This document describes the Discoverer Web Services API. It also provides reference information to help you use the API to access Discoverer connections and workbooks.
Preface

Intended Audience
Documentation Accessibility
Oracle BI Discoverer Keyboard Navigation
Related Documents
Conventions
JGoodies License Agreement

1 Introducing the Discoverer Web Services API

What is the Simple Object Access Protocol?
What are the Oracle BI Discoverer Web Services?
What is the SOAP endpoint URL for Discoverer Web Services?
What is the WSDL format?
About authentication and authorization
What modes of connecting to Discoverer are supported?
About maintaining Discoverer Web Service sessions
About managing the Discoverer session pool size
What are the availability requirements?
About error messages
How do you diagnose problems?
What is required to invoke the Discoverer Web Services?
Defining a trusted user to access Discoverer Web Services
Verifying access to the Discoverer Web Services
Setting values for Discoverer Web Services using Oracle Fusion Middleware Control
Creating web service client stubs (web service proxy)
Setting the Session Maintain Property
Writing a client application to invoke web services using generated web service client stubs

What is a typical flow of events for accessing Discoverer Web Services API to return results?

Typical flow of events: Detailed task examples
Provide the credentials to invoke Discoverer Web Services
Use the identifier to invoke login()
Inspect the connection information by invoking getConnectionList()
Select a particular connection and invoke getFolderEntryList()
Select a particular worksheet by invoking getWorksheetList()
Get the layout of the worksheet by invoking getLayoutMetadata() ............................................... 1-10
Select the parameter metadata for a worksheet by invoking getParameterMetadata() .......... 1-10
Select a worksheet parameter and obtain its LOVs by invoking getParameterValueList(). 1-10
Submit a worksheet query (including parameters) by invoking submitWorksheetQuery() 1-11
Check the status of the query by invoking getQueryStatus()....................................................... 1-11
View worksheet data on completion of the query by invoking getWorksheetData() .......... 1-12
Log out, by invoking Logout() ..................................................................................................... 1-13

Example of a Java class that invokes Discoverer Web Services ................................................. 1-13

2 Discoverer Web Services API Reference

getConnectionList ......................................................................................................................... 2-1
getFolderEntryList ....................................................................................................................... 2-2
getLayoutMetaData ....................................................................................................................... 2-3
getParameterMetaData .................................................................................................................... 2-4
getParameterValueList .................................................................................................................. 2-4
getQueryStatus ............................................................................................................................. 2-5
getVersion ....................................................................................................................................... 2-5
getViewerURL ................................................................................................................................ 2-6
getWorksheetData .......................................................................................................................... 2-6
getWorksheetList ............................................................................................................................ 2-8
isSessionValid ................................................................................................................................. 2-8
login .................................................................................................................................................. 2-8
logout ............................................................................................................................................... 2-9
requestQueryCancel ....................................................................................................................... 2-9
submitWorksheetQuery .................................................................................................................. 2-10
Welcome to the Oracle Fusion Middleware User’s Guide for Oracle Business Intelligence Discoverer Web Services API!

Intended Audience

This guide is intended for developers of applications that use Discoverer data through Discoverer Web Services. Readers are assumed to have a working knowledge of Web services.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.
Oracle BI Discoverer Keyboard Navigation

Oracle BI Discoverer supports standard keyboard navigation. Standard keyboard navigation includes the use of the tab key, mnemonics (using the Alt key and the underlined character), and accelerators (such as Alt+F4 to exit a window).

Related Documents

You can access the documents referenced in this guide, and other information about Oracle Business Intelligence (for example, whitepapers, best practices, documentation updates, other collateral) on Oracle Technology Network at:

http://www.oracle.com/technology

Conventions

Conventions used in this manual are shown in the table below:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><strong>monospace</strong></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
<tr>
<td><code>&lt; &gt;</code></td>
<td>Angle brackets enclose user-supplied names or values.</td>
</tr>
<tr>
<td><code>[]</code></td>
<td>Square brackets enclose optional clauses from which you can choose one or none.</td>
</tr>
<tr>
<td>Menu name</td>
<td>Command</td>
</tr>
</tbody>
</table>

For more information about the Discoverer Web Service API methods, see "Discoverer Web Services API Reference".

JGoodies License Agreement

Oracle Business Intelligence includes the JGoodies software, whose License Agreement follows:

Copyright© 2003 JGoodies Karsten Lentzsch. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of JGoodies nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
Introducing the Discoverer Web Services API

This chapter introduces the Discoverer Web Services API and describes how to use the API. It contains the following topics:

- "What is the Simple Object Access Protocol?"
- "What are the Oracle BI Discoverer Web Services?"
- "What is the SOAP endpoint URL for Discoverer Web Services?"
- "What is the WSDL format?"
- "About authentication and authorization"
- "What modes of connecting to Discoverer are supported?"
- "About maintaining Discoverer Web Service sessions"
- "About managing the Discoverer session pool size"
- "What are the availability requirements?"
- "About error messages"
- "How do you diagnose problems?"
- "What is required to invoke the Discoverer Web Services?"
- "What is a typical flow of events for accessing Discoverer Web Services API to return results?"
- "Typical flow of events: Detailed task examples"
- "Example of a Java class that invokes Discoverer Web Services"

**What is the Simple Object Access Protocol?**

SOAP (Simple Object Access Protocol) is a World Wide Web Consortium (W3C) recommendation for an XML protocol for exchanging information on the Web.

**What are the Oracle BI Discoverer Web Services?**

The Oracle BI Discoverer Web Services are part of an Application Programming Interface (API) that enables a client to do the following:

- Obtain Discoverer connections, workbooks, and worksheets
- Execute worksheet queries
- Obtain worksheet content using the SOAP protocol (version 1.1 with JAX-WS/document wrapped format)
What is the SOAP endpoint URL for Discoverer Web Services?

The SOAP endpoint URL for Discoverer Web Services is
http(s)://<host>:<port>/discoverer/wsi.

What is the WSDL format?

The WSDL (Web Services Definition Language) format is an industry standard that
formally defines services and methods, and is used to define Discoverer Web Service
APIs. Proxy classes for the services can be generated automatically.

Notes

■ The WSDL can be accessed from
http(s)://<host>:<port>/discoverer/wsi?wsdl
■ This guide assumes that WSDL, SOAP technologies, and the method of generating
client code from WSDL is known to a web services developer.

About authentication and authorization

The Discoverer Web Services are accessible only by trusted users. You must obtain the
trusted user name and password set up by the Discoverer middle-tier administrator, to
use these credentials in your code.

For a code example that includes user credentials, see Example 1-4, "Set credentials to
access the protected web service".

For information about how to create trusted users, see Defining a trusted user to access
Discoverer Web Services.

What modes of connecting to Discoverer are supported?

The Discoverer Web Services support Oracle Single Sign-On, Oracle Applications
Single Sign-On, and public connections in the current release.

