI Server XML Procedures

Oracle Business Intelligence provides several Oracle BI Server XML procedures that you can use to call the Oracle BI Server via JDBC or ODBC. You can only use the Oracle BI Server XML procedures with repositories in binary (RPD) format.

This topic contains detailed information on the following procedures:

- Section 4.4.1, "Extract Project Procedure"
- Section 4.4.2, "Modify Metadata Procedure"
- Section 4.4.3, "Query Metadata Procedure"
- Section 4.4.4, "Query Projects Procedure"

4.4.1 Extract Project Procedure

Use the Extract Project procedure to extract a project from the repository. The output results set contains a row for each project defined in the repository.

This procedure has the following structure:

```
NQSExtractMetadataProject
Input
   1.PROJECT_NAMES
   2.RPD_NAME (optional)
Output
   1.RPD_OBJECT
```

Table 4–4 contains the Extract Project procedure's input and output parameters and their descriptions.

Table 4–4 Extract Projects Procedure Parameters and Descriptions

Parameter	Description
PROJECT_NAMES	A delimited list of project names that will be sent to the procedure for extraction.
RPD_NAME	(Optional) This parameter is currently not used. It is reserved for future use.
RPD_OBJECT	Contains a single BLOB object with the binary representation for the extracted repository.

4.4.2 Modify Metadata Procedure

Use the Modify Metadata procedure to apply changes to the repository. The output result set will contain zero or one row. If the output returns one row, that row will contain error messages.

Note that you can modify metadata using other methods, such as command line XML utilities, for example, biserverxmlgen, biserverxmlexec, and biserverxmlcli, and by copying and pasting within the Administration Tool. For more information about biserverxmlgen, biserverxmlexec, and biserverxmlcli command-line tools, see "Generating and Executing XML" in *Oracle Fusion Middleware XML Schema Reference for Oracle Business Intelligence Enterprise Edition*

This procedure has the following structure:

```
NQSModifyMetadata
Input
   1.XUDML_TEXT
   2.ORIGINAL_RPD (optional)
   3.RPD_NAME  (optional)
   4.IGNORE_XUDML_ERRORS  (optional)
Output
   1.ERROR_MESSAGES
```

Table 4–5 contains the Modify Metadata procedure's input and output parameters and their descriptions.

Table 4–5 Modify Metadata Procedure Parameters and Descriptions

Parameter	Description
XUMDL_TEXT	The full Oracle BI Server XML text that will be applied to the server. In general, the metadata definitions will be applied directly, overwriting existing definitions, unless the original repository is set in the input parameter.
ORIGINAL_RPD	(Optional) The name of the repository to which Oracle Business Intelligence compares the Oracle BI Server XML. Oracle Business Intelligence compares the Oracle BI Server XML fragment to the original RPD that you specify, and if Oracle Business Intelligence detects a conflict, it does not apply the Oracle BI Server XML. If no conflict is detected, Oracle Business Intelligence applies the Oracle BI Server XML.
RPD_NAME	(Optional) This parameter is currently not used. It is reserved for future use.
IGNORE_XUDML_ ERRORS	(Optional) Ignore all non-fatal errors. Examples of non-fatal errors are unresolved objects, duplicated objects, and broken expressions. Set this parameter to true. Note that the value is case-insensitive.
ERROR_MESSAGES	Contains any returned errors resulting from the query execution.

4.4.3 Query Metadata Procedure

Use the Query Metadata procedure to query the repository for metadata. The output result set will contain as many rows as there are object matches. You must concatenate the results to obtain the fully formed Oracle BI Server XML document.

This procedure has the following structure:

```
NQSQueryMetadataObjects
Input
   1.OBJECT_TYPES
   2.OBJECT_NAME  (optional)
   3.OBJECT_PARENT (optional)
   4.CHILD_FLAG  (optional)
   5.QNAME_QUERY_FLAG (optional)
   6.RPD_NAME  (optional)
   7.CHILD_TYPE (optional)
Output
   1.XUDML_TEXT
   2.ERROR_MESSAGES
```

Table 4–6 contains the Query Metadata procedure's input and output parameters and their descriptions.

Table 4–6 Query Metadata Procedure Parameters and Descriptions

Parameter	Description
OBJECT_TYPES	A list of object types for which to search. You can query for multiple object types by adding a comma between object types. For example, you can query for subject area and presentation catalog objects by setting the object_types to a value of '2000,4004'. In this example, the subject area object has a type ID value of 2000 and presentation catalog object has a type ID value of 4004.
OBJECT_NAME	(Optional) Filter the objects matched by a name. This name is not the fully qualified name. For example, to query for all subject area named "AtomicStar", you would set the OBJECT_TYPES parameter to "2000" and the OBJECT_NAME parameter to "AtomicStar". OBJECT_NAME cannot be empty when OBJECT_PARENT is being used.
OBJECT_PARENT	(Optional) Filter the objects based on the name of the parent object. Use this parameter to specify a fully qualified name.
CHILD_FLAG	(Optional) If this parameter is set to true, then the query returns all the child objects of the queried objects. For example, if you specify the object type parameter as Business Model and set the child flag parameter to true, then all objects under the Business Model in the tree are returned.
QNAME_QUERY_FLAG	(Optional) If this parameter is set to true, then the query returns qualified names instead of XUDML fragments. If this parameter is not set, then the query returns XUDML fragments by default.
RPD_NAME	(Optional) This parameter is currently not used. It is reserved for future use.
CHILD_TYPE	(Optional) If this parameter is set to true, then the query filters the list of children by the types provided in the comma separated list of type IDs and returns only those types of children. If this parameter is left empty by default, then all types of children objects are returned. This parameter is applicable only when the CHILD_FLAG parameter is set to true.
XUDML_TEXT	Contains the returned XML representation of all the metadata. If the XML data is too large to fit in one row, it will be split and passed back in multiple rows. You must concatenate the results to obtain the fully formed Oracle BI Server XML document.
ERROR_MESSAGES	Contains any returned errors resulting from the query execution.

4.4.4 Query Projects Procedure

Use the Query Projects procedure to determine what projects are in the repository. The output results set contain the list of projects defined in the repository.

This procedure has the following structure:

```
NQSQueryMetadataProjects
Input
   1.RPD_NAME  (optional)
Output
   1.PROJECT_NAME
```

Table 4–7 contains the Query Projects procedure's input and output parameters and their descriptions.

Table 4–7 Query Projects Procedure Parameters and Descriptions

Parameter	Description
RPD_NAME	(Optional) This parameter is currently not used. It is reserved for future use.
PROJECT_NAME	Contains a rowset for each project. Note that it is the responsibility of the caller to collect the entire rowset.

5

Using Actions to Integrate Oracle BI EE with External Systems

This chapter describes the Oracle BI EE Action Framework and the configuration required to enable the Action Framework. Describes the credentials, privileges, and permissions required to add actions to Oracle BI EE content. Describes how to set up targets by action type (for example, Navigate to Siebel CRM and Invoke a Web Service).

This chapter includes the following sections:

- Section 5.1, "What is the Action Framework?"
- Section 5.2, "Overview of the Action Framework Configuration"
- Section 5.3, "Configuring the Action Framework"
- Section 5.4, "Overview of Action Security"
- Section 5.5, "Adding and Maintaining Credentials for Use With the Action Framework"
- Section 5.6, "Target Functionality for Actions"

5.1 What is the Action Framework?

The Action Framework is a component of the Oracle BI EE architecture. It consists of the following items:

- Actions web services for creating and invoking actions that are deployed in the application server.
- Components that reside within the Presentation Server and Scheduler Services.
- Actions-specific JavaScript in the presentation tier for creating actions and invoking certain action types directly from the browser.

To prepare the Action Framework for use in Oracle BI Presentation Services, you must perform the following tasks:

- Configure the Action Framework. For more information, see Section 5.2, "Overview of the Action Framework Configuration" and Section 5.3, "Configuring the Action Framework".
- Secure Actions. For more information, see Section 5.4, "Overview of Action Security" and Section 5.5, "Adding and Maintaining Credentials for Use With the Action Framework".

- Set up action targets. For more information, see Section 5.6, "Target Functionality for Actions".

5.1.1 What Functionality is Provided by the Action Framework?

The Action Framework provides functionality for creating, managing, and invoking actions. Actions provide functionality to:

- Navigate to related content.
- Invoke operations, functions, or processes in external systems.

Actions are created and managed in the Oracle BI Presentation Services user interface. Actions can be included within analyses, dashboards, agents, KPIs, and Scorecard objectives. There are several different types of actions, for example, Navigate to a Web Page, Invoke a Web Service, and Invoke a Browser Script.

For a list of action types, their descriptions, and information about creating and using actions, see "Working with Actions" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

5.1.2 Action Types and Action Execution

The actions that are available in Oracle BI EE are categorized into two groups: those actions that navigate to related content; and those actions that invoke operations, functions, or processes in external systems. Actions are further categorized into action types based on the technology they invoke (for example, URL or web service).

For a description of each action types and information about adding Action links to business intelligence content, see "Working with Actions" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

For information about the Contextual Event action type, which developers can use when adding Oracle BI EE objects to Oracle ADF applications, see "Passing Business Intelligence Content with the Oracle BI EE Contextual Event Action" in *Oracle Fusion Middleware Developer's Guide for Oracle Business Intelligence Enterprise Edition*.

Table 5–1, " System Components that Execute Actions" shows each action type and the system components that execute them. For more information about how a specific action type is executed and the other system components that are used in the process, see Section 5.6, "Target Functionality for Actions".

Table 5–1 System Components that Execute Actions

Action Type	Executed by...
Navigate to BI Content	Browser
Navigate to a Web Page	Browser
Navigate to EPM Content	Browser
Navigate to E-Business Suite	Browser
Navigate to Siebel CRM	Browser
Invoke a Web Service	AES
Invoke a Java Method (EJB)	AES
Invoke a Browser Script	Browser
Invoke an HTTP Request	AES
Invoke Server Script	Scheduler

Table 5–1 (Cont.) System Components that Execute Actions

Action Type	Executed by...
Invoke Agent	Scheduler
Java Jobs	Javahost

5.2 Overview of the Action Framework Configuration

Some action types are automatically available when Oracle BI EE is installed, while others require specific configuration to make them available. For action types requiring additional configuration, you must provide information about the external systems hosting functionality to be invoked by the actions, including the location of the target functionality and access details.

Where security policies are applied to action targets (for example, a target web service that is secured using a SAML-based policy), you must create security policy files in the same location.

A keystore is also required for security policies that include encryption or signing. For information about policy files and keystores, see Section 5.4, "Overview of Action Security".

Certain actions require credentials to be entered into the credential store. These credentials are used for either browsing for action targets when creating an action or when invoking the target action. For information about additional credentials for actions, see Section 5.5, "Adding and Maintaining Credentials for Use With the Action Framework".

5.2.1 Configuration Checklist by Action Type

This section summarizes the configuration you must perform to use each action type. Table 5–2, " Configuration Requirements by Action Type" lists each action type and its required confirmation. Note the following column descriptions:

- Column 1 contains the name of each action type displayed by the Oracle BI EE user interface.
- Column 2 indicates whether an entry in the configuration file is required so that the Oracle BI EE Presentation Services action menus displays the action type.
- Column 3 indicates whether a registry entry can be added to the configuration file to enable a browsable list of action targets in the web front-end.
- Column 4 indicates whether additional credentials are required in the credential store.
- Column 5 indicates the action types that can implement various security policies and therefore require policy files and, where necessary, additional keystore information.

Table 5–2 Configuration Requirements by Action Type

Action Type	Requires Configuration Entry?	Supports Registries?	Additional Credentials Required?	Policy Enabled?
Navigate to BI Content	No	No (Browse for navigation targets enabled by default.)	No	No
Navigate to Web Page	No	No	No	No
Navigate to EPM Content	Yes (Registry)	Yes (Mandatory)	Yes	No
Navigate to E-Business Suite	Yes	No	No (Requires Oracle E-Business Suite security integration.)	No
Navigate to Siebel CRM	Yes	No	No (Requires Oracle's Siebel CRM integration.)	No
Invoke a Web Service	No	Yes (Optional)	No (Optional)	Yes
Invoke a Java Method (EJB)	Yes (Registry)	Yes (Mandatory)	Yes	No
Invoke a Browser Script	No	No (Browse for navigation targets enabled by default.)	No	No
Invoke an HTTP Request	No	No	No	No
Invoke Server Script	No	No	No	No
Invoke Agent	No	No	No	No
Java Job	No	No	No	No

5.2.2 Overview of Targets

All actions require a target. A target is something to navigate to or an operation, function, or process to invoke. Before you create an action, you should confirm that the target for the action is in place. For example, a target URL should be available for a Navigate to a Web Page action, and a target web service should be in place for an Invoke Web Service Action.

Some action types have targets within the Oracle Business Intelligence system (for example, Navigate to BI Content), while other action types are primarily for invoking functionality or navigating to content in external systems (for example, Invoke a Java Method). In all cases, the process of creating an action assumes that a suitable target already exists for the action to consume.

For example, suppose that a content designer is building a dashboard that requires an action that the user clicks to book a meeting room. To accomplish this task, you might use Oracle JDeveloper to create and deploy a web service named Room Booking Service with an operation named bookRoom. This is the target web service operation to be invoked by the action. After you create and deploy the web service, the content designer is able to create an "Invoke a Web Service" action.

5.3 Configuring the Action Framework

The Oracle BI EE installation contains a configuration file named ActionFrameworkConfig.xml. You manually edit this configuration file to specify how you want the Action Framework to behave. This configuration file is located by default in the following location:

```
<Oracle Middleware Home>\user_projects\domains\bifoundation_
domain\config\fmwconfig\biinstances\coreapplication
```

The configuration file contains several elements. Table 5–3, " Action Framework Configuration Elements" describes each element.

Table 5–3 Action Framework Configuration Elements

Element Name	Description
location-alias	Use to enable actions to refer to a location alias rather than a fixed URL. Setting up aliases can make migrating between test and production systems easier. For more information about this element, see Section 5.3.1, "Aliases".
registry	Use to provide information about the pre-configured registries for web services, EJBs, and so on. that provide predefined sets of targets for actions. For more information about this element, see Section 5.3.2, "Registries".
content-type	Use to provide information about the types of content for which a registry returns information. For more information about this element, see Section 5.3.3, "Content Types".
account	Use to provide information about gateway accounts. Use for authenticating to action targets. For more information about this element, see Section 5.3.4, "Accounts".
policy	Use to provide information about the location of Oracle Web Services Manager (OWSM). For more information about this element, see Section 5.3.5, "Policies".
proxy	Use to provide information about proxy settings for accessing web services or URLS. For more information about this element, see Section 5.3.6, "Proxy".
ebusinesssuiteconfig	Use to specify whether the custom action "Navigate to E-Business Suite" displays in the Oracle BI Presentation Services user interface. For more information about this element, see Section 5.3.7, "ebusinesssuiteconfig".
siebelcrmconfig	Use to specify whether the custom action "Navigate to Siebel CRM" displays in the Oracle BI Presentation Services user interface. For more information about this element, see Section 5.3.8, "siebelcrmconfig".

A sample action configuration file is available in Appendix A, "Sample Files". You can use the samples contained in this appendix along with information in this section to understand the various ways to configure actions.

After you modify the configuration file, you need to restart the Managed Server in WebLogic Server that is hosting your Oracle BI EE environment. For general information about how to restart the Managed Server in WebLogic Server, see "Confirming If the Managed Server is Running and Starting It" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

5.3.1 Aliases

This topic describes the location alias element. Location aliases provide a substitution mechanism so that actions refer to an alias rather than a fixed URL. Aliases are useful if you want to switch between test and production systems. The aliases element can contain zero or more location-alias elements. Note the following example:

```
<location-alias>
     <alias>webservicehost</alias>
     <actual>myserver.mycompany.com:7001</actual>
</location-alias>
```

Based on the `webservicehost` alias in this example, an Oracle BI EE user is able to create an action to invoke a target web service through the following WSDL:

```
http://msyserver.mycompany.com:7001/MyWebService/myservice.wsdl
```

The action saved to the catalog stores the reference to the WSDL as:

```
http://@[webservicehost]/MyWebService/myservice.wsdl
```

When this action is invoked, the Action Framework substitutes the alias for the actual value before invoking the service.

The Administrator can later change the "actual" element value within the `location-alias` element so that the saved action points to a different URL without requiring an update the action.

An alias can refer to the server and the application path. Therefore, the following example is also valid. Note that the alias should not include the full path to the target WSDL.

```
<location-alias>
     <alias>webservicehost</alias>
     <actual>myserver.mycompany.com:7001/MyWebService/</actual>
</location-alias>
```

5.3.2 Registries

This topic describes the registry element. A registry defines how the Action Framework should access a browsable library of action targets. For more information about action types and their configuration requirements, see Section 5.2, "Overview of the Action Framework Configuration".

You can create registry definitions to support the following action types:

- Navigate to Oracle's Hyperion Enterprise Performance Management (EPM) content
- Invoke a Web Service
- Invoke a Java Method (EJB)

5.3.2.1 Navigate to EPM Content Action Type Registry Example

The Hyperion registry contains no additional elements beyond the standard registry elements. The location element should be used to define the Hyperion URL to interrogate. You will need to specify an account with access to the Hyperion targets. The following example is an entry for an EPM content action type. See Table 5–4, " Registry Entry Elements and Descriptions" for a description of each element.

```
<registry>
    <id>HDPreg</id>
    <name>Hyperion Directory Provider</name>
    <content-type>
      <typename>epm</typename>
      <displayname>Hyperion Applications</displayname>
      <actionType>URLActionType</actionType>
    </content-type>
```

```
    <provider-class>oracle.bi.action.registry.epm.HDPRegistry
    </provider-class>
    <description>Hyperion Financial Reports Registry</description>
    <location>
      <path>http://epms.mycompany.com:1901/workspace/browse/listXML</path>
    </location>
    <service-access>
      <account>EPM</account>
    </service-access>
</registry>
```

5.3.2.2 Invoke a Java Method Action Type Registry Example

The following example is an entry for an EJB registry of Java methods. The ejb-targets element, which is embedded within the custom-config element, tells the EJB registry specifically about the application server hosting the EJBs and the EJBs that should be exposed as registries from that server.

See Table 5–4, " Registry Entry Elements and Descriptions" for a description of each element.

```
<registry>
   <id>reg03</id>
   <name>Sample EJBs</name>
   <content-type>java</content-type>
   <provider-class>oracle.bi.action.registry.java.EJBRegistry</provider-class>
   <description>Custom Java classes which can be invoked as action
      targets</description>
   <location>
      <path/>
   </location>
   <custom-config>
      <ejb-targets>
         <appserver>
           <context-factory>weblogic.jndi.WLInitialContextFactory
           </context-factory>
           <jndi-url>t3://localhost:9704</jndi-url>
           <server-name>localhost</server-name>
           <account>WLSJNDI</account>
           <ejb-exclude>mgmt</ejb-exclude>
           <ejb-exclude>PopulationServiceBean</ejb-exclude>
         </appserver>
         <ejb-app>
           <server>localhost</server>
           <app-context>ActionSamplesEJB</app-context>
         </ejb-app>
      </ejb-targets>
   </custom-config>
</registry>
```

5.3.2.3 Invoke a Web Service Action Type Registry Example

The following example is an entry for a WSIL registry of web services. See Table 5–4, " Registry Entry Elements and Descriptions" for a description of each element.

```
<registry>
    <id>reg1</id>
    <name>Sample Web Services</name>
    <content-type>webservices</content-type>
    <provider-class>oracle.bi.action.registry.wsil.WSILRegistry</provider-class>
    <description></description>
```

```
    <location>
       <path>http://localhost:9704/ActionSamples/inspection.wsil</path>
    </location>
    <service-access>
       <path>/Sample Web Services/Rating Service</path>
       <policy>SAMLPolicy</policy>
       <propagateIdentity>true</propagateIdentity>
    </service-access>
    <service-access>
       <path>/Sample Web Services/Customer Rating Service</path>
       <account>PayrollUser</account>
       <policy>userNamePolicy</policy>
       <propagateIdentity>false</propagateIdentity>
    </service-access>
</registry>
```

5.3.2.4 Registry Elements Descriptions

Table 5–4, " Registry Entry Elements and Descriptions" contains each registry element and its description.

Table 5–4 Registry Entry Elements and Descriptions

Element	Description
registry	This element is the outer-most element that contains the registry definition.
id	This element holds an internal identifier that must be unique for each registry.
name	This element is the display name that is used by the ActionRegistryService when displaying the list of registries. The name that you specify in this element is also used as the root path for any targets within this registry.
content-type	You can use Oracle supplied values only. This element points to a content-type, which is another element defined in the Action Framework configuration file. For more information about the content types you can specify in this element, see Section 5.3.3, "Content Types". Note that if you specify EPM in this element, you must add the `<typename>`, `<displayname>`, and `<actionType>` sub-elements. If you do not correctly enter this element, then the Navigate to EPM action type will not display in the Oracle BI Presentation Services user interface. For an example of how to properly enter this element for EPM, see Section 5.3.2.1, "Navigate to EPM Content Action Type Registry Example."
provider-class	You can use Oracle-supplied values, only. Table 5–5, " Provider-Class Element Values" contains the valid values for this element.
description	This element is reserved for future use.
location	This element specifies the path to the registry that may contain aliased elements. For example, a WSIL file.

Table 5–4 (Cont.) Registry Entry Elements and Descriptions

Element	Description
service-access	This element is optional. You can define zero or more service-access elements in a registry definition. The service-access element specifies the authentication required to access the registry and, where specified, sub-paths within that registry. Note the following: ■ If a registry element does not have any service-access elements, no authentication is used when accessing services from that registry. ■ The service-access elements that specify a path element apply the account element only to services found under the specified path within the registry. ■ If a registry element contains a service-access element with just an account element that points to an account defined in the accounts element, then that account is used to access all services within that registry that are not specified by specific path-based service-access elements.
path	This element is optional. This element is located within the service-access element and specifies a path to a subset of the services in the registry. The path is not a physical location. The path hierarchy starts with the registry name and can be found in the user interface by expanding the nodes that lead to a particular action target when creating a new action.
account	This element is optional. This element is located within the service-access element and points to an instance of an account element defined elsewhere in the configuration file. If a registry element contains a service-access element with just an account element, then that account is used to access all services within that registry that are not specified by specific path-based service-access elements. For webservices registries, the credentials held against the account referenced are used by the Action Framework to invoke target actions if the propagateIdentity element has a value of false. For EPM registries, the credentials held against the account are used by the Action Framework to browse for EPM content.
policy	This element is optional. Use this element for webservices registries only. This element is located within the service-access element and points to a policy defined in the policies section elsewhere in the configuration file. This element is required for accessing target web services that define a WS-Security policy. Note that when invoking secured web services in this way, you need to correctly configure OWSM to work with Actions. For more information, see Section 5.5.3, "Configuring Oracle Web Services Manager".
propogateIdentity	Use this element for web services registries only. This element must be set if the policy element is set. This element is located within the service-access element and can be set to true or false. A value of true is valid when used in conjunction with policies that assert identity (for example, SAML-based policies) to enable propagating the identity of the user who initially invoked the action. If the propagateIdentity element is set to true, the account element becomes redundant and instead the identity of the user who invoked the action (for example, in Oracle BI Presentation Services or run as in scheduler) is used to invoke the target web service. Setting this element to true in conjunction with a username/password token policy is not supported.
custom-config	Use this element for java registries only. This element is used to specify how to access the EJBs on the target application server.

Table 5–4 (Cont.) Registry Entry Elements and Descriptions

Element	Description
ejb-targets	Use this element for java registries only. This element defines the application server where EJBs are deployed in the appserver element, and one or more J2EE applications containing EJBs on the specified appserver that is exposed in the ejb-app elements.
appserver	Use this element for java registries only. This element is located within the ejb-target element and defines the connection and authentication information to connect to the application server where EJBs are deployed. This element contains the following elements: ■ context-factory – This element contains name of the class used to do JNDI lookups for the application server on which the Action Framework web services are deployed. You must set this element to `weblogic.jndi.WLInitialContextFactory`. ■ jndi-url – This element contains the URL used in JNDI lookups to query the JNDI directory on the application server hosting the target EJBs. ■ server-name – This element contains the name used internally to refer back to this appserver element, which is located in the server element of subsequent ejb-app elements. ■ account – This element contains the name of an account defined elsewhere in the configuration file. This element must point to credentials that have sufficient permissions to query the JNDI directory on the application server that is hosting target EJBs. ■ ejb-exclude – This element is used to exclude EJBs or applications from a given application or application server. When the EJB registry finds a match for this value in the JNDI hierarchy, it stops searching for EJBs down that branch of the JNDI tree.
ejb-app	Use this element for java registries only. This element points to one or more J2EE applications containing EJBs. This element contains the following elements: ■ server - This element's value needs to match the server-name element located in the appserver element. ■ app-context - This element contains the application context where the target EJBs were deployed.

5.3.2.5 Valid Values for the Provider-Class Element

Table 5–5, " Provider-Class Element Values" contains the valid values for the provider-class element. For more information about the provider-class element, see Table 5–4, " Registry Entry Elements and Descriptions".

Table 5–5 Provider-Class Element Values

Related Action Type	Content Type	SPI Class
Navigate to EPM Content	EPM	oracle.bi.action.registry.epm.HDPRegistry
Invoke a Java Method (EJB)	Java	oracle.bi.action.registry.java.EJBRegistry
Invoke a Web Service	Web Services	oracle.bi.action.registry.wsil.WSILRegistry

5.3.3 Content Types

This topic describes the content-type element. Each content-type element provides meta-information about the types of services to which the registries are connecting. The content types provided during installation should not be modified. You can use only the content types shown in the following example:

```
<content-types>
    <content-type>
```

```
      <typename>webservices</typename>
      <displayname>Web Services and BPEL Processes</displayname>
      <actionType>WebServiceActionType</actionType>
    </content-type>
    <content-type>
      <typename>epm</typename>
      <displayname>Hyperion Applications</displayname>
      <actionType>URLActionType</actionType>
    </content-type>
    <content-type>
      <typename>misc</typename>
      <displayname>Mixed Services</displayname>
      <actionType>URLActionType</actionType>
    </content-type>
    <content-type>
      <typename>java</typename>
      <displayname>Java Actions</displayname>
      <actionType>JavaActionType</actionType>
    </content-type>
</content-types>
```

5.3.4 Accounts

This topic describes the account element. The account element defines gateway accounts, which are used for authentication to action targets. The registry element's service-access element then references the accounts that you have specified. For more information about the registry section, see Section 5.3.2, "Registries".

The following example is an account entry. Table 5–6, " Account Elements and Descriptions" describes each element.

```
<account>
    <name>SAMLTest</name>
    <description>Test SAML Account</description>
    <adminonly>false</adminonly>
    <credentialkey>SAMLTest</credentialkey>
</account>
```

5.3.4.1 Account Elements Descriptions

Table 5–6, " Account Elements and Descriptions" contains each account element and its description. For information about storing credentials, see Section 5.5, "Adding and Maintaining Credentials for Use With the Action Framework".

Table 5–6 Account Elements and Descriptions

Element	Description
name	This element specifies the name referenced in registry service-access elements, or the appserver element for Java registries. Each account must have a unique name within the Action Framework configuration.
description	This element is reserved for future use.
adminonly	This element is reserved for future use. You must set this element to false.

Table 5–6 (Cont.) Account Elements and Descriptions

Element	Description
credentialkey	This element is used for looking up credentials in Oracle's Credential Store Framework (CSF). A set of credentials with the specified key must exist in the domain credential store within a credential map called oracle.bi.actions. Note that this map does not exist by default and needs to be created. See "Adding and Maintaining Credentials for Use With the Action Framework".

