Oracle® Discussions

Application Developer's Guide

10*g* Release 1(10.1.2)

B28208-02

March 2006

ORACLE®

Oracle Discussions Application Developer's Guide, 10*g* Release 1(10.1.2)

B28208-02

Primary Author: Madhubala Mahabaleshwar

Contributors: Shreedhar Patwari, Krishna Rayapuram, Raymond Gallardo, Nick Taylor, Tehmurasp Ghyara, Meeta Gupta

Contents

Preface

This preface contains the following topics:

Audience

Oracle Discussions Application Developer's Guide is intended for the programmers and the developers who want to use Oracle Discussions Web Services to create custom applications.

Documentation Accessibility

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

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

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

Related Documentation

Printed documentation is available for sale in the Oracle Store at

http://oraclestore.oracle.com/

To download free release notes, installation documentation, white papers, or other collateral, visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

http://www.oracle.com/technology/membership/

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

http://www.oracle.com/technology/documentation/

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.

1

Overview

This chapter gives an overview of Oracle Discussions Software Development Kit (SDK) and Oracle Discussions Web Services.

Oracle Discussions Software Development Kit

Oracle Discussions SDK consists of the following components:

- Category and Forum Management Application Programming Interface (API): Developers can use this API to create or delete categories and forums, edit the properties of existing categories and forums, and set the access control lists on these categories and forums.
- Topic Management API: Developers can use this API to post new topics, read messages in a topic and move or delete.
- Message Management API: Developers can use this API to post new messages and edit or hide or delete the messages in a topic.
- Favorites and Subscriptions API: Developers can use this API to manage favorites and subscriptions to categories, forums and topics.
- Other Oracle Discussions internal classes

Oracle Discussions Web Services

The Oracle Discussions Web Services layer is on the top of the Oracle Discussions SDK layer. The methods in the Web Service class are actually operations or services that are supported by Oracle Discussions SDK. These methods are categorized into ten service classes as follows:

- Authentication Service

  This service provides methods for the user login and logout operations. It has to be invoked before the user performs the first operation on the Web Services and after the user finishes using the Oracle Discussions Web Services.

- Container Service

  This service provides operations that are common to both Category and Forum Services. Container Service is a logical parent service to both Category and Forum Services.

- Category Service

  A category is an abstraction of a container that holds other categories and forums. A category cannot contain messages directly under it. The Category Service provides methods for operations on Oracle Discussions Category. This service also provides methods for bulk-create and bulk-delete operations to operate on multiple categories in one invocation.

- Forum Service

  A forum in Oracle Discussions represents a container, to which topics or messages are posted. The Forum Service service provides methods for operations on Oracle Discussions Forum. In addition, it provides methods for bulk-create and bulk-delete operations to operate on multiple forums in one invocation.

- Topic Service

  A topic is the subject around which a group of messages are centred. It is the original message that starts the thread for other messages. The Topic Service provides methods for operations on Oracle Discussions Topic. This service also provides methods for bulk-create and bulk-delete operations to operate on multiple topics in one invocation.

- Message Service

  The replies to a topic are called messages. A message can have attachments and text. The Message Service provides methods that operate on these messages. This service also provides methods for bulk-create and bulk-delete operations to operate on multiple messages in one invocation. In addition, it also provides operations for retrieving messages from byte arrays, creating messages and editing them.

- MyDiscussions Service

  This service provides methods that operate on the following:

  – Favorites
  – My Posts
  – Replies to my posts
  – Popular topics
  – Recent posts

  Favorites show the Oracle Discussions elements such as categories, forums and messages that the user has bookmarked.

- Subscription Service

  This service provides methods to subscribe and unsubscribe to a container and a topic. It also provides methods to update, list and retrieve subscriptions.

- User Service

  This service provides methods to retrieve users of Oracle Discussions based on the user attributes.

- S2S Authentication Service

  This service facilitates other services to proxy authenticate an user to Oracle Discussions by authentic ating themselves with Oracle Discussions. The service proxy, that authenticates the user, should provide the login id of the user when asserting its own credentials to Oracle Discussions..

2

Overview of Oracle Discussions SDK

Oracle Discussions SDK contains Java API that allow you to create and manage Oracle Discussions elements. These API are exposed through the package **oracle.discussions.sdk**.

This chapter describes the elements of Oracle Discussions SDK and the Java classes that represent them. It also describes other classes that help manage and configure Oracle Discussions elements.

Some elements and functions in Oracle Discussions are represented with a new terminology in Oracle Discussions SDK. The following table describes the names of Oracle Discussions elements and functions as they appear in the Web client, and the corresponding name that is used in the Oracle Discussions SDK.

Table 2–1 Names Used in Oracle Discussions Web Client

Name Used in Web Client	Name Used in SDK
Category	Facility
Forum	Board
Topic	Thread
Hide	Delete
Replies to my posts	**TdMyFollowups**

Elements of Oracle Discussions SDK

Oracle Discussions SDK facilitates the creation of the following Oracle Discussions elements:

- Stores
- Users
- Containers
- Facilities
- Boards
- Announcements Board
- Threads
- Messages

Stores

A store allows a particular Oracle Discussions user to create, delete, and access Oracle Discussions facilities and boards. A store provides methods for the following:

- Creating and deleting containers
- Initializing global settings
- Initializing various managers such as **TdGlobalAdminManager** and **TdSubscriptionManager**
- Searching Oracle Discussions facilities and boards

Oracle Discussions will create a store at login time for every valid Oracle Discussions user. This store will be closed when the user logs out. A store is an abstraction of a Discussions root container. Every user that logs into Oracle discussions has a store associated with him or her.

The Oracle Discussions store is represented by **TdStore**.

Users

The **TdUser** class represents Oracle Discussions users. These users must be provisioned for a valid email domain to be able to access Oracle Discussions. A user is anyone who is provisioned with an Oracle Collaboration Suite account.

The factory class **TdUserFactory** factory class retrieves Oracle Discussions users from Oracle Internet Directory, based on a user attribute such as email address or global user identifier (GUID).

The **TdUserProfile** class represents an Oracle Discussion user's profile. This profile includes personal information (such as address, phone number, and picture), and basic statistics (such as last login time and number of posts). Access a user's profile with the **TdUserProfileManager** class.

Containers

An Oracle Discussions container is represented by the **TdContainer** class. This class is the super class of **TdFacility** and **TdBoard**. It stores attributes and properties common to both facilities and boards.

TdContainerPermissions represents an Oracle Discussions user's permissions on a container. It specifies the actions and operations that a user can perform on the container. The **TdContainerSettings** class represents the container attributes defined when the container is created.

Facilities

A facility is a container that holds other facilities or boards. It may not contain threads or messages. However, the boards stored in a facility may contain threads or messages. The **TdFacility** class represents a facility. Facilities are uniquely identified among all other facilities at the same level by their name and display name.

The **TdFacilityPermissions** class represents the actions the logged in user may perform on a facility.

Boards

A board is a container that holds threads and messages. It may not contain facilities or other boards. The **TdBoard** class represents a board. Boards are uniquely identified among all other boards at the same level by their name and display name.

The **TdBoardPermissions** class represents an Oracle Discussions user's permissions on a board. It specifies the actions and operations that a user can perform on the board. The **TdBoardSettings** class represents the board attributes defined when the board is created, such as maximum attachment size and board email address.

By configuring a board's settings, it can be viewed as an IMAP folder in an email client. As a result, any messages posted from the email client and addressed to the board will appear as Oracle Discussions messages in the board, and threads and messages posted to the board will appear as email messages in the user's inbox.

Announcements Board

The **TdAnnouncementsBoard** class represents special board to which messages can be posted only by the board moderator. An announcements board is present in the Oracle Discussions root container by default. When creating any other facility or board, you may specify whether or not it will contain an announcements board.

An announcements board is implicit. It always has a name ".Announcements" (note the period before this name). It is always public, meaning that any user who is a member of the container in which announcements board is present can read the messages in it. Once created, an announcements board cannot be edited. Announcements board cannot be locked like any other board.

Threads

The **TdThread** class represents a thread in a board. A root topic in a board is a thread and the subsequent replies to this root topic are messages. This class extends the **TdMessage** class.

The **TdThreadPermissions** class represents an Oracle Discussions user's permissions on a thread. It specifies the actions and operations that a user can perform on the thread. These actions and operations depend on the user's role as defined in the thread's parent container.

Messages

The **TdMessage** class represents a message in a board. This class provides methods to move and delete a message and retrieve information about a message, such as the message's parent thread, parent board, attachments, and body.

The **TdMessagePermissions** class specifies the actions a user may perform on a message. These depend on the user's role as defined in the parent container of the message and whether the user is the owner of the message.

To obtain a list of attachments of a particular message, call **TdMessage.getAttachments()**. This will return a list of **TdAttachment** objects. This class holds metadata about the attachment, including the attachment's name and size. Instead of holding the actual content of the attachment, it holds the attachment's ID, which can be passed to the **TdMessage.getAttachmentById()** method to fetch the attachment itself.

The **TdFlag** represents status flags for messages. This class represents the following flags:

- NEW_SINCE_LAST_VISIT: This flag is set if the message is new since the time the user last logged out.
- NEW_SINCE_SESSION: This flag is set if the message is new since the user's current session.

Retrieving Messages Associated with a Particular User

To retrieve all the messages posted by a user throughout the Oracle Discussions system and in all forums, call the **TdStore.getMyPosts()** method. This returns an instance of **TdMyPosts**.

To retrieve the last posted message (most recently posted message) at a particular container level, call **TdFacility.getLastPost()** or **TdBoard.getLastPost()**. To retrieve the most recently posted message in the entire Oracle Discussions system, call the **TdStore.getLastPost()**.

To retrieve the most recent posts or most popular threads of a particular user the replies to messages of a particular user, call **TdStore.getLastMessagePosts()** or **TdStore.getPopularThreads()**, respectively. Both these methods return iterators. These calls are also available at container levels.

To retrieve the replies to the user-posted messages, call **TdStore.getMyFollowups()**. The method returns the replies posted by other Oracle Discussions users to the logged in user's posts.

Management Features of Oracle Discussions SDK

The following features allow you to manage and administer the elements of Oracle Discussions SDK:

- Roles
- Iterators
- Favorites
- Searches
- Subscriptions
- Annotations
- Exceptions

Roles

The following access control roles have been defined:

- Global administrator: Users granted this role represent the super users of the system. They are granted the permission of executing all tasks on all objects in the system.
- Facility creator: These users can create new facilities and boards within the facility for which they are granted this role. Facility creators are automatically granted facility administrator and board administrator roles for the facilities and boards they create.
- Facility administrator: Facility administrators have permission to administer only the facilities and boards within the assigned facility. Facility administrators can post announcements for the facility they administer.

- Facility writer: These users automatically have board writer privileges for all the board within the facility for which they are granted this role.
- Facility reader: These users automatically have facility reader privileges for all the boards within the facility for which they are granted this role.
- Board creator: Board creators on a given facility have the privilege of creating boards within that facility.
- Board administrator: Users granted this role on a given board have the authority of managing the content within that board. Board administrators can post announcements for the forum they moderate.
- Board writer: Board writers can post new topics or reply to existing ones. They may be able to edit messages that they posted (depending on board policies), but never messages posted by other users.
- Board reader: Board readers will only have read access to boards on which they have been granted the role. Board readers do not have permission to post new messages or reply to existing topics.

The **TdRole** class represents a role. To grant or revoke a role to a user, call the grant and revoke methods in the **TdGlobalAdminManager** class.

To retrieve a list of users and their roles for a particular facility or board, or to retrieve users and their roles from the entire Oracle Discussions system, call the **listGranteeRoles()** method from the respective **TdFacility**, **TdBoard**, or **TdStore** instance. This will return a list of **TdGranteeRole** objects. This class associates a user with a role.

Iterators

When you call a method that returns a collection of Oracle Discussion elements, such as **TdBoard.getThreads()**, which retrieves all the threads of a particular board, the method will return an iterator. Use this iterator to traverse through the list of returned Oracle Discussion elements.

Favorites

A favorite is an Oracle Discussions element of user's interest. These favorites are listed at the global level and at each container level, which allows users quick access to them.

The **TdContainer** and **TdThread** classes implement the **TdFavorite** interface. Therefore, you may add or remove a container (facility or board) or thread from the user's favorites by calling the element's **addToFavorites()** or **removeFromFavorites()** methods.

Retrieve a complete list of a user's favorite threads in the entire Oracle Discussions system by using an instance of **TdFavoriteManager**, retrieved from a **TdStore** instance. This class also gives access to the user's favorite containers in the entire system.

Searches

To search for a message in a particular facility or board, call the **search()** method. This method takes a **oracle.discussions.sdk.TdSearchTerm** object as an argument to represent the query criteria of a search. It returns an array of **TdMessage** objects.

To search for a message globally, call the **TdStore.search()** method. This method takes a **TdSearchTerm** object as an argument to represents the query criteria of a search. The **TdStore.search()** method returns a **TdIterator**.

Subscriptions

Subscriptions allow users to monitor the changes to Oracle Discussions elements without actually logging into Oracle Discussions. When a user subscribes to an Oracle Discussions element, he or she will receive a notification email whenever that element is changed until that user unsubscribes from that Oracle Discussions element, or that element is deleted.

To subscribe to an Oracle Discussions element, obtain an instance of **TdSubscriptionManager** from **TdStore** and call the manager's **subscribe()** method.

The **TdSubscription** class represents a user's subscription to a container. The **TdThreadSubscription** class represents a user's subscription on a thread. To obtain an instance of either of these classes, call **TdSubscriptionManager.getSubscription()** with the subscribed container or thread as an argument.

Annotations

An annotation represents an Oracle Discussions operation performed on a message or thread. It represents a message modification history. An annotation stores information such as the user performing the operation, the operation being performed, and the date on which the operation is being performed.

The **TdAnnotation** class represents an annotation. Retrieve all the annotations of a message with the **TdMessage.getAnnotations()** method.

Exceptions

Oracle Discussions returns the following exceptions:

- **TdRuntimeException**: Represents general Oracle Discussions runtime exceptions
- **TdUnsupportedOperationException**: Represents runtime exceptions thrown for JavaMail operations not supported by Oracle Discussions
- **TdException**: Represents general Oracle Discussions exceptions
- **TdConfigurationException**: Thrown when supplied configuration parameters to the Discussions initial context are not correct.

Configuration Features of Oracle Discussions SDK

These features help configure Oracle Discussions:

- Initial Contexts
- System-Level Settings

Initial Contexts

The **TdInitialContext** class represents the initialization parameters provided to Oracle Discussions SDK. This class inherits from **java.util.HashMap**. The supplied properties will determine the behavior of Oracle Discussions in terms of infrastructure services to use such as logging and Oracle Internet Directory authentication.

System-Level Settings

Grant or revoke global administrator's or root level facility creation privileges to a user with the **TdGlobalAdminManager** class. Obtain an instance of this class from **TdStore.getGlobalAdminManager()**.

Change system-level properties with the **TdGlobalSettings** class. See the section Tuning Oracle Discussions in the chapter "Monitoring and Tuning Oracle Collaboration Suite Performance" in *Oracle Collaboration Suite Administrator's Guide* for a list of these properties.