About maintaining Discoverer Web Service sessions

The Discoverer Web Services are stateful in nature. In other words, every instance of
the web service client stub needs to have a single HTTP session with the Discoverer
Web Service. This HTTP session can be used for all web service operations, for all
client application users. For every user, the client application needs to call the login
method, to inform the Discoverer Web Service about a new user session. Every user
session is associated with a dedicated Discoverer session on the server side which is
either created or reused from the Discoverer session pool. The associated Discoverer
session is used in all interactions with the user. The client application must ensure that
it calls the logout method after completing all operations for the user.

The HTTP session established between the web service client and the server is tracked
by cookies and is managed by the web service.

About managing the Discoverer session pool size

It is useful to know the Discoverer session pool size, because every login() call and
subsequent data fetching operation requires a Discoverer session, and every logout() call releases the session.
The maximum session pool size value can be configured (in configuration.xml) using Oracle Fusion Middleware Control. The Discoverer session clean-up operation runs periodically removing any stale or inactive DiscovererSession objects from the pool.

The SSousername, Connectionkey, Workbook, Worksheet, and Locale methods are all used to determine which Discoverer sessions to pick up, to optimize allocation of Discoverer sessions for new login requests.

**What are the availability requirements?**

The Discoverer Web Services can be distributed across a cluster and have no dependency on a single point of failure. Transparent failover is not supported; therefore in the event of a failure, each client must authenticate again to create new user sessions.

**About error messages**

A web service API call can result in an exception that must be handled by the client. Error messages are displayed using the locale that was selected during login.

**How do you diagnose problems?**

You can diagnose problems from log files and from web service exceptions. You can view log entries using Oracle Fusion Middleware Control. For more information, see Oracle Fusion Middleware Configuration Guide for Oracle Business Intelligence Discoverer.

**What is required to invoke the Discoverer Web Services?**

To invoke the Discoverer Web Services you must comply with the following:

- "Defining a trusted user to access Discoverer Web Services"
- "Verifying access to the Discoverer Web Services"
- "Setting values for Discoverer Web Services using Oracle Fusion Middleware Control"
- "Creating web service client stubs (web service proxy)"
- "Setting the Session Maintain Property"
- "Writing a client application to invoke web services using generated web service client stubs"

**Defining a trusted user to access Discoverer Web Services**

Only trusted users can access Discoverer Web Services. You create trusted users by using the WebLogic Server Administration Console. For more information about creating users, see the "Create Users" topic in the WebLogic Administration Console Online Help.
What is required to invoke the Discoverer Web Services?

---

**Note:** By default, the **Administrators** group is assigned a scoped security role for the Discoverer application. To provide access to Discoverer Web Services, you can add the new user to the **Administrators** group.

For more information about adding users to groups, see "Add users to groups topic" in the WebLogic Administration Console Online Help.

For the new user to access Discoverer Web Services, you must assign the Discoverer scoped security role to the new user. For more information about adding users to security roles, see the "Add users to roles" topic in the WebLogic Administration Console Online Help.

---

**Verifying access to the Discoverer Web Services**

Before you create a Java class to invoke the Discoverer Web Services you must ensure that the Discoverer Web Services are installed and configured.

To verify that the Discoverer Web Services are installed and configured, you access the endpoint URL. If you cannot access the URL, contact the Discoverer manager.

Navigate to the link `http://<host>:<port>/discoverer/wsi`.

You should be prompted for the user/password created in the earlier steps.

**Note:** You can also use the user name and password of the ‘weblogic’ user that was entered during installation (for more information, see your middle tier administrator).

---

**Setting values for Discoverer Web Services using Oracle Fusion Middleware Control**

The `maxSessions` setting for Discoverer Web Services should be configured using Fusion Middleware Control. This setting specifies the maximum number of Discoverer sessions that can be active at the same time (the recommended value is 20).

For more information about configuring Discoverer Web Services using Fusion Middleware Control, see the Oracle Fusion Middleware Configuration Guide for Oracle Business Intelligence Discoverer.

---

**Creating web service client stubs (web service proxy)**

You can obtain a set of proxy/client files by generating them from a set of web service client libraries. Oracle Web Services provide libraries for this purpose. For more information, see Oracle Fusion Middleware Developer’s Guide for Oracle Web Services.

To generate the Discoverer Web Service client from the Discoverer WSDL URL:

1. Display a Web browser.
2. Access the Discoverer WSDL URL:
   
   `http://<host>:<port>/discoverer/wsi?wsdl`

   For more information, see "What is the WSDL format?", and "Writing a client application to invoke web services using generated web service client stubs".

---

**Setting the Session Maintain Property**

You must set the value of the `SESSION_MAINTAIN_PROPERTY` in the web service library. The `SESSION_MAINTAIN_PROPERTY` specifies whether sessions are stateful,
What is required to invoke the Discoverer Web Services?

and because Discoverer Web Services sessions are stateful, this property must be set to True as follows:

```java
Map requestContext = ((BindingProvider)proxy).getRequestContext();
requestContext.put(BindingProvider.SESSION_MAINTAIN_PROPERTY, true);
```

**Note:** Each supported web service library has an equivalent `SESSION_MAINTAIN_PROPERTY` for maintaining stateful sessions. For more information, see your web service documentation.

Writing a client application to invoke web services using generated web service client stubs

The following code examples provide an illustration of a basic client application:

**Example 1–1 Define a Java class**

```java
class webserviceclient {
    public static void main(String[] args)
    {
    }
}
```

**Example 1–2 Instantiate the web service client stub**

```java
class webserviceclient {
    @WebServiceRef
    private static Wsi_Service wsi_Service;
    public static void main(String[] args)
    {
        wsi_Service = new Wsi_Service();
        wsiProxy proxy = wsi_Service.getWsi();
    }
}
```

**Example 1–3 Ensure the client maintains the session**

This is needed as the Discoverer web service is stateful.

```java
class webserviceclient {
    @WebServiceRef
    private static Wsi_Service wsi_Service;
    public static void main(String[] args)
    {
        wsi_Service = new Wsi_Service();
        wsiProxy proxy = wsi_Service.getWsi();
        Map requestContext = ((BindingProvider)proxy).getRequestContext();
        requestContext.put(BindingProvider.SESSION_MAINTAIN_PROPERTY, true);
    }
}
```