5.3.5 Policies

The policy element is used to describe the location of OWSM client policy files used in accessing secured web services as action targets. Each policy element provides a name to reference the policy that may then be used by service-access elements as a shorthand to specify that the AES should apply the policy file referred to when invoking an action target covered by that service-access element.

> **Note:** To invoke a secured web service as described above, you must configure OWSM to work with Actions. For more information, see Section 5.5.3, "Configuring Oracle Web Services Manager".

The following example is a policy entry. Table 5–7, " Policy Elements and Descriptions" describes each element.

```
<policy>
    <name>SAMLPolicy</name>
    <policyfile>ActionsSAMLPolicy.xml</policyfile>
</policy>
```

5.3.5.1 Policy Elements Descriptions

Table 5–7, " Policy Elements and Descriptions" contains each policy element and its description.

Table 5–7 Policy Elements and Descriptions

Element	Description
name	This element specifies the name referenced in service-access elements. Each policy must have a unique name within the Action Framework configuration. For more information about the service-access element, see Section 5.3.2.4, "Registry Elements Descriptions".
policyfile	Use this element to enable actions to invoke web services that have WS policies applied to them. This element specifies the name of a file stored in the same directory as the ActionFrameworkConfig.xml file. This file contains an OWSM policy that OWSM applies at runtime when a web service action that was created from a registry path using this policy is executed. For more information about the service-access element, see Section 5.3.2.4, "Registry Elements Descriptions"

5.3.5.2 Policy Files

You need to manually create a separate Action Framework policy file for each distinct client security policy being used to secure target web services. An Action Framework policy file contains a reference to a web services client policy defined in OWSM, which

provides a standard WS-Policy PolicyReference element used to invoke a target web service.

To manually create or copy these files, use the sample files, located in Appendix A, "Sample Files".

After you create or copy the files, save them in the same folder as the main ActionFrameworkConfig.xml.

See the following example of an Action Framework Policy file's contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<oracle-webservice-clients>
   <webservice-client>
     <port-info>
        <policy-references>
          <policy-reference uri="oracle/log_policy" category="management"/>
          <policy-reference uri="oracle/wss11_saml_token_with_message_
               protection_client_policy" category="security"/>
        </policy-references>
     </port-info>
   </webservice-client>
</oracle-webservice-clients>
```

The Action Framework policy file needs to point to the appropriate client policy in order to invoke a web service secured by a service policy.

For example, if a target web service is secured by the policy "oracle/wss11_saml_token_with_message_protection_service_policy," then the web service client (that is, in this case the Action Framework) needs to use the counterpart client policy to invoke this web service. The appropriate client policy in this example is "oracle/wss11_saml_token_with_message_protection_client_policy."

You should enter the name of the appropriate client policy into the Action Framework policy file against the "policy-reference uri" element with a category of "security."

The service and client web services policies available through OWSM can be viewed through Fusion Middleware Control by selecting **Web Services** and then **Policies** from the WebLogic domain. For more information about the predefined policies available through OWSM, see "Predefined Policies" in *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

5.3.6 Proxy

The proxy element specifies proxy settings. The proxy settings are used for accessing items such as web services and URLs that would ordinarily be inaccessible from the network containing the WebLogic server where the Action Framework web services are deployed.

The following example is a proxy entry. Table 5–8, " Proxy Elements and Descriptions" describes each element.

```
<proxy>
    <host>proxyserver.mycompany.com</host>
    <port>80</port>
    <userid>jsmith</userid>
    <password>johsmi</password>
    <nonProxyHosts>localhost|*.mycompany.com|10.1.10.78
    </nonProxyHosts>
 </proxy>
```

5.3.6.1 Proxy Elements Descriptions

Table 5–8, " Proxy Elements and Descriptions" contains each proxy element and its description.

Table 5–8 Proxy Elements and Descriptions

Element	Description
host	This element specifies the host name of the server where the proxy server is located.
port	This element specifies the proxy server's port number.
userid password	These element specifies the userid and password for the proxy server. Use these elements for a proxy that requires authentication. Leave these elements blank for a non-authenticating proxy.
nonProxyHosts	This element allows for a pipe character (|) separated list of the server, domain names, and patterns to exclude from the proxy. Use this element so the system does not attempt to use the proxy to access internal resources.

5.3.7 ebusinesssuiteconfig

The ebusinesssuiteconfig element specifies that an Oracle E-Business Suite system is available. If this element is present in the configuration file, Oracle BI Presentation Services displays the menu option that allows users to create a Navigate to E-Business Suite action. Users must have the proper privileges to access E-Business Suite from Oracle BI Presentation Services. For information about this integration configuration, see Chapter 10, "Integrating with Oracle E-Business Suite Security".

The following example is an ebusinesssuiteconfig entry.

```
<ebusinesssuiteconfig>
    <visible>true</visible>
</ebusinesssuiteconfig>
```

5.3.8 siebelcrmconfig

The siebelcrmconfig element specifies that an Oracle Siebel CRM system is available. If this element is present in the configuration file, Oracle BI Presentation Services displays the menu option that allows users to create a Navigate to Siebel CRM action. Users must have the proper privileges to access Siebel CRM from Oracle BI Presentation Services.

Note that the Oracle BI Server must be integrated with the Siebel CRM server before users can invoke a Navigate to Siebel CRM action type from Oracle BI Presentation Services. For more information about this integration, see Chapter 11, "Embedding Oracle BI EE In Oracle's Siebel CRM".

The following example is a siebelcrmconfig entry.

```
<siebelcrmconfig>
    <visible>true</visible>
</siebelcrmconfig>
```

5.4 Overview of Action Security

Action Framework and actions security is determined by credentials, privileges, and permissions.

5.4.1 Oracle BI EE Credentials

Credentials are security-related attributes used to authenticate or authorize users and systems requesting access to Oracle BI EE resources such as actions. If you have configured one or more registries in the Action Framework to use credentials in the credential store for either browsing for actions targets or invoking actions (namely, if you have specified account elements in the ActionFrameworkConfig.xml) you need to enter a credential into a the credential store to match each account element in the ActionFramworkConfig.xml file. All credentials referenced by account elements must be in a credential map called oracle.bi.actions.

For example, if you have the following account defined in the configuration file, you need to enter a credential for JNDIUser into a map called oracle.bi.actions.

```
<account>
    <name>WLSJNDI</name>
    <description>Account used to access WLS JNDI.</description>
    <adminonly>false</adminonly>
    <credentialkey>JNDIUser</credentialkey>
</account>
```

For more information about how enter a credential into the credential store, see Section 5.5, "Adding and Maintaining Credentials for Use With the Action Framework".

5.4.2 Oracle BI EE Privileges

Actions privileges control the rights that users have to access the actions features and functionality in Oracle BI Presentation Services. The Oracle BI EE Administrator grants or denies privileges to specific application roles, individual users, and Catalog groups. The actions privileges are:

- Create Navigate Actions
- Create Invoke Actions
- Save Actions containing embedded HTML

For more information about the action privileges, see "Managing Presentation Services Privileges" in *Oracle Fusion Middleware Security Guide for Oracle Business Intelligence Enterprise Edition*.

5.4.3 Oracle BI Presentation Catalog Permissions

Action permissions are authorizations that the action's owner grants to users or roles. Permissions determine what tasks a user can perform on an action (for example, execute the action). The Oracle BI EE Administrator must give an action's owner the necessary privileges before the owner can assign permissions to action catalog objects. The action permissions are:

- Delete
- Execute
- Read
- Write

For more information about assigning permissions to action object stored in the catalog, see "Permission Definitions" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

5.5 Adding and Maintaining Credentials for Use With the Action Framework

This section contains the following procedures:

- Section 5.5.1, "Adding a Credential Map and Credential Key to the Credential Store"
- Section 5.5.3, "Configuring Oracle Web Services Manager"
- Section 5.5.2, "Creating a Default Keystore"

Use these procedures to enter a credential into the credential store to match each account element in the ActionFrameworkConfig.xml file. For more information about setting up credentials with Action Framework, see Section 5.4.1, "Oracle BI EE Credentials".

5.5.1 Adding a Credential Map and Credential Key to the Credential Store

Use the following procedure to add a credential map and credential key to the credential store.

1. Log into Fusion Middleware Control. The default location is

   ```
   http://<your host name>:7001/em
   ```

2. Expand the tree menu to view your domain, and then right-click on the domain name. An options list displays.
3. Highlight the **Security** option and from the list, select **Credentials**. The "Credentials" dialog displays.
4. Click **Create Map** and add a new map called oracle.bi.actions. Note that the only map Actions can reference is oracle.bi.actions, so you need to create a map with this name and add credentials to it for use with Actions.
5. Click **Create Key** and add a key against the oracle.bi.actions credential map you created.
6. Save the key and map.

5.5.1.1 Example of Creating the Credential Map and Credential Key

Note the following account element.

```
<account>
    <name>SecureTest</name>
    <description>Test Secure Account</description>
    <adminonly>false</adminonly>
    <credentialkey>SecureTest</credentialkey>
</account>
```

If you have this account element in your ActionFrameworkConfig.xml and the following conditions are true:

- this account element is referenced by a registry to invoke a web service secured using a username and password policy
- the propagateIdentity element is set to false,

then, for the example account element above, you must enter a username and password that is valid for invoking the target web service. This should be added to a credential map called oracle.bi.actions against a credentialkey of *SecureTest*.

5.5.2 Creating a Default Keystore

Use the following procedure to create a default self-signed keystore. Note that creating a default keystore is suitable for demonstration or development use, but is not suitable for production use. In production, a keystore that was created by importing a valid, correctly signed certificate should be used as described in the keytool documentation. For security policies that involve signing or encryption, you must also add a certificate to the bifoundation_domain keystore.

For more information about keytool and Solaris/Linux, go to

`http://java.sun.com/javase/6/docs/technotes/tools/solaris/keytool.htm`

For information about keytool and Windows, go to

`http://java.sun.com/javase/6/docs/technotes/tools/windows/keytool.html`

1. Confirm that your installer installed the JDK bin directory and that it displays in your path. For example, `<MIDDLEWARE_HOME>/jdk160_11/bin/`.
2. Open a command prompt at `<MIDDLEWARE_HOME>/user_projects/domains/bifoundation_domain/config/fmwconfig`.
3. Run the following command to create a default keystore.

   ```
   keytool -genkeypair -keyalg RSA -alias orakey -keypass orakey_passphrase
   -keystore default-keystore.jks -storepass store_passphrase -validity 3600
   ```

   The command creates a keystore with the name default-keystore.jks (if it does not already exist) and adds a new private key entry with alias "orakey" and password as "orakey." You can change the alias, password, and storepass in the command, but they must match the OWSM credentials added to the credential store in the Section 5.5.3, "Configuring Oracle Web Services Manager" procedure.
4. When prompted, answer the questions. Enter responses relevant to your organization. See the example. Note that in the example, the user "weblogic" refers to the System Administrator user created during the install. If you chose a user other than "weblogic," enter that username instead.

   ```
   What is your first and last name?
   [Unknown]:  weblogic
   What is the name of your organizational unit?
   [Unknown]:  J2EE Test Encryption Purposes Only
   What is the name of your organization?
   [Unknown]:  Oracle
   What is the name of your City or Locality?
   [Unknown]:  US
   What is the name of your State or Province?
   [Unknown]:  US
   What is the two-letter country code for this unit?
   [Unknown]:  US
   Is CN=weblogic, OU=J2EE Test Encryption Purposes Only, O=Oracle, L=US, ST=US,
   C=US correct?
   [no]:  yes
   ```

5. Confirm that the keystore was generated correctly by using the following command to list its contents.

   ```
   keytool -list -v -keystore default-keystore.jks -storepass welcome1
   ```

   Running this command produces a response similar to the following example.

   ```
   Keystore type: JKS
   ```

```
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: orakey
Creation date: 16-Sep-2009
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=weblogic, OU=J2EE Test Encryption purposes Only, O=Oracle, L=US,
ST=US
, C=US
Issuer: CN=weblogic, OU=J2EE Test Encryption purposes Only, O=Oracle, L=US,
ST=U
S, C=US
Serial number: 4ab0ee4e
Valid from: Wed Sep 16 14:55:26 BST 2009 until: Fri Jul 26 14:55:26 BST 2019
Certificate fingerprints:
     MD5:  84:0E:F4:F4:F3:30:0B:FF:4C:D4:E5:E6:BE:AE:64:DF
     SHA1: E4:73:80:4D:96:A6:9F:DE:06:0E:82:3B:D3:18:86:57:FE:CD:C6:37
     Signature algorithm name: SHA1withRSA
     Version: 3


*******************************************
*******************************************
```

6. Check if the default-keystore.jks file exists at the path. If it does not exist, copy it there.

```
<MIDDLEWARE_HOME>/user_projects/domains/bifoundation_domain/config/fmwconfig
```

5.5.3 Configuring Oracle Web Services Manager

Use the following procedure to configure Oracle Web Services Manager (WSM) to work with actions. Perform this procedure so that you can create an action that invokes a web service where the target web service has a WS security policy applied to it.

For security policies that involve signing or encryption, you must also add a certificate to a keystore. Note that the enc-csf-key, keystore-csf-key, and sign-csf-key credentials should match the corresponding passphrases given in the keypass and storepass arguments when running keytool.

1. In Fusion Middleware Control, add a credential map named oracle.wsm.security. For instructions on how to add a credential map, see Section 5.5.1, "Adding a Credential Map and Credential Key to the Credential Store".
2. Add the following keys to the oracle.wsm.security map.

Table 5–9 oracle.wsm.security Map Keys

Keyname	Type	User Name	Password
basic.credentials	Password	weblogic*	welcome1
enc-csf-key	Password	orakey	welcome1
keystore-csf-key	Password	owsm	welcome1
sign-csf-key	Password	orakey	welcome1

* This username refers to the System Administrator user created during the installation. If you chose a username other than "weblogic," enter that user name instead. Similarly, you should use the password for that account during install.

3. Save the map.

5.6 Target Functionality for Actions

This section contains further information about setting up target functionality in external systems. The action links added to business intelligence content invokes these target functionalities. The action types explained in this section are:

- Section 5.6.1, "Navigate to EPM Content"
- Section 5.6.2, "Navigate to E-Business Suite"
- Section 5.6.3, "Navigate to Siebel CRM"
- Section 5.6.4, "Invoke a Web Service"
- Section 5.6.6, "Invoke a Java Method (EJB)"
- Section 5.6.7, "Invoke a Browser Script"
- Section 5.6.8, "Invoke a Server Script"
- Section 5.6.9, "Invoke Agent"
- Section 5.6.10, "Java Job"

5.6.1 Navigate to EPM Content

This action type allows users to browse for target content in the EPM repository and then create an action to navigate to the selected content. Only navigation to Oracle Hyperion Financial Reporting content is currently supported.

For more information about creating this action type in Presentation Services, see "Working with Actions" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

5.6.1.1 Prerequisites for This Action Type

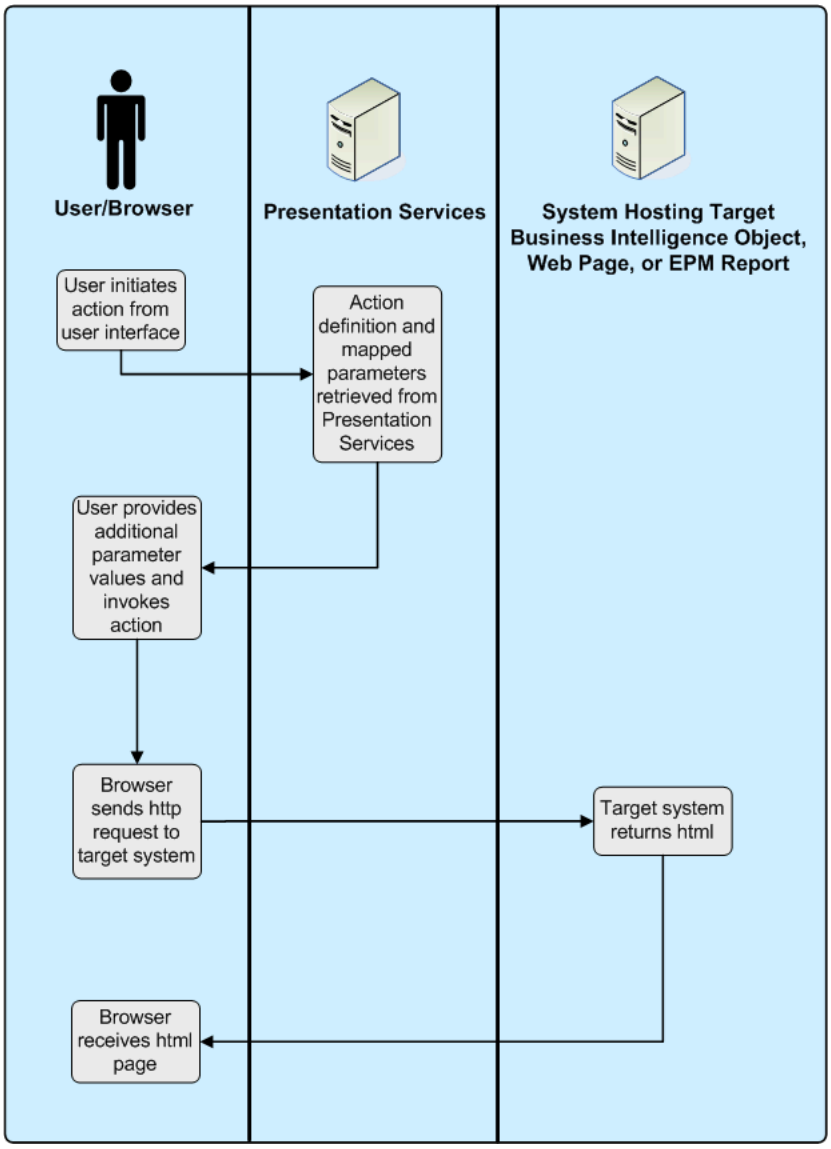
In order for the Action Framework to provide this action type, you must have performed the following tasks:

- Modified the configuration file's registry element. For information about this required configuration, see Section 5.3.2.1, "Navigate to EPM Content Action Type Registry Example".
- Added a credential to the credential store to enable the browse for EPM content functionality. For more information, see Section 5.4.1, "Oracle BI EE Credentials" and Section 5.5, "Adding and Maintaining Credentials for Use With the Action Framework".

5.6.1.2 What Happens When This Action Type is Invoked?

Figure 5–1, "Navigate to EPM Content Process Flow" illustrates the Navigate to EPM Content action type's process flow.

Figure 5–1 Navigate to EPM Content Process Flow



5.6.2 Navigate to E-Business Suite

This action type allows users to navigate from Oracle BI EE to Oracle E-Business Suite. The Oracle BI EE session holds the context of the user's Oracle E-Business Suite session, including the current Oracle E-Business Suite responsibility in the Oracle BI EE session variables.

A Navigate to Oracle E-Business Suite action takes two parameters:

- Connection Pool - This parameter contains the name of the BI connection pool that connects to the target Oracle E-Business Suite environment as defined in the repository.
- Function - This parameter contains the name of the target Oracle E-Business Suite function to which to navigate. The Oracle E-Business Suite administrator needs to provide the target function ID.

Note that before users can invoke a Navigate to E-Business Suite action, they must have privileges to execute direct database requests against the Oracle E-Business Suite connection pool.

Note that for a user to successfully invoke a Navigate to E-Business Suite action, the target Oracle E-Business function must be accessible from the user's current Oracle E-Business Suite.

For information about this action type's required security integration configuration, see Chapter 10, "Integrating with Oracle E-Business Suite Security".

5.6.2.1 Overview of Passing Context to Oracle E-Business Suite Java Forms

You can use Oracle Forms Builder to customize a target form and add one or more new, custom parameters to be populated from Oracle BI EE. For information about completing this task, see the Oracle Forms Developer and the Oracle E-Business Suite documentation.

You can use Oracle E-Business Suite's Forms Personalization to map the value or values from these new custom parameters into the form fields used by Oracle E-Business Suite for searching for transactions. For information about completing this task, see the Oracle E-Business Suite documentation.

When you create an action to navigate to the target Oracle E-Business Suite function, you must manually add new parameters with the same names as the custom parameters that you added to the Oracle E-Business Suite form. Note that the new parameters that you add are in addition to the two default parameters (Connection Pool and Function).

After you add the parameters and you invoke the action, Oracle BI EE passes the parameters to the target Oracle E-Business Suite form. Finally, the Form Personalizations take the passed parameter values and uses them to retrieve the required target transaction.

5.6.3 Navigate to Siebel CRM

This action type allows users to navigate from Oracle BI EE to a view (such as an opportunity) in a Siebel CRM application. You use this type of action to allow users to navigate from a dashboard that is embedded in a Siebel CRM application to a record in a view in the CRM application.

A Navigate to Siebel CRM action takes three parameters:

- View - This parameter specifies the name of the view that contains the record to which to navigate. For example, Opportunity List View.
- Applet - This parameter specifies the name of the parent applet in the view that contains the record to which to navigate. For example, Opportunity List Applet.
- Pass Value - The row number of the record to which to navigate. For example, 3SIA-2O5VU.

For information about determining the name of the view, applet, and record row number, see the Oracle's Siebel CRM application documentation.

Before you can use Navigate to Siebel CRM actions, you must embed Oracle Business Intelligence in the Oracle's Siebel CRM application. For more information, see Chapter 11, "Embedding Oracle BI EE In Oracle's Siebel CRM"

5.6.4 Invoke a Web Service

This action type allows users to browse for target web services operations and then create an action to invoke the selected content. For information about how Action Framework supports calls to web services, see Section 5.6.5, "Supported Functionality for Calling Web Services".

For more information about creating this action type in Presentation Services, see "Working with Actions" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

5.6.4.1 Prerequisites for This Action Type

In order for the Action Framework to provide this action type and before a user can create an Invoke a Web Services action, you must have performed the following tasks:

- Confirmed that a target web service with a URL to the WSDL for that web service exists.
- If you want to allow users to browse for web services, you must have added a WSIL document that points to one or more WSDL documents. The WSIL document must be accessible via a URL. For more information, see Section 5.6.4.2, "Example of a WSIL Document".
- If you want to browse to web services operations or apply security policies, you must have modified the configuration file's registry element so that it references the target WSIL document. For information about this required configuration, see Section 5.3.2.3, "Invoke a Web Service Action Type Registry Example".
- Confirmed whether you need to add credentials to the credential store to invoke web services. If the target web service is secured, appropriate configuration needs to be added to pass credentials to the web service operation when it is invoked. A suitable keystore might also be required. For more information, see Section 5.4.1, "Oracle BI EE Credentials", Section 5.5, "Adding and Maintaining Credentials for Use With the Action Framework", and Section 5.5.2, "Creating a Default Keystore".

5.6.4.2 Example of a WSIL Document

WSIL defines an XML format for referencing web service descriptions. These references are contained in a WSIL document and refer to web service descriptions (for example, WSDL files) and to other aggregations of web services (for example, another WSIL document or a UDDI registry). Note the following WSIL example:

```
<?xml version="1.0" encoding="UTF-8"?>
<inspection xmlns="http://schemas.xmlsoap.org/ws/2001/10/inspection/">
    <service>
       <abstract>Web Service: Order Services</abstract>
       <name>Order Service</name>
       <description referencedNamespace="http://schemas.xmlsoap.org/
         wsdl/"location="http://localhost:9704/ActionSamples/
         OrderProcessPort?wsdl"/>
    </service>
</inspection>
```

5.6.4.3 Troubleshooting Actions to Invoke a Web Service

Use the following list to troubleshoot actions to invoke a web service.

1. Confirm that you can invoke the target web service from a test client such as the HTTP Analyzer in JDeveloper.

2. If the target web service has a security policy, check the ActionFrameworkConfig.xml file to make sure that the Action Execution Service (AES) is using the appropriate client policy.
3. Go to Fusion Middleware Control to check the logs and diagnostics for Action Services. To access the log files, go to the Fusion Middleware Control, access the **Business Intelligence** navigation tree, and select **coreapplications** to display information about the Oracle BI instance. Select the **Diagnostics** tab and select the **Log Messages** sub tab. In the View/Search Log Files area, click **Action Services** log.

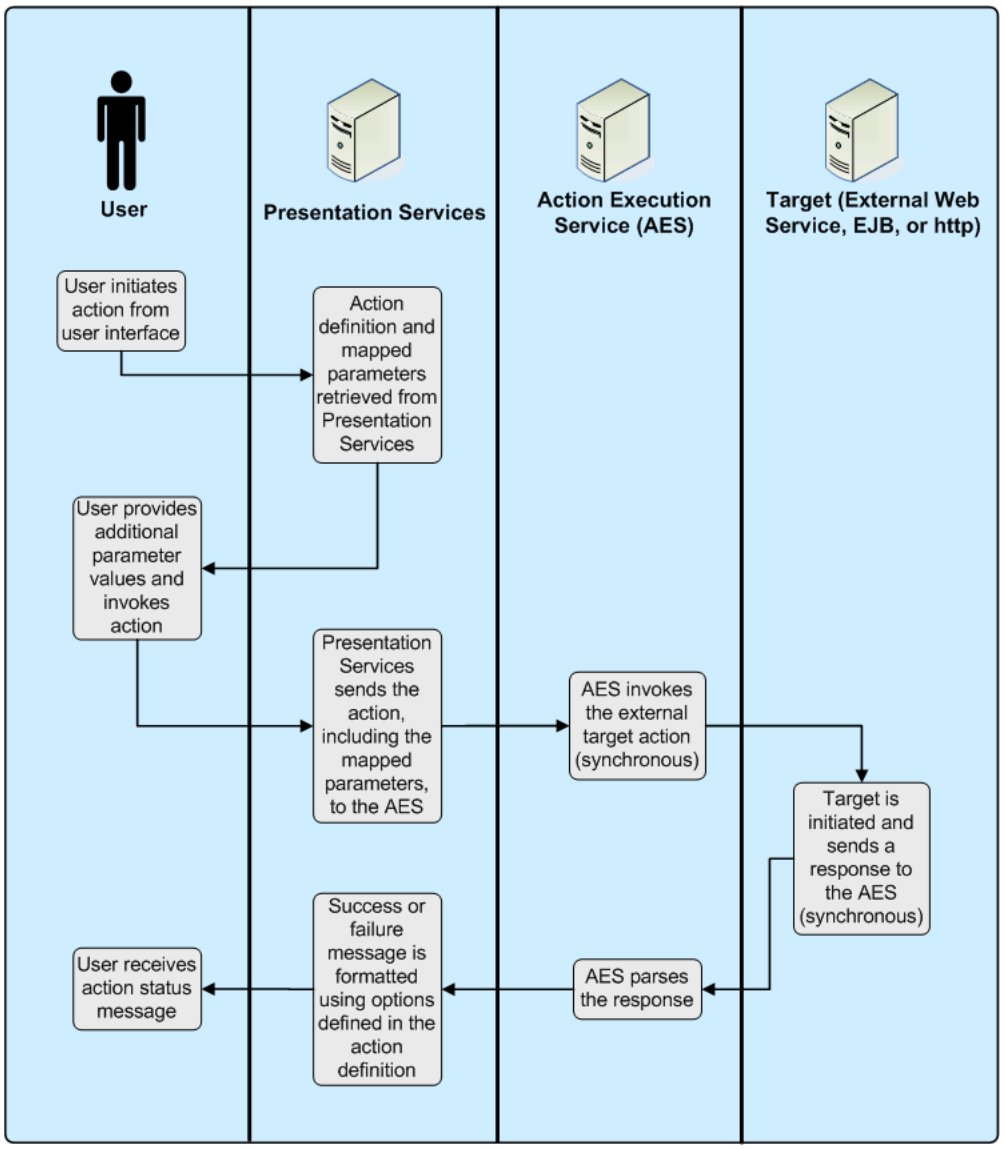
   You can enable verbose logging for WebLogic by adding the following entries to the JAVA_OPTIONS variable in the startWebLogic.cmd/*.sh file. After you have modified the JAVA_OPTIONS variable, you must restart WebLogic.

   ```
   -Doracle.multitenant.enabled=true -Dweblogic.wsee.verbose=*
   -Dweblogic.log.RedirectStdoutToServerLogEnabled=true
   -Dweblogic.webservice.client.verbose=true"
   ```
4. Check the ActionFrameworkConfig.xml file for mistakes or omissions.
5. Thoroughly check any associated policy files and make sure they are in the correct location.
6. If the target web service applies a security policy that includes either message protection or encryption, thoroughly check the additional OWSM credentials in the Credential Store.

5.6.4.4 What Happens When This Action Type is Invoked?

Figure 5–2, "Invoking a Web Service Process Flow" illustrates the web service action type's process flow.

Figure 5–2 Invoking a Web Service Process Flow



5.6.5 Supported Functionality for Calling Web Services

This section describes the supported technologies and limitations used by the Action Framework when it invokes web services.

5.6.5.1 Transport

Note the following items:

- The Action Framework supports HTTP for both WDSL browsing and calling web services through Actions.
- The Action Framework supports HTTPS for WSDL browsing of web services.
- The Action Framework supports HTTPS for sending messages over a secure channel between AES and the target web service. However, Action Framework does not currently support the use of digital certificates for authenticating between AEService and the target web service.

5.6.5.2 Messaging

Note the following items:

- A web service called from the Action Framework needs to be SOAP-based.
- The Action Framework supports synchronous, request/response messages.
- The Action Framework does not support WS-Addressing.

5.6.5.3 SOAP

The Action Framework supports the following data types in outgoing and incoming SOAP 1.1 and 1.2 messages.

Note that for an out-going SOAP message, you cannot input multiple values for a parameter even when the underlying schema specifies that the parameter should accept multiple values (for example, an array or collection).

- xsd:any
- xsd:base64Binary
- xsd:string
- xsd:date
- xsd:time
- xsd:dateTime
- xsd:double
- xsd:decimal
- xsd:int
- xsd:short
- xsd:long
- xsd:byte
- xsd:Boolean
- xsd:float

5.6.5.4 Response Document

The SOAP response document is made available to the user interface and can be formatted using XPATH and html. The document can retrieve multiple occurrences of the same parameter, but you must map each occurrence to a specific XPATH parameter.

5.6.5.5 Service Description

Note the following items:

- The Action Framework supports WSDL.
- The Action Framework supports synchronous, request/response messages.

5.6.5.6 Discovery Services

Note the following items:

- The Action Framework supports WSIL.
- The Action Framework does not currently support UDDI.

5.6.5.7 Security

The Action Framework supports WS-Security based on the policies available to the OWSM.

5.6.5.8 Reliable Messaging and Transactions

The Action Framework does not currently support any mechanism for guaranteeing message delivery or maintaining the integrity of transactions.

5.6.6 Invoke a Java Method (EJB)

This action type allows users to browse for target Java methods deployed in Enterprise Java Beans (EJBs) and then create an action to invoke the selected method.

For more information about creating this action type in Presentation Services, see "Working with Actions" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

5.6.6.1 Prerequisites for This Action Type

In order for the Action Framework to provide this action type and before a user can create an Invoke a Java Method (EJB) action, you must have performed the following tasks:

- Added a target EJB with a URL to the JNDI location for that EJB. Note that the Action Framework expects EJBs to be deployed to the default location of `/ejb`.
- Modified the configuration file's registry element so that it references the target EJB. For information about this required configuration, see Section 5.3.2.2, "Invoke a Java Method Action Type Registry Example".
- Added a credential to the credential store to invoke the EJB method. The appropriate configuration needs to be added to pass credentials to the EJB method when it is invoked. For more information, see Section 5.4.1, "Oracle BI EE Credentials" and Section 5.5, "Adding and Maintaining Credentials for Use With the Action Framework".

5.6.6.2 Parameters for the EJB

The Action Framework finds the Java methods exposed through the remote interface of the EJB. The parameters of exposed methods become the parameters for an action.

When deploying an EJB method, you may import, but not modify, the oracle.bi.action.annotation.OBIActionParameter class found in Actionframework-common.jar. This class, which was deployed with the installation of Oracle Business Intelligence, allows you to annotate parameters for use by the Action Framework when creating actions.

The following example shows code to define a remote interface exposing a method called ArchiveReport. Note that the import statement for oracle.bi.action.annotation.OBIActionParameter and the annotation @ OBIActionParameter. This annotation allows you to specify a parameter name and a prompt value to display when creating an action.