3

Building Applications with Oracle Discussions SDK

This chapter shows you how to configure your environment in order to compile and execute applications that use Oracle Discussions SDK. It provides some example code for you to use.

This chapter assumes that you have access to an Oracle Collaboration Suite installation, and you know to set the Java classpath and compile and execute Java programs.

Requirements

The following section shows you how to configure your environment. It describes the following tasks:

- Setting Classpath
- Passing Initial Context Parameters
- Setting JAZN Permissions

Setting Classpath

Include the following jar files in your classpath in order to compile and execute the sample code in this chapter. The path names in this list are relative to `$ORACLE_HOME`:

- `discussions/lib/discussions.jar`
- `j2ee/home/jazn.jar`
- `j2ee/home/jazncore.jar`
- `j2ee/home/lib/jms.jar`
- `j2ee/OC4J_OCSClient/applications/discussions/lib/rtcSDK.jar`
- `jdbc/lib/classes12.jar`
- `jdbc/lib/ojdbc14.jar`
- `jlib/collabsuiteuser.jar`
- `jlib/esadmin.jar`
- `jlib/escommon.jar`
- `jlib/esldap.jar`
- `jlib/esmail_sdk.jar`

- jlib/ldapjclnt10.jar
- jlib/ocsaddressbook.jar
- jlib/ocscommon.jar
- jlib/ojmisc.jar
- jlib/orai18n.jar
- jlib/orai18n-js.jar
- jlib/orai18n-ocs.jar
- jlib/repository.jar
- lib/activation.jar
- lib/dms.jar
- lib/mail.jar
- lib/xmlparserv2.jar
- workspaces/lib/workspaces_share.jar

Passing Initial Context Parameters

See the following Java options according to your deployment. These initial context parameters control the behavior of Oracle Discussions.

Table 3–1 Initial Context Parameters as Java Options

Initial Context Parameter	Description
-Doracle.home=*C:\TdDemo*	Base directory of your Oracle Discussions application
-Doracle.ocs.ldappool.host=*www.example.com*	Oracle Internet Directory hostname]
-Doracle.ocs.ldappool.port=*389*	Oracle Internet Directory port
-Doracle.ocs.ldappool.user_dn=*cn=orcladmin*	Oracle Internet Directory login user name
-Doracle.ocs.ldappool.user_password=*welcome1*	Oracle Internet Directory login user password
-Doracle.ocs.ldappool.initialsize=*1*	Oracle Internet Directory LDAP pool size. Do not change the size. Default size is 1.
-Doracle.ocs.ldappool.connectssl=*false*	Specifies if the Oracle Internet Directory connection is to be secured. Default is false.
-Doracle.mail.sdk.esmail.driver_type=*thin*	Driver type to be used to connect to Oracle Mail store.
-Doracle.mail.sdk.esmail.encryption=*false*	Specifies whether or not encryption will be used by Oracle Mail
-Doracle.discussions.event.enabled=*false*	If set to false, the Oracle Discussions system will not post events
-Doracle.discussions.tdadmin.email= *discussions_ocs@example.com*	Oracle Discussions internal administrator email ID. This will be in the following form: `discussions_<ocs-apps-instance-name> @<email-domain>` Log on to the Oracle Collaboration Suite Identity Management Self-Service Console to verify the existence of this user. The Applications tier instance name and the email domain are set when Oracle Collaboration Suite is installed. Refer to the Oracle Collaboration Suite installation log file for these details.

Setting JAZN Permissions

Oracle Discussions uses Oracle Mail, which uses JAZN file protection for .jar file protection. Create or add to the following XML files in the indicated directories.

Ensure that you edit these files so that the Java library files that appear in the jazn-data.xml files point to the same Java library files in your classpath or else a **SecurityException** will be thrown.

Example 3–1 $ORACLE_HOME/j2ee/home/config/jazn.xml

```
<jazn provider="XML" location="./jazn-data.xml"/>
```

Example 3–2 $ORACLE_HOME/j2ee/home/config/jazn-data.xml

```
<jazn-data>
  <!-- JAZN Realm Data -->
  <jazn-realm>
    <realm>
      <name>jazn.com</name>
      <users/>
      <roles/>
    </realm>
  </jazn-realm>
  <!-- JAZN Policy Data -->
  <jazn-policy>
    <grant>
      <grantee>
        <codesource>
          <url>file:C:\TdDemo\lib\esldap.jar</url>
        </codesource>
      </grantee>
      <permissions>
        <permission>
          <class>
            oracle.ias.repository.schemaimpl.CheckRepositoryPermission
          </class>
          <name>connectAs</name>
        </permission>
        <permission>
          <class>
            oracle.ias.repository.schemaimpl.CheckRepositoryPermission
          </class>
          <name>makeNewOIDEntry</name>
        </permission>
        <permission>
        <class>
          oracle.ias.repository.schemaimpl.CheckRepositoryPermission
        </class>
        <name>getIASProperty</name>
        </permission>
        <permission>
        <class>
          oracle.ias.repository.schemaimpl.CheckRepositoryPermission
        </class>
        <name>getOIDConnect</name>
        </permission>
      </permissions>
    </grant>
  </jazn-policy>
</jazn-data>
```

Example 3–3 $ORACLE_HOME/oes/jazn/jazn.xml

```
<jazn provider="XML" location="./jazn-data.xml"/>
```

Example 3–4 $ORACLE_HOME/oes/jazn/jazn-data.xml

```
<jazn-data>
  <!-- JAZN Realm Data -->
  <jazn-realm>
    <realm>
      <name>jazn.com</name>
      <users/>
      <roles/>
    </realm>
  </jazn-realm>
  <!-- JAZN Policy Data -->
  <jazn-policy>
    <grant>
      <grantee>
        <codesource>
          <url>file:C:\TdDemo\lib\discussions.jar</url>
        </codesource>
      </grantee>
      <permissions>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>logon</name>
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>minfo</name>
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>procmparams</name>
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>umparams</name>
        </permission>
      </permissions>
    </grant>

    <grant>
      <grantee>
        <codesource>
          <url>file:C:\TdDemo\lib\esldap.jar</url>
        </codesource>
      </grantee>
      <permissions>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>logon</name>
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>minfo</name>
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>procmparams</name>
```

```
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>umparams</name>
        </permission>
      </permissions>
    </grant>

    <grant>
      <grantee>
        <codesource>
          <url>file:C:\TdDemo\lib\collabsuiteuser.jar</url>
        </codesource>
      </grantee>
      <permissions>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>logon</name>
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>minfo</name>
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>procmparams</name>
        </permission>
        <permission>
          <class>oracle.security.jazn.JAZNPermission</class>
          <name>umparams</name>
        </permission>
      </permissions>
    </grant>
  </jazn-policy>
</jazn-data>
```

Sample Code

To create Oracle Discussions elements using Oracle Discussions SDK, first initialize a **TdStore** object for a particular user. This user must be a valid Oracle Discussions user and be provisioned for the same email domain as the Oracle Discussions internal administrator. Once a **TdStore** for a user is created, invoke Oracle Discussions API through that store. Each Oracle Discussions store is associated with a message store where Oracle Discussions elements are saved.

Example: Create Facility, Board, Thread, and Message

Before executing this example, create a user provisioned for the same domain as the Oracle Discussions internal administrator (in this example, the domain would be example.com). Grant this user global administrator role. This example uses a user with ID "tdga".

The following example creates a facility, creates a board inside the facility, posts a topic to the board, and replies to the topic.

Example 3–5 TdSampleClient.java

```
package oracle.discussions.demo;

import java.io.*;
import java.util.*;
import javax.mail.internet.*;
import oracle.discussions.sdk.*;
import oracle.discussions.util.log.LogHandlerIf;
import oracle.discussions.util.log.logger.WriterLogHandler;

/**
  * Oracle Discussions Sample Code.
  */

 public class TdSampleClient
 {
   TdStoreManager _tdStoreManager = null ;
   TdStoreFactory _tdStoreFactory = null ;

   public TdSampleClient()
     throws TdException
   {
     LogHandlerIf logh = null;
     try {

       // Initialize a file writer
       // to create a log handler
       logh = new WriterLogHandler( new PrintWriter(System.out));
     }
     catch(Throwable t) {
       System.err.println(
         "ThreadedDiscussions: Could not initialize log handler: "+t);
     }

     // Initialize a new TdStoreManager
     // Set the logging level to the lowest
     Properties sysProps = System.getProperties();
     Map mTdicProps = new HashMap(sysProps);

     // Change the following properties according to your deployment
     mTdicProps.put("oracle.ocs.ldappool.host", "www.example.com");
     mTdicProps.put("oracle.ocs.ldappool.port", "389");
     mTdicProps.put("oracle.ocs.ldappool.user_dn", "cn=orcladmin");
     mTdicProps.put("oracle.ocs.ldappool.user_password","welcome1");
     mTdicProps.put("oracle.ocs.ldappool.initialsize","1");
     mTdicProps.put("oracle.ocs.ldappool.connectssl","false");
     mTdicProps.put("oracle.discussions.log.level","debug");
     mTdicProps.put("oracle.discussions.tdadmin.email",
       "discussions_ocs@example.com");
     mTdicProps.put("oracle.discussions.event.enabled", "false");
     mTdicProps.put(TdInitialContext.LOGGING_LEVEL,        "debug");
     mTdicProps.put(TdInitialContext.LOG_HANDLER_INSTANCE, logh);
     mTdicProps.put(TdInitialContext.APPLICATION_NAME,     "td test");

     TdInitialContext tdic = new TdInitialContext(mTdicProps);
     _tdStoreManager = TdStoreManager.createInstance(tdic);
     _tdStoreFactory = _tdStoreManager.getTdStoreFactory();

     System.out.println("Created a handle to Store Manager succesfully");
   }
```

```
public TdStore getTdStore(String username)
  throws TdException
{
  //
  //
  TdUserFactory tduf = TdUserFactory.getInstance();
  TdUser      tduser = tduf.getTdUserByNickname(username);

  return _tdStoreFactory.getTdStore(tduser);
}



public static void main(String argv[])
  throws Exception
{

  // Initialize Oracle Discussions by getting a TdStoreManager
  TdSampleClient tdsc = new TdSampleClient();

  // Get a TdStore for a given user.
  // Ensure that this user exists and is provisioned for Oracle Mail
  // with the same domain as the Oracle Discussions internal administrator.
  TdStore tds = tdsc.getTdStore("tdga");
  try {

    // Get a Facility
    TdFacility tdf = tds.getFacility("SampleFacility");
    if (tdf == null) {
      tdf = tds.createFacility(null,
        "SampleFacility",
        "sample facility",
        "This is a facility created by Oracle Discussions sample code");
    }

    // Create a board
    TdBoard tdb = tds.getBoard("/SampleFacility/SampleBoard");
    if (tdb != null) {
      tdf.deleteBoard(tdb);
    }
    tdb = tdf.createBoard("SampleBoard",
                          "Sample Board",
                          "Board created by Discussions sample code");

    // Use Board settings to specify board configuration options
    TdBoardSettings tdbs = tdb.getBoardSettings();
    tdbs.setOutgoingEmailDL("marco.carrer@oracle.com");
    tdb.updateContainerSettings(tdbs);

    // Post some of new threads
    MimeMessage mm = tdb.createMessageTopic();
    mm.setSubject("New Thread Posted by Td Sample Client");
    mm.setText("Hello!! This is a message posted " +
      "by the Oracle Discussions Sample Client");
    tdb.appendMessage(mm);
    mm = tdb.createMessageTopic();
    mm.setSubject("This is the latest thread posted " +
```

```
        "by the Oracle Discussions Sample Client");
      mm.setText("Hello!! This is a new post");
      tdb.appendMessage(mm);

      // Get all threads and append a reply to each one
      int i=0;
      TdIterator tdi = tdb.getThreads();
      while (tdi.hasNext()) {

        TdThread tdt = (TdThread) tdi.next();
        System.out.println("Thread "+(i++) +": "+tdt.getSubject());

        // Post a new reply on each thread
        MimeMessage mmr = tdb.createMessageReply(tdt);
        mmr.setText("Hello to you too. This is my reply!!");
        tdb.appendMessage(mmr);
      }

      // Favorites operations:
      // Add a favorite thread and get all threads
      tdi = tdb.getThreads();
      TdThread tdt = (TdThread) tdi.next();
      if (!tdt.isFavorite()) {
       tdt.addToFavorites();
      }
      TdFavoriteManager tdfm = tds.getFavoriteManager();
      TdIterator  tdiFavTdts = tdfm.getFavoriteThreads();
      int j=0;
      while (tdiFavTdts.hasNext()) {

        TdThread tdtFav = (TdThread) tdiFavTdts.next();
        TdLastPost tdlp = tdtFav.getLastPost();
        System.out.println("Fav Thread "+(j++) +
                           ": "+tdtFav.getSubject()+
                           ". Last Post by "+tdlp.getLastPostAuthor()+
                           " on "+tdlp.getLastPostDate());
      }
    }
    finally {
      if (tds != null) {
        tds.close();
      }
    }
  }
}
```

Configuring Properties

Set properties according to your deployment with the **TdInitialContext** class. The TdSampleClient.java example sets these properties in a hash map called mTdicProps. The following is a description of properties that have not been described in "Passing Initial Context Parameters":

Table 3–2

Property	Description
oracle.discussions.log.level	Sets the logging level of Oracle Discussions. Set the log level to "info". This can be set to "debug" when debugging the application. The TdSampleClient.java example has set it to "debug".
TdInitialContext.LOG_HANDLER_ INSTANCE	Creates a log handler and passes it to the Oracle Discussions server so that various activities can be logged. Refer to TdSampleClient.java to see how to create a logger.
TdInitialContext.APPLICATION_NAME	Passes the application name to Oracle Discussions. Various statements in the Oracle Discussions log file are generated using this application name.

Example: List Threads

The following example lists information about each thread in a particular board. It uses the example TdSampleClient.java.

Example 3–6 TdThreadListingDemo.java

```
package oracle.discussions.demo;

import java.io.*;
import java.text.NumberFormat;
import java.text.SimpleDateFormat;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import oracle.discussions.sdk.*;
import oracle.discussions.util.log.LogHandlerIf;
import oracle.discussions.util.log.logger.WriterLogHandler;

public class TdThreadListingDemo
{
  public static void main(String argv[])
   throws Exception
  {
    // Initialize Oracle Discussions by gettign a TdStoreManager
    // from TdSampleClient
    TdSampleClient tdsc = new TdSampleClient();

    // Get a TdStore for a given user
    TdStore tds = tdsc.getTdStore("tdga");
    try {

      // Get a Board
      TdBoard tdb = tds.getBoard("/Samplefacility/Sampleboard");
      System.out.println("Board appears to be null! :" + (tdb == null) );
      if (tdb != null)
      {
        System.out.println("Board appears to be null!");
        TdThread tdt = null;
        TdIterator tdit  = tdb.getThreads();
        if (tdit != null)
```

```
        {
          for (int i = 0; i < tdit.size(); i++)
          {
            tdt = (TdThread)tdit.next();
            System.out.println("Thread Subject :" + tdt.getSubject());
            System.out.println("Is locked :" + tdt.isLocked());
            System.out.println("Number of messages in this thread :" +
              tdt.getMessageCount());
            System.out.println("Number of replies to this thread :" +
              tdt.getNumberOfReplies());
            System.out.println("Is this thread a favorite for this user :" +
              tdt.isFavorite());
            System.out.println("Thread Content :" +  tdt.getBodyPlainText());
          }
        }
      }
    }
    catch(TdException tdex)
    {
      tdex.printStackTrace();
    }
  }
}
```

Example: List Grantee Roles

The following lists the grantee roles of a particular user. It uses the example TdSampleClient.java.