**Example 1–4 Set credentials to access the protected web service**

```java
class webserviceclient {
    @WebServiceRef
    private static Wsi_Service wsi_Service;
    public static void main(String[] args)
    {
    }
}
```
What is required to invoke the Discoverer Web Services?

```java
{  
    wsi_Service = new Wsi_Service();  
    wsiProxy proxy = wsi_Service.getWsi();  
    Map requestContext = ((BindingProvider)proxy).getRequestContext();  
    requestContext.put(BindingProvider.USERNAME_PROPERTY, "username");  
    requestContext.put(BindingProvider.PASSWORD_PROPERTY, "password");  
    requestContext.put(BindingProvider.SESSION_MAINTAIN_PROPERTY, true);  
}

Example 1–5  Set the web service endpoint

class webserviceclient
{
    @WebServiceRef
    private static Wsi_Service wsi_Service;  
    public static void main(String[] args)
    {  
        wsi_Service = new Wsi_Service();  
        wsiProxy proxy = wsi_Service.getWsi();  
        Map requestContext = ((BindingProvider)proxy).getRequestContext();  
        requestContext.put(BindingProvider.USERNAME_PROPERTY, "username");  
        requestContext.put(BindingProvider.PASSWORD_PROPERTY, "password");  
        requestContext.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, "http://host:port/discoverer/wsi");  
        requestContext.put(BindingProvider.SESSION_MAINTAIN_PROPERTY, true);  
    }
}

Example 1–6  Perform any web service operations on the stub

class webserviceclient
{
    @WebServiceRef
    private static Wsi_Service wsi_Service;  
    public static void main(String[] args)
    {  
        wsi_Service = new Wsi_Service();  
        wsiProxy proxy = wsi_Service.getWsi();  
        Map requestContext = ((BindingProvider)proxy).getRequestContext();  
        requestContext.put(BindingProvider.USERNAME_PROPERTY, "username");  
        requestContext.put(BindingProvider.PASSWORD_PROPERTY, "password");  
        requestContext.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, "http://host:port/discoverer/wsi");  
        requestContext.put(BindingProvider.SESSION_MAINTAIN_PROPERTY, true);  
        DisplayName displayname = new DisplayName();  
        displayname.setUser("DISPLAYNAME");  
        Identifier idfr = new Identifier();  
        idfr.setId("IDENTIFIER");  
        UserCredential uc = new UserCredential();  
        uc.setDisplayName(displayname);  
        uc.setIdentifier(idfr);  
        LocaleBean locale = new LocaleBean();  
        locale.setCountry("US");  
        locale.setLanguage("en");  
        locale.setVariant("" );  
        SessionKey sKey = proxy.login(uc, locale);  
}
```
What is a typical flow of events for accessing Discoverer Web Services API to return results?

The Discoverer Web Services API can be used by a client application to obtain XML data related to Discoverer connections and worksheets. A typical flow of events is suggested in the following flow (for more information, see "Typical flow of events: Detailed task examples"):

**Typical flow of events**

1. "Provide the credentials to invoke Discoverer Web Services"
2. "Use the identifier to invoke login()"
   This starts a user session and provides a valid session key.
3. "Inspect the connection information by invoking getConnectionList()"
   This returns a list of connections for clients to use for display and selection.
4. "Select a particular connection and invoke getFolderEntryList()"
   This returns a list of workbooks for user display and selection.
   In case of OLAP connections there can be folders along with workbooks.
5. "Select a particular worksheet by invoking getWorksheetList()"
   This returns just a list of worksheets with empty parameter information.
6. "Get the layout of the worksheet by invoking getLayoutMetadata()"
7. "Select the parameter metadata for a worksheet by invoking getParameterMetadata()"
8. "Select a worksheet parameter and obtain its LOVs by invoking getParameterValueList()"
   This returns the LOV data in chunks which is used by clients for user display and selection.
   **Note:** If the worksheet does not contain parameters, then the client should not invoke getParameterValueList().
9. "Submit a worksheet query (including parameters) by invoking submitWorksheetQuery()"
   The client performs a submit, passing parameters if required.
   A queryKey is obtained when a worksheet query request is submitted.
10. "Check the status of the query by invoking getQueryStatus()"
This returns the current status of the query for user display and selection.

11. "View worksheet data on completion of the query by invoking getWorksheetData()"

12. "Log out, by invoking Logout()"

Note: Queries must be executed sequentially, so that when the client application performs a single login call for a user, it must ensure that the query submission for a worksheet is made only after the previous query submission has yielded results.

Note: For OLAP worksheets, the XML can contain aggregate totals. In that case, if the API client creates a total for an OLAP worksheet with Aggregate totals, it would yield wrong results.

Typical flow of events: Detailed task examples
The following flow of events shows some typical Java class entries for each task example.

Provide the credentials to invoke Discoverer Web Services
The following code must be executed before you can successfully perform a login() API call.

```java
wsi_Service = new Wsi_Service();
Wsi proxy = wsi_Service.getWsi();
Map requestContext = ((BindingProvider)proxy).getRequestContext();
requestContext.put(BindingProvider.USERNAME_PROPERTY,"weblogic");
requestContext.put(BindingProvider.PASSWORD_PROPERTY,"weblogic");
requestContext.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,testEndpoint);
requestContext.put(BindingProvider.SESSION_MAINTAIN_PROPERTY,true);
```

Use the identifier to invoke login()
You must provide the user credentials to invoke the Discoverer Web Services before you can use the login() call. For more information, see "Provide the credentials to invoke Discoverer Web Services".

Note: The login() call must be invoked before invoking any other Discoverer Web Services.

```java
DisplayName displayname = new DisplayName();
displayname.setUser("DISPLAYNAME");
Identifier idfr = new Identifier();
idfr.setId("IDENTIFIER");

UserCredential uc = new UserCredential();
uc.setDisplayName(displayname);
uc.setIdentifier(idfr);

LocaleBean locale = new LocaleBean();
locale.setCountry("US");
locale.setLanguage("en");
locale.setVariant(""");

System.out.println("Invoke the login() WS API");
SessionKey sKey = proxy.login(uc, locale);
System.out.println("Session Key :"+sKey.getKey());
```
Note: To access SSO-based connections, you can provide the GUID and SSOUsername as the Identifier and DisplayName respectively.

Inspect the connection information by invoking getConnectionList()  
Once logged in, the user can invoke the getConnectionList() API to obtain a list of available connections as shown below:

```java
System.out.println("Invoke the getConnectionList() API");
ConnectionList cl = proxy.getConnectionList(sKey);
List conns = cl.getConnections();
System.out.println("Discoverer connections:");
for(int i=0; i<conns.size(); i++)
{
    System.out.println("Name:" + conns.get(i).getConnectionName().getName());
    System.out.println(" Key:" + conns.get(i).getConnectionKey().getKey());
    System.out.println(" Desc:" + conns.get(i).getConnectionDesc().getDesc());
}
```

Select a particular connection and invoke getFolderEntryList()  
Before you can use the getFolderEntryList() API, you must provide credentials to invoke the Discoverer Web Services, and have a valid session key (by performing a login() call). For more information, see "Provide the credentials to invoke Discoverer Web Services", and "Use the identifier to invoke login()".

```java
ConnectionKey cKey = new ConnectionKey();
cKey.setKey("CONNECTIONKEY");
FolderEntryPath fPath = new FolderEntryPath();
fPath.setPath("");
System.out.println("Invoke the getFolderEntryList() API");
FolderEntryList fList = proxy.getFolderEntryList(sKey, cKey, fPath);
List fEntries = fList.getFolderEntries();
for(int i=0; i<fEntries.size(); i++)
{
    System.out.println("Name:" + fEntries.get(i).getName().getName());
    System.out.println(" Path:" + fEntries.get(i).getPath().getPath());
    System.out.println(" Desc:" + fEntries.get(i).getDesc().getDesc());
    System.out.println(" Type:" + fEntries.get(i).getType().getType());
}
```

Notes:
- The folder path is an empty string for relational database connections. For OLAP connections you can specify the complete folder name.
- For a relational database connection, the Type is always "Workbook". In case of an OLAP connection, you can see "Folder" as Type.