```
package project1;
import java.io.FileNotFoundException;
import java.io.IOException;
import javax.activation.DataHandler;
import javax.ejb.Remote;
import oracle.bi.action.annotation.OBIActionParameter;
```

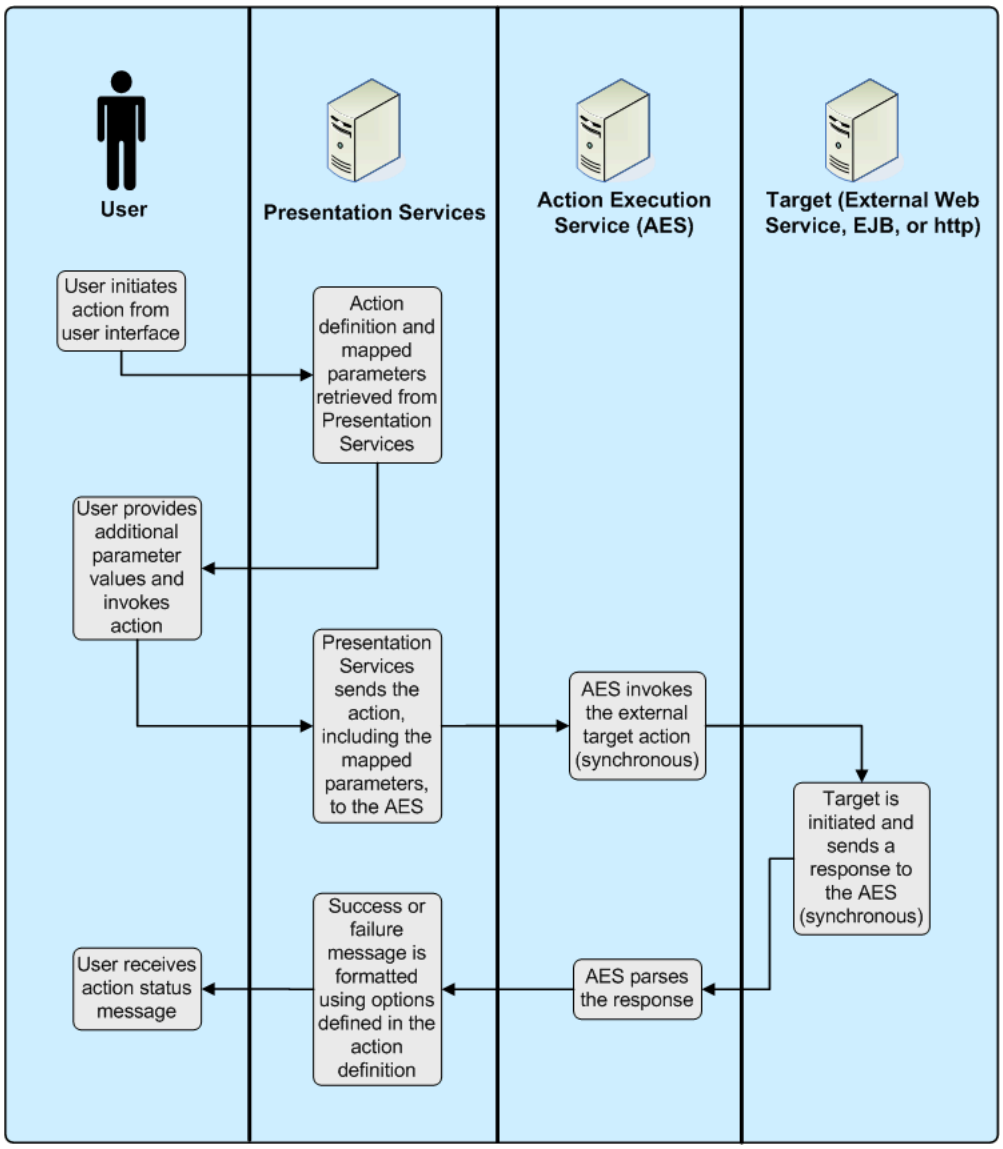
```
@Remote
public interface ArchiveReport {
String ArchiveReport( @OBIActionParameter (name = "Filename", prompt = "Enter
filename location:")
String filename, @OBIActionParameter (name = "Analysis", prompt = "Report to
Archive:") DataHandler document) throws
FileNotFoundException, IOException;
}
```

You can send documents from the catalog to a Java method in an EJB. The binary data for the document is sent using a specific Java data type. Therefore, if you want to create a Java method that accepts an analysis as a parameter, your Java method should include a specific Java data type (javax.activation.DataHandler) as a parameter. The Action Framework recognizes this data type as capable of receiving a document in one of the supported export formats (for example, PDF or HTML).

5.6.6.3 What Happens When This Action Type is Invoked?

Figure 5–3, "Invoke Java Method (EJB) Process Flow" illustrates the Invoke a Java Method (EJB) action type's process flow.

Figure 5–3 Invoke Java Method (EJB) Process Flow



5.6.7 Invoke a Browser Script

This action type allows users to browse target JavaScript functions and then create an action to invoke the selected function.

For more information about creating this action type in Presentation Services, see "Working with Actions" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

5.6.7.1 JavaScript Functions

Use the following information to help you create JavaScript functions.

- The target JavaScript functions that Actions use to Invoke a Browser Script call are hosted in the UserScripts.js file found in the following location:

  MW_HOME/user_projects/domains/bifoundation_domain/servers/bi_server1/tmp/_WL_user/analytics_11.1.1.2.0/<installation dependent folder>/war/res/b_mozilla/actions/UserScripts.js

- A sample UserScript.js file is located in Appendix A, "Sample Files".
- You can use the UserScripts.js file and add any user-defined JavaScript functions to invoke from a client-side script action.
- Custom functions should use the namespace USERSCRIPT so that no function name conflicts occur with the product code, and so that the Action Framework can identify functions when creating an action.
- After you add user-defined JavaScript functions to the UserScripts.js file, you must restart the Presentation Server, restart the Weblogic Server running analytics, and clear the browser cache.

5.6.7.2 UserScript.js

Custom functions in the UserScripts.js file consist of a function to contain the actual code to be called when the action is invoked, and, optionally, an associated "publish" object used by the Action Framework to browse for the function when creating the action and for mapping values to the function parameters.

5.6.7.2.1 JavaScript Example 1 The example shows a JavaScript function of USERSCRIPT.example_displayParameters that can be called by an action. Target functions receive a single parameter and a named array of values, indexed by the parameter name.

```
/** This is an example function to display all parameters that are received
 * @params {Array} aParams an array of values indexed by the parameter name
 */
USERSCRIPT.example_displayParameters = function(aParams)
{
    var sArgs = "";
    for( args in aParams )
    {
      var argName  = args;
      var argValue = aParams[argName];

      sArgs += "Parameter name: " + argName + "  Value: " + argValue;
      sArgs += "\n";
    }

    alert( sArgs.length == 0 ? "No Parameters" : sArgs );
};
```

5.6.7.2.2 JavaScript Example 2 The USERSCRIPT.parameter object is defined at the beginning of the UserScripts.js file. This object is used by the Action Framework to define parameters in custom JavaScript functions for use when creating an action to Invoke a Browser Script. Each parameter object includes a name, a prompt value holding the text to be displayed against the parameter when creating an action, and a default value.

```
/** This is the parameter object you can create to supply default parameters on
creation of Script action.
   * See the 'displayParameters' example below for usage.
   * @param {String} sName is the unique name of parameter.
   * @param {String} sPrompt is the display text used to prompt for the parameter
     value.
   * @param {String} sValue (Optional) is the default value for the parameter.
   */
USERSCRIPT.parameter = function(sName, sPrompt, sValue)
{
```

```
   this.name   = sName;
   this.prompt = sPrompt;
   this.value  = sValue;
};
```

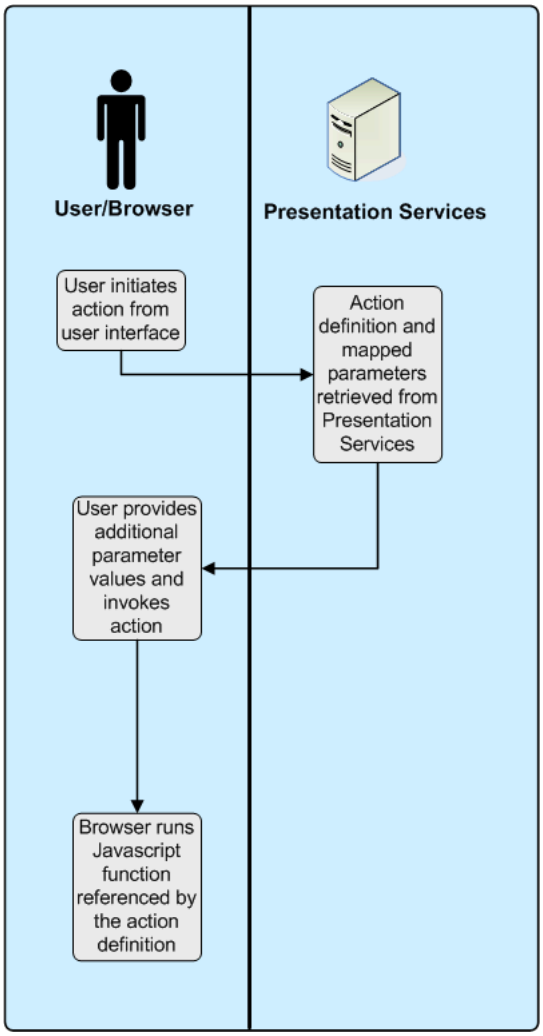
The example shows the definition of a "publish" object that defines parameters for the Action Framework. All functions within the USERSCRIPT namespace that include a "publish" object can be browsed for selection when creating an action. The publish object defines the array of parameters to be passed by an action to the target JavaScript function. Note that functions that do not have an associated "publish" object may still be invoked by an action, but the Action Framework is not able to browse to these private functions, and therefore parameters need to be added to the action definition manually.

```
USERSCRIPT.example_displayParameters.publish =
{
    // The existence of this 'publish' object causes the 'USERSCRIPT.example_
displayParameters' function to be
    // shown when browing the available user script functions (during creation of
a Script action).

    // If you wish the Script function to have parameters automatically created on
selection of the function,
    // create a 'parameters' object as shown below.
    // You can have any number of parameters, with each parameter requiring a
       unique name, prompt and an
    // optional value.
    parameters :
   [
      new USERSCRIPT.parameter( 'p1', 'Enter value for Param 1', 'p1 default
      value' ),
      new USERSCRIPT.parameter( 'p2', 'Enter value for Param 2', 'p2 default
      value' ),
      new USERSCRIPT.parameter( 'p3', 'Enter value for Param 3'  )
   ]

    // If no generated parameters are required, either create an empty array
    //    parameters : []
    // or don't declare the 'parameters' object at all.
};
```

5.6.7.3 What Happens When This Action Type is Invoked?

Figure 5–4, "Invoke a Browser Script Process Flow" illustrates the Invoke a Browser Script action type's process flow.

Figure 5–4 Invoke a Browser Script Process Flow



5.6.8 Invoke a Server Script

This action type allows users to specify the filename of a custom script to execute (on Microsoft Windows) when the current agent completes. The custom script type can be either JavaScript or VBScript. By default, the scripts are stored in the following directory:

ORACLE_INSTANCE\bifoundation\OracleBISchedulerComponent\coreapplication_obisch*n*\scripts\common

For example:

D:\OBI11g\instances\instance1\bifoundation\OracleBISchedulerComponent\coreapplication_obisch1\scripts\common

However, the Administrator can change this default path. For more information about the configuration setting that controls this default path, see "Configuring and Managing Agents" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* and "Introducing Oracle BI Scheduler" in *Oracle Fusion Middleware Scheduling Jobs Guide for Oracle Business Intelligence Enterprise Edition*.

When setting up this action type, you can select options for whether results are passed to the script, as well as desired formats. You can also manually add additional

parameters. Depending upon the type of content to be passed (from either the conditional request or the delivery content), results may be passed in some of the following formats:

- PDF
- MHTML (MIME HTML used in email)
- Plain Text
- XML
- CSV
- Excel
- Powerpoint

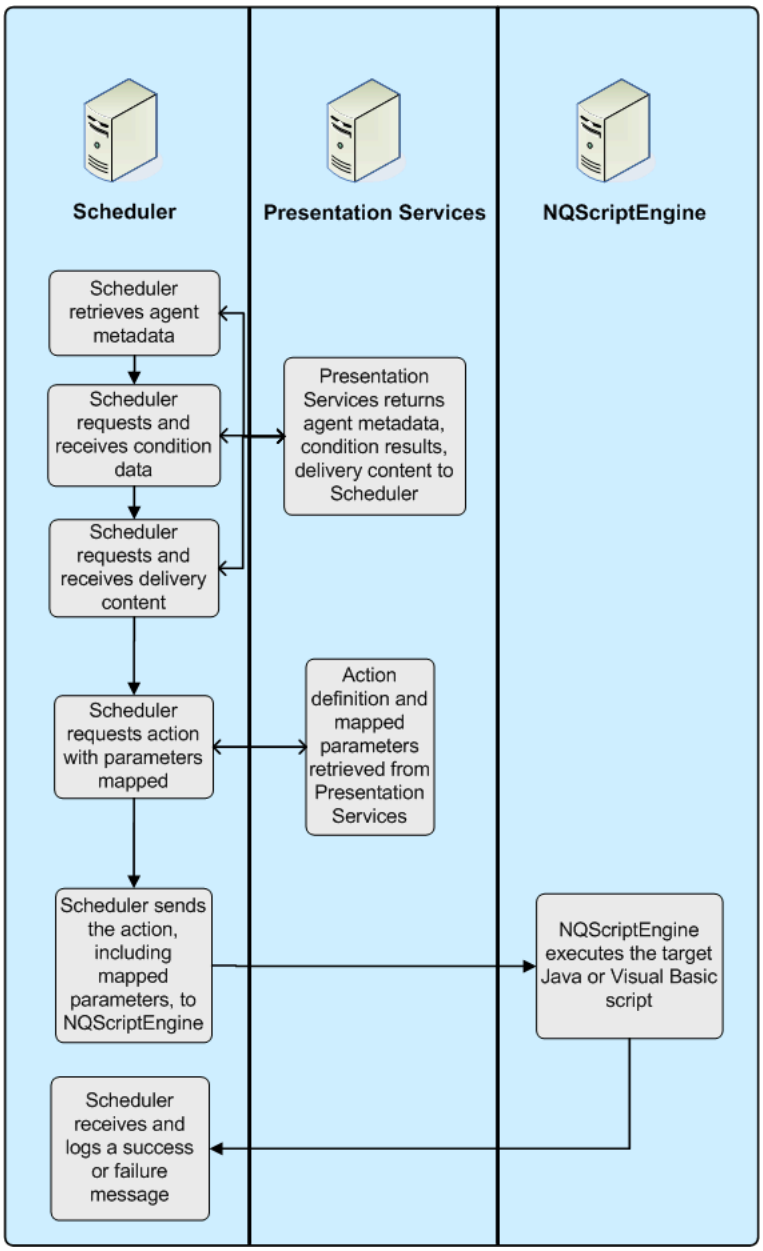
5.6.8.1 Prerequisites for This Action Type

In order for the user to create an Invoke Server Script action, you must confirm that the custom script file resides on the same server as the Oracle BI Delivers server (the Scheduler). See the above topic for more information about the directory where the scripts are stored by default.

5.6.8.2 What Happens When This Action Type is Invoked?

Figure 5–5, "Invoke a Server Script Process Flow" illustrates the Invoke a Server Script action type's process flow.

Figure 5–5 Invoke a Server Script Process Flow



5.6.9 Invoke Agent

Agents can execute actions depending on whether a data condition is met. For detailed conceptual information about agents and procedures for creating agents and adding actions to them, see "Delivering Content" and "Working with Actions" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

5.6.9.1 How Filters Work in Invoke Agent Actions

If you add an Invoke Agent action to an agent and the following conditions exist, then the child agent will be invoked once with filter values generated by the results of the parent agent.

- The parent agent to which you are adding the Invoke Agent action uses a condition to determine whether the parent agent delivers its content and runs its associated actions.
- The Invoke Agent action invokes a child agent that also uses a condition to determine whether the child agent delivers its content and runs its associated actions, *and* that condition is based on an analysis that filters its data by a column prompt.

For example, suppose you have the following:

- A parent agent that uses a condition based on an analysis that uses the Region, District, City, and Sum (Sales) columns
- A child agent that uses a condition based on an analysis that uses the Region, District, City, and Sum (Sales) columns and filters on Region and City.

And suppose when the parent agent runs, the following results are generated:

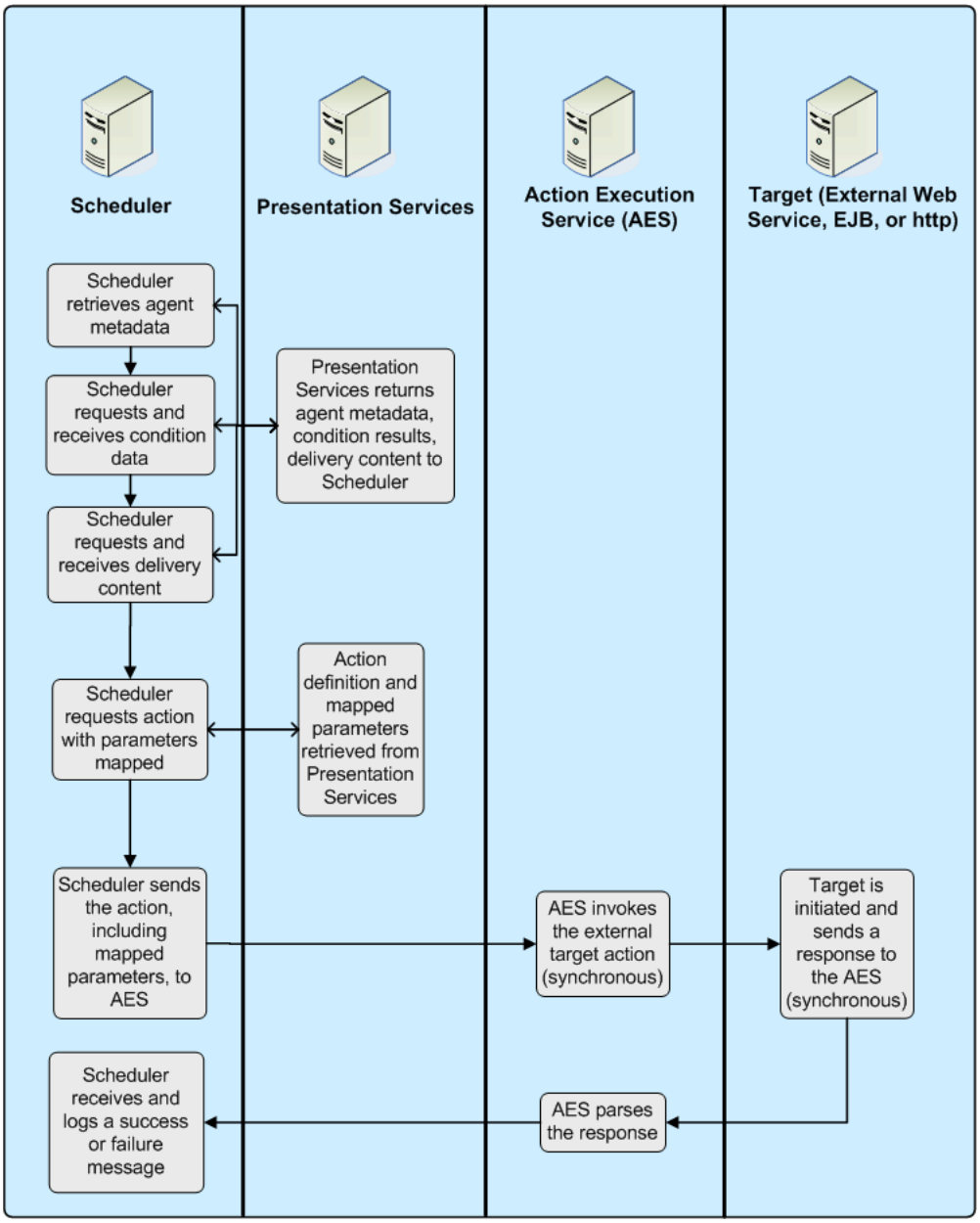
Region	District	City	Sum (Sales)
Central	A	C1	100
Western	B	C2	200

The child agent will run once using the filter values generated by the results of the parent agent, that is, Region = Central and City = C1, or Region = Western and City = C2.

5.6.9.2 What Happens When This Action Type is Invoked?

Figure 5–6 illustrates the Invoke Agent action type's process flow.

Figure 5–6 Invoke Agent Process Flow



5.6.10 Java Job

In Oracle BI EE 10*g*, you could specify a custom Java program to execute on Windows and UNIX when an iBot completed. When migrating to the current Oracle BI EE release, these actions are upgraded to read-only Java Job actions.

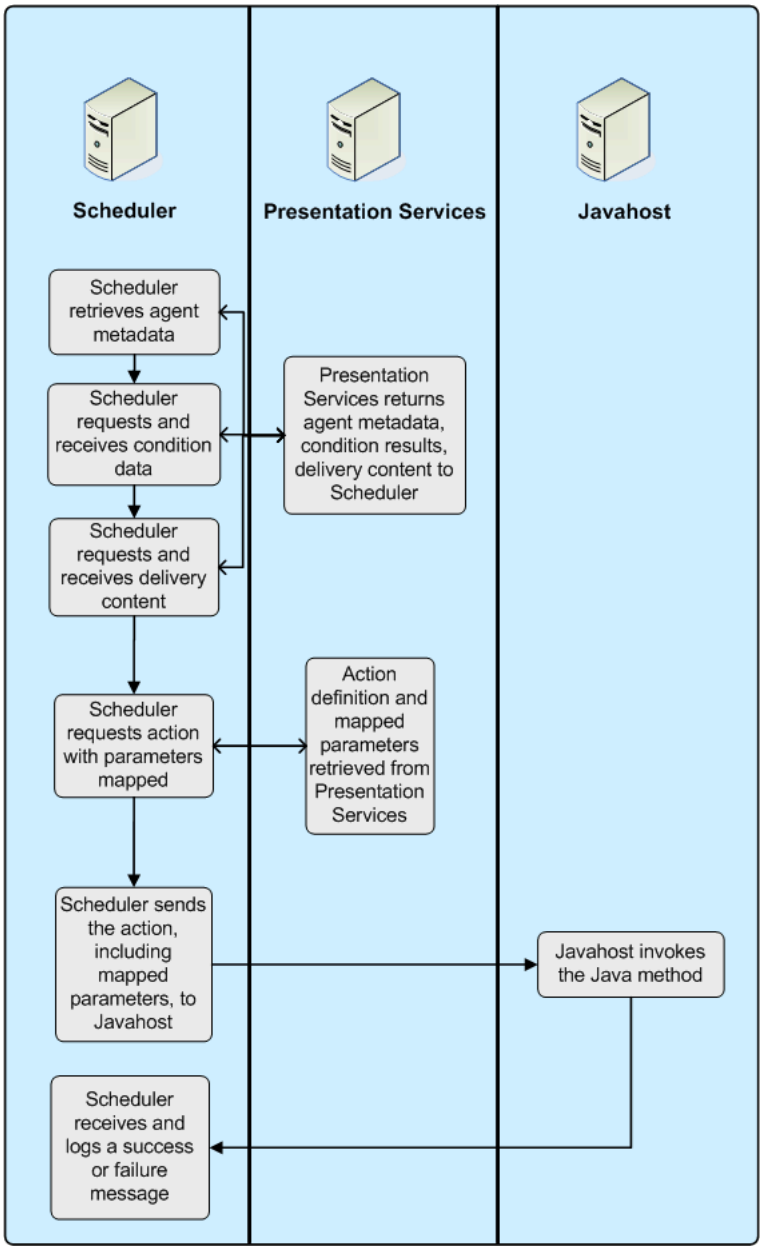
If you are creating new actions to run Java methods, you should use the Invoke a Java Method (EJB) action type. In order to use Java Job actions that have upgraded from 10g, you need to copy the jar file that includes the target java class into the default user Jar file path, as specified for the Oracle BI Javahost. Where the target java class imports Oracle BI EE classes, you should re-compile your target java class referencing the 11g Oracle BI EE classes in the classpath and make sure these classes are available to the target class when it is invoked. For more information, see "Introducing Oracle BI

Scheduler" in *Oracle Fusion Middleware Scheduling Jobs Guide for Oracle Business Intelligence Enterprise Edition*.

5.6.10.1 What Happens When This Action Type is Invoked?

Figure 5–7, "Java Job Process Flow" illustrates the Java Job action type's process flow.

Figure 5–7 Java Job Process Flow



6

Integrating Oracle BI Presentation Services into Corporate Environments Using HTTP and JavaScript

This chapter describes the HTTP and JavaScript methods used to integrate Oracle BI Presentation Services into a corporate environment. Describes Go URL, how to use Go URL to use SQL to pass filters, and Dashboard URL.

This chapter includes the following sections:

- Section 6.1, "Incorporating Oracle Business Intelligence Results into External Portals or Applications"
- Section 6.2, "Referencing Dashboard Content in External Portals or Applications"
- Section 6.3, "Using the Oracle BI Presentation Services Go URL to Issue SQL and Pass Filters"
- Section 6.4, "Example of an Oracle Business Intelligence Third-Party SQL Tool Integration"
- Section 6.5, "Retrieving Links to Dashboard Pages Using Scripts"

6.1 Incorporating Oracle Business Intelligence Results into External Portals or Applications

This section describes how to use the Oracle BI Presentation Services Go URL to incorporate results into external portals or applications. It contains the following topics:

- Section 6.1.1, "About the Oracle BI Presentation Services Prompted URL"
- Section 6.1.2, "About the Oracle BI Presentation Services Go URL"
- Section 6.2.2, "Structure of the Basic Oracle BI Presentation Services Dashboard URL"
- Section 6.1.4, "Optional Parameters for the Oracle BI Presentation Services Go URL"

6.1.1 About the Oracle BI Presentation Services Prompted URL

The Oracle BI Presentation Services Prompted URL command allows you to incorporate the path to a dashboard page and a simplified presentation of the dashboard prompts into external portals or applications. For more information on the

basic syntax, see "What Are Prompted Links?" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

6.1.2 About the Oracle BI Presentation Services Go URL

The Oracle BI Presentation Services Go URL command is for use in incorporating specific Oracle Business Intelligence results into external portals or applications. The Go URL is used when you add a result to your favorites, or add a link to a request to your dashboard or an external web site. It has a number of forms and optional arguments that can be used to control its behavior.

Note that you can use frame busting to prevent attackers from framing an application in an inline frame. For more information about using this method of security, see "Protecting Pages in Oracle BI EE from Attack" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

You can post the Go URL as a Form or issue it as a URL. If you are issuing parameters as part of a URL, they need to be escaped properly. You need to replace spaces with plus (+) signs, and so on. For example, to pass East Region as a value, type East+Region.

When called from within an Oracle BI Presentation Services screen, such as a dashboard or an HTML result view, the URL should begin with the following characters:

```
saw.dll?Go
```

When called from another screen on the same web server, the URL should begin with the following characters:

```
/analytics/saw.dll?Go
```

When referenced from a screen on a different server (or sent through email, and so on), the URL should begin with the fully qualified server name or IP address:

```
http://server_name_or_ip_address/analytics/saw.dll?Go
```

When invoking Go URL, you must make sure that a User-Agent string header is set.

To test these commands, you can enter the fully qualified version into the Address field of the browser.

6.1.3 Structure of the Basic Oracle BI Presentation Services Go URL

The basic Go URL command needs the full catalog path to the request to execute. It returns the default result view, which is defined in the request.

For example, the following go URL command returns the default result view as defined in the request, where SB2 is the name of the request to execute.

```
saw.dll?Go&Path=%2Fshared%2FTest%2FSB2
```

6.1.4 Optional Parameters for the Oracle BI Presentation Services Go URL

You can modify the behavior of the Go URL command by adding one or more of the following parameters. If an invalid URL is specified (for example, you type a parameter incorrectly), the browser displays a "The page cannot be found" error with the detailed text of "HTTP 400 - Bad Request."

Note: In parameter descriptions, SB2 is the name of the request to execute.

- **User ID and Password.** The user is prompted for user ID and password if this information has been omitted from the request.

  This is the format, where uuu is the user ID and ppp is the password:

  ```
  &NQUser=uuu&NQPassword=ppp
  ```

  Example:

  ```
  saw.dll?Go&Path=/Shared/Test/SB2&NQUser=user1&NQPassword=rock
  ```

  This logs on as user1 with a password of rock, and executes the request.

- **Link Options.** The results will include links.

  This is the format:

  ```
  &Options=x
  ```

  The x can be one or more of the following letters:

  - m = Modify Request
  - f = Printer Friendly
  - d = Download to Excel
  - r = Refresh Results

  Example:

  ```
  saw.dll?Go&Path=/Shared/Test/SB2&Options=md
  This displays results with the links Modify Request and Download.
  ```

- **Printer Friendly.** Results are in a printer-friendly format, without the paging controls, hot links, and so on.

  This is the format:

  ```
  &Action=print
  ```

  Example:

  ```
  saw.dll?Go&Path=/Shared/Test/SB2&Action=Print
  ```

- **Application Friendly.** Results are displayed in an application-friendly format, such as for Microsoft Excel, without the paging control, hot links, and so on.

  This is the format:

  ```
  &Action=Extract
  ```

  Example:

  ```
  saw.dll?Go&Path=/Shared/Test/SB2&Action=Extract
  ```

  The Extract action also acts as a Navigate action (read Section 6.3.2, "Passing Filters to the Oracle BI Presentation Services Go URL Through a URL (Navigation)") so you can filter the results that are returned by the call.

- **Specific View.** This shows an individual result view rather than the default compound view.

This is the format, where xx is the name of the view:

```
saw.dll?Go&Path=/Shared/Test/SB2&ViewName=xx
```

Example:

```
saw.dll?Go&Path=/Shared/Test/SB2&ViewName=Chart
```

Assuming that the request contains a Chart view named Chart, this displays just the Chart view.

- **Specific Style.** This shows the results using a specified style. If the style does not exist, the default is used.

  This is the format, where xx is the name of the style:

  ```
  saw.dll?Go&Path=/Shared/Test/SB2&Style=xx
  ```

  Example:

  ```
  saw.dll?Go&Path=/Shared/Test/SB2&Style=Lime
  ```

  This uses the style named Lime to show the results.

- **Result Format.** This controls the format of the results.

  This is the format, where xx is XML, HTML, or CSV:

  ```
  saw.dll?Go&Path=/Shared/Test/SB2&Format=xx
  ```

  Example:

  ```
  saw.dll?Go&Path=/Shared/Test/SB2&Format=XML
  ```

  This shows results in XML.

6.1.4.1 Displaying All Records in a Table

There are two ways to display all the records in the table:

- Set the Rows per Page property on the Table view to 10,000, and then use the basic Go. This is the easier of the two methods.
- Issue the following URL:

  ```
  saw.dll?Go&Path=/users/Administrator/a&Action=Scroll&P5=-1&ViewID=o:go~r:report
  ~v:compoundView!1~v:tableView!1
  ```

6.2 Referencing Dashboard Content in External Portals or Applications

This section describes how to use the Oracle BI Presentation Services Dashboard URL. It contains the following topics:

- "About the Oracle BI Presentation Services Dashboard URL"
- "Structure of the Basic Oracle BI Presentation Services Dashboard URL"
- "Optional Commands and Parameters for the Oracle BI Presentation Services Dashboard URL"

6.2.1 About the Oracle BI Presentation Services Dashboard URL

The Oracle BI Presentation Services Dashboard URL is for use in incorporating or referencing the content of a specific dashboard in external portals or applications. It has a number of forms and optional arguments that can be used to control its behavior.

You can post the Dashboard URL command as a Form or issue it as a URL. If you are issuing parameters as part of a URL, they need to be escaped properly. You need to replace spaces with plus (+) signs, and so on. For example, to pass East Region as a value, type East+Region.

When called from within an Oracle BI Presentation Services screen, such as a dashboard or an HTML result view, the URL should begin with this:

```
saw.dll?Dashboard
```

When called from another screen on the same web server, the URL should begin with this:

```
/analytics/saw.dll?Dashboard
```

When referenced from a screen on a different server (or sent through email, and so on), the URL should begin with the fully qualified server name or IP address:

```
http://server_name_or_ip_address/analytics/saw.dll?Dashboard
```

To test these commands, you can enter the fully qualified version into the Address field in the browser.

6.2.2 Structure of the Basic Oracle BI Presentation Services Dashboard URL

The basic Dashboard URL command needs no parameters. It displays the user's default portal after authenticating the user.

The following is the format of the basic Dashboard URL command:

```
http://server_name_or_ip_address/analytics/saw.dll?Dashboard&PortalPath=path of your dashboard
```

Note that PortalPath is the path of the dashboard and that when the user clicks the URL, the user is taken to the first page of the dashboard.

Example

```
http://localhost:9704/analytics/saw.dll?Dashboard&PortalPath=/shared/Paint Demo/_portal/Paint Dashboard
```

You can add the optional Page argument to specify which dashboard page displays when the user clicks the URL. Use the page number from the dashboard page tab as the value for this argument. This is the format of the URL including the Page argument.

```
http://server_name_or_ip_address/analytics/saw.dll?Dashboard&PortalPath=path of your dashboard&Page=page of your dashboard
```

Example

```
http://localhost:9704/analytics/saw.dll?Dashboard&PortalPath=/shared/Paint Demo/_portal/Paint Dashboard&Page=page 3
```

6.2.3 Optional Commands and Parameters for the Oracle BI Presentation Services Dashboard URL

You can add optional commands and parameters to the Dashboard URL. The below examples show and explain various uses of commands and parameters.

User ID and Password Example

You can modify the behavior of the Dashboard URL command by adding the user ID and password parameters. If the parameters are omitted, then the user is prompted for user ID and password information, unless the user chose the option to have logon information remembered when last logged on. If using a Session ID or Ticket, pass it as the NQUser parameter. In the parameter description, SB2 is the name of the request to execute.

This is the format where the user ID is uuu and the password is ppp:

```
&NQUser=uuu&NQPassword=ppp
```

Example URL:

```
http://localhost:9704/analytics/saw.dll?Dashboard&PortalPath=/shared/Paint Demo/_
portal/Paint Dashboard&NQUser=user1&NQPassword=rock
```

PortalPages Example

You can modify the behavior of the Dashboard URL command by adding the PortalPages command. This command opens a dashboard page without the common header. You can use this command to create a link or image with the specified dashboard page as the destination and open the dashboard page in a new browser window.

This is the format where the catalog path portal page is xxx and the dashboard page name is yyy.

```
PortalPages&PortalPath=xxx&Page=yyy&Done=close
```

Note the following command descriptions:

- PortalPath – This parameter contains the dashboard's catalog path.
- Page – (Optional) This parameter contains the dashboard page name.

Example URL:

```
http://localhost:8080/analytics/saw.dll?PortalPages&PortalPath=%2fshared%2fdashboa
rdfolder1%2f_portal%2fdefault&Page=page%202&Done=close
```

Go Example

You can modify the behavior of the Dashboard URL command by adding the Go command and specifying the catalog path to an analysis. This command opens an analysis without the common header. You can use this command to create a link or image with the specified analysis as the destination and open the analysis in a new browser window.

This is the format where the path is xxx and Done is close.

```
Go&Path=xxx&Done=close
```

Note the following parameter descriptions:

- Path – This parameter contains the analysis' catalog path.
- Done – This parameter is obsolete for the analysis' path.

Example URL:

```
http://localhost:8080/analytics/saw.dll?Go&Path=%2fusers%2fadministrator%2fdashboa
rd_actionlink_report&Action=Prompt&Done=close
```

PortalGo Example

You can modify the behavior of the Dashboard URL command by adding the PortalGo parameter and specifying the catalog path to the analysis. This parameter opens an analysis in a dashboard. You can use this parameter to create a link or image with an analysis as the destination, and open the analysis in the existing browser window.

This is the format where the portal path is xxx, the path is yyy, and Done is zzz.

```
PortalGo&PortalPath=xxx&Path=yyy&Done=zzz
```

Note the following parameter descriptions:

- PortalPath – This parameter is the dashboard's catalog path. It is in this dashboard that the analysis is displayed.
- Path – This parameter is the analysis' catalog path.
- Done – This parameter is used by the Return link and contains the return location.

Example URL:

```
http://localhost:8080/analytics/saw.dll?PortalGo&PortalPath=%2fusers%
   2fadministrator%2f_portal&Path=%2fusers%2fadministrator%2fSalesInMarket
```

6.3 Using the Oracle BI Presentation Services Go URL to Issue SQL and Pass Filters

This section explains how to use the Go URL command to issue SQL, and how to pass filters to be used for navigation. It contains the following topics:

- Section 6.3.1, "Issuing SQL Commands and Passing Filters"
- Section 6.3.2, "Passing Filters to the Oracle BI Presentation Services Go URL Through a URL (Navigation)"

6.3.1 Issuing SQL Commands and Passing Filters

The Go URL command can be used to issue Oracle Business Intelligence SQL. These forms of the Go URL return tabular results. The basic options from &Style= and &Options= can be used here as well.

To issue Oracle Business Intelligence's simplified SQL, include the escaped SQL as a parameter to the Go URL. For example:

```
saw.dll?Go&SQL=select+Region,Dollars+from+SupplierSales
```

where the FROM clause is the name of the Subject Area to query.

Alternatively, the command IssueRawSQL can be used to bypass the web processing and issue SQL directly against the BI Server.

6.3.2 Passing Filters to the Oracle BI Presentation Services Go URL Through a URL (Navigation)

The Go URL can also be used to pass context such as filters to a destination request. This is done by adding additional parameters to the call. You need to make sure that any columns you are passing are set up in the destination with Is Prompted filters, or specific default filters.

6.3.2.1 Navigation Parameters

The basic syntax of the navigation command is the same as presented in the section Section 6.2.2, "Structure of the Basic Oracle BI Presentation Services Dashboard URL", but with the addition of the Action=Navigate parameter, and then population of the P1 - Pn parameters, as necessary.

By default, you can add up to 100 parameters to the URL. However, you can adjust the number of parameters by modifying the Prompts/MaxPromptedURLParams setting in instanceconfig.xml. For more information see "Using a Text Editor to Update Configuration Settings" in the *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

```
&Action=Navigate
```

`&P0=n` where n is the number of columns you wish to filter, currently 1 - 6.

`&P1=op` where op is one of the following operators.

Operator	Meaning
eq	Equal to or in.
neq	Not equal to or not in.
lt	Less than.
gt	Greater than.
ge	Greater than or equal to.
le	Less than or equal to.
bwith	Begins with.
ewith	Ends with.
cany	Contains any (of the values in &P3).
call	Contains all (of the values in &P3).
like	You need to type %25 in place of the usual % wildcard. See the examples that follow.
top	&P3 contains 1+n, where n is the number of top items to display.
bottom	&P3 contains 1+n, where n is the number of bottom items to display.
bet	Between (&P3 must have two values).
null	Is null (&P3 must be 0 or omitted).
nnul	Is not null (&P3 must be 0 or omitted).
&P2=ttt.ccc	In this parameter, ttt is the table name and ccc is the column name. If the table or column contains spaces, it must be quoted with double-quotes. Spaces should be escaped as %20, for example, Measures."Dollar%20Sales".
&P3=n+xxx+yyy+...+zzz	In this parameter, n is the number of values, and xxx, yyy, and zzz are the actual values. **Note:** If the value of P3 begins with a numeric character, the entire value must be enclosed in quotes. For example: `saw.dll?Go&Path=/Shared/Test/SB2&Action=Navigate&P0=1&P1=top&P2=Customers.Region&P3="7West"`

Note: The settings for &P1,&P2, and &P3 are repeated for &P4-P6, &P7-P9, &P10-P12, &P13-P15, and &P16-P18 as necessary, depending on the value of &P0.

6.3.2.2 Navigation Examples

This returns records for the East and Central regions:

```
Saw.dll?Go&Path=/Shared/Test/SB2&Action=Navigate&P0=1&P1=eq&P2=Customers.Region&P3
=2+Central+East
```

This returns records for like Regions E....t:

```
saw.dll?Go&Path=/Shared/Test/SB2&Action=Navigate&P0=1&P1=like&P2=Customers.Region&
P3=1+E%25t
```

This returns the top two regions by dollars sold:

```
saw.dll?Go&Path=/Shared/Test/SB2&Action=Navigate&P0=1&P1=top&P2="Sales%20Facts".Do
llars&P3=1+2
```

This is an example where the number of arguments is not included in the syntax:

```
saw.dll?Go&Path=/Shared/Test/SB2&Action=Navigate&P0=1&P1=top&P2=Customers.Region&P
3=Central
```

Note: You can omit the number of arguments only if just one argument value is included.

This returns records with between 2,000,000 and 2,500,000 in sales:

```
saw.dll?Go&Path=/Shared/Test/SB2&Action=Navigate&P0=1&P1=top&P2="Sales%20Facts".Do
llars&P3=2+2000000+2500000
```

This returns records for Regions beginning with the letter E:

```
saw.dll?Go&Path=vate&P0=1&P1=bwith&P2=Customers.Region&P3=1+E
```

This returns records for Regions containing the letter E and having more than 20 million in sales:

```
saw.dll?Go&Path=/Shared/Test/SB2&Action=Navigate&P0=2&P1=cany&P2=Customers.Region&
P3=1+e&P4=gt&P5="Sales%20Facts".Dollars&P6=1+20000000
```

Oracle BI Presentation Services navigation is currently supported from charts, table and pivot table views, HTML views, and external applications and web pages. The destination search should have filters defined on columns for which it wants to receive context. These can be specific filters or, usually, the Is Prompted filter. In addition to the Table.Column value specifically referenced in the navigation call, all filters from the source request that have corresponding table.columns in the destination, are applied to the destination. Therefore, the appropriate context from a source can be passed to the destination.

6.3.2.3 Navigation Using JavaScript

Navigation can currently be accomplished using the custom text/date formatter for a column. The central concept is that you add a column you want to navigate from to your search. You then choose Custom Text Format from the properties for the column,

and enter HTML that calls one of the two provided JavaScript functions. This technique can be used to perform many actions, including sorting columns, calling custom JavaScript functions, and so on.

Oracle BI Presentation Services includes two JavaScript functions that enable navigation from Table and Pivot views: GoNav and PortalNav. (These functions are located in ORACLE_HOME/bifoundation/web/app/res/b_mozilla/viewhelper.js.) The former handles navigation to a specific search. The latter handles navigation to a specific dashboard. A description of their syntax follows, along with example Custom Text formats that you can use to implement navigation.

Note: To control the look of the navigable text using the style sheet, Oracle BI Presentation Services is standardized on the class=Nav.

GoNav function

```
function GoNav(event, sPath, sTbl, sCol, sVal, sTarget)
```

where:

`event = event indicator.`

`sPath` = the catalog path of the destination search.

`sTbl` = the logical table name to filter.

`sCol` = the logical column name to filter.

`sVal` = the value to filter by.

`sTarget` (optional) = "_blank" to open a new browser window with the results.

Sample Custom Text Format for GoNav Call

The GoNav and PortalNav calls can be wrapped in an HTML statement (include the quotes):

```
[html]"<font class=nav onclick=\"JavaScript:GoNav(event,
'/shared/topaz/performance/transaction details','
    Transaction','Quality','"@"');\">"@"</font>
```

Table 6–1 explains the elements of this example.

Table 6–1 Elemental Analysis of a GoNav Call

Element	Description
[html]	Tells Oracle BI Presentation Services to interpret the following text as HTML. Note that every "less than" character (<) must be preceded by a double quote (") if the intent is to use it in an HTML tag.
"<font	An HTML tag that a JavaScript call can be attached to. You could potentially use , , <a>, and so on.
class=nav	The CSS style class used for formatting of the HTML tag.
onclick=\"JavaScript:GoNav('event, /shared/topaz/performance/transaction details','Transaction','Quality','"@"');\"	The method to call a JavaScript function. When the user clicks on the contents of this HTML tag, then the JavaScript function is called.
>	The end of the font tag.

Table 6–1 (Cont.) Elemental Analysis of a GoNav Call

Element	Description
"@"	Instructs Oracle BI Presentation Services to replace the at sign (@) with the actual column value. When [html] is used, the @ symbol must be enclosed with double quotation marks (").
	The closing tag to match the <font> tag.

This example of GoNav places this HTML on the dashboard:

```
<a href="javascript:GoNav(event, '/shared/topaz/performance/transaction
details','Transaction','Quality','Some value');">Click here to navigate to
Transaction Details with ''Some value'</a>
```

PortalNav Function

```
function PortalNav(event, sPortal,sTbl,sCol,sVal)
```

`event = event indicator.`

`sPortal` = the catalog path of the destination portal.

`sTbl` = the logical table name to filter.

`sCol` = the logical column name to filter.

`sVal` = the value to filter by.

Sample Custom Text Format for PortalNav Call

Make sure to include the quotes exactly as shown.