Example 3–7 TdGranteeRoleListingDemo.java

```
package oracle.discussions.demo;

import java.io.*;
import java.text.NumberFormat;
import java.text.SimpleDateFormat;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import oracle.discussions.sdk.*;
import oracle.discussions.util.log.LogHandlerIf;
import oracle.discussions.util.log.logger.WriterLogHandler;

public class TdGranteeRoleListingDemo
{
  public static void main(String argv[])
    throws Exception
  {

    // Initialize Oracle Discussions by getting a TdStoreManager
    // from TdSampleClient
    TdSampleClient tdsc = new TdSampleClient();

    // Get a TdStore for a given user
    TdStore tds = tdsc.getTdStore("tdga");

    try {
      List granteeRoles = tds.getGlobalAdminManager().listGranteeRoles();
      if(granteeRoles != null)
      {
```

```
        TdGranteeRole tdgr = null;
        for (int i = 0 ; i < granteeRoles.size(); i++)
        {
          tdgr = (TdGranteeRole) granteeRoles.get(i);
          System.out.println("The User holding role :" +
            tdgr.getGrantee().getUserNickname());
          System.out.println("The role :" + tdgr.getTdRole().toString());
          System.out.println("Is this role inherited :" + tdgr.isInherited());
        }
      }
    }
    catch(TdException tdex)
    {
      tdex.printStackTrace();
    }
  }
}
```

Executing Sample Code

Compile each example with the following command:

```
javac -classpath $CLASSPATH <Fully qualified Java file name>
```

$CLASSPATH is the classpath environment variable to which all the Java library files listed in "Setting Classpath" are added.

<*Fully qualified Java file name*> is one of the samples in this chapter.

Execute each example with the following command:

```
java -cp $CLASSPATH <Fully qualified class file name>
```

<*Fully qualified class file name*> is one of the samples in this chapter qualified with its package name.

Troubleshooting

oracle.discussions.sdk.TdException: Cannot connect to Oracle Java Mail Store: javax.mail.NoSuchProviderException: esmail.

If you receive this exception, ensure that ojdbc.jar appears in your classpath.

oracle.discussions.sdk.TdException: The name for this Category is taken. Please choose a different name for the Category at oracle.discussions.sdk.impl.handler.ContainerHandler.createFacility

If you receive this exception, check the facility and board names and display names. The container you are trying to create might be the same as those already existing in the system. Change the name of the container you are trying to create and re-run the program. Both the name and display name must be unique for all containers at the same level.

java.lang.SecurityException: access denied

Ensure that the Java library files that appear in the jazn-data.xml files point to the same Java library files in your classpath.

Oracle Discussions SDK Usage Tips

The following are some programming tips when using Oracle Discussions SDK

- Avoid hard-coding the Oracle Discussions internal administrator email address. Instead, retrieve this from Oracle Internet Directory.
- Avoid granting the orclguest user the role of Oracle Discussions administrator.
- You cannot use orcladmin to login to Oracle Discussions because orcladmin is not provisioned for any email domain.
- Carefully double check the system parameters passed to the initial context of Oracle Discussions. Oracle Discussions behavior might vary depending on the initial context parameters passed to it.
- Double-check JAZN permissions before executing the example code.

4

Building Applications with Oracle Discussions Web Services

This chapter describes the requirements and the tasks for building applications with Oracle Discussions Web Services. For details about Web Services specification and activity, please refer to http://www.w3.org/2002/ws. This chapter consists of the following sections:

- Requirements
- Invoking a Service

Requirements

This guide assumes that you are using Apache Axis as your Simple Object Access Protocol (SOAP) client. The version of Apache Axis should be at least 1.2.0. Invoking Oracle Discussions Web Services would be similar if any other Web Services client toolkit is used. For details on SOAP specification, refer to http://www.w3.org/TR/SOAP.

Accessing Deployed Web Services

Discussions Web services description and Web Service Description Language (WSDL) files will be available at the location `http://<<hostname:port>>/discussions/ws` on the instance of Oracle Collaboration Suite Applications tier.

WSDL describes Web Services in a structured way. WSDL describes the data types that the service uses, and the location of the service to the interface of the service. Axis supports WSDL generation using a Java2WSDL tool.

When a Web Service is deployed in an Axis engine, users may then access the URL of the service with a standard web browser and by appending `?WSDL` at the end of the URL. An automatically generated WSDL document that describes the service is displayed on the browser screen.

For example, to access Oracle Discussions AuthenticationService, enter the following:

```
http://<<hostname>>:<<port>>/discussions/ws/AuthenticationService?wsdl
```

or

```
http://<<hostname>>:<<port>>/discussions/ws/axis/services/AuthenticationService
```

If the service URL is accessed in a browser, then a message indicating that the endpoint is an Axis service is displayed. If `?wsdl` is appended at the end of the URL,

then Axis will automatically generate a service description for the deployed service, and return it as XML in the browser. The resulting description may be saved or used as input to proxy generation. The WSDL-generated URL can be given to the online partners, and they will be able to use it to access the Web Services with toolkits like .NET, SOAP::Lite, or any other software which supports WSDL.

A WSDL document will contain the appropriate WSDL types, messages, portType, bindings, and service descriptions to support a SOAP RPC and encoding Web Service. If the methods of the Web Services interface reference other classes, then the Java2WSDL tool will generate the appropriate XML types to represent the classes and any nested or inherited types.

For details about the WSDL specification, please refer to the following URL: http://www.w3.org/TR/wsdl.

Generating Client-Side Stub Classes

To access Web Services, invoke a remote call on SOAP or HTTP. For this, the client must have the Web Service endpoint location and a handle to the Web Services interface. These remote interfaces, client side service locators, and stubs are generated by WSDL2Java tool that is provided by Axis.

Generating the Client Stubs

Download the axis related jar files from the location http://ws.apache.org/axis.

The downloaded axis jars are to be unzipped into /usr/axis (Change the directory accordingly). To set the classpath, follow the instructions below (it is assumed that you use bash shell).

```
set AXIS_HOME=/usr/axis
set AXIS_LIB=$AXIS_HOME/lib
set AXISCLASSPATH=$AXIS_LIB/axis.jar:$AXIS_LIB/commons-discovery.jar
  $AXIS_LIB/commons-logging.jar:$AXIS_LIB/jaxrpc.jar:$AXIS_LIB/saaj.jar
  $AXIS_LIB/log4j-1.2.8.jar:$AXIS_LIB/xml-apis.jar:$AXIS_LIB/xercesImpl.jar
```

Along with these, include http_client.jar, mail.jar, activation.jar into the classpath (classpath of Axis).

Invoking WSDL2Java tool to generate client side stubs

Axis WSDL2Java tool can be found in org.apache.axis.wsdl.WSDL2Java. The basic invocation form looks like this:

```
%    $JAVA_HOME/bin/java -cp <<AXISCLASSPATH>>org.apache.axis.wsdl.WSDL2Java
--output <<GEN_DIR >>
--deployScope Session --NStoPkg  <<NAMESPACE-TO-PACKAGE-MAPPING>>
<<WSDL-FILE-URL>>
```

The description of the preceding code is as follows:

- JAVA_HOME is the location, where Java Development Kit (JDK) is installed.
- GEN_DIR is the directory in which the client stubs are to be generated.
- NAMESPACE-TO-PACKAGE-MAPPING is the mapping between the XML namespace to Java files. While converting the XML files (WSDL files are XML documents) to corresponding Java classes (client stubs), the XML namespace is converted into the provided package structure. For example, if the namespace to package mapping is provided as following:

  ```
  "http://xmlns.oracle.com/discussions/ws"="oracle.discussions.ws"
  ```

Then the generated client stubs are placed in the `oracle/discussions/ws/package` structure.

- `WSDL-FILE-URL` is the URL at which the WSDL files are available. For example, if Oracle Discussions Web Services are deployed on machine `stacx01.us.oracle.com` at port 8888, then Authentication Service WSDL-FILE-URL would be as follows:

  ```
  http://stacx01.us.oracle.com: 8888/discussions/ws/AuthenticationService?wsdl
  ```

Compiling the client stubs

The client stubs are to be compiled by invoking the following command:

```
Javac -classpath $AXISCLASSPATH -d <<DESTINATION-CLASSES-DIRECTORY>>  *java
```

The invocation is to be done in the directory in which the client stub source files are present.

`DESTINATION-CLASSES-DIRECTORY` is the directory in which the class files are to be placed.

Once the client stubs classes are generated, include the directory containing the stub classes into the classpath. Now you are ready to compile and run a web services java client, to access Oracle Discussions Web Services.

Invoking a Service

The following is an example for invoking a Web Service. This example does the following operations:

- Log in as `td_superuser`, by supplying username and password in plain text. (`td_superuser` and `welcome1` are the user name and password considered in the Example 4–1).
- Retrieve the cookie on successful login.
- Set the retrieved cookie on the Category Service client stub.
- Prepare the required input parameters and invoke a Web Service call on the category service.
- Print the properties on the returned category (and invoke any other Web Services).
- Log out the user.

Example 4–1 Invoking a Web service

```
package oracle.discussions.ws;

import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
```

```
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.CategoryServiceSoapBindingStub;
import oracle.discussions.ws.CategoryServiceServiceLocator;
import oracle.discussions.ws.CategoryService;

/*
 * This method invokes various web services methods, to retrieve the results and
 * check if the results are fine.
 */
public class WSClient
{
  public static void main(String args[])
  {
     try
     {
        //STEP - 1: User Login
        //Create the service locator for authentication service. The locator
        //contains information about the webservices endpoint.
         AuthenticationServiceServiceLocator assl = new
                  AuthenticationServiceServiceLocator();

        //Get the handle to the actual Web Services interface
         AuthenticationService as = assl.getAuthenticationService();
        //Indicate to the server that the client is interested in maintaining
        //session.
        //HTTP is stateless and the user state is maintained in the session.
        //Unless the user is willing to participate in a session, user state
        //cannot be preserved.
         ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);

        //Invoke the actual login Web Services call
         as.login("td_superuser","welcome1");
        //Retrieve the cookie. The cookie is set on successful authentication.
         String cookie = (String)((AuthenticationServiceSoapBindingStub)as).
                  _getCall().getMessageContext()
                  .getProperty(HTTPConstants.HEADER_ COOKIE);
         System.out.println("Cookie : " + cookie);

        //STEP - 2: Category Creation
        //Create an instance of the category service locator.
         CategoryServiceServiceLocator catssl = new
                  CategoryServiceServiceLocator();
        //Get the handle to the category service interface.
         CategoryService cats = catssl.getCategoryService();
        //Indicate to the server that the user is willing to participate in the
        // session.
         ((CategoryServiceSoapBindingStub)cats).setMaintainSession(true);
        //Set the cookie retrieved in the call to login webservice.
        //The cookie asserts user's identity to the server.
         ((javax.xml.rpc.Stub)cats)._setProperty(HTTPConstants.HEADER_
                COOKIE,cookie);

        //Define a category definition bean. The bean defines the category
        //properties.
         CategoryDefinition cd = new CategoryDefinition();
         cd.setName("Test Category");
         cd.setDisplayName("Test Category");
         cd.setDescription("Category created for testing ws java client");
         cd.setWithAnnouncementBoard(true);
        //Invoke the webservices call on the web services interface.
```

```
 Category category = cats.create(-1,cd);

//If the category creation is successful, a category bean is returned,
//depicting the category object created in the TD repository.
 if(category != null)
 {
     System.out.println("Creation Date :" +
           category.getCreationDate().getTime());
     System.out.println("Creator :" + category.getCreator());
     System.out.println("Description :" + category.getDescription());
     System.out.println("Display Path :" + category.getDisplayPath());
     System.out.println("Display Name :" + category.getDisplayName());
     LastPost lp = category.getLastPost();
     if(lp != null)
            {
                System.out.println("Author :" + lp.getLastpostAuthor());
                System.out.println("Date :" + lp.getLastpostDate());
                System.out.println("Forum ID :" +
                          lp.getLastpostForumId());
                System.out.println("Topic id :" +
                          lp.getLastpostTopicId());
                System.out.println("Message Id :" +
                          lp.getLastpostMessageId());
            }

            System.out.println("Is Favorite :" + category.isFavorite());
            System.out.println("Is New :" + category.isNewFolder());
            System.out.println("Is New Since Last Visit :" +
                    category.isNewSinceLastVisit());
            System.out.println("Is New Since Session :" +
                    category.isNewSinceSession());
            System.out.println("Name :" + category.getName());
            System.out.println("Id :" + category.getId());
            System.out.println("is category type :" +
                    category.isCategoryType());
            System.out.println("Web ui URL :" + category.getWebuiURL());
            System.out.println("Rss URL :" + category.getRssURL());
            System.out.println("Has ann forum :" +
                    category.isAnnouncementsBoardPresent());
            System.out.println("Parent id :" + category.getParentId());
      }

     //Invoke as many web service operations as you want..
     //.........................
     //.........................

     //STEP - 2: User Logout
      System.out.println("User logout.");
     //Once all the webservices operations are done, invoke logout
     //operation.
     //On logout, user state is destroyed and the cookie is
     //invalidated.
      as.logout();
      System.out.println("Done");
  }
  catch(TdWSException ex)
  {
        System.out.println("Error Code :" + ex.getErrorCode());
        System.out.println("Error Message :" + ex.getErrorMessage());
        String[] trace = ex.getServerStackTrace();
```

```
                if(trace != null)
                {
                   for(int i = 0 ; i < trace.length ; i++)
                   System.out.println("trace[" + i + "]" + trace[i]);
                }
            }
    catch(Exception ex)
    {
       System.out.println("Never went into tdex :" + ex.getMessage());
       ex.printStackTrace();
    }
  }
}
```

Compile the Web Services Java program by invoking the following command:

```
Javac -classpath $AXISCLASSPATH  WSClient.java
```

`WSClient .java` is the name of the program in Example 4–1. Append the directory that contains the compiled class file to the `AXISCLASSPATH`.

The compiled Java program is run by using the following command:

```
Java -cp $AXISCLASSPATH  WSClient
```

The sample output of the program code above is given below

```
Cookie : JSESSIONID=8c57046a22b882c3e6c4ee72400ebbcfd8eeb777becf
Creation Date :Thu Feb 23 23:50:46 PST 2006
Creator :td_superuser
Description :Category created for testing ws java client
Display Path :/WS Category
Display Name : WS Category
Is Favorite :false
Is New :false
Is New Since Last Visit :false
Is New Since Session :false
Name : WS Category
Id :85764
is category type :true
Web ui URL :http://stacx01:8888/discussions/app/mainAction.do?fid=85764
Rss URL :http://stacx01:8888/discussions/rss/facility?fid=85764
Has ann forum :true
Parent id :-1
User logout.
Done
```

Note: Please ensure that the users mentioned in the sample program above are created and are granted relevant roles (`td_superuser` is granted a system administrator role in the provisioning console). Also please ensure that the category name or the display name are not same as in an already existing category in the same container in which the current category is being created. Please verify the user password before executing the program.