Select a particular worksheet by invoking getWorksheetList()  
You can use the getWorksheetList() API to get a list of worksheets for a given workbook.

```java
WorkbookKey wbKey = new WorkbookKey();
```
wbKey.setConnKey(cKey);
wbKey.setKey("ANALYTIC_FUNCTION_EXAMPLES");

System.out.println("Invoke the getWorksheetList() API");
WorksheetList wsList = proxy.getWorksheetList(sKey, wbKey);
List wsheets = wsList.getWorksheets();
    for(int i=0; i<wsheets.size(); i++)
    {
        System.out.println(\"Name:\" +wsheets.get(i).getName().getName());
        System.out.println(\"    Key:\"+wsheets.get(i).getKey().getKey());
    }

The Key that you obtained from the above call can be used as the worksheet key for the subsequent web service calls.

Get the layout of the worksheet by invoking getLayoutMetadata()

This API provides information about the dimensions of a Discoverer worksheet.

WorksheetKey wsKey = new WorksheetKey();
wsKey.setWbKey(wbKey);
wsKey.setKey("ANALYTIC_FUNCTION_EXAMPLES/1817");

System.out.println("Invoke the getLayoutMetadata() API");
Layout layout = proxy.getLayoutMetadata(sKey, wsKey);
List dimensions = layout.getDimensions();
List measures = layout.getMeasures();
    for(int i=0; i<dimensions.size(); i++)
    {
        System.out.println("Dimension :\"+dimensions.get(i).getName());
    }
    for(int i=0; i<measures.size(); i++)
    {
        System.out.println("Measure :\"+measures.get(i).getName());
    }

Select the parameter metadata for a worksheet by invoking getParameterMetadata()

Use this API to check whether any parameter metadata is available for the worksheet.

System.out.println("Invoke the getParameterMetaData() API");
ParameterList pList = proxy.getParameterMetadata(sKey, wsKey);
List parameters = pList.getParameters();
    for(int i=0; i<parameters.size(); i++)
    {
        System.out.println("Name:\" + parameters.get(i).getName().getName());
        System.out.println("    Key:\"+parameters.get(i).getKey().getKey());
    }

This API returns an empty list if the worksheet does not contain any parameter.

Select a worksheet parameter and obtain its LOVs by invoking getParameterValueList()

ParameterKey pKey = new ParameterKey();
pKey.setKey("ANALYTIC_FUNCTION_EXAMPLES/1817/36211");
pKey.setWsKey( wsKey);
ParameterValueList pvList = proxy.getParameterValueList(sKey,pKey,new Integer(50));
List pvs = pvList.getParamValues();
Submit a worksheet query (including parameters) by invoking submitWorksheetQuery()

The parameters for this API are sessionKey, workSheetKey, parameterSelectList and QueryOption.

ParameterSelectList helps you to pass a list of selected parameters. The code below is for multiple parameters and multiple values for each parameter.

```java
ParameterValue[] pval = new ParameterValue[1];
pval[0] = new ParameterValue();
pval[0].setValue("Aladdin");
pval[0].setDescriptorKey(null);
pval[1] = new ParameterValue();
pval[1].setValue("A Few Good Men");
pval[1].setDescriptorKey(null);

ParameterSelect[] pselect = new ParameterSelect[1];
pselect[0] = new ParameterSelect();
pselect[0].setKey(pKey);
ArrayList pvallist = new ArrayList();
pvallist.add(pval[0]);
pvallist.add(pval[1]);
pselect[0].getValues().addAll(pvallist);
ParameterSelectList pselectList = new ParameterSelectList();
ArrayList pselectlist = new ArrayList();
pselectList.getSelParams().addAll(pselectlist);
```

The QueryOption helps you to control the properties of the result set. The supported result types are XMLROWSET, PDF, HTML, and XLS. For the XMLROWSET type, you can specify the number of rows per fetch.

From the QueryOptions that you specify, the applicable options are considered; the others are discarded.

```java
ResultType rt = new ResultType();
rt.setType("XMLROWSET");

QueryOption qo = new QueryOption();
qo.setChunkSize(1021);
qo.setNoOfrows(25);
qo.setResultType(rt);
qo.setUserCredential(uc);
```

The code below submits the query.

```java
QueryKey qKey = proxy.submitWorksheetQuery(sKey, wsKey, pselectList, qo);
System.out.println("Query Key : " + qKey.getKey());
```
Typical flow of events: Detailed task examples

String resultsReady = "Results Ready";
String cancelled = "Cancelled";
QueryStatus qs = null;
while(!status.equalsIgnoreCase(resultsReady) &&
!status.equalsIgnoreCase(cancelled))
{
 qs = proxy.getQueryStatus(sKey, qKey);
 status = qs.getStatus();
 System.out.println("Status :"+qs.getStatus());
 Thread.sleep(5);
}

View worksheet data on completion of the query by invoking getWorksheetData()

Once you get the query status as Ready, you can get the worksheet data using the
getWorksheetData() API. The code below shows how to get the worksheet data
for XMLROWSET result type.

if(qs!= null && qs.getStatus().equalsIgnoreCase(resultsReady))
{
 System.out.println("Invoking getWorksheetData() API");
 QueryResult qr = null;
 do
 { qr = proxy.getWorksheetData(sKey, qKey, false);
   String data = qr.getData();
   if (qo.getResultType().getType().matches("XMLROWSET") )
   { System.out.println("-----------Data block begin-----------");
      System.out.println(data);
      System.out.println("-----------Data block end-----------");
   }
   while(qr.isFinished()== false);
 }

The following code sample explains how to get the worksheet data for a binary data
type (for example, XLS).

if(qs!= null && qs.getStatus().equals(resultsReady))
{
 System.out.println("Invoking getWorksheetData() API");
 QueryResult qr = null;
 do
 { qr = proxy.getWorksheetData(sKey, qKey, false);
   String data = qr.getData();
   if(qo.getResultType().getType().matches("XLS") )
   { String opFilename = "/tmp/out.xls" ;
     else if
     Base64Decoder myBase64Decoder = new Base64Decoder();
     byte _myTempByteArray [] = null;
     FileOutputStream fout = null;
     String opFilename = "/tmp/out.xls" ;
     try
     {
       fout =new FileOutputStream(opFilename);
     }
     catch(FileNotFoundException e)
     { System.out.println("Error Opening Output File");
   }
Example of a Java class that invokes Discoverer Web Services