```
"<font class=nav onclick=\"JavaScript:PortalNav(event, '/shared/topaz/_
portal/transaction analysis','
    Transaction','Type','"@"');\">"@"</font>"
```

6.3.2.4 Navigation from HTML Results

This is the same as described in Section 6.3.2.3, "Navigation Using JavaScript", but rather than using a custom formatter, type in the HTML syntax with static values in place of the @ signs.

6.4 Example of an Oracle Business Intelligence Third-Party SQL Tool Integration

This section illustrates the requirements for integrating a third-party SQL tool with Oracle Business Intelligence by describing an example integration, using Microsoft Access. Because Oracle Business Intelligence is designed as a middleware platform for enterprise data access and integration, common report writers and business intelligence tools can communicate natively with the BI Server.

Most third-party SQL tools require the user to include join conditions within queries to avoid cross-joins. A cross-join occurs when a request does not have a WHERE clause, which, in turn creates a Cartesian product of the tables involved in the join. The size of a Cartesian product is the number of rows in the first table multiplied by the number of rows in the second table.

To integrate Microsoft Access with the BI Server, the BI Server Administrator must expose the keys within the Presentation layer of the Oracle BI Administration Tool.

6.4.1 Example of integrating a third-party SQL tool

1. Drag and drop the keys from the Business Model and Mapping layer to the Presentation layer and save the repository.
2. Open Microsoft Access, select the option **Blank Access Database**, type the name oracle-analytics.mdb when prompted, and click **Create**.
3. After creating the new Microsoft Access database, right-click in the white section of the screen and select **Link Tables**.
4. From the Files of Type drop-down list box, select **ODBC Databases**.

   The Select a Source Dialog appears, and prompts you for a Data Source Name.
5. Click the **Machine Data Source** tab, locate the Analytics_Web DNS, and click **OK**.

   The Oracle BI Server requires a login.
6. Type your user ID and password.

   The Import Objects dialog box appears.
7. Click the **Select All** button, or highlight the desired logical tables from Oracle Business Intelligence.

   The import may take a while to complete.
8. When the import completes, right-click in the white section of the screen and select **Relationships**:
   a. Add the desired tables and drag and drop the keys from the dimension tables (Period, Market, Product) to the fact table (Sales Measures).
   b. Drag and drop Period Key over the perkey column, and repeat for each corresponding key to create the joins.

   Now, you can test and run a request.
9. Select **Create query in Design view** from the **Queries** button:
   a. Select Markets, Products and Sales Facts.
   b. Add Region, Brand, Units and Dollars, respectively, and then click Run.

6.5 Retrieving Links to Dashboard Pages Using Scripts

You can retrieve links (both bookmark links and prompted links) to dashboard pages by using the JavaScript functions described in Table 6–2 in your custom scripts on your dashboard pages.

> **Note:** For these JavaScript functions to work, the HardenXSS element must be set to false. For more information, see "Making Advanced Configuration Changes for Presentation Services" in the *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

For more information about bookmark links and prompted links, see "About Creating Links to Dashboard Pages" in *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition*.

Table 6–2 JavaScript Functions for Retrieving Links to Dashboard Pages

JavaScript Function	Description
`LinkToPage(bInlineDrill, bGetBookmarkOnly)`	Sets the value of the variable `saw.bookmarkURL` to the bookmark link. Set the arguments as follows: ■ `bInlineDrill` — Set to true if the **Page Options** button is displayed on the page; otherwise set to false. ■ `bGetBookmarkOnly` — Set to true to prevent the bookmark URL from being refreshed in the Address Bar of the browser. Set to false (or omit) to allow the bookmark URL to be refreshed in the Address Bar of the browser.
`GetPURL()`	Returns the prompted link as a string.

7

Oracle Business Intelligence Systems Management API

This chapter describes the Oracle Business Intelligence Systems Management API, which enables a developer to programmatically perform system administration tasks for Oracle Business Intelligence.

This chapter includes the following sections:

- Section 7.1, "What is the Oracle Business Intelligence Systems Management API?"

7.1 What is the Oracle Business Intelligence Systems Management API?

This Oracle Business Intelligence Systems Management API is a programming interface which provides access to Business Intelligence JMX Admin MBeans, enabling a developer to programmatically carry out system administration tasks for Oracle Business Intelligence. The system administration tasks that can be carried out using the BI Systems Management API, are also available using Fusion Middleware Control.

For more information, see "Introducing the BI Systems Management API" in *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition*.

8

Integrating Other Clients with Oracle Business Intelligence

This chapter describe how to configure an ODBC DSN to connect to the Oracle BI Server on Windows and UNIX platforms.

This chapter includes the following topics:

- Section 8.1, "Overview of Integrating with Oracle Business Intelligence"
- Section 8.2, "About Integrating with the Oracle BI Server as a Data Source"
- Section 8.3, "ODBC Conformance Level"
- Section 8.4, "Configuring an ODBC DSN for the Oracle BI Server on Windows"
- Section 8.5, "Configuring an ODBC DSN for the Oracle BI Server on Linux or UNIX"

8.1 Overview of Integrating with Oracle Business Intelligence

You can connect to the Oracle BI Server with a wide variety of ODBC-compliant query and reporting tools, as well as other clients such as remote Administration Tool clients.

> **Note:** Oracle generally does not confirm or test third-party client tool compatibility with the Oracle BI Server ODBC interface. Check the Oracle BI EE certification document for any specific certification information on third-party tools and the Oracle BI Server ODBC interface. For more information about the certification document, see "System Requirements and Certification".

To connect with a remote client, you configure an ODBC DSN for the Oracle BI Server on the remote client computer, and then use that DSN to connect to a repository from the query tool.

Oracle BI Presentation Services clients also connect to the Oracle BI Server using an ODBC DSN. A default ODBC DSN for Presentation Services is created and configured for you during Oracle Business Intelligence installation.

Administration Tool clients on the same host as a Presentation Services instance can also use the default DSN to connect to the Oracle BI Server. Or, you can create a separate DSN for the Administration Tool to use.

The connection parameters for the Cluster Controller and the SSL parameters in the default DSN are centrally managed by Fusion Middleware Control. Do not update

these parameters. If you attempt to manually update the centrally managed parameters, the values will be overwritten the next time the system is started.

In addition, do not change the name of the centrally managed default DSN. The default DSN has a name similar to "coreapplication_OH*id_number*," where *id_number* is a number specific to an installed Oracle home.

This chapter explains how you can integrate with the Oracle BI Server as a data source using ODBC on both Windows and UNIX platforms.

8.2 About Integrating with the Oracle BI Server as a Data Source

Open Database Connectivity (ODBC) is an industry standard interface for connecting to databases. A Data Source Name (DSN) is used to store the information about connecting to a given database as a given database user over ODBC.

You use the Oracle BI Server ODBC driver to configure a DSN to connect to a set of clustered Oracle BI Servers through the Cluster Controllers, or to an individual Oracle BI Server. The DSN you configure can be used with any ODBC-compliant query and reporting tool.

The Presentation layer lets you configure the presentation of a business model to be consistent with the rules and conventions of your tools to take advantage of the analytical engine and data abstraction of the Oracle BI Server. This makes it much easier to include columns involving complex aggregation and calculation rules in queries and reports. Also, if your organization is currently using query and reporting tools, using the Oracle BI Server as a data source makes these tools more valuable and simplifies the work entailed when using them.

Note: In a non-English environment, you can use a direct ODBC connection from a remote client to the Oracle BI Server. However, the translation and localization option that is provided with Oracle BI Presentation Services is not available. Instead, the translation and localization included in the Oracle BI Server metadata (the Oracle BI repository) is available. For example, if Presentation layer metadata objects are defined in French, they are displayed in French through a third-party application.

8.2.1 About Routing Requests to the Physical Layer

Oracle BI Server clients can create a dedicated physical connection to the Oracle BI Server over which all `SELECT` queries are treated as `SELECT_PHYSICAL` queries. `SELECT_PHYSICAL` queries directly query objects in the Physical layer of the metadata repository, bypassing the Presentation layer and the Business Model and Mapping layer. To enable this direct connection for your ODBC client, select **Route Requests To Physical Layer** in the DSN configuration.

Note that you cannot have a single ODBC connection that sends queries to both the Presentation layer, and the Physical layer. Instead, you can create one regular connection that sends requests to the Presentation layer, and one connection that sends requests to the Physical layer.

See "Syntax and Usage Notes for SELECT_PHYSICAL" in *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition* for more information about `SELECT_PHYSICAL` queries.

8.2.2 About Integrating with the Oracle BI Server Using JDBC

In addition to using ODBC, you can also integrate with the Oracle BI Server using JDBC. For full information, see the README.TXT file contained in the bijdbc.jar file. You can find the bijdbc.jar file in the following directory:

ORACLE_HOME/bifoundation/jdbc

8.3 ODBC Conformance Level

The Oracle BI Server supports the following ODBC calls from client applications:

- SQLAllocConnect
- SQLAllocEnv
- SQLAllocStmt
- SQLBindCol
- SQLCancel
- SQLColumns
- SQLConnect
- SQLDescribeCol
- SQLDisconnect
- SQLDriverConnect
- SQLError
- SQLExecDirect
- SQLExecute
- SQLExtendedFetch
- SQLFetch
- SQLFreeConnect
- SQLFreeEnv
- SQLFreeStmt
- SQLGetConnectOption
- SQLGetCursorName
- SQLGetData
- SQLGetFunctions
- SQLGetInfo
- SQLGetStmtOption
- SQLGetTypeInfo
- SQLColAttributes
- SQLNumResultCols
- SQLPrepare
- SQLRowCount
- SQLSetConnectOption

- SQLSetStmtOption
- SQL Tables

Oracle Business Intelligence ODBC supports full scrollable cursors with static, dynamic, forward only, and key set driven cursors. Oracle Business Intelligence ODBC also supports asynchronous and synchronous processing and cancellation.

8.4 Configuring an ODBC DSN for the Oracle BI Server on Windows

This section explains how to create an ODBC DSN for the Oracle BI Server on Windows to enable remote client access. You use the Oracle BI Server DSN Configuration Wizard to set up an ODBC DSN that you can use to connect to a repository through the Oracle BI Server.

The Oracle BI Server ODBC driver (installed with the BI Admin Tool) is 32-bit or 64-bit, depending on the bitness of the BI Admin Tool installed. The 32-bit version of ODBC Data Source Administrator is located at C:\Windows\SysWOW64\odbcad32.exe, and the 64-bit version is located at C:\Windows\system32\odbcad32.exe.

To create a DSN for the Oracle BI Server on Windows:

1. Open the Windows Control Panel by selecting **Start > Settings > Control Panel**, double-click **Administrative Tools**, and then double-click **Data Sources (ODBC)**.
2. In the ODBC Data Source Administrator dialog, click the System DSN tab, and then click **Add**.
3. In the Create New Data Source dialog, select the driver **Oracle BI Server 11g_ OH*id_number*** (where ***id_number*** is a number specific to an installed Oracle home), and then click **Finish**. The first page of the Oracle BI DSN Configuration wizard is displayed.
4. Type a name for the data source in the **Name** field.

   Note: Do not change the name of the centrally managed default DSN for Presentation Services and the Administration Tool.
5. Optionally, enter a description in the **Description** field.
6. To connect to a single Oracle BI Server that is not part of a cluster, in the **Server** field, select the computer on which the Oracle BI Server is running.

   If the server name does not appear in the list, then type the computer name in the **Server** field.

   Note: This field is not used for the default DSN created for Presentation Services, because the Oracle Business Intelligence system is clustered by default.
7. To connect to a set of clustered Oracle BI Servers through the Cluster Controllers, do the following:
   a. Select **Clustered DSN**.
   b. Enter information for the primary and secondary Cluster Controllers, as follows:
      – In the **Primary Controller** field, enter the name of the computer that is specified as the primary Cluster Controller. Then, enter the port number for the Primary Controller in the appropriate **Port** field.
      – If a secondary Cluster Controller has been set up, then type the name of the host where the Secondary Controller is running in the **Secondary**

Controller field. Then, enter the port number for the Secondary Controller in the appropriate **Port** field.

You can find information about the Primary and Secondary Controller hosts and ports on the Availability tab of the Capacity Management page in Fusion Middleware Control.

c. To test the connection to the Cluster Controllers, click **Test Cluster Connect**.

If the test is not successful, then correct any errors identified in the message and test the connection again.

Note: Do not change the Primary Controller, Secondary Controller, and Port parameters of the centrally managed default DSN for Presentation Services.

8. Select **Route Requests To Physical Layer** to create a dedicated physical connection to the Oracle BI Server for clients using this DSN. All `SELECT` queries over this connection will be treated as `SELECT_PHYSICAL` queries.

   `SELECT_PHYSICAL` queries directly query objects in the Physical layer of the metadata repository, bypassing the Presentation layer and the Business Model and Mapping layer.

9. Select **Use Forward Only Cursor** to change the ODBC cursor from its default scrollable mode to forward only.

10. To configure this DSN to communicate over SSL, select **Use SSL**.

    To configure advanced settings for SSL, see "Configuring the ODBC DSN for Advanced SSL Settings" for additional instructions.

    Note: Do not change the SSL parameters of the centrally managed default DSN for Presentation Services.

11. Click **Next**. The second page of the Oracle BI DSN Configuration wizard is displayed.

12. Optionally, for **Login ID** and **Password**, enter a user name and corresponding password for the Oracle BI Server.

13. To save the user name in the Windows registry for this client, select **Save login ID**. If you select this option, you will not have to enter your user name each time you connect. You will still have to enter a password.

14. Enter a port number for the Oracle BI Server in the **Port** field. This option is enabled when **Clustered DSN** has not been selected on the first page of the wizard.

    You can find information about the Oracle BI Server port on the Availability tab of the Capacity Management page in Fusion Middleware Control.

15. If you want to connect to a repository other than the default repository, then select **Change the default repository to**, and then type the logical name of the repository to which you want to connect.

    You can only use this setting if you have configured the Oracle BI Server to host multiple repositories. Note that hosting multiple repositories on a single Oracle BI Server is not recommended for production deployments.

Note: For Presentation Services clients, each Presentation Services instance can only access a single repository. To configure multiple Presentation Services instances to access multiple repositories, follow these steps:

1. Install Presentation Services on one computer for each repository instance. For example, if you have three repositories running on your Oracle BI Server, install three Presentation Services instances on three separate computers.
2. Adjust the setting **Change the default repository to** in the DSN for each Presentation Services instance.
3. As a web client end user, you can choose which repository to access from your client browser by substituting the appropriate web server computer name or IP address in the URL for the Presentation Services instance that points to the repository you want to work with:

   ```
   http://host_name_or_IP_address/analytics/saw.dll?
   ```

16. Optionally, select **Connect** to obtain default settings for the additional configuration options.

    If you select this option, the wizard will attempt to connect to the server to obtain default values for the configuration settings on the next screen. If you do not select this option, then you can still configure the DSN by manually entering the information in the next screen.

    If you select this option, you must provide values for **Login ID** and **Password**.

17. Click **Next**. The third page of the Oracle BI DSN Configuration wizard is displayed.

18. To change the name of the default subject area, select **Change the default subject area to** and then enter the name of the subject area. If you selected the **Connect** option in the previous screen, you can choose a name from the list.

    Do not select this option for the DSN used by Oracle BI Presentation Services.

    If you do not select this option, the default subject area is the one defined in the internal call query metadata. You can use the `DATABASE` SQL function to determine the default subject area; see *Oracle Fusion Middleware Metadata Repository Builder's Guide for Oracle Business Intelligence Enterprise Edition* for more information.

19. To change the default error message language, select **Change the default error message language to** and then select the language you want to use for error messages. See *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for more information about choosing the error message language.

20. If needed, select **Use Regional Settings when outputting dates and times**. When this option is enabled, data in Date, Time, or DateTime format is displayed using Windows regional settings, rather than the default format for the Oracle BI Server. To see the Windows regional settings, open the Control Panel and double-click Regional and Language Options.

21. If the repository has been set up for database-specific login IDs and passwords, you can enter the user names and passwords for the underlying databases. The database-specific credentials allow privileged users to connect to the underlying

databases at the level of authority granted to those users in the databases. To enter user names and passwords for the underlying databases, follow these steps:

a. Select the database for which you want to enter a user name and password and click **Edit**.

b. Enter the login ID and password corresponding to the database you selected and click **OK**.

c. Repeat these steps for other databases if necessary.

22. Click **Finish** to save the DSN configuration.

8.4.1 Configuring the ODBC DSN for Advanced SSL Settings

Follow the steps in this section to configure additional advanced settings for SSL.

Note: Do not change the SSL parameters of the centrally managed default DSN for Presentation Services.

To configure the Oracle BI Server DSN for advanced SSL settings:

1. On the first page of the Oracle BI DSN Configuration wizard, after selecting **Use SSL**, click the **Configure SSL** button. The Secure Socket Layer Configuration dialog appears.
2. For **Certificate File**, enter the path and file name of the Client Certificate file, or click **Select** to browse for the file. For example:

   ORACLE_HOME\ssl\client-cert.pem

3. For **Certificate Private Key File**, enter the path and file name of the Client Private Key file, or click **Select** to browse for the file. For example:

   ORACLE_HOME\ssl\client-key.pem

4. For **File Containing Passphrase**, enter the path and file name of the passphrase file for the Client Key, or click **Select** to browse for the file. For example:

   ORACLE_HOME\ssl\clientpwd.txt

5. Select **Verify Peer**.
6. If you are using the hashed version of the CA certificate, enter the directory where the hashed file is located in the **CA Certificate Directory** field. For example:

   ORACLE_HOME\ssl

7. If you are using the CA certificate, provide the path and file name of the CA Certificate file in the **CA Certificate File** field. For example:

   ORACLE_HOME\ssl\cacert.pem

8. For **Cipher List**, enter the list of ciphers to be used. For example:

   EXP-DES-56-SHA

9. For **Certificate Verification Depth**, specify 1.
10. For **Trusted Peer Distinguished Names**, enter DNs of servers that will allowed to connect. For example:

    C=US/ST=CA/L=Redwood Shores/O=Oracle/ OU=BI/CN=servercertificate

11. Click **OK**.

12. Complete the additional steps to configure the DSN, starting from Step 11 of the procedure in the previous section.

After you complete these steps, copy the client certificate, client private key, and passphrase files (for example, client-cert.pem, client-key.pem, and clientpwd.txt) to the directory specified in the parameters. If you have set the CA Certificate File parameter, you must also copy the CA certificate file (for example cacert.pem) to the directory specified. If you have set the CA Certificate Directory parameter, copy the hash version of the CA certificate to the directory specified.

8.5 Configuring an ODBC DSN for the Oracle BI Server on Linux or UNIX

This section explains how to create an ODBC DSN for the Oracle BI Server on Linux or UNIX to enable remote client access.

On Linux and UNIX systems, the file odbc.ini contains the standard or clustered Oracle BI ODBC connection details that are used by Oracle BI Presentation Services and `nqcmd` processes.

Do not change the DSN logical name, the Primary and Secondary Cluster Controller and Port parameters, or the SSL parameters for the centrally managed default DSN.

To create an ODBC DSN for the Oracle BI Server on Linux or UNIX:

1. Log on as a separate telnet session.
2. Open the odbc.ini file for editing. You can find this file at:

   ORACLE_INSTANCE/bifoundation/OracleBIApplication/coreapplication/setup/odbc.ini

3. In the section [ODBC Data Sources], add the new data source name you want to create and define it as an Oracle BI Server data source. For example:

   ```
   [ODBC Data Sources]
   my_new_dsn = Oracle BI Server
   ```

4. Add a section for the new DSN and provide parameters as follows:

   ```
   [my_new_dsn]
   Driver = ORACLE_HOME/bifoundation/server/bin/call_interface.library_suffix
   ServerMachine = local
   Port = bi_server_port_number
   ForwardOnlyCursor = No
   SelectPhysical = No
   Regional = Yes
   ```

   Note the following:

   - For call_interface, enter nqsodbc for ODBC35 (the default), or nqsodbc20 for ODBC20.
   - For *library_suffix*, use the library suffix appropriate for your operating system. For example, use `.so` for Linux, Solaris, or AIX, or use .sl for HP-UX.
   - For Port, enter the Oracle BI Server port number. Do not provide this option if you want this data source to connect to clustered Oracle BI Servers.

     You can find information about the Oracle BI Server port number on the Availability tab of the Capacity Management page in Fusion Middleware Control.
   - Set ForwardOnlyCursor to Yes to change the ODBC cursor from its default scrollable mode to forward only.

- Set SelectPhysical to Yes to enable integrating clients to create a dedicated physical connection to the Oracle BI Server over which all `SELECT` queries are treated as `SELECT_PHYSICAL` queries.
- Set Regional to Yes to display data in Date, Time, or DateTime format using the regional settings for your operating system, rather than the default format for the Oracle BI Server.

5. If you want this data source to connect to clustered Oracle BI Servers, supply the following additional parameters. Note that the primary CCS and secondary CCS should not be on the same computer.

```
IsClusteredDSN=Yes
PrimaryCCS=primary_cluster_controller_name
PrimaryCCSPort=primary_cluster_controller_port_number
SecondaryCCS=secondary_cluster_controller_name
SecondaryCCSPort=secondary_cluster_controller_port_number
```

   Note the following:

   - If you want this data source to connect to a single Oracle BI Server, set IsClusteredDSN to No.
   - You can find information about the Primary and Secondary Controller hosts and ports on the Availability tab of the Capacity Management page in Fusion Middleware Control.

6. To configure SSL for this data source, supply the following additional parameters as appropriate for your SSL deployment:

```
SSL = Yes
SSLCertificateFile = path_to_client_certificate_file
SSLPrivateKeyFile = path_to_client_private_key_file
SSLPassphraseFile = path_to_passphrase_file_for_client_key
SSLCipherList = list_of_cipers
SSLVerifyPeer = Yes
SSLCACertificateDir = path_to_hashed_version_of_ca_certificate_file
SSLCACertificateFile = path_to_ca_certificate_file
SSLTrustedPeerDNs = list_of_server_DNs_allowed_to_connect
SSLCertVerificationDepth = 1
```

7. Save and close the file.
8. If you are updating the DSN used by Presentation Services, you must restart Presentation Services.

9

Using Discoverer Data in Applications

This chapter describes exposing Oracle Business Intelligence Discoverer worksheets in third-party client applications. Describes converting Discoverer metadata for use in Oracle BI EE.

This chapter includes the following sections:

- Section 9.1, "Exposing Discoverer Worksheets in Applications"
- Section 9.2, "Converting Discoverer Metadata to Use in Oracle BI EE"

9.1 Exposing Discoverer Worksheets in Applications

You can expose Discoverer worksheets for use in third-party client applications, using Oracle Business Intelligence Discoverer Web Services API (Application Programming Interface). A third-party client application can use the Discoverer Web Services API to programmatically obtain Discoverer connections, workbooks, and worksheets; to execute worksheet queries; and to display worksheet content.

For information, see *Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Discoverer Web Services API*.

9.2 Converting Discoverer Metadata to Use in Oracle BI EE

You can convert Discoverer metadata into a format that can be used in Oracle BI EE by running the metadata conversion assistant tool. The metadata conversion assistant tool is a command line utility that accelerates the migration of Discoverer metadata from the End User Layer (EUL) to Oracle BI EE. The command line utility takes a Discoverer EEX export file and generates an RPD file that can be used in Oracle BI EE. Note that this RPD file is compatible with Oracle BI EE 10*g*, but you can use the Oracle BI EE Upgrade Assistant to upgrade this RPD file for use with Oracle BI EE 11*g*.

You can find the metadata conversion assistant (MigrateEUL.exe) in the following location:

ORACLE_HOME\bifoundation\server\bin

You can find documentation that describes how to use the metadata conversion assistant in the following location:

ORACLE_HOME\bifoundation\server\document

Note: This utility migrates only Discoverer metadata and only for relational data sources.

10

Integrating with Oracle E-Business Suite Security

This chapter describes setting up Oracle Business Intelligence to use Oracle E-Business Suite security to authenticate actions users who navigate to Oracle E-Business Suite. Describes creating a database object and connection pool for the database, setting up authentication, and embedding a link in Oracle E-Business Suite that opens Oracle Business Intelligence dashboards.

This chapter includes the following sections:

- Section 10.1, "Creating a Database Object and Connection Pool for the Oracle E-Business Suite Database"
- Section 10.2, "Setting Up Authentication"
- Section 10.3, "Embedding Links to Oracle Business Intelligence in Oracle E-Business Suite"

10.1 Creating a Database Object and Connection Pool for the Oracle E-Business Suite Database

To enable integration with Oracle E-Business Suite, you must create a database object and connection pool for the Oracle E-Business Suite database in the Oracle BI repository.

To create a database object and connection pool for the Oracle E-Business Suite database:

1. In the Administration Tool, open the repository that you want to integrate with Oracle E-Business Suite.
2. Right-click in the Physical layer and select **New Database**.
3. Enter a name for the new database (for example, Oracle E-Business Suite 12).
4. For **Database**, select the appropriate Oracle Database type for your Oracle E-Business Suite database (for example, Oracle 10g R2 or Oracle 11g).
5. Click **OK**.
6. Right-click the new database object you just created and select **New Object**, then select **Connection Pool**.
7. Enter a name for the connection pool (for example, Oracle E-Business Suite 12).

   You must provide a unique name for this connection pool. Because of this requirement, do not name the object 'Connection Pool.'

8. For **Call interface**, select OCI 10g/11g.
9. For **Data source name**, enter the TNS name of the Oracle E-Business Suite database.
10. For **User name** and **Password**, enter the user name and password of the Oracle E-Business Suite super user.
11. Select the Connection Scripts tab.
12. Click **New** for **Execute on connect**.
13. Enter the following physical SQL, and then click **OK**:

```
call APP_SESSION.validate_icx_session('valueof(NQ_SESSION.ICX_SESSION_COOKIE)')
```

    Check that this script is enabled.
14. Click **OK** in the "Connection Pool dialog."
15. Save the repository.

10.2 Setting Up Authentication

This section explains how to set up shared authentication between Oracle Business Intelligence and Oracle E-Business Suite.

> **Note:** If you set up shared authentication using the EBS ICX authentication cookie as described in this section, you cannot use single sign-on to seamlessly navigate from Oracle Business Intelligence to Oracle BI Publisher.

This section contains the following topics:

- Section 10.2.1, "Setting Up Session Variables for Authentication"
- Section 10.2.2, "Updating authenticationschemas.xml"
- Section 10.2.3, "Updating instanceconfig.xml"

10.2.1 Setting Up Session Variables for Authentication

To set up proper authentication for your integrated environment, you must set up several session variables and an initialization block in the Oracle BI repository.

To set up session variables for authentication in the Administration Tool:

1. In the Administration Tool, open the repository that you want to integrate with Oracle E-Business Suite.
2. Select **Manage**, then select **Variables**.
3. From the **Action** menu, select **New**, then **Session**, then **Initialization Block**.
4. Enter a name for the initialization block (for example, Oracle E-Business SSO).
5. Click **Edit Data Source**.
6. For **Default initialization string**, enter the following:

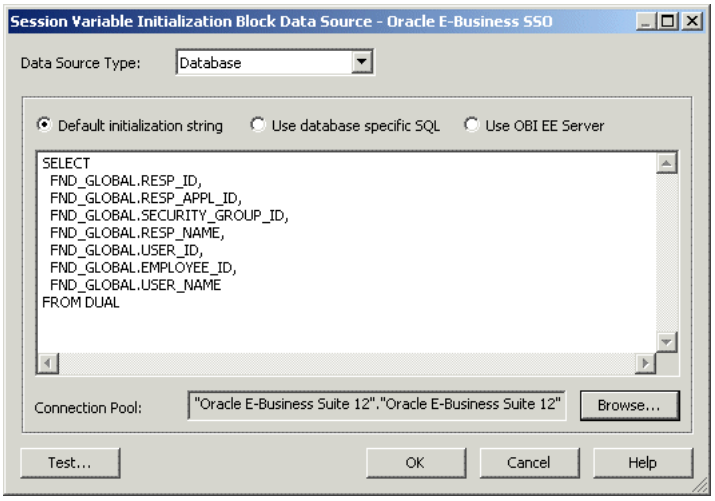
```
SELECT
  FND_GLOBAL.RESP_ID,
  FND_GLOBAL.RESP_APPL_ID,
  FND_GLOBAL.SECURITY_GROUP_ID,
```

```
  FND_GLOBAL.RESP_NAME,
  FND_GLOBAL.USER_ID,
  FND_GLOBAL.EMPLOYEE_ID,
  FND_GLOBAL.USER_NAME
FROM DUAL
```

7. For **Connection Pool**, click **Browse**, select the connection pool you created for the Oracle E-Business Suite database (for example, Oracle E-Business Suite 12), and click **Select**.

   Figure 10–1 shows the "Session Variable Initialization Block Data Source dialog," with example values for **Default initialization string** and **Connection Pool**.

Figure 10–1 Session Variable Initialization Block Data Source Dialog



8. Click **OK**.
9. Click **Edit Data Target**.
10. Create the following session variables:

```
EBS_RESP_ID
EBS_RESP_APPL_ID
EBS_SEC_GROUP_ID
EBS_RESP_NAME
EBS_USER_ID
EBS_EMPLOYEE_ID
USER
```

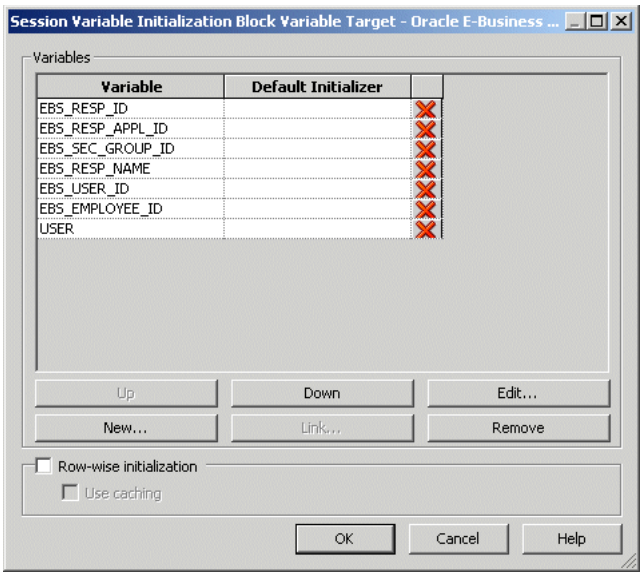
    To do this, click **New**, enter the variable name, and then click **OK**. Click **Yes** when you receive a warning about the special purpose of the `USER` variable.

    Optionally, you can select **Security Sensitive** before clicking **OK** for each variable. See *Oracle Fusion Middleware System Administrator's Guide for Oracle Business Intelligence Enterprise Edition* for more information about this option.

    You must ensure that the variables are listed in the given order. If necessary, select a variable and click **Up** or **Down** to reorder the list.

    Figure 10–2 shows the "Session Variable Initialization Block Variable Target dialog," with the required variables displayed in the required order.

Figure 10–2 Session Variable Initialization Block Variable Target Dialog



11. Click **OK** in the "Session Variable Initialization Block Variable Target dialog".
12. In the "Session Variable Initialization Block dialog," select **Required for authentication**.
13. Click **OK**.
14. Save the repository.

10.2.2 Updating authenticationschemas.xml

You must update the authenticationschemas.xml file to add the name of the EBS ICX authentication cookie.

To update authenticationschemas.xml:

1. Open the file authenticationschemas.xml for editing. You can find this file at:

   ORACLE_HOME/bifoundation/web/display

2. Find the following element:

   ```
   <AuthenticationSchema name="EBS-ICX">
   ```

3. Locate the sub-element RequestVariable source="cookie" and change the value of the nameInSource attribute from ICX_SESSION to the name of the EBS ICX authentication cookie prefix. For example:

   ```
   <RequestVariable source="cookie" type="auth" nameInSource="VIS" biVariableName=
   "NQ_SESSION.ICX_SESSION_COOKIE" />
   ```

   Do not update the RequestVariable source="url" sub-element.

Note: Ask your Oracle E-Business Suite administrator for the name of the EBS ICX authentication cookie if you do not know it. Alternatively, follow these steps to find the cookie name:

1. Log in to Oracle E-Business Suite.
2. Enter the following text in the address bar of your browser:

   ```
   javascript:document.writeln(document.cookie);
   ```

3. The cookie is displayed. For example:

   ```
   ORA_BIPS_LBINFO=1262d6a5f9a; ORA_MOS_LOCALE=en%7CUS; ORA_UCM_
   INFO=3~000271477666646140522702168700 92~LastName~FirstName~Firs
   tname.LastName@mycompany.com~USA~en~~~~~1; ORA_UCM_VER=
   %2FMP%2F8kgic%2Cr_ wjmp%3Emp_ajc%2CamkMP%2F8iega*p%5Duhkn%3Ckn
   %5D_ha*_kiMP%2F8%2F26%2C65%2C7%2C22MP%2F8-04*43*5*00; ORA_UCM_
   SRVC=3*OTN~1~0~//~null~*OPN~1~0~//~SE1%3ASE1%3ASE1%3ASE1%3ASE1%
   3ASE1%3ASE1%3ASE1%3A~*EMP~1~0~/34/~null~*GMO~1~0~//~null; ORA_
   TAHITI_PREFS=~0--------------; VIS=ZcEJeoLNVqcHGiGYvCpzTx3N:S;
   ADMINCONSOLESESSION=0yQmLP2D67vJKgtXLxsNl534QTWlThYkyvXfR0fjFK0
   LPsD3Hh83!1322564050
   ```

   The value you need to provide in authenticationschemas.xml is the prefix of the EBS ICX authentication cookie. In the previous example, the EBS ICX authentication cookie is `VIS=ZcEJeoLNVqcHGiGYvCpzTx3N:S;`, and the prefix is `VIS`.

4. In the same entry (RequestVariable source="cookie"), ensure that the value of the biVariableName attribute is the same as the value you entered as part of the connection script when you created the connection pool for the Oracle E-Business Suite database. See Step 13 of Section 10.1, "Creating a Database Object and Connection Pool for the Oracle E-Business Suite Database" for more information.
5. Find the following element:

   ```
   <SchemaKeyVariable source="cookie">
   ```

6. Change the value of the nameInSource attribute from ICX_SESSION to the name of the EBS ICX authentication cookie prefix (often `VIS`). For example:

   ```
   <SchemaKeyVariable source="cookie" nameInSource="VIS" forceValue="EBS-ICX"/>
   ```

7. Save and close the file.

10.2.3 Updating instanceconfig.xml

You must update the instanceconfig.xml file to add EBS ICX as one of the enabled schemas, and set it as the default. You must update the instanceconfig.xml file to configure login and logout information.

To update instanceconfig.xml:

1. Configure Oracle Business Intelligence to use an SSO provider of "Custom," as described in "Enabling SSO Authentication Using Fusion Middleware Control" in *Oracle Fusion Middleware Security Guide for Oracle Business Intelligence Enterprise Edition*.
2. Open the file instanceconfig.xml for editing. You can find this file at:

   ORACLE_INSTANCE/config/OracleBIPresentationServicesComponent/coreapplication_obips*n*

3. Locate the Authentication element.
4. Include EBS ICX in the list of enabled schemas. For example:

```
<EnabledSchemas>UidPwd,Impersonate,UidPwd-soap,Impersonate-soap,EBS-ICX</Enable
dSchemas>
```

   Ignore the comment in instanceconfig.xml that says this setting is centrally managed. EBS ICX must be manually added to the EnabledSchemas element.
5. Save and close the file.
6. Restart Oracle Business Intelligence.

10.3 Embedding Links to Oracle Business Intelligence in Oracle E-Business Suite

To embed a link in Oracle E-Business Suite that opens Oracle Business Intelligence dashboards, you need to create a form function and then assign menus and responsibilities.

Before you begin, log in to Oracle E-Business Suite as the system administrator (for example, sysadmin). Then, select the **System Administrator** responsibility from the responsibility navigator pane on the left. The available menus appear on the right.

Follow the steps in this section to create the following objects, in sequence:

- Function
- Menu
- Responsibility
- User
- Profile

This section contains the following topics:

- Section 10.3.1, "Domain Prerequisites"
- Section 10.3.2, "Creating a Form Function"
- Section 10.3.3, "Creating a Menu That Invokes the Form Function"
- Section 10.3.4, "Assigning the Menu to a Responsibility"
- Section 10.3.5, "Assigning the Responsibility to a User"
- Section 10.3.6, "Setting Up a Profile"

10.3.1 Domain Prerequisites

If the URL entry points to your Oracle E-Business Suite instance and your Oracle Business Intelligence instance are configured using different internet domains, you must use an appropriate mechanism to create a single logical domain for the two URLs. This step is required to configure a link between Oracle E-Business Suite and Oracle Business Intelligence that is capable of passing the Oracle E-Business Suite session information to Oracle Business Intelligence.

For example, if Oracle E-Business Suite is available on

```
http://ebshost.ebsdomain.com:8001
```

and Oracle Business Intelligence is available on

```
http://bihost.bidomain.com:9704
```

an appropriate mechanism needs to be put in place so that a browser can access both URLs using the same domain.

One way to achieve this is to install a supported HTTP server such as Oracle HTTP Server. In the example above, the HTTP server against Oracle Business Intelligence could be configured so that a request for

```
http://bihost.ebsdomain.com:9704
```

maps to

```
http://bihost.bidomain.com:9704
```

In this way a client browser can link to

```
http://bihost.ebsdomain.com:9704
```

and under the covers, the http server will point to the real location.

For information about configuring and securing Oracle HTTP Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

10.3.2 Creating a Form Function

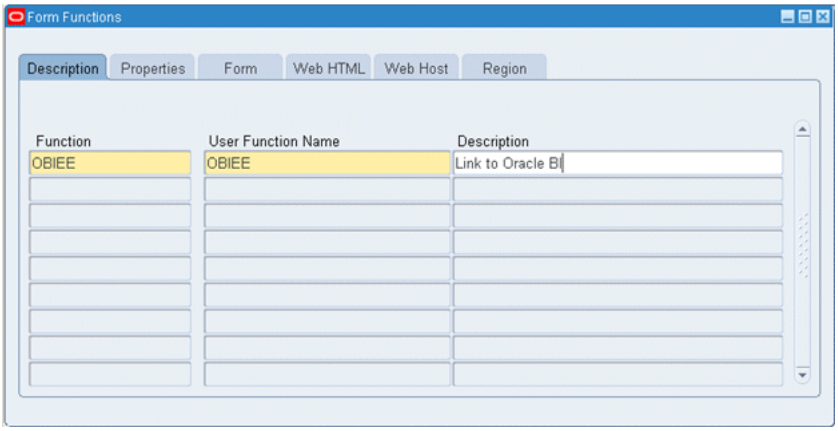
This section explains how to create a form function in Oracle E-Business Suite.

To create a form function:

1. From the **Application** menu, select **Function**. The "Form Functions dialog" is displayed.

   Figure 10–3 shows the "Form Functions dialog."

Figure 10–3 Form Functions Dialog



2. Enter the name of the function in the **Function** field (for example, OBIEE).
3. Enter the user function name (for example, OBIEE).
4. Enter a description (for example, Link to Oracle BI).
5. Save your changes using the **Save** button on the toolbar.
6. Select the Properties tab.

7. For **Type**, enter SSWA jsp function.
8. Select the Web HTML tab.
9. For **HTML Call**, enter one of the following options:
   - To link to Answers, enter:

     OracleOasis.jsp?mode=OBIEE&function=Answers
   - To link to Dashboards, enter:

     OracleOasis.jsp?mode=OBIEE&function=Dashboard
10. Save your changes using the **Save** button on the toolbar, and then close the "Form Functions dialog."

10.3.3 Creating a Menu That Invokes the Form Function

This section explains how to create a menu that invokes the form function in Oracle E-Business Suite. Note that menus are compiled whenever they are updated.

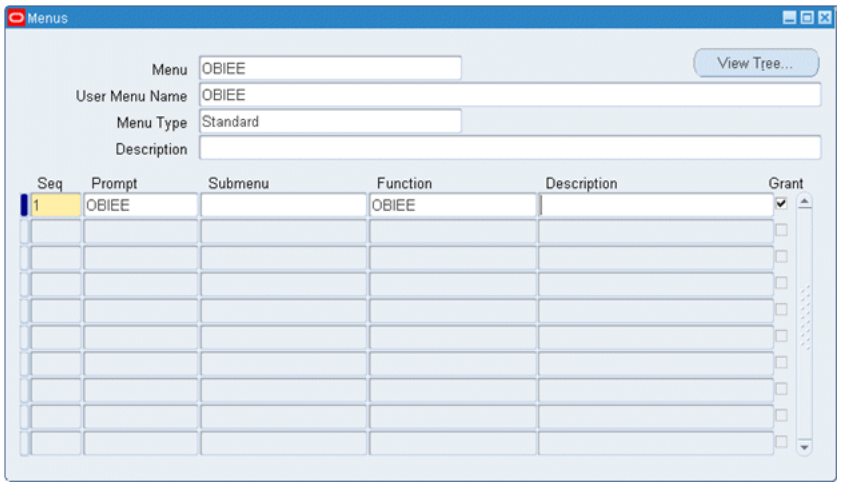
To create a menu that invokes the form function:

1. From the **Application** menu, select **Menu**. The "Menus dialog" is displayed.

   Tip: If you are already in Forms, you can select Menu from the Top Ten list.

   Figure 10–4 shows the "Menus dialog."

Figure 10–4 Menus Dialog



2. Enter the name of the menu in the **Menu** field (for example, OBIEE).
3. Enter a **User Menu Name** (for example, OBIEE).
4. For **Menu Type**, enter Standard.
5. For **Function**, enter the name of the function you created in Section 10.3.2, "Creating a Form Function."
6. Save your changes using the **Save** button on the toolbar, and then close the "Menus dialog."

Note that if a menu has only one function, then that function is selected by default for the user. If this is the case, intermediate steps like displaying the function may be skipped.

10.3.4 Assigning the Menu to a Responsibility

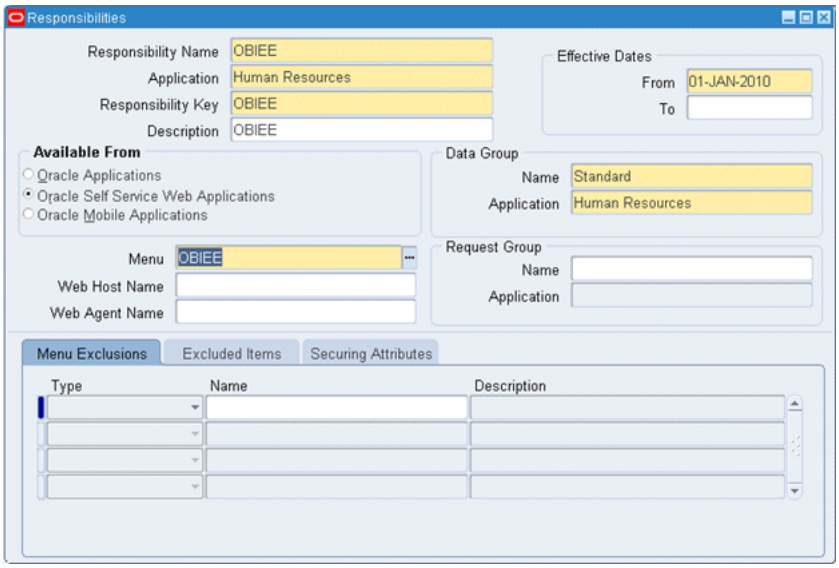
The menu that you created in Section 10.3.3, "Creating a Menu That Invokes the Form Function" must be associated with a responsibility. You can reuse an existing responsibility, or create a responsibility, as shown in the following procedure.

To assign the menu to a new responsibility:

1. Select **Responsibilities** from the Top Ten List.

   Figure 10–5 shows the "Responsibilities dialog."

Figure 10–5 Responsibilities Dialog



2. Enter a name for the responsibility (for example, OBIEE).
3. For **Application**, enter the application for which you created the menu.
4. For **Responsibility Key**, define any unique value. To ensure that this value is unique, the Responsibility Key is not translated.
5. For **Available From**, select **Oracle Self Service Web Applications**.
6. For **Data Group**, enter Standard for **Name** and re-enter the application name for **Application**.
7. For **Menu**, enter the name of the menu you created in Section 10.3.3, "Creating a Menu That Invokes the Form Function."
8. Save your changes using the **Save** button on the toolbar, and then close the "Responsibilities dialog."

10.3.5 Assigning the Responsibility to a User

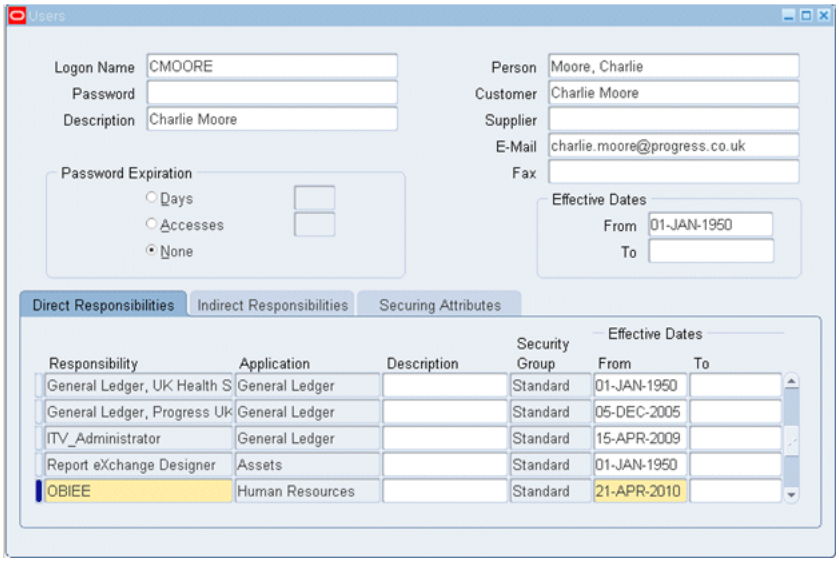
You must assign the responsibility that is associated with the menu to a user. You can create a user, or assign the responsibility to an existing user, as shown in the following procedure.

To assign the responsibility to an existing user:

1. Select **Users** from the Top Ten List.

   Figure 10–6 shows the "Users dialog."

Figure 10–6 Users Dialog



2. For the user you want to edit, enter the responsibility details in the Direct Responsibilities tab. Enter the responsibility name, application, security group (Standard), and effective dates.
3. Save your changes using the **Save** button on the toolbar, and then close the "Users dialog."

10.3.6 Setting Up a Profile

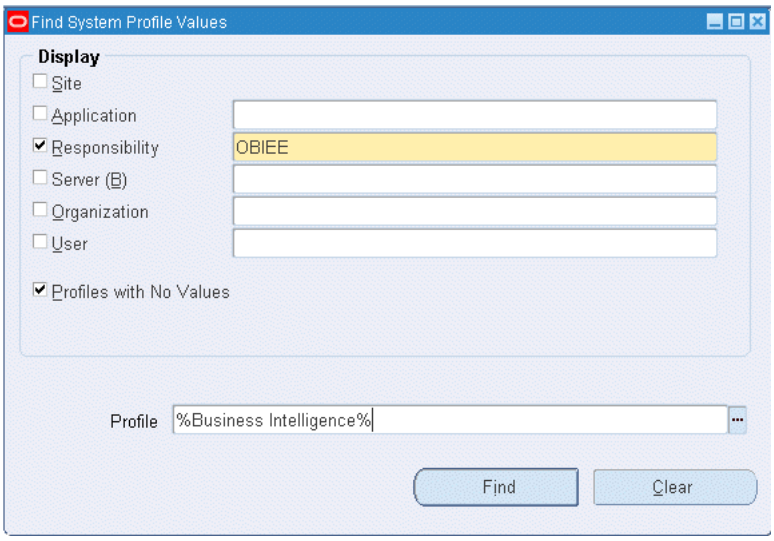
You need to enter the URL of the Oracle BI Server as part of a profile. You can set a profile for a responsibility, a user, or a site. The following procedure shows how to set profile options for a responsibility.

To set profile options for a responsibility:

1. From the **Application** menu, select **Profile**.

   Figure 10–7 shows the "Find System Profile Values dialog."

Figure 10–7 Find System Profile Values Dialog



2. Select **Responsibility**, and then enter the name of the responsibility to which you assigned the menu in Section 10.3.4, "Assigning the Menu to a Responsibility."
3. Enter %Business Intelligence% in the **Profile** field.
4. Click **Find**.
5. On the resulting screen, under **Responsibility**, enter the Oracle Business Intelligence URL. For example:

   ```
   http://my_server.domain.com:port/analytics
   ```

   You should use a fully-qualified host name.domain name rather than an IP address or just a host name. The Oracle Business Intelligence domain needs to be the same as the Oracle E-Business Suite domain. This is required so that the EBS ICX cookie will be visible to Oracle Business Intelligence from the user's browser. For more information, see "Domain Prerequisites".

   For port, enter the web server port where Oracle Business Intelligence is running (for example, 9704).
6. Save your changes using the **Save** button on the toolbar.

11

Embedding Oracle BI EE In Oracle's Siebel CRM

This chapter describes how to embed Oracle BI EE in Oracle's Siebel CRM. Describes configuring the Oracle HTTP server's proxy settings, configuring the Siebel application's HTTPS server, and configuring Siebel's symbolic URLs to locate the Oracle BI EE directory.

This chapter includes the following sections:

- Section 11.1, "Overview of Embedding Oracle BI EE in Oracle's Siebel CRM"
- Section 11.2, "Configuring Oracle HTTP Server"
- Section 11.3, "Configuring the Siebel Application to Find Oracle BI Through HTTP Server"
- Section 11.4, "Modifying the Siebel URLs to Reference the /analytics Directory"

11.1 Overview of Embedding Oracle BI EE in Oracle's Siebel CRM

If, for example, you plan to run Oracle's Siebel Web Extension (SWE) and Oracle Business Intelligence Presentation Services on different web servers, you must use some kind of networking or load balancing mechanism to create a single logical domain (or virtual IP address) for the two machines. One way to achieve this is to install a supported HTTP Server such as Oracle HTTP Server. You then configure the HTTP server against both the Siebel CRM application (SWE) and the entry point to Oracle Business Intelligence. In this way, you configure a reverse proxy that makes it appear to client browsers that both Oracle's Siebel CRM and Oracle Business Intelligence are being served from the same machine.

After modifying the HTTP server's proxy settings, you must update Oracle's Siebel CRM administration settings to map the HTTP server to the NQHOST and NQHOSTHOME variables. Finally, you must confirm that each URL contains a properly formatted reference to the Oracle BI EE analytics directory.

11.2 Configuring Oracle HTTP Server

Use this procedure to configure Oracle HTTP Server's proxy settings to set up a reverse proxy. This is so the client browser thinks the Oracle BI EE and Oracle's Siebel CRM applications are on the same machine. The HTTP server routes requests from the client browser to the appropriate machine. Note that the HTTP server must run on the same port at Oracle's Siebel CRM server.

This procedure assumes that you are using Oracle HTTP Server, but you can use any supported HTTP server to embed Oracle BI EE in Oracle's Siebel CRM.

If you are working in a production environment, Oracle recommends that you add security policies to the HTTP server so that users cannot directly access the underlying business intelligence objects.

For information about installing Oracle HTTP Server, see *Oracle Fusion Middleware Quick Installation Guide for Oracle Web Tier*. For information about configuring and securing Oracle HTTP Server, see *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

To configure the Oracle HTTP Server's proxy settings

1. Open Oracle HTTP Server's mod_wl_ohs.conf file.
2. Insert the following proxy request information into the configuration file. For more information, see the following "Example".

```
ProxyRequest Off
<Proxy*>
  Order deny,allow
  Allow from all
</Proxy*>

Location/Oracle's Siebel CRM subdirectory
  ProxyPass http://location of Oracle's Siebel CRM server/
     Oracle's Siebel CRM subdirectory

  ProxyPassReverse http://location of Oracle's Siebel CRM server/
     Oracle's Siebel CRM subdirectory
/Location

Location/Oracle BI EE subdirectory
  ProxyPass http://location of Oracle BI EE server/
     Oracle BI EE subdirectory
  ProxyPassReverse http://location of Oracle BI EE server/
     Oracle BI EE subdirectory
/Location
```

3. Save and close the configuration file.
4. Test the configuration by using a fully-qualified name to log into Oracle's Siebel CRM through Oracle HTTP Server.

Example

Note the following proxy configuration example.

```
ProxyRequests Off
<Proxy *>
  Order deny,allow
  Allow from all
</Proxy>

<Location /sales_enu>
  ProxyPass http://myserver.mycompany.com/sales_enu
  ProxyPassReverse http://myserver.mycompany.com/sales_enu
</Location>

<Location /analytics>
  ProxyPass http://myserver:9704/analytics
  ProxyPassReverse http://myserver:9704/analytics
</Location>
```

11.3 Configuring the Siebel Application to Find Oracle BI Through HTTP Server

Use this procedure to update the Siebel application's settings to map the HTTP server to the NQHOST and NQHOSTHOME variables.

For more information about working with the administration settings, see the Siebel product documentation.

To configure the Siebel application

1. Navigate to the Siebel application's **Administration - Integration** tab.
2. Navigate to the **Host Administration** frame.
3. In the **Virtual Name** column, located NQHOST and NQHOSTNAME.
4. In the **Name** column, update the name to the host name of the HTTP server that is serving the Oracle BI Presentation Services. This is the location name that you specified in the Oracle HTTP Server's mod_wl_ohs.conf file. For information about this location, see "Configuring Oracle HTTP Server".

   Note that if the HTTP server is running on a port other than 80, then you need to include the port number.

11.4 Modifying the Siebel URLs to Reference the /analytics Directory

Use this procedure to confirm that each symbolic URL located in Oracle's Siebel CRM contains a properly formatted reference to the Oracle BI EE /analytics directory.

To modify the Siebel URLs

1. Navigate to the Siebel application's **Administration - Integration** tab.
2. Navigate to the **Symbolic URL Administration** frame.
3. Click Query and in the **URL** column, search for *Analytics*. The search returns a list of URLs that match the search criteria.
4. Modify any business intelligence object URLs that contain /Analytics (with an upper case "A") to /analytics (with a lower case "a").

A

Sample Files

This appendix contains the contents of sample files that enable you to configure actions and to enable actions to invoke browser script calls.

This appendix contains the following section:

- Section A.1, "Sample Action Framework Configuration File"
- Section A.2, "Sample Java Script File"

A.1 Sample Action Framework Configuration File

Use the ActionFrameworkConfig.xml sample file with information given in Chapter 5, "Using Actions to Integrate Oracle BI EE with External Systems" to understand the various ways to configure actions.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <obi-action-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="afconfig.xsd">
- <aliases>
 - <location-alias>
    <alias>actionsrv</alias>
    <actual>localhost:9704</actual>
   </location-alias>
 - <location-alias>
    <alias>biserver</alias>
    <actual>http://localhost:9704/analytics/saw.dll?WSDL</actual>
   </location-alias>
   </aliases>
- <registries>
 - <registry>
    <id>reg1</id>
    <name>Sample Web Services</name>
    <content-type>webservices</content-type>
    <provider-class>oracle.bi.action.registry.wsil.WSILRegistry</provider-class>
    <description />
  - <location>
     <path>http://localhost:9704/ActionSamples/inspection.wsil</path>
    </location>
  - <service-access>
     <path>/Sample Web Services/Rating Service</path>
     <policy>SAMLPolicy</policy>
     <propagateIdentity>true</propagateIdentity>
    </service-access>
   </registry>
 - <registry>
    <id>reg1b</id>
```

```
    <name>BI EE Web Services for SOA</name>
    <content-type>webservices</content-type>
    <provider-class>oracle.bi.action.registry.wsil.WSILRegistry</provider-class>
    <description />
  - <location>
     <path>http://localhost:9704/biservices/inspection?wsil</path>
    </location>
  - <service-access>
     <account>wsil.browsing</account>
     <policy>wss_username_token_policy</policy>
     <propagateIdentity>false</propagateIdentity>
     </service-access>
   </registry>
 - <registry>
     <id>reg1c</id>
     <name>Built-in BI EE Web Services</name>
     <content-type>webservices</content-type>
     <provider-class>oracle.bi.action.registry.wsil.WSILRegistry</provider-class>
     <description />
   - <location>
      <path>http://localhost:9704/ActionSamples/Builtin.wsil</path>
     </location>
    </registry>
  - <registry>
     <id>reg2</id>
     <name>Secure Web Services Sample</name>
     <content-type>webservices</content-type>
     <provider-class>oracle.bi.action.registry.wsil.WSILRegistry</provider-class>
     <description />
   - <location>
      <path>http://localhost:9704/ActionSamples/secure.wsil</path>
     </location>
   - <service-access>
      <path>/Secure Web Services
Sample/RatingService/CreditRatingService/CreditRatingPort/processRating</path>
     <policy>SAMLPolicy</policy>
     <propagateIdentity>true</propagateIdentity>
    </service-access>
   </registry>
 - <registry>
    <id>reg03</id>
    <name>Sample EJBs</name>
    <content-type>java</content-type>
    <provider-class>oracle.bi.action.registry.java.EJBRegistry</provider-class>
    <description>Custom Java classes which can be invoked as action
targets</description>
  - <location>
     <path />
    </location>
  - <custom-config>
   - <ejb-targets>
    - <appserver>
       <context-factory>weblogic.jndi.WLInitialContextFactory</context-factory>
       <jndi-url>t3://localhost:9704</jndi-url>
       <server-name>localhost</server-name>
       <account>WLSJNDI</account>
       <ejb-exclude>mgmt</ejb-exclude>
       <ejb-exclude>PopulationServiceBean</ejb-exclude>
      </appserver>
    - <ejb-app>
```

```
      <server>localhost</server>
      <app-context>ActionSamplesEJB</app-context>
     </ejb-app>
    </ejb-targets>
   </custom-config>
  </registry>
 </registries>
- <content-types>
 - <content-type>
    <typename>webservices</typename>
    <displayname>Web Services and BPEL Processes</displayname>
    <actionType>WebServiceActionType</actionType>
   </content-type>
 - <content-type>
    <typename>psft</typename>
    <displayname>PeopleSoft Applications</displayname>
    <actionType>URLActionType</actionType>
   </content-type>
 - <content-type>
    <typename>epm</typename>
    <displayname>Hyperion Applications</displayname>
    <actionType>URLActionType</actionType>
   </content-type>
 - <content-type>
    <typename>misc</typename>
    <displayname>Mixed Services</displayname>
    <actionType>URLActionType</actionType>
   </content-type>
 - <content-type>
    <typename>java</typename>
    <displayname>Java Actions</displayname>
    <actionType>JavaActionType</actionType>
   </content-type>
 </content-types>
- <accounts>
 - <account>
    <name>wsil.browsing</name>
    <description>Account for BI WS for SOA</description>
    <adminonly>false</adminonly>
    <credentialkey>browsing</credentialkey>
   </account>
 - <account>
    <name>WLSJNDI</name>
    <description>Account used to access WLS JNDI.</description>
    <adminonly>false</adminonly>
    <credentialkey>JNDIUser</credentialkey>
   </account>
 - <account>
    <name>EPM</name>
    <description>Account used to connect to EPM.</description>
    <adminonly>false</adminonly>
    <credentialkey>EPM</credentialkey>
   </account>
 </accounts>
- <policies>
 - <policy>
    <name>SAMLPolicy</name>
    <policyfile>ActionsSAMLPolicy.xml</policyfile>
   </policy>
 - <policy>
```

```
    <name>wss_username_token_policy</name>
    <policyfile>wss_username_token_policy.xml</policyfile>
   </policy>
  </policies>
- <!--
Uncomment this element to enable support for using proxy settings to enable
browsing/invocation
of external web services when running behind a firewall

  <proxy>
     <host>@proxy.host@</host>
     <port>@proxy.port@</port>
     <userid>@proxy.user@</userid>
     <password>@proxy.password@</password>
     <nonProxyHosts>localhost|*.oracle.com|@proxy.nonProxyHosts@</nonProxyHosts>
  </proxy>



  -->
- <ebusinesssuiteconfig>
    <visible>true</visible>
   </ebusinesssuiteconfig>
 </obi-action-config>
```

A.2 Sample Java Script File

Use the UserScripts.js sample file with information given in Chapter 5, "Using Actions to Integrate Oracle BI EE with External Systems" to understand how to enable actions to invoke browser script calls.

```
/
----------------------------------------------------------------------------------
---------------------------------
/** USER-DEFINED SCRIPTS
  *
  * Use this file to enter any user-defined JavaScript functions you wish to
invoke from a client-side Script action.
  * By using the namespace of USERSCRIPT (as shown in the example code below), you
will ensure that no function name
  * conflicts occur with with the product code, and all PUBLISHED functions (in
the USERSCRIPT namespace) can be
  * browsed for selection, at Action creation time.
  *
  * By default, all functions in here are PRIVATE, in so much that they are not
displayed within the Browse dialog
  * during Action creation.
  * Note - They ARE still usable, providing you know the correct function name.
  *
  * To publish (make visible) a function, simply add the 'publish' static member
as shown in the example below, along
  * with any parameters you want automatically created.
  *
  * ALL functions will receive ONE parameter; a named array of values, indexed by
the parameter name.
  */
//
----------------------------------------------------------------------------------
---------------------------------
```

```
USERSCRIPT = function(){};

//
--------------------------------------------------------------------------------
---------------------------------
/** This is the parameter object you can create to supply default parameters on
creation of Script action.
  * See the 'displayParameters' example below for usage.
  * @param {String} sName is the unique name of parameter.
  * @param {String} sPrompt is the display text used to prompt for the parameter
value.
  * @param {String} sValue (Optional) is the default value for the parameter.
  */
USERSCRIPT.parameter = function(sName, sPrompt, sValue)
{
   this.name   = sName;
   this.prompt = sPrompt;
   this.value  = sValue;
};

/** This is an example function to display all parameters that are recieved
  * @params {Array} aParams an array of values indexed by the parameter name
  */
USERSCRIPT.example_displayParameters = function(aParams)
{
   var sArgs = "";
   for( args in aParams )
   {
      var argName  = args;
      var argValue = aParams[argName];

      sArgs += "Parameter name: " + argName + "  Value: " + argValue;
      sArgs += "\n";
   }

   alert( sArgs.length == 0 ? "No Parameters" : sArgs );
};
USERSCRIPT.example_displayParameters.publish =
{
   // The existence of this 'publish' object causes the 'USERSCRIPT.example_
displayParameters' function to be
   // shown when browing the available user script functions (during creation of a
Script action).

   // If you wish the Script function to have parameters automatically created on
selection of the function,
   // create a 'parameters' object as shown below.
   // You can have any number of parameters, with each parameter requiring a
unique name, prompt and an
   // optional value.
   parameters :
   [
      new USERSCRIPT.parameter( 'p1', 'Enter value for Param 1', 'p1 default
value' ),
      new USERSCRIPT.parameter( 'p2', 'Enter value for Param 2', 'p2 default
value' ),
      new USERSCRIPT.parameter( 'p3', 'Enter value for Param 3'  )
   ]
```

```
   // If no generated parameters are required, either create an empty array
   //    parameters : []
   // or don't declare the 'parameters' object at all.
};
//global variables
var newWin;
var sArgs;


//
------------------------------------------------------------------------------
------------
/** This is an example of a private function, which is not shown when browsing
available
  * user-defined functions.  It is private as is does NOT have the 'publish'
static member declaration.
  * However, the function CAN still be called if you know the function name.
  */
USERSCRIPT.example_hideMeDuringBrowse = function()
{
   alert("But I can still be called");
};



//
------------------------------------------------------------------------------
------------
```

Glossary

This glossary defines terms for Oracle Business Intelligence Enterprise Edition. See also the Oracle Fusion Middleware Master Glossary for additional terms and definitions.

action

Provides functionality to navigate to related content or to invoke operations, functions or processes in external systems. You can include actions in analyses, dashboard pages, agents, scorecard objectives, scorecard initiatives, and KPIs.