5

Understanding the Authentication Service

The Authentication Service interface provides methods related to user login and logout. The `login()` method should be invoked before invoking any other Web Service operations. The `logout()` method is to be invoked after the user finishes with the Web Service operations.

This chapter addresses User Login and Logout.

User Login and Logout

The following is an example for logging in:

Example 5–1 Logging in

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;

public class WSClient
{
  private static AuthenticationService as = null;
  public static String getCookie() throws Exception
  {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about
      //the webservices endpoint.
       AuthenticationServiceServiceLocator assl = new
                          AuthenticationServiceServiceLocator();
      //Get the handle to the actual webservices interface.
       as = assl.getAuthenticationService();
      //Indicate to the server that the client is interested in maintaining
      //session
      //HTTP is stateless and the user state is maintained in the session.
      //Unless the user is willing to participate in a session, user state cannot
      //be preserved.
       ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
```

```
            //Invoke the actual login webservices call.
             as.login("td_superuser","welcome1");
            //Retrieve the cookie. The cookie is set on successful authentication.
             String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                      ._getCall().getMessageContext()
                                      .getProperty(HTTPConstants.HEADER_COOKIE);
             System.out.println("Cookie : " + cookie);
             return cookie;
         }

         public static void main(String args[])
        {
          try
          {
            //STEP - 0: User Login
             String cookie = getCookie();

            //STEP - 1: Invoke web service related operations.
            //.............................
            //.............................

            //STEP - 2: User Logout
             logout();
        }
        catch(TdWSException ex)
        {
          System.out.println("Error Code :" + ex.getErrorCode());
          System.out.println("Error Message :" + ex.getErrorMessage());
          String[] trace = ex.getServerStackTrace();
          if(trace != null)
          {
             for(int i = 0 ; i < trace.length ; i++)
                 System.out.println("trace[" + i + "]" + trace[i]);
          }
        }
      catch(Exception ex)
      {
        System.out.println("Never went into tdex :" + ex.getMessage());
        ex.printStackTrace();
      }
    }

    public static void logout() throws Exception
    {
      System.out.println("User logout.");
      //Once all the webservices operations are done, invoke logout
      //operation.
      //On logout, user state is destroyed and the cookie is invalidated.
      as.logout();
      System.out.println("Done");
    }
}
```

Note: tFor all the examples mentioned in this guide, the package structure need not be `oracle.discussions.ws`. The package structure will be same as the package-to-namespace-mapping provided during the WSDL2Java invocation or the package structure in which the compiled artifacts are present.

The `login()` method authenticates the user to be a valid Oracle Internet Directory user and if the authentication is successful, a cookie is returned to the user. The same cookie is to be passed between the user (client) and server for all Web Service invocations, until the user logs out.

Note: User nickname and password are to be supplied in plain text.

Note: Without setting the cookie and indicating the server to maintain the session, the user will not be able to invoke any operation despite logging in successfully.

The `logout()` method invalidates the user session. It is invoked after the user completes Web Services method invocations. Once logged out, the user will not be able to invoke any Web Service until the user logs in again. The cookie created during login is destroyed.

6

Understanding Service To Service Authentication

The Service to Service (S2S) authentication framework provides a means for a trusted partner application to establish user sessions with a trusting provider application on behalf of its users, without having to supply any credentials for the users individually. The trusting partner application is any application (registered or configured with Oracle Internet Directory) that is capable of establishing a client SOAP over HTTP Web Service connection with authentication headers.

Registering Partner Applications with Oracle Internet Directory

The following steps describe how to register a partner application with the S2S authentication framework.

1. Create the following entry in Oracle Internet Directory:

   ```
   dn: cn=MyApplicationProductName,cn=Products, cn=OracleContext
   objectClass: orclContainer
   objectClass: top
   ```

   MyApplicationProductName is the product name (or category) of your application.

2. Create the following entry in Oracle Internet Directory:

   ```
   dn: orclApplicationCommonName=MyAppName,
   cn=MyApplicationProductName,cn=Products,
   cn=OracleContext
   objectClass: orclApplicationEntity
   objectClass: top
   orclApplicationCommonName: MyAppName
   userpassword: ApplicationPassword
   ```

   MyAppName is the name of your application. *ApplicationPassword* is the password to access your application.

3. To the *MyAppName* entity you have just added, find the property `orcltrustedapplicationgroup`. Set the value of this proper to the name of the Trusted Applications group, which should be the following:

   ```
   cn=trusted applications,cn=groups,cn=oraclecontext
   ```

4. Find the following entry in Oracle Internet Directory:

   ```
   dn: cn=Trusted Applications,cn=Groups,cn=OracleContext
   ```

To the entry that you have just found, add the following line to `uniquemember` (all in lower case, line breaks added for clarity):

```
orclapplicationcommonname=myappname,
cn=myapplicationproductname,cn=products,cn=oraclecontext
```

Only application entities listed in `uniquemember` are allowed to authenticate using S2S. To complete the authentication, the application entity must have the member `orcltrustedapplicationgroup` in its Oracle Internet Directory entry containing the location of the trusted applications group. The password in the application entity entry will be used to authenticate.

Use Oracle Directory Manager or the command-line tools provided by Oracle Internet Directory to add and modify entries in Oracle Internet Directory. For more information about these tools, see Chapter 4, "Directory Administration Tools" in Oracle Internet Directory Administrator's Guide.

Setting Oracle-Specific Headers

On an HTTP response object, set the Oracle-specific header ORA_S2S_PROXY_USER to the value of a nickname of an Oracle Collaboration Suite user. The S2S authentication framework will use this user to authenticate the partner application.

Remember to set any other headers required by the HTTP authentication method that you are using. Oracle HTTP Server provides the HTTPClient package with which you can set this header. For more information about this package, see the HTTPClient API Reference Javadoc.

Logging in with S2S Authentication Service

Call the `login()` method of S2SAuthentication Service. The following code extract demonstrates how to do this:

Example 6–1 Logging In

```
  // Initialize the service locator for the S2SAuthentication service.
  // The service locator contains the Web service endpoint URL
  // in the form of
  // http://<midtier instance name>:<port>/
  // ocw/services/S2SAuthenticationService
   S2SAuthenticationServiceServiceLocator s2sassl =
                new S2SAuthenticationServiceServiceLocator();

  // Retrieve a reference to the Web service's remote interface
  // from the service locator
S2SAuthenticationService s2sas = s2sassl.getS2SAuthenticationService();

  // Indicate the client's willingness to participate in the session.
  // Unless it is set to true, the server assumes that client is not
  // participating in the session.

 // As HTTP is stateless, each Web service invocation will be
 // different and the user's state will not be
 // maintained across different Web service invocations.
 ((S2SAuthenticationServiceSoapBindingStub)s2sas).
                       setMaintainSession(true);
 // Invoke the login method to authenticate the user.
 // User nickname and password are sent as plain text.
  s2sas.login();
```

Retrieve the authentication cookie from the service as follows:

```
String cookie = (String)((S2SAuthenticationServiceSoapBindingStub)as).
_getCall().getMessageContext().
getProperty(HTTPConstants.HEADER_COOKIE);
```

Your partner application may now invoke any Oracle Discussions Web Service (with the authentication cookie) until the user session expires or the `logout()` method of S2S Authentication Service is called.

Using S2SAuthenticationServiceClient

A number of Web Service clients provide no published means to access the underlying HTTP transport to allow setting of the HTTP headers and so forth. Therefore, these clients are incapable of invoking the S2S Authentication Service directly by setting `PROXY HEADER` into the request. As a workaround, Oracle ships a S2S servlet that can be accessed through a regular HTTP client library, such as `HttpClient`. The Web Services client can invoke S2SAuthenticationServletClient, which can be configured to connect to the S2SServlet and obtain the cookie. For further Web Services, the Web Services client can then use the session cookie returned through authentication to the `S2SServlet`.

S2SAuthenticationServiceClient connects to the servlet on a HTTP connection. In the following sample code, the code in bold is to be replaced with appropriate text according to the environment on which the program is being tested:

```
import java.net.URL;
import oracle.discussions.ws.client.S2SAuthenticationServiceClient;

public class S2STest
{
  public static void main(String args[]) throws Exception
  {
     try
     {
        //Retrieve the caller application distinguished name from Oracle Internet
        //Directory
         String appName = "orclApplicationCommonName = om4c.stacx01.us.oracle.com,
                  cn=MidtierInstances,cn=CollaborativeWorkspaces,cn=Products,
                                                  cn=OracleContext";
        //appName should be in lowercase
         appName = appName.toLowercase();
         System.out.println("Using the distinguished name:"+appName);
         String appPwd = "I886TG5UVFjP";
        //Provide the relevant application server URL on whicn the Authentication
        //Service is deployed
         String appURL =
               "http://host:port/discussions/ws/S2SAuthenticationServlet";
        //Provide the user nickname, on behalf of whom the caller application
        //proxies. The user should be a valid Oracle Collaboration Suite user.
        //td_superuser is a sample user nickname.
         String userNickname = "td_superuser";
        //Invoke the init method on the Client program, to set the cookie
        // handling mechanism
              S2SAuthenticationServiceClient.init();
             //Invoke the method to connect to the http endpoint servlet and
             //retrieve the cookies
             String cookie = S2SAuthenticationServiceClient.getSessionCookies("td_
                           superuser",appName,appPwd,"",new URL(appURL));
```

```
            System.out.println("Got cookies :" + cookies);
          }
         catch
         {
            System.out.println("Error Message:"+tdex.getMessage());
          }
      }
}
```

To locate the Oracle Internet Directory node from which the `appName` and `appPwd` are to be taken, select **Entry Management**, **Oracle Context**, **Products**, **Collaborative Workspaces**, **MidtierInstances**. The `appName` should be taken from the first node in the **Midtier Instances** tab. The `appPwd` should be picked from the node (`orclResourceName = OIDProperties`), that is, beneath the first node. The realm is an empty string.

Note: Ensure that Axis related jars and `javax mail.jar` are in the classpath for executing the client.

7

Understanding the Container Service

The Container Service interface provides methods common to both Category and Forum Services. It acts as a logical parent service to both the services.

The Container Service interface consists of methods for the following operations:

- Retrieving parent category of a container
- Retrieving the information on the last post in a container
- Searching a container
- Retrieving announcements forum in a container
- Listing or updating roles of users on a container

This chapter addresses the following tasks:

- Searching Oracle Discussion Containers
- Updating Grantee Roles

Searching Oracle Discussion Containers

The following is an example to search Oracle Discussions Containers by using a search term as the search criterion:

Example 7–1 Searching Oracle Discussion Containers

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.ContainerServiceSoapBindingStub;
import oracle.discussions.ws.ContainerServiceServiceLocator;
import oracle.discussions.ws.ContainerService;

public class WSClient
{
  private static AuthenticationService as = null;
  public static String getCookie() throws Exception
```

```
{
  //STEP - 0: User Login
  //Create the service locator for authentication service. The locator
  //contains information about the web Services endpoint
   AuthenticationServiceServiceLocator assl = new
               AuthenticationServiceServiceLocator();
  //Get the handle to the actual webservices interface.
   as = assl.getAuthenticationService();
  //Indicate to the server that the client is interested in maintaining
  //session.
  //HTTP is stateless and the user state is maintained in the session.
  //Unless the user is willing to participate in a session, user state
  //cannot be preserved.
   ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
  //Invoke the actual login webservices call.
   as.login("td_superuser","welcome1");
  //Retrieve the cookie. The cookie is set on successful authentication.
   String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                              ._getCall().getMessageContext()
                              .getProperty(HTTPConstants.HEADER_COOKIE);
  System.out.println("Cookie : " + cookie);
  return cookie;
}
public static void main(String args[])
{
  try
  {
    //STEP - 0: User Login
     String cookie = getCookie();

    //STEP - 1: Locate the Web Services end point.
    //The Web Services locator contains the webservices end point address.
    //Hence instantiate the service locator.
     ContainerServiceServiceLocator fssl = new
                            ContainerServiceServiceLocator();

   //Get a handle to the actual web service.
    ContainerService fs = fssl.getContainerService();

   //Indicate to the server that the user is willing to participate in the
   //session.
   //Since HTTP is stateless, all the client data is maintained in the
   //session.
    ((ContainerServiceSoapBindingStub)fs).setMaintainSession(true);

   //Set the cookie retrieved in the call to login webservice.
   //The cookie asserts user's identity to the server.
    ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_
                                        COOKIE,cookie);

   //Invoke the actual web services call.
    SearchTerm st = new SearchTerm();
    st.setAuthor("td_superuser");
   //The container ID needs to be changed accordingly
    ForumMessage sr[] = fs.search(new long[]{15524},st,true);
    if(sr != null)
    {
      for(int i = 0; i < sr.length; i++)
      {
        System.out.println("Subject :" + sr[i].getSubject());
```

```
                System.out.println("Plain Text :" + sr[i].getBodyPlainText());
            String[] frmAdd = sr[i].getFromAddresses();
            if(frmAdd != null)
            {
                for(int j = 0 ; j < frmAdd.length ; j++)
                    System.out.println("From Address :" + frmAdd[j]);
            }

            System.out.println("Level :" + sr[i].getLevel());
            System.out.println("No of replies :" + sr[i].getNumberOfReplies());
            System.out.println("Oracle Msg Id :" + sr[i].getMessageId());
            System.out.println("is new since session :" +
                                    sr[i].isNewSinceSession());
            System.out.println("is new since last visit :" +
                                    sr[i].isNewSinceLastVisit());
            System.out.println("is new " + sr[i].isNewMessage());
            System.out.println("Has Attachments :" + sr[i].isHasAttachments());
            System.out.println("X Priority :" + sr[i].getXPriority());
            System.out.println("Sent date :" + sr[i].getSentDate());
            System.out.println("Size :" + sr[i].getSize());
            System.out.println("HTML ?  :" + sr[i].isHTMLContentType());
            System.out.println("WebUI URL :" + sr[i].getWebUIUrl());
          }
         }


         //STEP - 2: User Logout
         logout();
        }
        catch(TdWSException ex)
        {
            System.out.println("Error Code :" + ex.getErrorCode());
            System.out.println("Error Message :" + ex.getErrorMessage());
            String[] trace = ex.getServerStackTrace();
            if(trace != null)
            {
                for(int i = 0 ; i < trace.length ; i++)
                    System.out.println("trace[" + i + "]" + trace[i]);
            }
        }
        catch(Exception ex)
        {
            System.out.println("Never went into tdex :" + ex.getMessage());
            ex.printStackTrace();
        }
    }

    public static void logout() throws Exception
    {
          System.out.println("User logout.");
        //Once all the webservices operations are done, invoke logout
        //operation.
        //On logout, user state is destroyed and the cookie is invalidated.
         as.logout();
         System.out.println("Done");
    }
}
```

The `search()` method in the `ContainerService` interface searches the containers specified by `lContainerId` for search term `st`. If no container IDs are supplied then

all the Oracle Discussions containers, for which the user has access, are searched. If a container ID array is supplied, then only those containers are searched.