Introducing the Discoverer Web Services API

The following text is an example of a Java class (wsiClient.java) that might invoke Discoverer Web Services:

```java
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.util.List;
import java.util.Map;
import javax.xml.ws.BindingProvider;
import javax.xml.ws.WebServiceRef;
import jaxwstest.proxy.Wsi;
import jaxwstest.proxy.Wsi_Service;
import jaxwstest.proxy.types.ConnectionKey;
import jaxwstest.proxy.types.ConnectionList;
import jaxwstest.proxy.types.DisplayName;
import jaxwstest.proxy.types.FolderEntry;
import jaxwstest.proxy.types.FolderEntryList;
import jaxwstest.proxy.types.FolderEntryPath;
import jaxwstest.proxy.types.Identifier;
import jaxwstest.proxy.types.LocaleBean;
import jaxwstest.proxy.types.QueryKey;
import jaxwstest.proxy.types.QueryOption;
import jaxwstest.proxy.types.QueryResult;
import jaxwstest.proxy.types.QueryStatus;
import jaxwstest.proxy.types.ResultType;
import jaxwstest.proxy.types.SessionKey;
import jaxwstest.proxy.types.UserCredential;
import jaxwstest.proxy.types.WorkbookKey;
import jaxwstest.proxy.types.Worksheet;
import jaxwstest.proxy.types.WorksheetKey;
import jaxwstest.proxy.types.WorksheetList;
import oracle.net.www.Base64Decoder;

public class wsiClient {
    @WebServiceRef
    private static Wsi_Service wsi_Service;
    public static void main(String[] args) {
        try {
            // Create a new stub
```
Example of a Java class that invokes Discoverer Web Services

```java
wsi_Service = new Wsi_Service();
wsiProxy = new wsiProxy();

Map requestContext = ((BindingProvider)proxy).getRequestContext();
requestContext.put(BindingProvider.USERNAME_PROPERTY, "weblogic");
requestContext.put(BindingProvider.PASSWORD_PROPERTY, "weblogic");
requestContext.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, testEndpoint);
requestContext.put(BindingProvider.SESSION_MAINTAIN_PROPERTY, true);
// Add your code to call the desired methods

String version = proxy.getVersion();
System.out.println("Version =" + version);

DisplayName ssouser = new DisplayName();
ssouser.setUser("SSOUSER1");
Identifier idfr = new Identifier();
idfr.setId("IDENTIFIER");

UserCredential uc = new UserCredential();
uc.setDisplayName(ssouser);
uc.setIdentifier(idfr);
LocaleBean locale = new LocaleBean();
locale.setCountry("US");
locale.setLanguage("en");
locale.setVariant("");

System.out.println("Invoke the login() WS API");
SessionKey sKey = proxy.login(uc, locale);
System.out.println("Session Key :"+sKey.getKey());

System.out.println("-----------------------------");
// Get the list of connections accessible to this SSOUser
System.out.println("Invoke the getConnectionList() API");
ConnectionList cl = proxy.getConnectionList(sKey);
List conns = cl.getConnections();
System.out.println("Discoverer connections:");
for(int i=0; i<conns.size(); i++)
{
    System.out.println("Name:" + conns.get(i).getConnectionName().getName());
    System.out.println(" Key:" + conns.get(i).getConnectionKey().getKey());
    System.out.println(" Desc:" + conns.get(i).getConnectionDesc().getDesc());
}
System.out.println("-----------------------------");

System.out.println("Invoke the getFolderEntryList() API");
// Create a valid connectionKey object using one of the connection keys obtained
in the getConnectionList
ConnectionKey cKey = new ConnectionKey();
cKey.setKey("CONNECTIONKEY");
FolderEntryPath fPath = new FolderEntryPath();
fPath.setPath("");

System.out.println("Invoke the getFolderEntryList() API");
FolderEntryList fList = proxy.getFolderEntryList(sKey, cKey, fPath);
List fEntries = fList.getFolderEntries();
for(int i=0; i<fEntries.size(); i++)
{
    System.out.println("Name:" + fEntries.get(i).getName().getName());
    System.out.println(" Path:" + fEntries.get(i).getPath().getPath());
    System.out.println(" Desc:" + fEntries.get(i).getDesc().getDesc());
}
```

1-14  Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Discoverer Web Services API
Example of a Java class that invokes Discoverer Web Services

```java
System.out.println("Type:"+ fEntries.get(i).getType().getType());
} System.out.println("-----------------------------");
WorkbookKey wbKey = new WorkbookKey();
wbKey.setConnKey(cKey);
wbKey.setKey("ANALYTIC_FUNCTION_EXAMPLES");
System.out.println("Invoke the getWorksheetList() API");
WorksheetList wsList = proxy.getWorksheetList(sKey, wbKey);
List wsheets = wsList.getWorksheets();
for(int i=0; i<wsheets.size(); i++)
{
    System.out.println("Name:" +wsheets.get(i).getName().getName());
    System.out.println("    Key:"+wsheets.get(i).getKey().getKey());
}
System.out.println("-----------------------------");
WorksheetKey wsKey = new WorksheetKey();
wsKey.setWsKey(wsKey);
wsKey.setKey("ANALYTIC_FUNCTION_EXAMPLES/1817");
System.out.println("Invoke the getLayoutMetadata() API");
Layout layout = proxy.getLayoutMetadata(sKey, wsKey);
List dimensions = layout.getDimensions();
List measures = layout.getMeasures();
for(int i=0; i<dimensions.size(); i++)
{
    System.out.println("Dimension :"+dimensions.get(i).getName());
}
for(int i=0; i<measures.size(); i++)
{
    System.out.println("Measure :"+measures.get(i).getName());
}
System.out.println("-----------------------------");
System.out.println("Invoke the getParameterMetaData() API");
ParameterList pList = proxy.getParameterMetadata(sKey, wsKey);
List parameters = pList.getParameters();
for(int i=0; i<parameters.size(); i++)
{
    System.out.println("Name:"+ parameters.get(i).getName().getName());
    System.out.println("    Key:"+parameters.get(i).getKey().getKey());
}
System.out.println("-----------------------------");
ParameterKey pKey = new ParameterKey();
pKey.setKey("ANALYTIC_FUNCTION_EXAMPLES/1817/36211");
pKey.setWsKey(wsKey);
ParameterValueList pvList = proxy.getParameterValueList(sKey,pKey,new Integer(50));
List pvs = pvList.getParamValues();
System.out.println("Invoke the getParameterValueList() API");
System.out.println("Parameter Lovs:");
for(int i=0; i<pvs.size(); i++)
{
    System.out.println("Val :"+pvs.get(i).getValue()); // Lov
    System.out.println(" DescriptorKey :"+pvs.get(i).getDescriptorKey());
```