See also **action link**.

Action Framework

The Action Framework is a component of the Oracle BI EE architecture and includes a J2EE application called the Action Execution Service (AES) and actions-specific JavaScript functionality deployed as part of Oracle BI EE. The action framework also includes client-side functionality for creating actions and invoking certain action types directly from the browser.

action link

A link to an action that you have embedded in an analysis, dashboard page, scorecard objective, scorecard initiative, or KPI that, when clicked, runs an associated action.

See also **action**.

ADF Business Intelligence Component

Provides the developer the ability to include Oracle BI Presentation Catalog objects in ADF Applications. This component uses a SOAP connection to access the catalog.

Administration Server

Part of the WebLogic server domain and runs the processes that manage Oracle Business Intelligence components. The Administration Server includes the Oracle WebLogic Server Administration Console, Oracle Fusion Middleware Control, and JMX MBeans. For a Simple Install type, the Administration Server also includes Java components for Oracle Business Intelligence such as Oracle BI Publisher and Oracle Real-Time Decisions.

See also **Fusion Middleware Control**, **Java components** and **Managed Server**.

Administration Tool

See **Oracle BI Administration Tool**.

advanced trellis

A trellis view that can display multiple visualization types within its grid, for example, Spark Line graphs, Spark Bar graphs, and numbers. Each visualization type displays a different measure.

You can think of an advanced trellis as a pivot table, except that for each measure you add to the pivot table, you can optionally associate a dimension and render that dimension as a spark graph visualization.

See also **trellis** and **visualization**.

agent

Enables you to automate your business processes. You can use agents to provide event-driven alerting, scheduled content publishing, and conditional event-driven action execution.

Agents can dynamically detect information-based problems and opportunities, determine the appropriate individuals to notify, and deliver information to them through a wide range of devices (email, phones, and so on).

aggregate persistence

A feature that automates the creation and loading of aggregate tables and their corresponding Oracle Business Intelligence metadata mappings to enable aggregate navigation.

aggregate table

A table that stores precomputed results from measures that have been aggregated over a set of dimensional attributes. Each aggregate table column contains data at a given set of levels. For example, a monthly sales table might contain a precomputed sum of the revenue for each product in each store during each month. Using aggregate tables optimizes performance.

aggregation rule

In an Oracle BI repository, a rule applied to a logical column or physical cube column that specifies a particular aggregation function to be applied to the column data, such as `SUM`.

In Presentation Services, users can see the rules that have been applied in the repository. Users can also change the default aggregation rules for measure columns.

alias table

A physical table that references a different physical table as its source. You can use alias tables to set up multiple tables, each with different keys, names, or joins, when a single physical table must serve in different roles. Because alias table names are included in physical SQL queries, you can also use alias tables to provide meaningful table names, making the SQL statements easier to read.

analysis

A query that a user creates on the Criteria tab in Presentation Services. An analysis can optionally contain one or more filters or selection steps to restrict the results.

See also **filter** and **selection step**.

analysis criteria

Consists of the columns, filters, and selection steps that you specify for an analysis.

See also **analysis**.

analysis prompt

A prompt that is added to an analysis. When the user selects a prompt value, that value then determines the content that displays in the analysis that contains the prompt, only.

See **dashboard prompt** and **prompt**.

attribute

The details of a dimension in an Oracle BI repository. Attributes usually appear as columns of a dimension table.

attribute column

In Presentation Services, a column that holds a flat list of values that are also known as members. No hierarchical relationship exists between these members, as is the case for members of a hierarchical column. Examples include ProductID or City.

See **hierarchical column**.

BI Composer

BI Composer is a simple-to-use wizard that enables you to quickly and easily create, edit, or view analyses without the complexities of the Analysis editor.

BI domain

Contains configurable system components (the coreapplication) and Java components (the WebLogic server domain), and includes the web-based management tools and applications that use resources.

A BI domain can be a set of middleware homes spread across one or more physical servers.

See also **BI instance**.

BI instance

Refers to the system components (coreapplication) of a BI domain

See also **BI domain**.

BI object

A piece of business intelligence content that is created with Presentation Services and saved to the Oracle BI Presentation Catalog. Examples of BI objects include analyses, dashboards, dashboard pages, scorecards, and KPIs.

BI Search

A search tool that resides outside of Presentation Services. BI Search is available from the Home Page after the administrator adds a link to the BI Search URL. BI Search provides a mechanism for searching for objects in the Oracle BI Presentation Catalog that is similar to a full-text search engine.

bookmark link

Captures the path to a dashboard page and all aspects of the page state.

See **prompted link**.

bridge table

A table that enables you to resolve many-to-many relationships between two other tables.

briefing book

See **Oracle BI Briefing Books**.

business model

An object in the Oracle BI repository that contains the business model definitions and the mappings from logical to physical tables. Business models are always dimensional, unlike objects in the Physical layer, which reflect the organization of the data sources. Each business model contains logical tables, columns, and joins.

Business Model and Mapping layer

A layer of the Oracle BI repository that defines the business, or logical, model of the data and specifies the mapping between the business model and the Physical layer schemas. This layer can contain one or more business models.

The Business Model and Mapping layer determines the analytic behavior that is seen by users, and defines the superset of objects available to users. It also hides the complexity of the source data models.

business owner

The person responsible for managing and improving the business value and performance of a KPI or scorecard object, such as an objective, cause & effect map, and so on.

catalog

See **Oracle BI Presentation Catalog**.

cause & effect map

A component of a scorecard that lets you illustrate the cause and effect relationships of an objective or KPI.

See also **Oracle Scorecard and Strategy Management**.

chronological key

A column in a time dimension that identifies the chronological order of the members within a dimension level. The key must be unique at its level.

Cluster Controller

A process that serves as the first point of contact for new requests from Presentation Services and other clients. The Cluster Controller determines which Oracle BI Server in the cluster to direct the request to based on Oracle BI Server availability and load. It monitors the operation of servers in the cluster, including the Oracle BI Scheduler instances. The Cluster Controller is deployed in active-passive configuration.

column

In an Oracle BI repository, columns can be physical columns, logical columns, or presentation columns.

In Presentation Services, indicates the pieces of data that an analysis returns. Together with filters and selection steps, columns determine what analyses contain. Columns also have names that indicate the types of information that they contain, such as Account and Contact.

See also **analysis**, **attribute column**, **hierarchical column**, and **measure column**.

column filter

See **filter**.

column prompt

A type of filter that enables you to build specific value prompts on a data column to either exist alone on the dashboard or analysis or to expand or refine existing dashboard and analysis filters.

See also **prompt**.

complex join

A join in the Physical layer of an Oracle BI repository that uses an expression other than equals.

condition

Objects that return a single Boolean value based on the evaluation of an analysis or of a key performance indicator (KPI). You use conditions to determine whether agents deliver their content and execute their actions, whether actions links are displayed in dashboard pages, or whether sections and their content are displayed in dashboard pages.

See also **action**, **action link**, **agent** and **key performance indicator (KPI)**.

connection pool

An object in the Physical layer of an Oracle BI repository that contains the connection information for a data source.

See also **Physical layer**.

content designer

The user who creates business intelligence objects such as analyses, dashboards, and scorecards.

contextual event action

A predelivered action that uses the Action Framework to pass content from the business intelligence object to another region on an ADF page.

See also **action**, **Action Framework**, and **action link**.

criteria

See **analysis criteria**.

cube

An OLAP (online analytical processing) data structure that lets data be analyzed more quickly and with greater flexibility than structures in relational databases. Cubes are made up of measures and organized by dimensions. Cubes in multidimensional data sources roughly correspond to star schemas in relational database models.

currency prompt

A prompt that enables the user to change the currency type that displays in the currency columns on an analysis or dashboard.

See also **prompt**.

custom view

A component of a scorecard that lets you show a customized view of your business and strategy data.

See also **Oracle Scorecard and Strategy Management**.

dashboard

An object that provides personalized views of corporate and external information. A dashboard consists of one or more pages. Pages can display anything that you can access or open with a web browser, such as results of analyses, images, alerts from agents, and so on.

dashboard prompt

A prompt that is added to the dashboard. When the user selects a prompt value, that value then determines the content that displays in all analyses that are included on the dashboard.

See **analysis prompt** and **prompt**.

Dashboard URL

Used for incorporating or referencing the content of a specific dashboard in external portals or applications. It has several forms and optional arguments that you can use to control its behavior.

data source name (DSN)

A data structure that contains the information about a specific database, typically used by an ODBC driver to connect to the database. The DSN contains information such as the name, directory, and driver of the database.

Connection pool objects in the Physical layer of the Oracle BI repository contain DSN information for individual data sources.

database hint

Instructions placed within a SQL statement that tell the database query optimizer the most efficient way to execute the statement. Hints override the optimizer's execution plan, so you can use hints to improve performance by forcing the optimizer to use a more efficient plan. Hints are supported only for Oracle Database data sources.

dimension

A hierarchical organization of logical columns (attributes). One or more logical dimension tables might be associated with at most one dimension.

A dimension might contain one or more (unnamed) hierarchies. There are two types of logical dimensions: dimensions with level-based hierarchies (structure hierarchies), and dimensions with parent-child hierarchies (value hierarchies).

A particular type of level-based dimension, called a time dimension, provides special functionality for modeling time series data.

See also **hierarchy**.

dimension table

A logical table that contains columns used by a particular dimension. A dimension table cannot be a fact table.

See also **fact table**.

driving table

A mechanism used to optimize the manner in which the Oracle BI Server processes multi-database joins when one table is very small (the driving table) and the other table is very large.

DSN

See **data source name (DSN)**.

event polling table

Event polling tables (also called event tables) provide information to the Oracle BI Server about which physical tables have been updated. They are used to keep the query cache up-to-date. The Oracle BI Server cache system polls the event table, extracts the physical table information from the rows, and purges stale cache entries that reference those physical tables.

Essbase

A multidimensional database management system available from Oracle that provides a multidimensional database platform upon which to build business intelligence applications. Also referred to as Oracle's Hyperion Essbase.

fact table

In an Oracle BI repository, a logical table in the Business Model and Mapping layer that contains measures and has complex join relationships with dimension tables.

See also **dimension table**.

filter

Criteria that are applied to attribute and measure columns to limit the results that are displayed when an analysis is run. For measure columns, filters are applied before the query is aggregated and affect the query and thus the resulting values.

See also **prompt** and **selection step**.

focus node

The center circle of a strategy contribution wheel that represents the starting objective of the diagram.

See also **strategy contribution wheel** and **focus trail**.

focus trail

A series of chained circles that represent the node in the center of the strategy contribution wheel and any of its ancestors that are included in the diagram. Each circle displays the status color of its corresponding node.

See also **strategy contribution wheel** and **focus node**.

foreign key

A column or a set of columns in one table that references the primary key columns in another table.

fragmentation content

The portion, or fragment, of the set of data specified in a logical table source when the logical table source does not contain the entire set of data at a given level. Fragmentation content is defined by the logical columns that are entered in the **Fragmentation content** box in the Content tab of the Logical Table Source dialog box.

Fusion Middleware Control

Provides web-based management tools that enable you to monitor and configure Fusion Middleware components.

global header

An object in the user interface for Oracle BI Presentation Services that contains links and options that enable the user to quickly begin a task or locate a specific object within the Oracle BI Presentation Catalog. The global header always displays in the Presentation Services user interface, thus enabling users to quickly access links and search the catalog without having to navigate to the Home Page or Catalog page.

Go URL

Used to incorporate specific business intelligence results into external portals or applications. The Go URL is used when you add a result to your favorites or add a link to a request to a dashboard or external web site. It has several forms and optional arguments that you can use to control its behavior.

hierarchical column

In Presentation Services, a column that holds data values that are organized using both named levels and parent-child relationships. This column is displayed using a tree-like structure. Individual members are shown in an outline manner, with lower-level members rolling into higher-level members. For example, a specific day belongs to a particular month, which in turn is within a particular year. Examples include Time or Geography.

See also **attribute column**.

hierarchy

In an Oracle BI repository, a system of levels in a logical dimension that are related to each other by one-to-many relationships. All hierarchies must have a common leaf level and a common root (all) level.

Hierarchies are not modeled as separate objects in the metadata. Instead, they are an implicit part of dimension objects.

See also **dimension**, **logical level**, and **presentation hierarchy**.

hierarchy level

In Presentation Services, an object within a hierarchical column that either rolls up or is rolled up from other levels. Corresponds to a presentation level in an Oracle BI repository.

See also **presentation level**.

home page

Provides an intuitive, task-based entry way into the functionality of Presentation Services. The Home page is divided into sections that enable you to quickly begin specific tasks, locate an object, or access technical documentation.

image prompt

A prompt that provides an image with different areas mapped to specific values. The user clicks an image area to select the prompt value that populates the analysis or dashboard.

See also **prompt**.

initialization block

Used to initialize dynamic repository variables, system session variables, and nonsystem session variables. An initialization block contains the SQL statements that are executed to initialize or refresh the variables that are associated with that block.

initiative

Used in a scorecard, an initiative is a time-specific task or project that is necessary to achieve objectives. As such, you can use initiatives that support objectives as milestones as they reflect progress toward strategy targets.

See also **objective** and **Oracle Scorecard and Strategy Management**.

inner graph

A nested graph, inside the grid of a trellis graph. Each inner graph has its own dimensionality as specified in the Visualization area of the Layout pane.

See also **trellis** and **outer edge**.

Java components

Fusion Middleware Control components that are deployed as one or more Java EE applications (and a set of resources) and are managed by Node Manager.

See also **Node Manager**.

key performance indicator (KPI)

A measurement that defines and tracks specific business goals and strategic objectives. KPIs often times roll up into larger organizational strategies that require monitoring, improvement, and evaluation. KPIs have measurable values that usually vary with time, have targets to determine a score and performance status, include dimensions to allow for more precise analysis, and can be compared over time for trending purposes and to identify performance patterns.

See also **Oracle Scorecard and Strategy Management**.

KPI watchlist

A method of distributing KPIs to end users. A watchlist is a collection of KPIs that are built by adding the KPIs that are stored in the Oracle BI Presentation Catalog. After a KPI watchlist is built and saved, it is stored as a catalog object and can be added to dashboards and scorecards.

See also **key performance indicator (KPI)**, **watchlist**, and **Oracle Scorecard and Strategy Management**.

level

See **hierarchy level**.

logical display folder

Folders used to organize objects in the Business Model and Mapping layer of an Oracle BI repository. They have no metadata meaning.

logical join

Joins that express relationships between logical tables. Logical joins are *conceptual*, rather than physical, joins. In other words, they do not join to particular keys or columns. A single logical join can correspond to many possible physical joins.

logical layer

See **Business Model and Mapping layer**.

logical level

In an Oracle BI repository, a component of a level-based hierarchy that either rolls up or is rolled up from other levels.

Parent-child hierarchies have implicit, inter-member levels between ancestors and descendants that are not exposed as logical level objects in the metadata. Although parent-child hierarchies also contain logical level objects, these levels are system generated and exist to enable aggregation across all members only.

See also **dimension** and **hierarchy**.

Logical SQL

The SQL statements that are understood by the Oracle BI Server. The Oracle BI Server Logical SQL includes standard SQL, plus special functions (SQL extensions) like `AGO`, `TODATE`, `EVALUATE`, and others.

Clients like Presentation Services send Logical SQL to the Oracle BI Server when a user makes a request. In addition, Logical SQL is used in the Business Model and Mapping layer to enable heterogeneous database access and portability. The Oracle BI Server transforms Logical SQL into physical SQL that can be understood by source databases.

logical table

A table object in the Business Model and Mapping layer of an Oracle BI repository. A single logical table can map to one or more physical tables. Logical tables can be either fact tables or dimension tables.

See also **dimension table** and **fact table**.

logical table source

Objects in the Business Model and Mapping layer of an Oracle BI repository that define the mappings from a single logical table to one or more physical tables. The physical to logical mapping can also be used to specify transformations that occur between the Physical layer and the Business Model and Mapping layer, and to enable aggregate navigation and fragmentation.

Managed Server

An individual J2EE application container (JMX MBean container). It provides local management functions on individual hosts for Java components and system components contained within the local middleware home, and refers to the Administration Server for all of its configuration and deployment information.

See also **Administration Server** and **Fusion Middleware Control**.

MDS

Oracle Metadata Services. A core technology of the Application Development Framework. MDS provides a unified architecture for defining and using metadata in an extensible and customizable manner.

See also **MDS XML**.

MDS XML

An XML format that is compatible with Oracle Metadata Services. MDS XML is a supported format for the Oracle BI repository. It enables integration with third-party source control management systems for offline repository development.

MDS XML format is different from the XML format generated by the Oracle BI Server XML API.

See also **MDS**, **Oracle BI repository**, and **Oracle BI Server XML API**.

measure column

A column that can change for each record and can be added up or aggregated. Typical measures are sales dollars and quantity ordered. Measures are calculated from data sources at query time.

Measure columns are displayed in the Oracle BI repository, usually in fact tables, or in Presentation Services.

metadata

Data about data. Metadata objects include the descriptions of schemas (such as tables, columns, data types, primary keys, foreign keys, and so on) and logical constructs (like fact tables, dimensions, and logical table source mappings).

The Oracle BI repository is made up of the metadata used by the Oracle BI Server to process queries.

metadata dictionary

A static set of XML documents that describe metadata objects, such as a column, including its properties and relationships with other metadata objects. A metadata dictionary can help users obtain more information about metrics or attributes for repository objects.

microchart

A tiny graph displayed in a grid along with other tiny graphs and numbers, comprising the data cell contents of an advanced trellis view. In Oracle BI EE, a microchart is always a spark graph.

See also **advanced trellis** and **spark graph**.

mission statement

A statement in a scorecard that specifies the key business goals and priorities that are required to achieve your vision.

See also **vision statement** and **Oracle Scorecard and Strategy Management**.

multi-database join

A join between two tables in an Oracle BI repository, where each table resides in a different database.

Node Manager

A daemon process that provides remote server start, stop, and restart capabilities when Java processes become unresponsive or terminate unexpectedly.

See also **Java components**.

OCI

See **Oracle Call Interface (OCI)**.

ODBC

See **Open Database Connectivity (ODBC)**.

object properties

Information about an object and attributes that the owner can assign to an object. Examples of properties include name, description, date stamps, read-only access, and do not index flag.

See also **permissions**.

objective

A required or desired outcome in a scorecard that forms your corporate strategy.

See also **initiative** and **Oracle Scorecard and Strategy Management**.

offline mode

In the Oracle BI Administration Tool, a mode where a repository builder can edit a repository that is not loaded into the Oracle BI Server.

online mode

In the Oracle BI Administration Tool, a mode where a repository builder can edit a repository while it is available for query operations. Online mode also allows user session monitoring for users connected to the subject areas in the repository.

opaque view

A Physical layer table that consists of a `SELECT` statement. In the Oracle BI repository, opaque views appear as view tables in the physical databases, but the view does not actually exist.

Open Database Connectivity (ODBC)

A standard interface used to access data in both relational and nonrelational databases. Database applications can use ODBC to access data stored in different types of database management systems, even if each database uses a different data storage format and programming interface.

OPMN

See **Oracle Process Manager and Notification Server (OPMN)**.

Oracle BI Administration Tool

A Windows application that is used to create and edit Oracle BI repositories. The Administration Tool provides a graphical representation of the three parts of a repository: the Physical layer, the Business Model and Mapping layer, and the Presentation layer.

Oracle BI Briefing Books

A collection of static or updatable snapshots of dashboard pages, individual analyses, and BI Publisher reports. You can download briefing books in PDF or MHTML format for printing and viewing. You also can update, schedule, and deliver briefing books using agents.

Oracle BI JavaHost

A service that gives Presentation Services the ability to use functionality that is provided in Java libraries to support components such as graphs. The services are provided based on a request-response model.

Oracle BI Logical SQL View Object

Provides the developer the ability to create a Logical SQL statement to access the Oracle BI Server and fetch business intelligence data and bind it to native ADF components for inclusion on an ADF page. This view object uses a BI JDBC connection to the Oracle BI Server.

Oracle BI Presentation Catalog

Stores business intelligence objects, such as analyses and dashboards, and provides an interface where users create, access, and manage objects, and perform specific object-based tasks (for example, export, print, and edit). The catalog is organized into folders that are either shared or personal.

Oracle BI Presentation Services

Provides the framework and interface for the presentation of business intelligence data to web clients. It maintains a Presentation Catalog service on the file system for the customization of this presentation framework. It is a standalone process and communicates with the Oracle BI Server using ODBC over TCP/IP. It consists of components that are known as Answers, Delivers, and Interactive Dashboards.

See also **ODBC**; **Oracle BI Server**; **Oracle BI Presentation Catalog**; **Oracle BI Presentation Services server**.

Oracle BI Presentation Services server

The Oracle BI web server that exchanges information and data with the Oracle BI Server.

Oracle BI Publisher

A J2EE application that provides enterprise-wide publishing services in Oracle Business Intelligence. It generates highly formatted, pixel-perfect reports.

See also **report**.

Oracle BI Publisher report

See **report**.

Oracle BI repository

The set of Oracle Business Intelligence metadata that defines logical schemas, physical schemas, physical-to-logical mappings, aggregate table navigation, and other constructs. Oracle BI repositories can be in binary (RPD) format, in which repository metadata is contained in a single file with an extension of .rpd, or in a set of MDS XML documents. MDS XML format repositories are used for offline development only and cannot be loaded into the Oracle BI Server. Oracle BI repositories in both formats can be edited using the Oracle BI Administration Tool.

See also **metadata** and **Oracle BI Administration Tool**.

Oracle BI Scheduler

An extensible scheduling application for scheduling results to be delivered to users at specified times. It is the engine behind the Oracle BI Delivers feature.

See also **results**.

Oracle BI Server

A standalone process that maintains the logical data model that it provides to Presentation Services and other clients through ODBC. Metadata is maintained for the

data model in a local proprietary file called the repository file. The Oracle BI Server processes user requests and queries underlying data sources.

Oracle BI Server XML API

Provides utilities to create a generic, XML-based representation of the Oracle BI repository metadata. You can use this XML file version of the repository to programmatically modify the metadata. The Oracle BI Server XML API objects correspond to metadata repository objects in an RPD file. These objects differ from XML objects in the Oracle BI Presentation Catalog.

The XML generated by the Oracle BI Server XML API is different from the MDS XML format used for Oracle BI repositories integrated with third-party source control management systems.

See also **MDS XML**.

Oracle Business Intelligence Mobile

Oracle Business Intelligence Mobile allows you to view Oracle BI EE content on supported mobile devices such as the Apple iPhone and Apple iPad.

Using Oracle Business Intelligence Mobile, you can view and analyze BI content such as analyses and dashboards, BI Publisher content, scorecard content, and content delivered by agents.

Oracle Business Intelligence Web Services

See Oracle Business Intelligence Session-Based Web Services and Oracle Business Intelligence Web Services for SOA.

Oracle Business Intelligence Session-Based Web Services

An API that implements SOAP. These web services are designed for programmatic use, where a developer uses one web service to invoke many different business intelligence objects. These web services provide functionality on a wide range of Presentation Services operations. These web services enable the developer to extract results from Oracle BI Presentation Services and deliver them to external applications, perform Presentation Services management functions, and execute Oracle Business Intelligence alerts (known as Intelligent Agents).

See also Oracle Business Intelligence Web Services for SOA.

Oracle Business Intelligence Web Services for SOA

Contains three web services, ExecuteAgent, ExecuteAnalysis, and ExecuteCondition, which are hosted by the bimiddleware J2EE application. These web services are designed to enable developers to use third-party web services clients (for example, Oracle SOA Suite) to browse for and include business intelligence objects in service oriented architecture components.

See also Oracle Business Intelligence Session-Based Web Services.

Oracle Call Interface (OCI)

A connection interface that the Oracle BI Server can use to connect to Oracle Database data sources. You should always use OCI when importing metadata from or connecting to an Oracle Database.

Oracle OLAP

Oracle Database has an OLAP Option that provides an embedded, full-featured online analytical processing server.

Oracle Business Intelligence supports Oracle OLAP as a data source. When you import metadata from an Oracle OLAP source, the Oracle OLAP objects appear in the Physical layer of the Administration Tool. Oracle OLAP objects include Analytic Workspaces, which are containers for storing related cubes.

Oracle Process Manager and Notification Server (OPMN)

A process management tool that manages all system components (server processes), and supports both local and distributed process management, automatic process recycling and the communication of process state (up, down, starting, stopping). OPMN detects process unavailability and automatically restarts processes).

See also **system components**.

Oracle Scorecard and Strategy Management

A performance management tool that lets you describe and communicate your business strategy. You can drive and assess your corporate strategy and performance from the top of your organization down, or from the bottom up.