The method also accepts a boolean parameter that indicates if the announcements are also to be searched. Wild-card searches can also be performed. This method has the following three attributes:

- `lContainerId` is the array of container IDs specifying search scope.
- `st` is the search term specifying search context.
- `bSearchAnnouncements` is the boolean value indicating if announcements are also to be searched.

`The ContainerService.search()` returns `sr` which is an array of forum message beans that represents search results.

The sample output for Example 7–1 is as follows:

```
Subject :Topic-1
Plain Text :Topic-1
From Address :Admin <td_superuser@stacx01.us.oracle.com>
Level :0
No of replies :0
Oracle Msg Id :161673
is new since session :false
is new since last visit :false
is new false
Has Attachments :false
X Priority :1
Sent date: Sun Feb 26 19:54:18 PST 2006
Size :404
HTML ?  :false
WebUI URL
:http://stacx01:8888/discussions/app/mainAction.do?fid=15524&tid=161673&mid=161673
#161673
User logout.
Done
```

Updating Grantee Roles

The following is an example for updating and listing grantee roles:

Example 7–2 Updating and Listing Grantee Roles

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.ContainerServiceSoapBindingStub;
import oracle.discussions.ws.ContainerServiceServiceLocator;
import oracle.discussions.ws.ContainerService;

public class WSClient
```

```
{
  private static AuthenticationService as = null;

  public static String getCookie() throws Exception
   {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about the Web Services endpoint
       AuthenticationServiceServiceLocator assl = new
                        AuthenticationServiceServiceLocator();
      //Get the handle to the actual webservices interface.
       as = assl.getAuthenticationService();
      //Indicate to the server that the client is interested in maintaining
      //session.
      //HTTP is stateless and the user state is maintained in the session.
      //Unless the user is willing to participate in a session, user state cannot
      //be preserved.
       ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
      //Invoke the actual login webservices call.
       as.login("td_superuser","welcome1");
      //Retrieve the cookie. The cookie is set on successful authentication.
       String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                        ._getCall().getMessageContext()
                                  .getProperty(HTTPConstants.HEADER_COOKIE);
       System.out.println("Cookie : " + cookie);
       return cookie;
   }

   public static void main(String args[])
   {
     try
     {
       //STEP - 0: User Login
        String cookie = getCookie();

       //STEP - 1: Locate the Web Services end point.
       //The web services locator contains the webservices end point address.
       //Hence instantiate the service locator.
        ContainerServiceServiceLocator fssl = new
                              ContainerServiceServiceLocator();

       //Get a handle to the actual web service.
        ContainerService fs = fssl.getContainerService();

       //Indicate to the server that the user is willing to participate in the
       //session.
       //Since HTTP is stateless, all the client data is maintained in the
       //session.
        ((ContainerServiceSoapBindingStub)fs).setMaintainSession(true);
       //Set the cookie retrieved in the call to login webservice.
       //The cookie asserts user's identity to the server.
        ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_COOKIE,cookie);

       //Invoke the actual web services call.
        Role role1 = new Role();
        role1.setId(1);
        role1.setName("oracle.discussions.roles.board_reader");
        GranteeRole[] grArray = new GranteeRole[3];
        GranteeRole gr1 = new GranteeRole();
       //The user ID and the email ID neeeds to be changed accordingly
```

```
                gr1.setEmail("tduser1@stacx01.us.oracle.com");
                gr1.setRole(role1);

                Role role2 = new Role();
                role2.setId(8);
                role2.setName("oracle.discussions.roles.board_moderator");
                GranteeRole gr2 = new GranteeRole();
                gr2.setEmail("tduser@stacx01.us.oracle.com");
                gr2.setRole(role2);
                Role role3 = new Role();
                role3.setId(8);
                role3.setName("oracle.discussions.roles.board_moderator");
                GranteeRole gr3 = new GranteeRole();
                gr3.setEmail("td_superuser@stacx01.us.oracle.com");
                gr3.setRole(role3);
                grArray[0] = gr1;
                grArray[1] = gr2;
                grArray[2] = gr3;

                GranteeRole[] gr = fs.updateGranteeRoles(6732,grArray);
                if(gr != null)
                {
                   for(int i = 0 ; i < gr.length ; i++)
                   {
                     System.out.println(gr[i].getRole().getName());
                     System.out.println(gr[i].getEmail());
                   }
                 }
               //STEP - 2: User Logout
                logout();
            }
            catch(TdWSException ex)
            {
              System.out.println("Error Code :" + ex.getErrorCode());
              System.out.println("Error Message :" + ex.getErrorMessage());
              String[] trace = ex.getServerStackTrace();
              if(trace != null)
              {
                for(int i = 0 ; i < trace.length ; i++)
                   System.out.println("trace[" + i + "]" + trace[i]);
              }
            }
           catch(Exception ex)
          {
            System.out.println("Never went into tdex :" + ex.getMessage());
            ex.printStackTrace();
          }
      }

      public static void logout() throws Exception
      {
          System.out.println("User logout.");
         //Once all the webservices operations are done, invoke logout
         //operation.
         //On logout, user state is destroyed and the cookie is invalidated.
          as.logout();
          System.out.println("Done");
      }
  }
```

The `GranteeRole` class represents the role of a Oracle Discussions user in the container.

The `updateGranteeRoles()` method edits the roles of users in the container specified by `lContainerId`. While invoking the operation, it is important that the grantee role information of the user invoking the action is also specified. The roles inherited from the parent container cannot be edited at this container level. An exception is raised if a user tries to edit his own role. This method has the following two attributes:

- `lContainerId` is the ID that specifies the container on which the roles are to be edited.
- `grRoleArray` is an array of beans that represents grantee roles.

This method returns `grArrayRes` which is an array of beans representing updated grantee roles.

The `listGranteeRoles()` method returns the roles of all the users on the container specified by the `lContainerId`.

The sample output for Example 7–2 is as follows:

```
Cookie : JSESSIONID=8c57046a22b8ed03c740184043cca9fb9c7877bee658
Updating the grantee role information.
Updated grantee role information successfully.
```

8

Understanding the Category Service

The Category Service provides the following category-related operations:

- Creating a category
- Fetching category details, given the category ID
- Listing containers under a given category
- Updating the category definition
- Deleting a category

Apart from the preceding operations, it also provides bulk operations to create and delete multiple categories in a single invocation. This chapter addresses the task of Creating a Category.

Creating a Category

The following is an example for creating a category:

Example 8–1 Creating a Category

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.CategoryServiceSoapBindingStub;
import oracle.discussions.ws.CategoryServiceServiceLocator;
import oracle.discussions.ws.CategoryService;

public class WSClient
{
  private static AuthenticationService as = null;

  public static String getCookie() throws Exception
  {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about the Web Services endpoint.
```

```
     AuthenticationServiceServiceLocator assl = new
                     AuthenticationServiceServiceLocator();
    //Get the handle to the actual webservices interface.
     as = assl.getAuthenticationService();
    //Indicate to the server that the client is interested in maintaining
    //session.
    //HTTP is stateless and the user state is maintained in the session.
    //Unless the user is willing to participate in a session, user state
    //cannot be preserved.
     ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
    //Invoke the actual login webservices call.
     as.login("td_superuser","welcome1");
    //Retrieve the cookie. The cookie is set on successful authentication
     String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                           ._getCall().getMessageContext()
                          .getProperty(HTTPConstants.HEADER_COOKIE);
     System.out.println("Cookie : " + cookie);
     return cookie;
 }

public static void main(String args[])
{
   try
   {
     //STEP - 0: User Login
      String cookie = getCookie();

     //STEP - 1: Locate the web services end point.
     //The web services locator contains the webservices end point
     //address
     //Hence instantiate the service locator.
      CategoryServiceServiceLocator fssl = new
                       CategoryServiceServiceLocator();

     //Get a handle to the actual web service.
      CategoryService fs = fssl.getCategoryService();
     //Indicate to the server that the user is willing to participate in
     //the session.
     //Since HTTP is stateless, all the client data is maintained in the
     //session.
      ((CategoryServiceSoapBindingStub)fs).setMaintainSession(true);

     //Set the cookie retrieved in the call to login webservice.
     //The cookie asserts user's identity to the server.
      ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_
                                          COOKIE,cookie);

     //Invoke the actual web services call.
      CategoryDefinition cDefn = new CategoryDefinition();
      cDefn.setName("My Category");
      cDefn.setDisplayName("My Category");
      cDefn.setDescription("My Category");
      cDefn.setWithAnnouncementBoard(true);
     //- 1 indicates that a root level category is created
      Category aContainer = fs.create(-1,cDefn);
      if(aContainer != null)
     {
       System.out.println("Container Type :" +
                            aContainer.getContainerType());
       System.out.println("Creation Date :" +
```

```
                                    aContainer.getCreationDate().getTime());
        System.out.println("Creator :" + aContainer.getCreator());
        System.out.println("Description :" + aContainer.getDescription());
        System.out.println("Display Path :" + aContainer.getDisplayPath());
        System.out.println("Display Name :" + aContainer.getDisplayName());
        LastPost lp = aContainer.getLastPost();
        if(lp != null)
        {
            System.out.println("Author :" + lp.getLastpostAuthor());
            System.out.println("Date :" + lp.getLastpostDate());
            System.out.println("Forum ID :" + lp.getLastpostForumId());
            System.out.println("Topic id :" + lp.getLastpostTopicId());
            System.out.println("Message Id :" + lp.getLastpostMessageId());
        }

        System.out.println("Is Favorite :" + aContainer.isFavorite());
        System.out.println("Is New :" + aContainer.isNewFolder());
        System.out.println("Is New Since Last Visit :" +
                                aContainer.isNewSinceLastVisit());
        System.out.println("Is New Since Session :" +
                                 aContainer.isNewSinceSession());
        System.out.println("Name :" + aContainer.getName());
        System.out.println("Id :" + aContainer.getId());
        System.out.println("is category type :" + aContainer.isCategoryType());
        System.out.println("Web ui URL :" + aContainer.getWebuiURL());
        System.out.println("Rss URL :" + aContainer.getRssURL());
        System.out.println("Has ann forum :" +
                               aContainer.isAnnouncementsBoardPresent());
        System.out.println("Parent id :" + aContainer.getParentId());
     }

    //STEP - 2: User Logout
     logout();
   }
   catch(TdWSException ex)
   {
       System.out.println("Error Code :" + ex.getErrorCode());
       System.out.println("Error Message :" + ex.getErrorMessage());
       String[] trace = ex.getServerStackTrace();
       if(trace != null)
       {
           for(int i = 0 ; i < trace.length ; i++)
               System.out.println("trace[" + i + "]" + trace[i]);
       }
   }
   catch(Exception ex)
   {
       System.out.println("Never went into tdex :" + ex.getMessage());
       ex.printStackTrace();
   }
}
public static void logout() throws Exception
{
     System.out.println("User logout.");
    //Once all the webservices operations are done, invoke logout
    //operation.
    //On logout, user state is destroyed and the cookie is invalidated.
     as.logout();
     System.out.println("Done");
}
```

```
}
```

The `create()` method of the `categoryService` interface creates a new category in the repository. If the value of `lParentId` is `-1` then a root-level category is created. Otherwise, a category is created under the container represented by `lParentId`. An exception is raised if `lParentId` does not pertain to a valid category.

The properties of the category to be created are specified in the category definition bean (`cDefn` in this example).

> **Note:** The category name and the category display name are mandatory. They are unique arguments in the category definition bean and hence they cannot be null.

The sample output for Example 8–1 is as follows:

```
Cookie : JSESSIONID=8c57046a22b8a4e4d7408338486da27bb6988d0322f2
Container Type :1
Creation Date :Sun Feb 26 20:44:53 PST 2006
Creator :td_superuser
Description :My Category
Display Path :/My Category
Display Name :My Category
Is Favorite :false
Is New :false
Is New Since Last Visit :false
Is New Since Session :false
Name :My category
Id :85770
is category type :true
Web ui URL :http://stacx01:8888/discussions/app/mainAction.do?fid=85770
Rss URL :http://stacx01:8888/discussions/rss/facility?fid=85770
Has ann forum :true
Parent id :-1
User logout.
Done
```

9

Understanding the Forum Service

The Forum Service interface represents consists of methods that operate on Oracle Discussions Forum. The Web Services interface provides the following operations:

- Creating a forum
- Updating a forum
- Retrieving or updating forum settings
- Listing contents of the forum
- Deleting a forum

This interface also provides bulk operations to create or delete multiple forums in a single invocation.

This chapter addresses the following tasks:

Creating a Forum

The following is an example for creating a forum with forum definition bean:

Example 9–1 Creating a Forum with a Valid Forum Definition Bean

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.ForumServiceSoapBindingStub;
import oracle.discussions.ws.ForumServiceServiceLocator;
import oracle.discussions.ws.ForumService;

public class WSClient
{
```

```
      private static AuthenticationService as = null;
      public static String getCookie() throws Exception
      {
        //STEP - 1: User Login
        //Create the service locator for authentication service. The locator
        //contains information about the Web Services endpoint
         AuthenticationServiceServiceLocator assl = new
                       AuthenticationServiceServiceLocator();
        //Get the handle to the actual webservices interface.
         as = assl.getAuthenticationService();
        //Indicate to the server that the client is interested in maintaining
        //session.
        //HTTP is stateless and the user state is maintained in the session.
        //Unless the user is willing to participate in a session, user state
        //cannot be preserved.
         ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
        //Invoke the actual login webservices call.
         as.login("td_superuser","welcome1");
        //Retrieve the cookie. The cookie is set on successful
        //authentication.
         String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                               ._getCall().getMessageContext()
                               .getProperty(HTTPConstants.HEADER_COOKIE);
         System.out.println("Cookie : " + cookie);
         return cookie;
       }

      public static void main(String args[])
      {
try
{
  //STEP - 1: User Login
   String cookie = getCookie();

  //STEP - 2: Locate the web services end point.
  //The web services locator contains the webservices end point address.
  //Hence instantiate the service locator.
   ForumServiceServiceLocator fssl = new ForumServiceServiceLocator();

  //Get a handle to the actual web service.
   ForumService fs = fssl.getForumService();

  //Indicate to the server that the user is willing to participate in the
  //session.
  //Since HTTP is stateless, all the client data is maintained in the
  //session.
   ((ForumServiceSoapBindingStub)fs).setMaintainSession(true);
  //Set the cookie retrieved in the call to login webservice.
  //The cookie asserts user's identity to the server.
   ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_COOKIE,cookie);

  //Invoke the actual web services call.
   ForumDefinition fDefn = new ForumDefinition();
   fDefn.setName("My Forum");
   fDefn.setDisplayName("My Forum");
   fDefn.setDescription("ForumCreated");
   fDefn.setEmailAddress("myforum@stacx01.us.oracle.com");
   fDefn.setWithAnnouncementBoard(false);
   fDefn.setPublicForum(true);
   fDefn.setEmailRoutingPolicy(2);
```