Introducing the Discoverer Web Services API 1-15
Example of a Java class that invokes Discoverer Web Services

```java
// parameter descriptor key. This is valid if the parameter is an indexed type;

ParameterValue[] pval = new ParameterValue[1];
pval[0] = new ParameterValue();
pval[0].setValue("Aladdin");
pval[0].setDescriptorKey(null);
pval[1] = new ParameterValue();
pval[1].setValue("A Few Good Men");
pval[1].setDescriptorKey(null);

ParameterValue[] pval = new ParameterValue[1];
pval[0] = new ParameterValue();
pval[0].setValue("Aladdin");
pval[0].setDescriptorKey(null);
pval[1] = new ParameterValue();
pval[1].setValue("A Few Good Men");
pval[1].setDescriptorKey(null);

ParameterSelect[] pselect = new ParameterSelect[1];
pselect[0] = new ParameterSelect();
pselect[0].setKey(pKey);
ArrayList pvallist = new ArrayList();
pvallist.add(pval[0]);
pvallist.add(pval[1]);
pselect[0].getValues().addAll(pvallist);
ParameterSelectList pselectList = new ParameterSelectList();
pselectlist.add(pselect[0]);
pselectList.getSelParams().addAll(pselectlist);

ResultType rt = new ResultType();
rt.setType("XMLROWSET");

QueryOption qo = new QueryOption();
qo.setChunkSize(1024);
qo.setNoOfrows(25);
qo.setResultType(rt);
qo.setUserCredential(uc);

QueryKey qKey = proxy.submitWorksheetQuery(sKey, wsKey, pselectList, qo);
System.out.println("Query Key :" + qKey.getKey());

String status="";
String resultsReady = "Results Ready";
String cancelled = "Cancelled";
QueryStatus qs = null;
while(!status.equalsIgnoreCase(resultsReady) &&
  !status.equalsIgnoreCase(cancelled))
{
    qs = proxy.getQueryStatus(sKey, qKey);
    status = qs.getStatus();
    System.out.println("Status :" + qs.getStatus());
    Thread.sleep(5);
}
if(qs!= null && qs.getStatus().equalsIgnoreCase(resultsReady))
{
    System.out.println("Invoking getWorksheetData() API");
    QueryResult qr = null;
do
    {
      qr = proxy.getWorksheetData(sKey, qKey, false);
      String data = qr.getData();
      if (qo.getResultType().getType().matches("XMLROWSET"))
      {
        System.out.println("------------Data block begin------------");
        System.out.println(data);
        System.out.println("------------Data block end------------");
      }
    }
    System.out.println("----------Data block begin----------");
    System.out.println(data);
    System.out.println("----------Data block end----------");
```
Example of a Java class that invokes Discoverer Web Services

```java
while(qr.isFinished() == false);
proxy.logout(sKey);
catch(Exception ex){
ex.printStackTrace();
}
```

Introducing the Discoverer Web Services API
This chapter provides detailed reference information for the following Discoverer Web Services API.

- "getConnectionList"
- "getFolderEntryList"
- "getLayoutMetaData"
- "getParameterMetaData"
- "getParameterValueList"
- "getQueryStatus"
- "getVersion"
- "getViewerURL"
- "getWorksheetData"
- "getWorkSheetList"
- "isSessionValid"
- "login"
- "logout"
- "requestQueryCancel"
- "submitWorksheetQuery"

### getConnectionList

This API call obtains a list of connections. It accepts SessionKey and returns the Connection list object. In single sign-on mode the list of public connections and private connections is returned. The connection can be of type Relational, OLAP, or APPS.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public ConnectionList getConnectionList(SessionKey aSessionKey) ( )</td>
</tr>
<tr>
<td>Input:</td>
<td>■ SessionKey</td>
</tr>
<tr>
<td>Output</td>
<td>■ ConnectionList</td>
</tr>
<tr>
<td>Exceptions</td>
<td>■ java.rmi.RemoteException</td>
</tr>
</tbody>
</table>

The following table lists the fields of the structures.
getFolderEntryList

This API call uses a connection to obtain the list of non-scheduled workbooks and FolderEntries. It accepts SessionKey, ConnectionKey, and FolderEntryPath and provides a list of non-scheduled workbooks and FolderEntries data that is accessible from the connection for the FolderEntryPath. The list of non-scheduled workbooks includes all of the shared workbooks for the database user that was used in the connection.

When the connection is to relational data, workbooks are stored in a single level in the EUL. The only valid value for FolderEntryPath is "". An empty string specifies the root folder entry in both Discoverer Plus Relational and Discoverer Plus OLAP.

When the connection is to OLAP data, FolderEntries and Workbooks are stored in the Discoverer Catalog. There can be multiple levels of FolderEntries, which can be queried by providing the appropriate FolderEntryPath. Clients must make successive calls to this method to browse FolderEntries.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionList</td>
<td>■ Array of Connection objects</td>
</tr>
</tbody>
</table>
| Connection | ■ ConnectionName  
 ■ ConnectionKey  
 ■ ConnectionDescription  
 ■ ConnectionEUL  
 ■ ConnectionDBIdentifier  
 ■ ConnectionAccessType |
| ConnectionName | ■ String name |
| ConnectionKey | ■ String key |
| ConnectionDescription | ■ String description |
| ConnectionEUL | ■ String eulname |
| ConnectionDBIdentifier | ■ String database name |
| ConnectionAccessType | ■ String accesstype |

The following table lists the fields of the structures.
getLayoutMetaData

This API call obtains the layout metadata for a selected worksheet. It accepts SessionKey and WorksheetKey, and it provides the Layout information for the worksheet that is identified by WorksheetKey.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>FolderEntryList</td>
<td>■ Array of FolderEntry objects</td>
</tr>
<tr>
<td></td>
<td>■ String country</td>
</tr>
<tr>
<td></td>
<td>■ String variant</td>
</tr>
<tr>
<td>FolderEntry</td>
<td>■ FolderEntryName</td>
</tr>
<tr>
<td></td>
<td>■ FolderEntryPath</td>
</tr>
<tr>
<td></td>
<td>Fully qualified path to either the FolderEntry or the WorkbookEntry.</td>
</tr>
<tr>
<td></td>
<td>■ FolderEntryType</td>
</tr>
<tr>
<td></td>
<td>■ FolderEntryDescription</td>
</tr>
<tr>
<td>FolderEntryName</td>
<td>■ String name</td>
</tr>
<tr>
<td>FolderEntryPath</td>
<td>■ String path</td>
</tr>
<tr>
<td>FolderEntryDescription</td>
<td>■ String description</td>
</tr>
<tr>
<td>FolderEntryType</td>
<td>■ String type</td>
</tr>
<tr>
<td></td>
<td>Either FolderEntry or WorkbookEntry.</td>
</tr>
<tr>
<td></td>
<td>■ WorkbookKey</td>
</tr>
<tr>
<td></td>
<td>Valid if the type is WorkbookEntry.</td>
</tr>
<tr>
<td>WorkbookKey</td>
<td>■ String wbDevKey</td>
</tr>
<tr>
<td></td>
<td>■ ConnectionKey</td>
</tr>
</tbody>
</table>

The following table lists the fields of the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout</td>
<td>■ Array of Dimension objects</td>
</tr>
<tr>
<td></td>
<td>■ Array of Measure objects</td>
</tr>
<tr>
<td>Dimension</td>
<td>■ String label</td>
</tr>
<tr>
<td>Measure</td>
<td>■ String label</td>
</tr>
</tbody>
</table>
getParameterMetaData

This API call obtains the parameter metadata for a selected worksheet. It accepts SessionKey and WorksheetKey and provides the list of parameters for the specified worksheet that is identified by WorksheetKey.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public ParameterList getParameterMetaData(SessionKey aSessionKey, WorksheetKey aWorksheetKey) {}</td>
</tr>
</tbody>
</table>
| Input: | ■ SessionKey  
■ WorksheetKey |
| Output | ■ ParameterList |
| Exceptions | ■ java.rmi.RemoteException |