Oracle Technology Network (OTN)

A repository of technical information about Oracle's products where you can search for articles, participate in discussions, ask the user community technical questions, and search for and download Oracle products and documentation.

outer edge

The outer edges are the parts of a trellis view that border the inner graphs. These include the column and row headers, the section headers, and so on.

See also **trellis** and **inner graph**.

parent-child hierarchy

A hierarchy of members that all have the same type. All the dimension members of a parent-child hierarchy occur in a single data source. In a parent-child hierarchy, the inter-member relationships are parent-child relationships between dimension members.

See also **dimension**.

parent-child relationship table

A table with values that explicitly define the inter-member relationships in a parent-child hierarchy. Also called a closure table.

pass-through calculation

A calculation that is not computed by the Oracle BI Server but instead is passed to another data source. Enables advanced users to leverage data source features and functions without the need to modify the Oracle BI repository.

performance tile

A view type that displays a single aggregate measure value in a manner that is both visually simple and prominent, yet it immediately reveals summary metrics to the user that will likely be presented in more detail within a dashboard view. Performance tile views communicate status through simple formatting by using color, labels, and limited styles, or through conditional formatting of the background color or measure value to make the tile visually prominent.

permissions

Specify which users can access an object, and limit how users can interact with an object. Examples of permissions include write, delete, and change permissions.

See **object properties**.

perspective

A category in your organization with which to associate initiatives, objectives, and KPIs in a scorecard. A perspective can represent a key stakeholder (such as a customer, employee, or shareholder/financial) or a key competency area (such as time, cost, or quality).

See also **initiative**, **key performance indicator (KPI)**, **objective**, and **Oracle Scorecard and Strategy Management**.

physical catalog

An object in the Physical layer of an Oracle BI repository that groups different schemas. A catalog contains all the schemas (metadata) for a database object.

physical display folder

Folders that organize objects in the Physical layer of an Oracle BI repository. They have no metadata meaning.

physical join

Joins between tables in the Physical layer of an Oracle BI repository.

Physical layer

A layer of the Oracle BI repository that contains objects that represent physical data constructs from back-end data sources. The Physical layer defines the objects and relationships available for writing physical queries. This layer encapsulates source dependencies to enable portability and federation.

physical schema

An object in the Physical layer of an Oracle BI repository that represents a schema from a back-end database.

physical table

An object in the Physical layer of an Oracle BI repository, usually corresponding to a table that exists in a physical database.

See also **Physical layer**.

point of view area

An area within KPIs and scorecards that shows data of specific interest to you, such as the area of business for which you are responsible. The point of view area displays controls for the dimensions of KPIs that are used in the scorecard to measure the progress and performance of initiatives and objectives.

See also **initiative**, **key performance indicator (KPI)**, and **objective**.

presentation hierarchy

An object in the Presentation layer of an Oracle BI repository that provides an explicit way to expose the multidimensional model in Presentation Services and other clients. Presentation hierarchies expose analytic functionality such as member selection,

custom member groups, and asymmetric queries. Users can create hierarchy-based queries using presentation hierarchies.

In Presentation Services, presentation hierarchies are displayed as hierarchical columns.

See also **hierarchical column** and **presentation level**.

Presentation layer

Provides a way to present customized, secure, role-based views of a business model to users. It adds a level of abstraction over the Business Model and Mapping layer in the Oracle BI repository. The Presentation layer provides the view of the data seen by users who build analyses in Presentation Services and other client tools and applications.

See also **Business Model and Mapping layer**.

presentation level

In the Oracle BI repository, a component of a presentation hierarchy that either rolls up or is rolled up from other levels. Presentation levels are displayed as levels within hierarchical columns in Presentation Services.

See also **hierarchy level** and **presentation hierarchy**.

Presentation Services

See **Oracle BI Presentation Services**.

Presentation Services server

See **Oracle BI Presentation Services server**.

presentation table

An object in the Presentation layer of an Oracle BI repository that is used to organize columns into categories that make sense to the user community. A presentation table can contain columns from one or more logical tables. The names and object properties of the presentation tables are independent of the logical table properties.

primary key

A column (or set of columns) where each value is unique and identifies a single row of a table.

process instance

A unique process on an individual workstation that is associated with a BI instance.

See also **BI instance**.

prompt

A type of filter that enables the content designer to build and specify data values or the end user to choose specific data values to provide a result sets for an individual analysis or multiple analyses included on a dashboard or dashboard page. A prompt expands or refines existing dashboard and analysis filters.

The types of prompts are column prompts, currency prompts, image prompts, and variable prompts.

See also **column prompt**, **currency prompt**, **filter**, **image prompt**, and **variable prompt**.

prompted link

Captures the path to a dashboard page and a simplified presentation of the dashboard prompt.

See **bookmark link**.

query

Contains the underlying SQL statements that are issued to the Oracle BI Server. You do not have to know a query language to use Oracle Business Intelligence.

query cache

A facility to store query results for use by other queries.

ragged hierarchy

See **unbalanced hierarchy**.

report

The response returned to the user from the execution of a query created using Oracle BI Publisher. Reports can be formatted, presented on a dashboard page, saved in the Oracle BI Presentation Catalog, and shared with other users.

See also **analysis**.

repository

See **Oracle BI repository**.

repository variable

See **variable**.

results

The output returned from the Oracle BI Server for an analysis.

See also **analysis**.

scorecard

See **Oracle Scorecard and Strategy Management**.

selection step

A choice of values that is applied after the query is aggregated that affects only the members displayed, not the resulting aggregate values. Along with filters, selection steps restrict the results for an analysis.

See also **analysis** and **filter**.

session variable

See **variable**.

simple trellis

A trellis view that displays inner visualizations that are all the same type, such as all scatter graphs. The inner visualizations all use a common axis, also known as a synchronized scale.

See also **trellis**, **synchronized scale**, and **visualization**.

skip-level hierarchy

A hierarchy where some members do not have a value for a particular ancestor level. For example, in the United States, the city of Washington in the District of Columbia does not belong to a state. The expectation is that users can still navigate from the country level (United States) to Washington and below without the need for a state.

See also **hierarchy**.

smart watchlist

A smart watchlist is a view into a particular scorecard based on criteria that you specify. For example, a smart watchlist might show the top ten KPIs in a scorecard based on best performance or all the objectives, initiatives, and KPIs in a scorecard that are owned by a specific business owner.

See also **watchlist** and **Oracle Scorecard and Strategy Management**.

snowflake schema

A dimensional schema where one or more of the dimensions are partially or completely normalized.

spark graph

An embedded mini-graph that, in conjunction with other mini-graphs and numbers, illustrates a single trend. Spark graphs are also known as *sparks*.

Sparks do not include axes or labels; they get their context from the content that surrounds them. Each type of spark graph has only one measure, which is hidden; the scale is relative to itself only.

A spark graph can be of the graph subtype Spark Line, Spark Bar, or Spark Area.

See also **microchart**.

SQL

See **structured query language (SQL)**.

star schema

A relational schema that allows dimensional analysis of historical information. Star schemas have one-to-many relationships between the logical dimension tables and the logical fact table. Each star consists of a single fact table joined to a set of denormalized dimension tables.

strategy contribution wheel

A component of a scorecard that makes it easy to see the contribution (or impact) a specific objective or KPI has on a parent objective in a series of concentric rings (or wheel diagram). You use the strategy contribution wheel diagram to hierarchically view an objective and its supporting child objectives and KPIs.

See also **Oracle Scorecard and Strategy Management**.

strategy map

A component of a scorecard that shows how the objectives that have been defined for a scorecard and the KPIs that measure their progress are aligned by perspectives. It also shows cause and effect relationships.

See also **Oracle Scorecard and Strategy Management**.

strategy tree

A component of a scorecard that shows an objective and its supporting child objectives and KPIs hierarchically in a tree diagram.

See also **Oracle Scorecard and Strategy Management**.

structured query language (SQL)

A standard programming language for querying and modifying data.

See also **Logical SQL**.

subject area

In an Oracle BI repository, an object in the Presentation layer that organizes and presents data about a business model. It is the highest-level object in the Presentation layer and represents the view of the data that users see in Presentation Services. Oracle BI repository subject areas contain presentation tables, presentation columns, and presentation hierarchies.

In Presentation Services, subject areas contain folders, measure columns, attribute columns, hierarchical columns, and levels.

synchronized scale

(Applicable to simple trellis only) A synchronized scale means that all the visualizations within the trellis are viewed on the same scale, that is, they share a common axis. Having a common axis makes all graph markers easy to compare across rows and columns.

See also **simple trellis** and **visualization**.

system components

Server processes (not Java applications) that are managed by the Oracle Process Manager and Notification server (OPMN).

See also **Oracle Process Manager and Notification Server (OPMN)**.

transformation

Work that is performed on data when moving from a database to another location (sometimes another database). Some transformations are typically performed on data when it is moved from a transaction system to a data warehouse system.

trellis

Displays multidimensional data shown as a set of cells in a grid, where each cell represents a subset of data using a particular graph type. Data can be represented with graphs, microcharts, and numbers.

The trellis view has two subtypes: simple trellis and advanced trellis.

See also **advanced trellis** and **simple trellis**.

unbalanced hierarchy

A hierarchy where the leaves do not have the same depth. For example, an organization might choose to have data for the current month at the day level, data for the previous year at the month level, and data for the previous five years at the quarter level.

See also **hierarchy**.

value hierarchy

See **parent-child hierarchy**.

variable

Objects in an Oracle BI repository that are used to streamline administrative tasks and dynamically modify metadata content to adjust to a changing data environment.

Variables are of the following types:

- Repository variables have a single value at any point in time. There are two types of repository variables: static and dynamic.
- Session variables are created and assigned a value when each user logs on. There are two types of session variables: system and nonsystem.

variable prompt

Enables the user to select a value specified in the variable prompt to display on the dashboard. A variable prompt is not dependent upon column data, but enables you to manipulate, for example add or multiply, the column data on an analysis.

See also **prompt**.

virtual physical table

A physical table that is made from a stored procedure or a `SELECT` statement. Creating virtual tables can provide the Oracle BI Server and the underlying databases with the proper metadata to perform some advanced query requests.

vision statement

A short statement in a scorecard that describes what your organization wants to become sometime in the future. For example, it might be to become the most successful business in the South America Polypropylene Market.

See also **mission statement** and **Oracle Scorecard and Strategy Management**.

visualization

In the context of Oracle BI EE, a visualization is the choice of graph that appears within a data cell in a trellis view. There are many visualizations from which to choose when creating a trellis view, including bar graphs, scatter graphs, and spark graphs.

See also **trellis**.

watchlist

A table that lists scorecard objects (that is, initiatives, objectives, and KPIs) that are related to a particular aspect of a scorecard or that are grouped together for a particular purpose. There are different types for watchlists for example, KPI watchlists or smart watchlists.

See also **KPI watchlist**, **smart watchlist**, and **Oracle Scorecard and Strategy Management**.

waterfall graph

A graph type that lets you visualize how a value increases or decreases sequentially and cumulatively. Waterfall graphs focus the user's attention on how each measure contributes to the overall total and communicate through simple formatting by using color.

WebLogic server domain

Contains Java components that are configured to participate in the servicing of SOAP, HTTP, and other forms of requests.

WebLogic Scripting Tool (WLST)

A command-line scripting interface that enables you to configure, manage, and persist changes to WebLogic Server instances and domains and to monitor and manage server runtime events.

XML API

See **Oracle BI Server XML API**.

Index

Symbols

_import() method, 3-21

A

Access (Microsoft), example of integrating with Oracle Business Intelligence, 6-11
AccessControlToken structure, 2-2
account element, 5-11
Account structure, 2-2
ACL structure, 2-3
Action Framework
  about, 5-1
  adding and maintaining credentials, 5-16
  and credentials, 5-15
  and functionality provided, 5-2
  and security, 5-14
  and the account element, 5-11
  and the content type element, 5-10
  and the ebusinesssuiteconfig element, 5-14
  and the location alias element, 5-5
  and the policy element, 5-12
  and the proxy element, 5-13
  and the registry element, 5-6
  configuration checklist, 5-3
  configuration overview, 5-3
  configuring, 5-5
ActionFrameworkConfig.xml, 5-5
actions
  about, 5-2
  and permissions, 5-15
  and privileges, 5-15
  and security, 5-14
  and targets, 5-4, 5-19
  configuring OWSM for, 5-18
  execution of, 5-2
  Invoke a Browser Script, 5-29
  Invoke a Java Method (EJB), 5-26
  Invoke a Server Script, 5-32
  Invoke a Web Service, 5-22
  Invoke Agent, 5-34
  Java Job, 5-36
  Navigate to E-Business Suite, 5-21
  Navigate to EPM Content, 5-19
  types of, 5-2
addReportToPage() method, 3-4
AggregationRule Values, 2-11
APIs
  Dashboard URL, using, 6-4
  Go URL
  Oracle Business Intelligence Systems Management, 7-1
  SQL, issuing using the Go command, 6-7
  third-party SQL tool, integrating example, 6-11
application programming interfaces
  See APIs
applyReportParams() method, 3-23, 3-24
ArrayofGUIDS structure, 2-3
authentication, setting up for Oracle E-Business Suite, 10-2
authenticationschemas.xml file, updating for Oracle E-Business Suite integration, 10-4
AuthResult structure, 2-3

C

callback URLs
  modifying, 3-4
  replaced, 3-8
cancelJob() method, 3-13
cancelQuery() method, 3-43
CatalogItemsFilter structure, 2-4
CatalogObject structure, 2-4
certification information, xiv
checklist for Action Framework configuration, 5-3
clearQueryCache() method, 3-17
ConditionService
  service, 3-1
configuring
  web services for SOA, 1-3
connectivity
  ODBC data source names, configuring, 8-4, 8-8
  query and reporting tools, about using, 8-1
content type element, 5-10
copyItem() method, 3-35
copyItem2() method, 3-36
createFolder() method, 3-36
createLink() method, 3-36
credential key, 5-16
credential map, 5-16
credentials

about, 5-15
adding and maintaining, 5-16

D

Dashboard URL
about and format, 6-4
basic Dashboard URL, about and format, 6-5
PortalPath parameter, using, 6-5
User ID and Password parameter, using, 6-5
Data Source Name, about, 8-2
databases,supported, xiv
default keystore, 5-17
deleteIBot() method, 3-10
deleteItem() method, 3-37
deleteResultSet() method, 3-13
describeColumn() method, 3-17
describeSubjectArea() method, 3-18
describeTable() method, 3-19
Discoverer metadata, 9-1
Discoverer worksheets, 9-1
drilldown links, 3-4
driver, for Oracle BI DSN, 8-4

E

EBS ICX authentication cookie, 10-4
ebusinesssuiteconfig element, 5-14
enabling
SSL for web services communication, 1-3
web services for SOA, 1-3
endPage() method, 3-5
error message language, setting in ODBC DSN, 8-6
ErrorDetailsLevel structure, 3-35
ErrorInfo structure, 2-4
evaluateCondition() method, 3-2
evaluateInlineCondition() method, 3-2
ExecuteAgent, 1-2
ExecuteAnalysis, 1-2
ExecuteCondition, 1-2
executeIBotNow() method, 3-10
executeSQLQuery() method, 3-43
executeXMLQuery() method, 3-44
execution of actions, 5-2
export() method, 3-22
ExportFlags structure, 3-20

F

fetchNext() method, 3-45
files, ActionFrameworkConfig.xml, 5-5
filters
Go URL, passing to through URL,, 6-7
forgetAccount() method, 3-29
form functions, creating in Oracle E-Business Suite, 10-7

G

generateReportSQL() method, 3-24
getAccounts() method, 3-30
getCommonBodyHTML() method, 3-5
getConditionCustomizableReportElements() method, 3-3
getCounts() method, 3-13
getCurUser() method, 3-25
getGlobalPrivilegeACL() method, 3-30
getGlobalSAWPrivileges() method, 3-30
getGroups() method, 3-31
getHeadersHTML() method, 3-6
getHtmlforPageWithOneReport() method, 3-6
getHTMLForReport() method, 3-7
getItemInfo() method, 3-37
getJobInfo() method, 3-14
getMembers() method, 3-31
getPermissions() method, 3-31
getPrivilegesStatus() method, 3-32
getPromptedColumns() method, 3-14
getPromptedFilters() method, 3-45
getSessionVariable() method, 3-26
getSubItems() method, 3-37
GetSubItemsParams structure, 2-5
getSubjectAreas() method, 3-19
Go URL
about, 6-1
application-friendly format, displaying results in, 6-3
basic Go URL, about and example, 6-2
filters, about passing to through URL, 6-7
HTML results, navigation from, 6-11
JavaScript, navigation using, 6-9
link options, format and example, 6-2
navigation example, 6-9
navigation parameters syntax, 6-8
printer friendly format and example, 6-3
result format, controlling, 6-4
specific style, showing results in, 6-4
specify view, showing, 6-3
SQL, using to issue and passing filters, 6-7
tables, displaying all records in, 6-4
user ID and password, prompting for, 6-2
GoNav JavaScript function, about, 6-9

H

HtmlViewService
bridging, 3-4
service, 3-3
HTTPS
invoking web services, 1-3

I

iBotService
service, 3-9
impersonate() method, 3-26
impersonateex() method, 3-27
import() method, 3-21
ImportError structure, 2-5
ImportFlags structure, 3-21
instanceconfig.xml file, updating for Oracle

E-Business Suite integration, 10-6
integrating Oracle Business Intelligence
Dashboard URL, using, 6-4
filters, passing to the Go command through a URL, 6-7
Go URL, using, 6-1
SQL, issuing using the Go command, 6-7
third-party SQL tool, integrating example, 6-11
with Oracle E-Business Suite security, 10-1
with other clients, 8-1
Invoke a Browser Script action, 5-29
Invoke a Java Method (EJB) action, 5-26
Invoke a Server Script action, 5-32
Invoke a Web Service action, 5-22
Invoke Agent action, 5-34
invoking
web services with HTTPS, 1-3
isMember() method, 3-32
ItemInfo structure, 2-6

J

Java Job action, 5-36
JDBC, using to integrate with the Oracle BI Server, 8-3
JobManagementService
service, 3-12
joinGroups() method, 3-32

K

keepAlive() method, 3-27
keystore, 5-17

L

leaveGroups() method, 3-33
location alias element, 5-5
logoff() method, 3-28
logon() method, 3-28
logonex() method, 3-28

M

maintenanceMode() method, 3-38
markForReplication() method, 3-22
memory requirements, xiv
menus
assigning to responsibilities in Oracle E-Business Suite, 10-9
creating in Oracle E-Business Suite, 10-8
metadata conversion assistant tool, 9-1
metadata, Discoverer, 9-1
MetadataService
service, 3-17
method
getConditionCustomizableReportElements(), 3-3
methods
_import(), 3-21
addReportToPage(), 3-4
applyReportParams(), 3-23, 3-24
cancelJob(), 3-13
cancelQuery(), 3-43
clearQueryCache(), 3-17
copyItem(), 3-35
copyItem2(), 3-36
createFolder(), 3-36
createLink(), 3-36
deleteIBot(), 3-10
deleteItem(), 3-37
deleteResultSet(), 3-13
describeColumn(), 3-17
describeSubjectArea(), 3-18
describeTable(), 3-19
endPage(), 3-5
evaluateCondition(), 3-2
evaluateInlineCondition(), 3-2
executeIBotNow(), 3-10
executeSQLQuery(), 3-43
executeXMLQuery(), 3-44
export(), 3-22
fetchNext(), 3-45
forgetAccount(), 3-29
generateReportSQL(), 3-24
getAccounts(), 3-30
getCommonBodyHTML(), 3-5
getCounts(), 3-13
getCurUser(), 3-25
getGlobalPrivilegeACL(), 3-30
getGlobalSAWPrivileges(), 3-30
getGroups(), 3-31
getHeadersHTML(), 3-6
getHtmlforPageWithOneReport(), 3-6
getHTMLForReport(), 3-7
getItemInfo(), 3-37
getJobInfo(), 3-14
getMembers(), 3-31
getPermissions(), 3-31
getPrivilegesStatus(), 3-32
getPromptedColumns(), 3-14
getPromptedFilters(), 3-45
getSessionVariable(), 3-26
getSubItems(), 3-37
getSubjectAreas(), 3-19
impersonate(), 3-26
impersonateex(), 3-27
import, 3-21
isMember(), 3-32
joinGroups(), 3-32
keepAlive(), 3-27
leaveGroups(), 3-33
logoff(), 3-28
logon(), 3-28
logonex(), 3-28
maintenanceMode(), 3-38
markForReplication(), 3-22
moveIBot(), 3-10
moveItem(), 3-38
pasteItem2(), 3-39
prepareCache(), 3-14
purgeCache(), 3-15

purgeLog(), 3-23
readObject(), 3-39
removeFolder(), 3-40
renameAccounts(), 3-33
SASubjectAreaDetails(), 3-18
SATablesDetails(), 3-19
saveResultSet(), 3-15
sendMessage(), 3-11
sessionEnvironment(), 3-26
setBridge(), 3-7
setBridge(), using for callback URLs, 3-4
setItemAttributes(), 3-40
setItemProperty(), 3-41
setOwnership(), 3-41
startPage(), 3-8
subscribe(), 3-11
unsubscribe(), 3-12
updateGlobalPrivilegeACL(), 3-33
upgradeXML(), 3-45
writeIBot(), 3-9
writeListFiles(), 3-16
writeObjects(), 3-42
Microsoft
Access, example of integrating with Oracle Business Intelligence, 6-11
minimum disk space, xiv
moveIBot() method, 3-10
moveItem() method, 3-38

N

NameValuePair structure, 2-6
Navigate to E-Business Suite action, 5-21
Navigate to EPM Content action, 5-19

O

ODBC
about, 8-2
configuring advanced SSL settings, 8-7
creating dedicated connection to Physical layer, 8-2
data source names, configuring, 8-4, 8-8
localization considerations over, 8-2
query and reporting tools, about connecting with, 8-1
supported calls from client applications, 8-3
operating systems, supported, xiv
Oracle BI repository
configuring Physical layer for Oracle E-Business Suite integration, 10-1
setting session variables for Oracle E-Business Suite integration, 10-2
Oracle BI Server
about integrating with over JDBC, 8-3
about integrating with over ODBC, 8-2
Oracle Business Intelligence Systems Management API, 7-1
Oracle E-Business Suite
assigning menus to responsibilities, 10-9
assigning responsibilities to users, 10-10
configuring Oracle BI repository for, 10-1
creating form functions, 10-7
creating menus, 10-8
embedding links to Oracle Business Intelligence, 10-6
setting up profiles, 10-11
setting up shared authentication with, 10-2
OWSM, 5-18

P

pasteitem2() method, 3-39
PathMap structure, 2-7
permissions, 5-15
platforms, supported, xiv
policy element, 5-12
PortalNav JavaScript function, about, 6-9
prepareCache() method, 3-14
Privilege structure, 2-7
privileges, 5-15
profiles, setting up in Oracle E-Business Suite, 10-11
prompted URL
about, 6-1
proxy element, 5-13
purgeCache() method, 3-15
purgeLog() method, 3-23

Q

query tool, about connecting with, 8-1
QueryResults structure, 2-7

R

readObject() method, 3-39
ReadObjectReturnOptions
structure, 3-35
registry element, 5-6
removeFolder() method, 3-40
renameAccounts() method, 3-33
ReplicationService
service, 3-20
ReportEditingService
service, 3-23
ReportHTMLLinksMode structure, 2-8
ReportHTMLOptions structure, 2-8
reporting tool, about connecting with, 8-1
ReportParams structure, 2-8
ReportRef structure, 2-9
requirements, system, xiv
responsibilities, assigning to users in Oracle E-Business Suite, 10-10

S

SAColumn structure, 2-9
SADataType Values, 2-10
SASubjectArea structure, 2-11
SASubjectAreaDetails() method, 3-18
SATable structure, 2-11

SATablesDetails() method, 3-19
saveResultSet() method, 3-15
SAWLocale structure, 2-12
SAWSessionParameters structure, 2-12
SAWSessionService
  service, 3-25
securing
  web services for SOA, 1-3
SecurityService
  service, 3-29
SegmentationOptions structure, 2-13
SELECT_PHYSICAL, setting up ODBC connection for, 8-2
sendMessage() method, 3-11
services
  ConditionService, 3-1
  HtmlViewService, 3-3
  iBotService, 3-9
  JobManagementService, 3-12
  MetadataService, 3-17
  ReplicationService, 3-20
  ReportEditingService, 3-23
  SAWSessionService, 3-25
  SecurityService, 3-29
  WebCatalogService, 3-34
  XMLViewService, 3-42
session variables, setting for Oracle E-Business Suite integration, 10-2
session-based web services
  overview, 1-1
SessionEnvironment structure, 2-13
sessionEnvironment() method, 3-26
setBridge() method, 3-4, 3-7
setItemAttributes() method, 3-40
setItemProperty() method, 3-41
setOwnership() method, 3-41
shared authentication, with Oracle E-Business Suite, 10-2
single sign-on, limitations when using EBS-ICX authentication cookie, 10-2
SOA
  web services for SOA, 1-2
SQL
  example of integrating third-party tool, 6-11
  using Go command to issue SQL and passing filters, 6-7
SSL
  enabling for web services communication, 1-3
SSL, configuring in ODBC DSN, 8-7
startPage() method, 3-8
StartPageParams structure, 2-14
structures, 2-1
  AccessControlToken, 2-2
  Account, 2-2
  ACL, 2-3
  ArrayofGUIDS, 2-3
  AuthResult, 2-3
  CatalogItemsFilter, 2-4
  CatalogObject, 2-4
  ErrorDetailsLevel, 3-35
  ErrorInfo, 2-4
  ExportFlags, 3-20
  GetSubItemsParams, 2-5
  ImportError, 2-5
  ImportFlags, 3-21
  ItemInfo, 2-6
  NameValuePair, 2-6
  PathMap, 2-7
  Privilege, 2-7
  QueryResults, 2-7
  ReadObjectReturnOptions, 3-35
  ReportHTMLLinksMode, 2-8
  ReportHTMLOptions, 2-8
  ReportParams, 2-8
  ReportRef, 2-9
  SAColumn, 2-9
  SASubjectArea, 2-11
  SATable, 2-11
  SAWLocale, 2-12
  SAWSessionParameters, 2-12
  SegmentationOptions, 2-13
  SessionEnvironment, 2-13
  StartPageParams, 2-14
  TreeNodePath, 2-14
  UpdateACLMode, 2-15
  UpdateACLParams, 2-14
  UpdateCatalogItemACL, 3-41
  UpdateCatalogItemACLParams, 2-15
  Variable, 2-15
  XMLQueryExecutionOptions, 2-16
  XMLQueryOutputFormat, 3-43
subscribe() method, 3-11
supported installation types, xiv
system requirements, xiv

T

targets for actions, 5-4, 5-19
third-party SQL tool, integrating example, 6-11
tool, metadata conversion assistant, 9-1
TreeNodePath structure, 2-14

U

UNIX, configuring ODBC data source names on, 8-8
unsubscribe() method, 3-12
UpdateACLMode structure, 2-15
UpdateACLParams structure, 2-14
UpdateCatalogItemACL structure, 3-41
UpdateCatalogItemACLParams structure, 2-15
updateGlobalPrivilegeACL() method, 3-33
upgradeXML() method, 3-45
URLS, callback, 3-4

V

Variable structure, 2-15

W

web services

session-based overview, 1-1
SOA overview, 1-2
WebCatalogService service, 3-34
Windows, configuring ODBC data source names on, 8-4
worksheets, Discoverer, 9-1
writeIBot() method, 3-9
writeListFiles() method, 3-16
writeObjects() method, 3-42
WSDL
session-based web services, 1-1
webservices for SOA, 1-2
WSIL
web services for SOA, 1-2

X

XMLQueryExecutionOptions structure, 2-16
XMLQueryOutputFormat structure, 3-43
XMLViewService service, 3-42