```
 Forum forum = fs.create(-1,fDefn);
 if(forum != null)
 {
     System.out.println("Is Public :" + forum.isPublicForum());
     System.out.println("Forum is :" + forum.getId());
     System.out.println("Type :" + forum.getContainerType());
     System.out.println("Creation date :" +
                         forum.getCreationDate().getTime());
     System.out.println("Creator :" + forum.getCreator());
     System.out.println("Description :" + forum.getDescription());
     System.out.println("Display Path :" + forum.getDisplayPath());
     System.out.println("Displayname :" + forum.getDisplayName());
     LastPost lp = forum.getLastPost();
     if(lp != null)
     {
         System.out.println("Lastpost Author :" + lp.getLastpostAuthor());
         System.out.println("Last Posted date :" +
                             lp.getLastpostDate().getTime());
         System.out.println("Last post forum id :" +
                             lp.getLastpostForumId());
         System.out.println("Last post Topic id :" +
                             lp.getLastpostTopicId());
         System.out.println("Last post Message id :" +
                             lp.getLastpostMessageId());
     }
     System.out.println("is favorite :" + forum.isFavorite());
     System.out.println("is new :" + forum.isNewFolder());
     System.out.println("is new since last visit :" +
                             forum.isNewSinceLastVisit());
     System.out.println("is new since session :" +
                             forum.isNewSinceSession());
     System.out.println("name :" + forum.getName());
     System.out.println("Id :" + forum.getId());
     System.out.println("is locked :" + forum.isLocked());
     System.out.println("thread count :" + forum.getThreadCount());
     System.out.println("message count :" + forum.getMessageCount());
     System.out.println("Rss Url :" + forum.getRssURL());
     System.out.println("Webui URL :" + forum.getWebuiURL());
     System.out.println("is category type :" + forum.isCategoryType());
     System.out.println("Email Routing policy :"+
                             forum.getEmailRoutingPolicy());
  }

  //STEP - 3: User Logout
  logout();
}
catch(TdWSException ex)
{
    System.out.println("Error Code :" + ex.getErrorCode());
    System.out.println("Error Message :" + ex.getErrorMessage());
    String[] trace = ex.getServerStackTrace();
    if(trace != null)
    {
        for(int i = 0 ; i < trace.length ; i++)
            System.out.println("trace[" + i + "]" + trace[i]);
    }
}
catch(Exception ex)
{
    System.out.println("Never went into tdex :" + ex.getMessage());
```

```
                ex.printStackTrace();
            }
        }

        public static void logout() throws Exception
        {
             System.out.println("User logout.");
            //Once all the webservices operations are done, invoke logout
            //operation.
            //On logout, user state is destroyed and the cookie is invalidated.
             as.logout();
             System.out.println("Done");
         }
}
```

The `create()` method creates forums under the container represented by `lParentId`, with the properties specified in the `fDefn` bean array.

This method returns a forum bean that represents the forum object created on the server side.

- If the forum creation is successful, then the forum bean representing the forum is created.
- If the forum creation failed, then the details of the error, such as error message, error code, and stack trace, are returned.

> **Note:** An exception is raised if two forums at the same level have same name or same display name. After a forum is created, the forum name cannot be changed. Default settings are applied to the forum and you can update them later.

The sample output for Example 9–1 is as follows:

```
Cookie : JSESSIONID=8c57046a22b877e223ce6e5b4a4dae97fa9011e1fe8c
Is Public :true
Forum is :85769
Type :2
Creation date :Sun Feb 26 20:39:38 PST 2006
Creator :td_superuser
Description :ForumCreated
Display Path :/My Forum
Displayname :My Forum
is favorite :false
is new :false
si new since last visit :false
is new since session :false
name :My forum
Id :85769
is locked :false
thread count :0
message count :0
Rss Url :http://stacx01:8888/discussions/rss/board?fid=85769
Webui URL :http://stacx01:8888/discussions/app/mainAction.do?fid=85769
is category type :false
Email Routing policy :2
User logout.
Done
```

Note: The URLs mentioned in the sample output of Example 9–1 change as per the host on which the client is executing.

Updating Forum Settings

The following is an example for updating the forum settings:

Example 9–2 Updating Forum Settings

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.ForumServiceSoapBindingStub;
import oracle.discussions.ws.ForumServiceServiceLocator;
import oracle.discussions.ws.ForumService;

public class WSClient
{
   private static AuthenticationService as = null;

   public static String getCookie() throws Exception
    {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about the Web Services endpoint
       AuthenticationServiceServiceLocator assl = new
                         AuthenticationServiceServiceLocator();
      //Get the handle to the actual webservices interface.
       as = assl.getAuthenticationService();
      //Indicate to the server that the client is interested in maintaining
      //session.
      //HTTP is stateless and the user state is maintained in the session.
      //Unless the user is willing to participate in a session, user state cannot
      //be preserved.
       ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
      //Invoke the actual login webservices call.
       as.login("td_superuser","welcome1");
      //Retrieve the cookie. The cookie is set on successful authentication.
       String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                        ._getCall().getMessageContext()
                                    .getProperty(HTTPConstants.HEADER_COOKIE);
       System.out.println("Cookie : " + cookie);
       return cookie;
   }

   public static void main(String args[])
   {
     try
     {
```

```
//STEP - 1: User Login
String cookie = getCookie();

//STEP - 2: Locate the Web Services end point.
//The web services locator contains the webservices end point address.
      //Hence instantiate the service locator.
       ForumServiceServiceLocator fssl = new ForumServiceServiceLocator();

      //Get a handle to the actual web service.
       ForumService fs = fssl.getForumService();

      //Indicate to the server that the user is willing to participate in
      //the session.
      //Since HTTP is stateless, all the client data is maintained in the
      //session.
       ((ForumServiceSoapBindingStub)fs).setMaintainSession(true);

      //Set the cookie retrieved in the call to login webservice.
      //The cookie asserts user's identity to the server.
       ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_
                                             COOKIE,cookie);

      //Invoke the actual web services call. The ID needs to be changed
      //accordingly
       ForumSettings fSettings = fs.getForumSettings(6732);
       if(fSettings != null)
       {
         fSettings.setBeginQuotePrefix(fSettings.getBeginQuotePrefix());
         fSettings.setAttachmentMaxSize(1000000);
         fSettings.setCanAttach(true);
         fSettings.setComposeEditor(2);
         fSettings.setEditDeletePolicy(fSettings.getEditDeletePolicy());
         fSettings.setFlatViewPagingSize(fSettings.getFlatViewPagingSize());
         fSettings.setThreadView(2);
         fSettings.setEmailForwardPolicy(0);
         fSettings.setEmailInboundPolicy(0);
         fSettings.setOutgoingEmailDLSubjectFormat
                        (fSettings.getOutgoingEmailDLSubjectFormat());
         fSettings.setOutgoingEmailDLBodyFormat
                        (fSettings.getOutgoingEmailDLBodyFormat());
         fSettings.setReplyPrefix("Re: From Announcements forum");
         fSettings.setOriginalQuoted(fSettings.isOriginalQuoted());
         fSettings.setPublicAccessPolicy(3);
         fSettings.setEmailRoutingPolicy(2);

        //The ID needs to be changed accordingly
         fSettings = fs.getForumSettings(6732);
         if(fSettings != null)
         {
            System.out.println("Retrieved changed forum settings");
            System.out.println("Begin Quote Prefix :" +
                           fSettings.getBeginQuotePrefix());
            System.out.println("Attachment max size :" +
                           fSettings.getAttachmentMaxSize());
            System.out.println("Can Attach :" + fSettings.isCanAttach());
            System.out.println("Compose Editor :" +
                           fSettings.getComposeEditor());
            System.out.println("Edit delete policy :" +
                           fSettings.getEditDeletePolicy());
            System.out.println("Email Address " +
```

```
                                    fSettings.getEmailAddress());
                    System.out.println("Flat view paging size :" +
                                    fSettings.getFlatViewPagingSize());
                    System.out.println("Thread View :" + fSettings.getThreadView());
                    System.out.println("Email forward policy :" +
                                    fSettings.getEmailForwardPolicy());
                    System.out.println("Email inbound policy :" +
                                fSettings.getEmailInboundPolicy());
                System.out.println("Outgoing email DL :" +
                                fSettings.getOutgoingEmailDL());
                System.out.println("Outgoing email DL Subjet format: " +
                                fSettings.getOutgoingEmailDLSubjectFormat());
                System.out.println("Outgoing email DL Body format :" +
                                fSettings.getOutgoingEmailDLBodyFormat());
                System.out.println("Reply Prefix " + fSettings.getReplyPrefix());
                System.out.println("Is original quoted :" +
                                fSettings.isOriginalQuoted());
                System.out.println("Public access policy :" +
                                fSettings.getPublicAccessPolicy());
                System.out.println("Email routing policy :" +
                                fSettings.getEmailRoutingPolicy());
            }
         }

          //STEP - 2: User Logout
          logout();
        }
        catch(TdWSException ex)
        {
            System.out.println("Error Code :" + ex.getErrorCode());
            System.out.println("Error Message :" + ex.getErrorMessage());
            String[] trace = ex.getServerStackTrace();
            if(trace != null)
            {
                for(int i = 0 ; i < trace.length ; i++)
                    System.out.println("trace[" + i + "]" + trace[i]);
            }
        }
        catch(Exception ex)
        {
            System.out.println("Never went into tdex :" + ex.getMessage());
            ex.printStackTrace();
        }
    }
    public static void logout() throws Exception
    {
        System.out.println("User logout.");
        //Once all the webservices operations are done, invoke logout
        //operation.
        //On logout, user state is destroyed and the cookie is invalidated.
        as.logout();
        System.out.println("Done");
    }
}
```

The `updateForumSettings()` method updates the settings for the forum represented by `lForumId`, with the settings supplied in forum settings bean, `fs`.

The `fs` bean represents the settings of a Oracle Discussions Forum. These settings define the forum description, display name of the forum, and other properties.

The `setEmailAddress()` method sets the e-mail address of the forum. The e-mail domain should be same as the domain for which Oracle Discussions internal administration is provisioned.

The `setEmailForwardPolicy()` method sets the e-mail forward policy of the forum. The valid values for the parameter `policy` are as follows:

- 0 indicates that the e-mail messages are forwarded as original messages.
- 1 indicates that the e-mail messages are forwarded as notifications.

The `setEmailInboundPolicy()` method sets the e-mail inbound policy of the forum. The valid values for the parameter `policy` are as follows:

- Integer value 0 indicates that all the messages sent to the e-mail address of the forum are ignored.
- Integer value `1` indicates that e-mail messages sent to the forum e-mail address will be posted to the forum only if the author is a forum writer.
- Integer value 2 indicates that the e-mail messages sent to the forum e-mail address will always be posted to the forum

The sample output for Example 9–2 is as follows:

```
Cookie : JSESSIONID=8c57046a22b88562b8545c3b459bbcae9546e94a8c49
Retrieved changed forum settings
Begin Quote Prefix :On ${msg-sent-date}, ${msg-author}
[${msg-from}] wrote
${msg-subject}:
Attachment max size :1000000
Can Attach :true
Compose Editor :2
Edit delete policy :2
Email Address changed_forum@stacx01.us.oracle.com
Flat view paging size :10
Thread View :2
Email forward policy :0
Email inbound policy :0
Outgoing email DL :null
Outgoing email DL Subjet format: ${msg-subject} (${msg-board})
Outgoing email DL Body format :New reply to topic <A
href='${msg-thread-url}'>"${msg-thread}"</A> in Forum <A
href='${msg-board-url}'>"${msg-board}"</A>.
 <P>
 <B>From: </B><A href='mailto:${msg-from}'>${msg-author}</A><BR>
 <B>Subject: </B>${msg-subject}<BR>
 <B>Date: </B>${msg-sent-date}<BR>
 <BR>
 ${msg-body}<BR>
 <HR>
 View in <A href='${msg-url}'>Oracle Discussions</A>
Reply Prefix Re: From Announcements forum
Is original quoted :true
Public access policy :3
Email routing policy :1
User logout.
Done
```

Note: The e-mail ID in the sample output can change as per the e-mail address set to the forum of the actual user.

Listing the Topics in the Forum

The following is an example for listing the topics in the forum:

Example 9–3 Listing the Topics in the Forum

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.ForumServiceSoapBindingStub;
import oracle.discussions.ws.ForumServiceServiceLocator;
import oracle.discussions.ws.ForumService;

public class WSClient
{
  private static AuthenticationService as = null;
  public static String getCookie() throws Exception
  {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about the Web Services endpoint
       AuthenticationServiceServiceLocator assl = new
                         AuthenticationServiceServiceLocator();
      //Get the handle to the actual webservices interface.
       as = assl.getAuthenticationService();
      //Indicate to the server that the client is interested in maintaining
      //session.
      //HTTP is stateless and the user state is maintained in the session.
      //Unless the user is willing to participate in a session, user state cannot
      //be preserved.
       ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
      //Invoke the actual login webservices call.
       as.login("td_superuser","welcome1");
      //Retrieve the cookie. The cookie is set on successful authentication.
       String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                  ._getCall().getMessageContext()
                                  .getProperty(HTTPConstants.HEADER_COOKIE);
       System.out.println("Cookie : " + cookie);
       return cookie;
  }

  public static void main(String args[])
  {
     try
     {
       //STEP - 0: User Login
        String cookie = getCookie();

       //STEP - 1: Locate the web services end point.
       //The web services locator contains the webservices end point address.
       //Hence instantiate the service locator.
```

```
 ForumServiceServiceLocator fssl = new ForumServiceServiceLocator();

//Get a handle to the actual web service.
 ForumService fs = fssl.getForumService();

//Indicate to the server that the user is willing to participate in the
//session.
//Since HTTP is stateless, all the client data is maintained in the
//session
 ((ForumServiceSoapBindingStub)fs).setMaintainSession(true);

//Set the cookie retrieved in the call to login Web Service.
      //The cookie asserts user's identity to the server.
       ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_
                                              COOKIE,cookie);

      //Invoke the actual web services call. The ID needs to be changed
      // accordingly
       Topic[] topics = fs.listTopics(6732,5,5,true,false);
       if(topics != null)
       {
         System.out.println("Retrieved :" + topics.length + " topics");
         for(int i = 0 ; i < topics.length ; i++)
         {
           if(topics[i] == null)
           {
             System.out.println("Topic not existing..:" + i);
             continue;
            }
           System.out.println("Subject :" + topics[i].getSubject());
           System.out.println("Web ui url :" + topics[i].getWebUIUrl());
           System.out.println("Web ui url :" + topics[i].getRssURL());
           LastPost lp = topics[i].getLastPost();
           if(lp != null)
           {
             System.out.println("Lastpost Author :" +
                                lp.getLastpostAuthor());
             System.out.println("Last Posted date :" +
                                lp.getLastpostDate());
             System.out.println("Last post forum id :" +
                                lp.getLastpostForumId());
             System.out.println("Last post Topic id :" +
                                lp.getLastpostTopicId());
             System.out.println("Last post Message id :" +
                                lp.getLastpostMessageId());
           }
          System.out.println("Msg Count :" + topics[i].getMsgCount());
          System.out.println("Subject :" + topics[i].getSubject());
          System.out.println("is Favorite :" + topics[i].isFavorite());
          System.out.println("is locked :" + topics[i].isLocked());
          System.out.println("Plain Text :" +
                             topics[i].getBodyPlainText());
          String[] frmAdd = topics[i].getFromAddresses();
          if(frmAdd != null)
          {
            for(int j = 0 ; j < frmAdd.length ; j++)
               System.out.println("From Address :" + frmAdd[j]);
           }
          System.out.println("Level :" + topics[i].getLevel());
          System.out.println("No of replies :" +
```