The following table lists the fields of the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterList</td>
<td>■ Array of Parameter objects.</td>
</tr>
</tbody>
</table>

getParameterValueList

This API call obtains the parameter LOVs using a ParameterKey. It accepts SessionKey, ParameterKey, and numValues and provides the list of values (LOVs). You call this method for parameters that have LOVs. The LOVs are retrieved in chunks (the size is specified by numValues), and you can page through from start to finish in one direction only; there is no bi-directional paging support of parameters. The client application is responsible for caching the parameters.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public ParameterValueList getParameterValueList(SessionKey aSessionKey, ParameterKey aParamKey, int numValues) {}</td>
</tr>
</tbody>
</table>
| Input: | ■ SessionKey  
■ ParameterKey  
■ numValues  
An integer value that specifies the number of values that the web service should return in one call. If there are more values getParameterValueList should be called repeatedly. |
| Output | ■ ParameterValueList |
| Exceptions | ■ java.rmi.RemoteException |
| Notes | ■ Parameter linking is inactive. |

The following table lists the fields of the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
</table>
| ParameterValueList | ■ Array of ParameterValue Objects  
■ Boolean finished |
**getQueryStatus**

This API call returns the status of the query, such as executing or canceled. It accepts SessionKey and QueryKey.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public QueryStatus getQueryStatus(SessionKey aSessionKey, QueryKey aQueryKey) { }</td>
</tr>
</tbody>
</table>

Input:
- SessionKey
- QueryKey

Output:
- QueryStatus

Exceptions:
- java.rmi.RemoteException

The following table lists the fields of the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
</table>
| QueryStatus | String detail  
|           | String status  
|           | Status is one of the following:  
|           | QUERY_FAILED  
|           | QUERY_EXECUTING  
|           | QUERY_CANCELED  
|           | QUERY_RESULTS_READY  
|           | QUERY_NEEDS_EXECUTING  
|           | QUERY_NOT_DEFINED  
|           | QUERY_SCHEDULED  
|           | QUERY_VALIDATING |

**getVersion**

This API call provides the web service component version. Any change in the web service interface or API causes the version number to change. Use the version number to assist you in determining the functionality that is available for the web service.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public String getVersion( ) { }</td>
</tr>
</tbody>
</table>

Input:
- Void

Output:
- String

A version of 1.0 implies support for integration with Oracle BI Publisher.
A version of 2.0 implies support for integration with Oracle BI Publisher and interoperability with Oracle BI Enterprise Edition.

Exceptions:
- None
getViewerURL

This API call provides the URL of the Discoverer Viewer instance that is hosted on the same machine as the web service. This URL can be used by clients to launch Discoverer Viewer from within their application.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public String getViewerURL(SessionKey aSessionKey, WorksheetKey aWorksheetKey, List aParameterSelectList){ }</td>
</tr>
</tbody>
</table>
| Input:  | ■ SessionKey  
          ■ WorksheetKey  
          ■ Java.util.List of ParameterSelect |
| Output  | ■ String  
          URL to launch Discoverer Viewer. |
| Exceptions | ■ java.rmi.RemoteException |

getWorksheetData

This API call accepts SessionKey and QueryKey and provides the worksheet data in the format type that is specified in QueryOption in the submitWorksheetQuery method. If the result set is large and is Rowset XML, then by default, 25 rows of data are returned. For other format types, 1MB of data is returned. You can page through the remaining data by calling this method repeatedly. If the finished flag in QueryResults is false, then call the method again to continue paging through the data. You can page through from start to finish in one direction only; there is no bi-directional paging support.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public QueryResult getWorksheetData(SessionKey aSessionKey, QueryKey aQueryKey){ }</td>
</tr>
</tbody>
</table>
| Input:  | ■ SessionKey  
          ■ QueryKey |
| Output  | ■ QueryResult |
| Exceptions | ■ java.rmi.RemoteException |

The following table lists the fields of the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
</table>
| QueryResult | ■ String data  
             Rowset XML, HTML, PDF, XLS data. HTML, PDF, XLS data uses base64 encoding.  
             ■ Boolean finished  
             If finished is true, then do not call GetWorksheetData. |

The following table describes the export types.
For HTML, PDF, and XLS export types

- If the worksheet does not have page items, then all the data is exported.

- If the worksheet has page items, the output is different for OLAP and Relational connections as follows:
  - For Relational connections:
    HTML, PDF, and XLS export types - Returns worksheet data for the page items selected in the worksheet.
  - For OLAP connections:
    XLS export types - Returns worksheet data for all combination of page item values.
    HTML or PDF export types - Returns worksheet data for the selected page item.

This export behavior is similar to Viewer with the exception that the "isCurrentPageItemsExportForOLAP" setting in the configuration.xml file is not respected.

**Sample Rowset XML structure**

```xml
<BIData>
  <Query id="1">
    <PAGE>
      <ROWSET>
        <ROW>
          <Dimension L="Time Dimension Values">1996</D>
          < Dimension L="Product Dimension Values">TOTALPROD</D>
          < Dimension L="Geography Dimension Values">WORLD</D>
          < Dimension L="Channel Dimension Values">TOTALCHANNEL</D>
          <Measure L="Dollar Sales">1.18247112042864E8</M>
          <Measure L="Quota">5475441.87541972</M>
        </ROW>
        <ROW>
          < Dimension L="Time Dimension Values">1997</D>
          < Dimension L="Product Dimension Values">TOTALPROD</D>
          < Dimension L="Geography Dimension Values">WORLD</D>
          < Dimension L="Channel Dimension Values">TOTALCHANNEL</D>
          <Measure L="Dollar Sales">4.6412112042864E8</M>
          <Measure L="Quota">2917189.87541972</M>
        </ROW>
      </ROWSET>
    </PAGE>
  </Query>
</BIData>
```
**getWorkSheetList**

This API call accepts SessionKey and WorkbookKey and provides the list of worksheets within a workbook that is specified in WorkbookKey.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>Public WorksheetList getWorkSheetList(SessionKey aSessionKey, WorkbookKey aWorkbookKey ) {}</code></td>
</tr>
<tr>
<td>Input:</td>
<td></td>
</tr>
<tr>
<td>■</td>
<td>SessionKey</td>
</tr>
<tr>
<td>■</td>
<td>WorkbookKey</td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>■</td>
<td>WorksheetList</td>
</tr>
<tr>
<td>Exceptions</td>
<td></td>
</tr>
<tr>
<td>■</td>
<td>java.rmi.RemoteException</td>
</tr>
</tbody>
</table>

The following table lists the fields of the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorksheetList</td>
<td>■ Array of Worksheet objects</td>
</tr>
</tbody>
</table>
| Worksheet | ■ WorksheetName  
|           | ■ WorksheetKey  
|           | ■ Array of Parameter objects.  
|           | This is always a zero-sized array. |
| WorksheetName | ■ String name |
| WorksheetKey | ■ String wsDevKey  
|           | Identifies a worksheet. For Discoverer Plus Relational and Discoverer Plus OLAP, this is a string.  
|           | ■ WorkbookKey |

**isSessionValid**

This API call accepts a SessionKey and returns a Boolean value that indicates whether the specified session is valid.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>Public boolean isSessionValid( SessionKey sKey ) {}</code></td>
</tr>
<tr>
<td>Input:</td>
<td></td>
</tr>
<tr>
<td>■</td>
<td>SessionKey</td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>■</td>
<td>true if session is valid, false otherwise</td>
</tr>
<tr>
<td>Exceptions</td>
<td></td>
</tr>
<tr>
<td>■</td>
<td>java.rmi.RemoteException</td>
</tr>
</tbody>
</table>

**login**

This API call provides a mechanism for user identity propagation between the client application and the Discoverer Web Services. Authentication occurs during every invocation of the Discoverer Web Services. For more information, see "About authentication and authorization".

For every login made by a client application, the Discoverer Web Services create a new Discoverer user session and allocate it to the user.
This API call accepts UserCredential and LocaleBean information and returns a unique SessionKey, which is then passed to all other API calls. This API call does not create a new HTTP session if a session exists between the client application instance and the Discoverer Web Services.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public SessionKey login(UserCredential aUserCredential, LocaleBean aLocale) { }</td>
</tr>
</tbody>
</table>
| Input:            | UserCredential
                    This object must contain a valid login details of a Discoverer user.
                    LocaleBean
                    This object passes locale information to the webservice session. By default the Locale is English (en-US).
| Output            | SessionKey                                                             |
| Exceptions        | java.rmi.RemoteException                                                |

The following table lists the fields of the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
</table>
| LocaleBean      | String language
                    String country
                    String variant                      |
| UserCredential  | Identifier anIdentifier
                    DisplayName aDisplayName             |
| Identifier      | String id                            |
| DisplayName     | String user                          |
| SessionKey      | String key
                    Securely generated random alphanumerical sequence. |

**logout**

This API call informs the Discoverer Web Services API about the completion of the user session. This call frees the dedicated Discoverer session that was associated with the user session and returns the Discoverer session to the session pool.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public void logout(SessionKey aSessionKey) { }</td>
</tr>
<tr>
<td>Input:</td>
<td>SessionKey</td>
</tr>
<tr>
<td>Output</td>
<td>Void</td>
</tr>
<tr>
<td>Exceptions</td>
<td>java.rmi.RemoteException</td>
</tr>
</tbody>
</table>

**requestQueryCancel**

This API call cancels a query request and removes the resources that are associated with that request. Call this method for queries that have been submitted but whose
data has not yet been fetched. If the query data has been fetched, then the resources are removed. This asynchronous API call returns immediately.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public Void requestQueryCancel(SessionKey aSessionKey, QueryKey){ }</td>
</tr>
<tr>
<td>Input:</td>
<td>■ SessionKey</td>
</tr>
<tr>
<td>Output:</td>
<td>■ Void</td>
</tr>
<tr>
<td>Exceptions</td>
<td>■ java.rmi.RemoteException</td>
</tr>
</tbody>
</table>

submitWorksheetQuery

This API call submits the query for the specified worksheet for execution. This asynchronous API call returns immediately, without waiting for the query execution to complete.

<table>
<thead>
<tr>
<th>API</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Public QueryKey submitWorksheetQuery(SessionKey aSessionKey, WorksheetKey aWorksheetKey, List aParameterSelectList, QueryOption aQueryOption){ }</td>
</tr>
<tr>
<td>Input:</td>
<td>■ SessionKey</td>
</tr>
<tr>
<td></td>
<td>■ WorksheetKey</td>
</tr>
<tr>
<td></td>
<td>■ Java.util.List of ParameterSelect</td>
</tr>
<tr>
<td></td>
<td>■ QueryOption</td>
</tr>
<tr>
<td>Output:</td>
<td>■ QueryKey</td>
</tr>
<tr>
<td>Exceptions</td>
<td>■ java.rmi.RemoteException</td>
</tr>
</tbody>
</table>

The following table lists the fields of the structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryKey</td>
<td>■ String queryId</td>
</tr>
<tr>
<td></td>
<td>A secure and unique key that is randomly generated.</td>
</tr>
<tr>
<td>ParameterSelect</td>
<td>■ ParameterKey</td>
</tr>
<tr>
<td></td>
<td>■ Array of ParameterValue objects</td>
</tr>
<tr>
<td>QueryOption</td>
<td>■ int NoOfRows</td>
</tr>
<tr>
<td></td>
<td>The number of rows to return in Rowset XML. The default is 25.</td>
</tr>
<tr>
<td></td>
<td>■ int NoOfRows</td>
</tr>
<tr>
<td></td>
<td>The number of rows to return in Rowset XML. The default is 25.</td>
</tr>
<tr>
<td></td>
<td>■ int chunksize</td>
</tr>
<tr>
<td></td>
<td>The size of data chunks in kilobytes for PDF, XLS and HTML formats.</td>
</tr>
<tr>
<td></td>
<td>The default is 64 KB.</td>
</tr>
<tr>
<td></td>
<td>■ ResultType</td>
</tr>
<tr>
<td></td>
<td>■ UserCredential</td>
</tr>
<tr>
<td></td>
<td>The GUIDUsername or SSOUsername of the user for whom the query is run.</td>
</tr>
<tr>
<td></td>
<td>If no user name is specified, then the query is run for the user who</td>
</tr>
<tr>
<td></td>
<td>has logged in.</td>
</tr>
<tr>
<td>Structure</td>
<td>Fields</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| LocaleBean  | String type  
Valid types are:  
- XMLROWSET  
- HTML - base64 encoded content returned in data string  
- PDF - base64 encoded content returned in data string  
- XLS - base64 encoded content returned in data string |
API calls
- getConnectionList (), 2-1
- getFolderEntryList (), 2-2
- getLayoutMetaData (), 2-3
- getParameterMetaData (), 2-4
- getParameterValueList (), 2-4
- getQueryStatus (), 2-5
- getVersion (), 2-5
- getViewerURL (), 2-6
- getWorksheetData (), 2-6
- getWorkSheetList (), 2-8
- isSessionValid (), 2-8
- login (), 2-8
- logout (), 2-9
- requestQueryCancel (), 2-9
- submitWorksheetQuery (), 2-10

authentication and authorization, 1-2
availability requirements, 1-3

creating web service client stubs, 1-4

diagnosing problems, 1-3
Discoverer Web Services API
- requirements for invoking, 1-3
- verifying access to, 1-4

Endpoint URL, 1-2
error messages, 1-3

Java class example that might invoke the Discoverer Web Services, 1-13

Oracle BI Discoverer Web Services
- about, 1-1

sessions
- maintaining, 1-2
- pool size, 1-2
- single sign-on, 1-2
SOAP endpoint URL, 1-2
SOAP protocol, 1-1

typical flow of events
- detailed task examples, 1-8
- list, 1-7

writing client application
- using generated web service client stubs, 1-5
- WSDL format, 1-2