```
                                topics[i].getNumberOfReplies());
                    System.out.println("Oracle Msg Id :" + topics[i].getMessageId());
                    System.out.println("is new since session :" +
                                topics[i].isNewSinceSession());
                    System.out.println("is new since last visit :" +
                                topics[i].isNewSinceLastVisit());
                    System.out.println("is new " + topics[i].isNewMessage());
                    System.out.println("Has Attachments :" +
                                topics[i].isHasAttachments());
                    System.out.println("X Priority :" + topics[i].getXPriority());
                    System.out.println("Sent dat :" + topics[i].getSentDate());
                    System.out.println("Size :" + topics[i].getSize());
                    System.out.println("Subject :" + topics[i].getSubject());
                }
            }

            //STEP - 2: User Logout
             logout();
           }
           catch(TdWSException ex)
           {
             System.out.println("Error Code :" + ex.getErrorCode());
             System.out.println("Error Message :" + ex.getErrorMessage());
             String[] trace = ex.getServerStackTrace();
             if(trace != null)
             {
                for(int i = 0 ; i < trace.length ; i++)
                 System.out.println("trace[" + i + "]" + trace[i]);
             }
           }
           catch(Exception ex)
           {
             System.out.println("Never went into tdex :" + ex.getMessage());
             ex.printStackTrace();
           }
       }

       public static void logout() throws Exception
       {
          System.out.println("User logout.");
          //Once all the webservices operations are done, invoke logout
          //operation.
          //On logout, user state is destroyed and the cookie is invalidated.
          as.logout();
          System.out.println("Done");
       }
}
```

The `listTopics()` method lists topics under the forum represented by `lForumId`. This method returns `null` if there are no topics under the forum. The listed topics can be sorted by creation date, by specifying `bSortByCreation` to `true`. The topics can also be sorted in the ascending order of the creation date, by specifying `bAscending` to `true`. This method has the following parameters:

- `lForumId` is the ID representing forum from which the topics are listed.
- `iFetchStart` is the start index from which topics are returned. The value of i`FetchStart` should always be greater than `0`.

- `iFetchMax` is used to limit the size of topics returned in an iteration. The maximum value accepted by `iFetchMax` is `500`. An exception is raised of the user tries to set a value greater than `500`.

  The number of topics returned would be starting from `iFetchStart` to `iFetchStart + iFetchMax`. If the number of topics in the forum is less than `iFetchStart + iFetchMax` count, then all the topics are returned in the same iteration. If there are no topics in the forum, then `null` is returned.
- `bSortByCreation` (`true` in this example) is the boolean value specifying if the returned topics are to be sorted by creation time.
- `bAscending` (`true` in this example) is the boolean value specifying if the returned topics are to be sorted in the ascending order of creation date.

The sample output for Example 9–3 is as follows:

```
Subject :Posted on 24thJan06988
Web ui url :http://stacx01:8888/discussions/app/mainAction.do?fid=6732&tid=123545
Rss url :http://stacx01:8888/discussions/rss/thread?fid=6732&tid=123545
Lastpost Author :Admin <td_superuser@stacx01.us.oracle.com>
Last Posted date : Sun Feb 26 19:54:18 PST 2006
Last post forum id :6732
Last post Topic id :123545
Last post Message id :123545
Msg Count :1
Subject :Posted on 24thJan06988
is Favorite :false
is locked :false
Plain Text :Posted on 24thJan06988
From Address :Admin <td_superuser@stacx01.us.oracle.com>
Level :0
No of replies :0
Oracle Msg Id :123545
is new since session :false
is new since last visit :false
is new false
Has Attachments :false
X Priority :1
Sent dat : Sun Feb 26 19:54:18 PST 2006
Size :438
Subject :Posted on 24thJan06988
```

Note: The URLs mentioned in the sample output of Example 9–3 change as per the host on which the client is executing.

10

Understanding the Topic Service

The Topic Service interface represents the methods that operate on Oracle Discussions Topics. The interface provides the following operations:

- Creating a topic
- Listing of messages under a topic
- Locking or unlocking a topic
- Retrieving a topic
- Moving a topic from one forum to another
- Editing the contents of a topic
- Deleting a topic

It also provides bulk operations to create or delete multiple topics in one invocation. This chapter addresses the task of Creating a Topic.

Creating a Topic

The following is an example for creating a topic from the byte array:

Example 10–1 Creating a Topic from the Byte Array

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.TopicServiceSoapBindingStub;
import oracle.discussions.ws.TopicServiceServiceLocator;
import oracle.discussions.ws.TopicService;

public class WSClient
{
  private static AuthenticationService as = null;
  public static String getCookie() throws Exception
  {
      //STEP - 0: User Login
```

```
    //Create the service locator for authentication service. The locator
    //contains information about the web Services endpoint
     AuthenticationServiceServiceLocator assl = new
                    AuthenticationServiceServiceLocator();
    //Get the handle to the actual webservices interface.
     as = assl.getAuthenticationService();
    //Indicate to the server that the client is interested in maintaining
    //session.
    //HTTP is stateless and the user state is maintained in the session.
    //Unless the user is willing to participate in a session, user state
    //cannot be preserved.
     ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
    //Invoke the actual login webservices call.
     as.login("td_superuser","welcome1");
    //Retrieve the cookie. The cookie is set on successful authentication.
     String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                             ._getCall().getMessageContext()
                            .getProperty(HTTPConstants.HEADER_COOKIE);
     System.out.println("Cookie : " + cookie);
     return cookie;
 }

public static void main(String args[])
{
  try
  {
     //STEP - 0: User Login
      String cookie = getCookie();

     //STEP - 1: Locate the Web Services end point.
     //The Web Services locator contains the Web Services end point
     //address
     //Hence instantiate the service locator.
      TopicServiceServiceLocator tssl = new TopicServiceServiceLocator();

     //Get a handle to the actual web service.
      TopicService ts = tssl.getTopicService();

     //Indicate to the server that the user is willing to participate in
     //the session.
     //Since HTTP is stateless, all the client data is maintained in the
     //session.
      ((TopicServiceSoapBindingStub)ts).setMaintainSession(true);

     //Set the cookie retrieved in the call to login webservice.
     //The cookie asserts user's identity to the server.
      ((javax.xml.rpc.Stub)ts)._setProperty(HTTPConstants.HEADER_
                                            COOKIE,cookie);

     //Invoke the actual web services call.
      byte[] bytes = null;
     //Create a MIME Compatible byte array
      if(bytes != null)
      {
       //The ID needs to be changed accordingly
        ForumMessage msg = ts.createTopicFromByte(15524,bytes);
        if(msg != null)
        {
         System.out.println("Subject :" + msg.getSubject());
         System.out.println("Web ui url :" + msg.getWebUIUrl());
```

```
            System.out.println("Subject :" + msg.getSubject());
            System.out.println("Plain Text :" + msg.getBodyPlainText());
            String[] frmAdd = msg.getFromAddresses();
            if(frmAdd != null)
            {
                for(int j = 0 ; j < frmAdd.length ; j++)
                    System.out.println("From Address :" + frmAdd[j]);
            }

            System.out.println("Level :" + msg.getLevel());
            System.out.println("No of replies :" + msg.getNumberOfReplies());
            System.out.println("Oracle Msg Id :" + msg.getMessageId());
            System.out.println("is new since session :" + msg.isNewSinceSession());
            System.out.println("is new since last visit :" +
                                        msg.isNewSinceLastVisit());
            System.out.println("is new " + msg.isNewMessage());
            System.out.println("Has Attachments :" + msg.isHasAttachments());
            System.out.println("X Priority :" + msg.getXPriority());
            System.out.println("Sent dat :" + msg.getSentDate());
            System.out.println("Size :" + msg.getSize());
            System.out.println("Subject :" + msg.getSubject());
          }
        }
        //STEP - 2: User Logout
         logout();
      }
      catch(TdWSException ex)
      {
          System.out.println("Error Code :" + ex.getErrorCode());
          System.out.println("Error Message :" + ex.getErrorMessage());
          String[] trace = ex.getServerStackTrace();
          if(trace != null)
          {
              for(int i = 0 ; i < trace.length ; i++)
                  System.out.println("trace[" + i + "]" + trace[i]);
          }
      }
      catch(Exception ex)
      {
          System.out.println("Never went into tdex :" + ex.getMessage());
          ex.printStackTrace();
      }
   }
   public static void logout() throws Exception
   {
       System.out.println("User logout.");
       //Once all the webservices operations are done, invoke logout
       //operation.
       //On logout, user state is destroyed and the cookie is invalidated.
       as.logout();
       System.out.println("Done");
   }
}
```

The `createTopicFromByte()` creates a topic from the specified byte array.

The `byte[]` array should be MIME compatible. It is useful in creating topics with attachments. The entire message, along with all the headers and attachments is

converted into a `byte[]`. The same `byte[]` can be used in creating a topic. This method accepts two parameters:

- `lForumId` is the ID representing a forum, in which topic is to be created.
- `msg` is the byte representing the topic to be created.

The method returns `ForumMessage` which is a bean that represents the newly created topic.

The sample output for Example 10–1 is as follows:

```
Cookie : JSESSIONID=8c57046a22b87bc7c77f14844facb3d6e912391d9f5d
Subject :Topic-1
Web ui url
:http://stacx01:8888/discussions/app/mainAction.do?fid=15524&tid=281488&mid=281488
#281488
Subject :Topic-1
Plain Text :Topic-1
From Address :Admin <td_superuser@stacx01.us.oracle.com>
Level :0
No of replies :0
Oracle Msg Id :281488
is new since session :true
is new since last visit :true
is new true
Has Attachments :false
X Priority :1
Sent date : Sun Feb 26 19:54:18 PST 2006
Size :405
Subject :Topic-1
User logout.
Done
```

11

Understanding Message Service

The Message Service interface represents the operations related to Oracle Discussions Message. Messages are either new posts into a board (and therefore topics) or replies to other messages. Messages may contain attachments. In general, the first message of a thread is also referred to as the thread itself, while all its replies are just simply referred as messages. The Web Service interface provides for the following operations:

- Replying to a message
- Editing a message
- Fetching the message details, given the message id
- Retrieving the message in a raw byte format
- Fetching parent topic, given a message
- Clipping a message and all its replies into another topic
- Hiding a message
- Unhiding a message
- Deleting a message

Apart from the preceding operations, it also provides bulk operations to hide or show multiple messages in a single invocation.

This chapter addresses the following tasks:

Replying to a Message

The following is an example for replying to a message:

Example 11–1 Replying to a Message

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
```

```
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.MessageServiceSoapBindingStub;
import oracle.discussions.ws.MessageServiceServiceLocator;
import oracle.discussions.ws.MessageService;

public class WSClient
{
  private static AuthenticationService as = null;
  public static String getCookie() throws Exception
 {
   //STEP - 0: User Login
   //Create the service locator for authentication service. The locator
   //contains information about the Web Services endpoint
    AuthenticationServiceServiceLocator assl = new
                    AuthenticationServiceServiceLocator();
   //Get the handle to the actual webservices interface.
    as = assl.getAuthenticationService();
   //Indicate to the server that the client is interested in maintaining
   //session
   //HTTP is stateless and the user state is maintained in the session.
   //Unless the user is willing to participate in a session, user state
   //cannot be preserved.
    ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
   //Invoke the actual login webservices call.
    as.login("td_superuser","welcome1");
   //Retrieve the cookie. The cookie is set on successful authentication.
    String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                  ._getCall().getMessageContext()
                               .getProperty(HTTPConstants.HEADER_COOKIE);
    System.out.println("Cookie : " + cookie);
    return cookie;
 }
 public static void main(String args[])
 {
   try
   {
     //STEP - 0: User Login
      String cookie = getCookie();

     //STEP - 1: Locate the web services end point.
     //The web services locator contains the webservices end point address.
     //Hence instantiate the service locator.
      MessageServiceServiceLocator fssl = new
                           MessageServiceServiceLocator();

     //Get a handle to the actual web service.
      MessageService fs = fssl.getMessageService();
     //Indicate to the server that the user is willing to participate in
     //the session.
     //Since HTTP is stateless, all the client data is maintained in the
     //session.
      ((MessageServiceSoapBindingStub)fs).setMaintainSession(true);

     //Set the cookie retrieved in the call to login webservice.
     //The cookie asserts user's identity to the server.
      ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_
                                            COOKIE,cookie);
```

```
    //Invoke the actual web services call.
     MessageDefinition mDefn = new MessageDefinition();
     mDefn.setSubject("Yahoo Email Solution");
     mDefn.setBody("<a href=\"http://www.yahoomail.com\">Dont Click</a>");
     mDefn.setHTMLContentType(true);
    //The IDs need to be changed accordingly
     ForumMessage topic = fs.reply(15524,281488,281488,mDefn);
     if(topic != null)
     {
        System.out.println("Subject :" + topic.getSubject());
        System.out.println("Plain Text :" + topic.getBodyPlainText());
        String[] frmAdd = topic.getFromAddresses();
        if(frmAdd != null)
        {
            for(int j = 0 ; j < frmAdd.length ; j++)
                System.out.println("From Address :" + frmAdd[j]);
        }
        System.out.println("Level :" + topic.getLevel());
        System.out.println("No of replies :" + topic.getNumberOfReplies());
        System.out.println("Oracle Msg Id :" + topic.getMessageId());
        System.out.println("is new since session :" +
                                  topic.isNewSinceSession());
        System.out.println("is new since last visit :" +
                                  topic.isNewSinceLastVisit());
        System.out.println("is new " + topic.isNewMessage());
        System.out.println("Has Attachments :" + topic.isHasAttachments());
        System.out.println("X Priority :" + topic.getXPriority());
        System.out.println("Sent dat :" + topic.getSentDate().getTime());
        System.out.println("Size :" + topic.getSize());
        System.out.println("Webui URL :" + topic.getWebUIUrl());
        System.out.println("HTML ? :" + topic.isHTMLContentType());
        System.out.println("Hidden :" + topic.isHidden());
    }

    //STEP - 2: User Logout
    logout();
   }
   catch(TdWSException ex)
   {
       System.out.println("Error Code :" + ex.getErrorCode());
       System.out.println("Error Message :" + ex.getErrorMessage());
       String[] trace = ex.getServerStackTrace();
       if(trace != null)
       {
           for(int i = 0 ; i < trace.length ; i++)
               System.out.println("trace[" + i + "]" + trace[i]);
       }
   }
   catch(Exception ex)
   {
       System.out.println("Never went into tdex :" + ex.getMessage());
       ex.printStackTrace();
   }
}

 public static void logout() throws Exception
 {
     System.out.println("User logout.");
     //Once all the webservices operations are done, invoke logout
```

```
        //operation.
              //On logout, user state is destroyed and the cookie is invalidated.
              as.logout();
              System.out.println("Done");
           }
}
```

The `reply()` method replies to the message specified by `iMsgId`. The reply posted will be a child message of the message specified. Only users with forum writer or higher role can post a reply, unless the forum is public, with anonymous posting setting enabled.

The sample output for Example 11–1 is as follows:

```
Cookie : JSESSIONID=8c57046a22b81295628fa8f249819566d10096888122
Subject :Yahoo Email Solution
Plain Text :<a href="http://www.yahoomail.com" target="_blank">Dont Click</a>
From Address :Admin <td_superuser@stacx01.us.oracle.com>
Level :1
No of replies :0
Oracle Msg Id :281489
is new since session :true
is new since last visit :true
is new true
Has Attachments :false
X Priority :1
Sent dat :Sun Feb 26 21:35:58 PST 2006
Size :579
Webui URL
:http://stacx01:8888/discussions/app/mainAction.do?fid=15524&tid=281488&mid=281489
#281489
HTML ? :true
Hidden :false
User logout.
Done
```

Creating the Message from Byte

The following is an example for creating the message from byte:

Example 11–2 Creating the Message from Byte

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.MessageServiceSoapBindingStub;
import oracle.discussions.ws.MessageServiceServiceLocator;
import oracle.discussions.ws.MessageService;

public class WSClient
{
  private static AuthenticationService as = null;
```

```
public static String getCookie() throws Exception
{
    //STEP - 0: User Login
    //Create the service locator for authentication service. The locator
    //contains information about the web Services endpoint
      AuthenticationServiceServiceLocator assl = new
                  AuthenticationServiceServiceLocator();
     //Get the handle to the actual webservices interface.
      as = assl.getAuthenticationService();
     //Indicate to the server that the client is interested in maintaining
     //session.
     //HTTP is stateless and the user state is maintained in the session.
     //Unless the user is willing to participate in a session, user state cannot
     //be preserved.
     ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
    //Invoke the actual login webservices call.
     as.login("td_superuser","welcome1");
    //Retrieve the cookie. The cookie is set on successful authentication.
     String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                            ._getCall().getMessageContext()
                            .getProperty(HTTPConstants.HEADER_COOKIE);
     System.out.println("Cookie : " + cookie);
     return cookie;
}

public static void main(String args[])
{
   try
   {
    //STEP - 0: User Login
     String cookie = getCookie();

    //STEP - 1: Locate the web services end point.
    //The web services locator contains the webservices end point address.
    //Hence instantiate the service locator.
     MessageServiceServiceLocator fssl = new MessageServiceServiceLocator();

    //Get a handle to the actual web service.
     MessageService fs = fssl.getMessageService();
    //Indicate to the server that the user is willing to participate in the
    //session.
    //Since HTTP is stateless, all the client data is maintained in the session.
     ((MessageServiceSoapBindingStub)fs).setMaintainSession(true);

    //Set the cookie retrieved in the call to login webservice.
    //The cookie asserts user's identity to the server.
     ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_COOKIE,cookie);

    //Invoke the actual web services call.
     byte[] bytes = fs.getMessageInByte(15524,281488,281488);
     if(bytes != null)
     {
    //The IDs will change accordingly
       ForumMessage topic = fs.createMessageFromByte(15524,281488,281488,bytes);
       if(topic != null)
       {
           System.out.println("Subject :" + topic.getSubject());
           System.out.println("Plain Text :" + topic.getBodyPlainText());
           String[] frmAdd = topic.getFromAddresses();
           if(frmAdd != null)
```

```
                {
                    for(int j = 0 ; j < frmAdd.length ; j++)
                        System.out.println("From Address :" + frmAdd[j]);
                }
                       System.out.println("Level :" + topic.getLevel());
                       System.out.println("No of replies :" +
                                                  topic.getNumberOfReplies());
                       System.out.println("Oracle Msg Id :" + topic.getMessageId());
                       System.out.println("is new since session :" +
                                           topic.isNewSinceSession());
                       System.out.println("is new since last visit :" +
                                           topic.isNewSinceLastVisit());
                       System.out.println("is new " + topic.isNewMessage());
                       System.out.println("Has Attachments :" +
                                            topic.isHasAttachments());
                       System.out.println("X Priority :" + topic.getXPriority());
                       System.out.println("Sent dat :" +
                                            topic.getSentDate().getTime());
                       System.out.println("Size :" + topic.getSize());
                       System.out.println("Webui URL :" + topic.getWebUIUrl());
                       System.out.println("HTML ? :" + topic.isHTMLContentType());
                       System.out.println("Hidden :" + topic.isHidden());
                    }
                }

                //STEP - 2: User Logout
                 logout();
              }
              catch(TdWSException ex)
              {
                System.out.println("Error Code :" + ex.getErrorCode());
                System.out.println("Error Message :" + ex.getErrorMessage());
                String[] trace = ex.getServerStackTrace();
                if(trace != null)
                {
                    for(int i = 0 ; i < trace.length ; i++)
                        System.out.println("trace[" + i + "]" + trace[i]);
                }
             }
            catch(Exception ex)
            {
                System.out.println("Never went into tdex :" + ex.getMessage());
                ex.printStackTrace();
            }
        }
        public static void logout() throws Exception
         {
             System.out.println("User logout.");
             //Once all the webservices operations are done, invoke logout
             //operation.
             //On logout, user state is destroyed and the cookie is invalidated.
             as.logout();
             System.out.println("Done");
         }
}
```

The reply posted will be a child message of the message specified. The `byte[]` array should be MIME-compatible. This is useful in creating messages with attachments. The entire message, along with all the headers and attachments, is converted into a byte array by `createMessageFromByte()`. The same byte array can be used in

creating the message. Only users with forum writer or higher role can post a reply, unless the forum is public, with the anonymous posting setting enabled.

> **Note:** Threaded Discussions maintains the threading information for messages posted to a board. Threading information is used to sort a discussion thread in thread order. Such an operation is based on the usage of the `References` message header which keeps the IDs of the parent messages in a comma separated list. If the references header is absent, then messages are threaded based on the message subject.

The sample output for Example 11–2 is as follows:

```
Cookie : JSESSIONID=8c57046a22b888e1210975ab43f4b75ca85fa663c175
Subject :Topic-1
Plain Text :Topic-1
From Address :Admin <td_superuser@stacx01.us.oracle.com>
Level :1
No of replies :0
Oracle Msg Id :281490
is new since session :true
is new since last visit :true
is new true
Has Attachments :false
X Priority :1
Sent dat :Sun Feb 26 21:42:44 PST 2006
Size :540
Webui URL
:http://stacx01:8888/discussions/app/mainAction.do?fid=15524&tid=281488&mid=281490
#281490
HTML ? :false
Hidden :false
User logout.
Done
```

Editing a Message

The following is an example for editing a message:

Example 11–3 Editing a Message

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.MessageServiceSoapBindingStub;
import oracle.discussions.ws.MessageServiceServiceLocator;
import oracle.discussions.ws.MessageService;

public class WSClient
{
```

```
private static AuthenticationService as = null;
public static String getCookie() throws Exception
{
          //STEP - 0: User Login
          //Create the service locator for authentication service. The locator
          //contains information about the Web Services endpoint
           AuthenticationServiceServiceLocator assl = new
                       AuthenticationServiceServiceLocator();
          //Get the handle to the actual webservices interface.
           as = assl.getAuthenticationService();
          //Indicate to the server that the client is interested in maintaining
          //session.
          //HTTP is stateless and the user state is maintained in the session.
          //Unless the user is willing to participate in a session, user state
          //cannot be preserved.
           ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
          //Invoke the actual login webservices call.
           as.login("td_superuser","welcome1");
          //Retrieve the cookie. The cookie is set on successful authentication.
           String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                  ._getCall().getMessageContext()
                                  .getProperty(HTTPConstants.HEADER_COOKIE);
           System.out.println("Cookie : " + cookie);
           return cookie;
      }

     public static void main(String args[])
    {
      try
      {
        //STEP - 0: User Login
         String cookie = getCookie();

        //STEP - 1: Locate the Web Services end point.
        //The Web Services locator contains the webservices end point address
        //Hence instantiate the service locator.
         MessageServiceServiceLocator fssl = new
                             MessageServiceServiceLocator();

        //Get a handle to the actual web service.
         MessageService fs = fssl.getMessageService();
        //Indicate to the server that the user is willing to participate in the
        //session.
        //Since HTTP is stateless, all the client data is maintained in the
        //session
         ((MessageServiceSoapBindingStub)fs).setMaintainSession(true);

        //Set the cookie retrieved in the call to login webservice.
        //The cookie asserts user's identity to the server.
         ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_
                                                  COOKIE,cookie);

        //Invoke the actual web services call. The IDs need to be changed
        //accordingly
         ForumMessage topic = fs.editMessage(15524,281488,281488,"Edited
                                        Subject","Edited Body");
         if(topic != null)
         {
           System.out.println("Subject :" + topic.getSubject());
           System.out.println("Plain Text :" + topic.getBodyPlainText());
```

```
                String[] frmAdd = topic.getFromAddresses();
                if(frmAdd != null)
                {
                 for(int j = 0 ; j < frmAdd.length ; j++)
                   System.out.println("From Address :" + frmAdd[j]);
              }
              System.out.println("Level :" + topic.getLevel());
              System.out.println("No of replies :" + topic.getNumberOfReplies());
              System.out.println("Oracle Msg Id :" + topic.getMessageId());
              System.out.println("is new since session :" +
                                      topic.isNewSinceSession());
              System.out.println("is new since last visit :" +
                                     topic.isNewSinceLastVisit());
              System.out.println("is new " + topic.isNewMessage());
              System.out.println("Has Attachments :" + topic.isHasAttachments());
              System.out.println("X Priority :" + topic.getXPriority());
              System.out.println("Sent dat :" + topic.getSentDate().getTime());
              System.out.println("Size :" + topic.getSize());
              System.out.println("Webui URL :" + topic.getWebUIUrl());
              System.out.println("HTML ? :" + topic.isHTMLContentType());
              System.out.println("Hidden :" + topic.isHidden());
            }
           //STEP - 2: User Logout
           logout();
          }
          catch(TdWSException ex)
          {
              System.out.println("Error Code :" + ex.getErrorCode());
              System.out.println("Error Message :" + ex.getErrorMessage());
              String[] trace = ex.getServerStackTrace();
              if(trace != null)
              {
                  for(int i = 0 ; i < trace.length ; i++)
                      System.out.println("trace[" + i + "]" + trace[i]);
              }
          }
          catch(Exception ex)
          {
              System.out.println("Never went into tdex :" + ex.getMessage());
              ex.printStackTrace();
          }
      }
      public static void logout() throws Exception
       {
           System.out.println("User logout.");
           //Once all the webservices operations are done, invoke logout
           //operation.
           //On logout, user state is destroyed and the cookie is invalidated.
           as.logout();
           System.out.println("Done");
       }
}
```

The `editMessage()` method edits the message subject and content. A message should have either a subject or a body. The topic posted will be of text or plain content type. The `lForumId` and `iTopicId` should represent the parent forum and topic of the message respectively. The old values of the message subject and content will be overwritten with the values that are provided.

The sample output for Example 11–3 is as follows:

```
Cookie : JSESSIONID=8c57046a22b88d154290041d4bd0bc48bff029d632b1
```

```
Subject :Edited Subject
Plain Text :Edited Body
From Address :Admin <td_superuser@stacx01.us.oracle.com>
Level :0
No of replies :2
Oracle Msg Id :281488
is new since session :false
is new since last visit :false
is new false
Has Attachments :false
X Priority :1
Sent dat :Sun Feb 26 20:13:13 PST 2006
Size :472
Webui URL
:http://stacx01:8888/discussions/app/mainAction.do?fid=15524&tid=281488&mid=281488
#281488
HTML ? :false
Hidden :false
User logout.
Done
```

The following is an example for editing a message with a byte array:

Example 11–4 Editing a Message with a Byte Array

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.MessageServiceSoapBindingStub;
import oracle.discussions.ws.MessageServiceServiceLocator;
import oracle.discussions.ws.MessageService;

public class WSClient
{
  private static AuthenticationService as = null;
  public static String getCookie() throws Exception
  {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about the Web Services endpoint
       AuthenticationServiceServiceLocator assl = new
                       AuthenticationServiceServiceLocator();
      //Get the handle to the actual webservices interface.
       as = assl.getAuthenticationService();
      //Indicate to the server that the client is interested in maintaining
      //session.
      //HTTP is stateless and the user state is maintained in the session.
      //Unless the user is willing to participate in a session, user state cannot
      //be preserved.
       ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
      //Invoke the actual login webservices call.
```

```
    as.login("td_superuser","welcome1");
   //Retrieve the cookie. The cookie is set on successful authentication.
    String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                    ._getCall().getMessageContext()
                     .getProperty(HTTPConstants.HEADER_COOKIE);
   System.out.println("Cookie : " + cookie);
   return cookie;
 }

public static void main(String args[])
{
  try
  {
   //STEP - 0: User Login
    String cookie = getCookie();

   //STEP - 1: Locate the Web Services end point.
   //The Web Services locator contains the webservices end point address.
   //Hence instantiate the service locator.
    MessageServiceServiceLocator fssl = new MessageServiceServiceLocator();

   //Get a handle to the actual web service.
    MessageService fs = fssl.getMessageService();

   //Indicate to the server that the user is willing to participate in the
   //session.
   //Since HTTP is stateless, all the client data is maintained in the session.
    ((MessageServiceSoapBindingStub)fs).setMaintainSession(true);

   //Set the cookie retrieved in the call to login webservice.
   //The cookie asserts user's identity to the server.
    ((javax.xml.rpc.Stub)fs)._setProperty(HTTPConstants.HEADER_COOKIE,cookie);

   //Invoke the actual web services call. The IDs need to be changed
   //accordingly
    Byte[] bytes = fs.getMessageInByte(15524,281488,281488);

   //The Ids need to be changed accordingly
    ForumMessage topic = fs.editMessageWithByte(15524,281488,281488,bytes);
    if(topic != null)
    {
       System.out.println("Subject :" + topic.getSubject());
       System.out.println("Plain Text :" + topic.getBodyPlainText());
       String[] frmAdd = topic.getFromAddresses();
       if(frmAdd != null)
       {
           for(int j = 0 ; j < frmAdd.length ; j++)
               System.out.println("From Address :" + frmAdd[j]);
       }
       System.out.println("Level :" + topic.getLevel());
       System.out.println("No of replies :" + topic.getNumberOfReplies());
       System.out.println("Oracle Msg Id :" + topic.getMessageId());
       System.out.println("is new since session :" +
                               topic.isNewSinceSession());
       System.out.println("is new since last visit :" +
                               topic.isNewSinceLastVisit());
       System.out.println("is new " + topic.isNewMessage());
       System.out.println("Has Attachments :" + topic.isHasAttachments());
       System.out.println("X Priority :" + topic.getXPriority());
       System.out.println("Sent dat :" + topic.getSentDate().getTime());
```

```
            System.out.println("Size :" + topic.getSize());
            System.out.println("Webui URL :" + topic.getWebUIUrl());
            System.out.println("HTML ? :" + topic.isHTMLContentType());
            System.out.println("Hidden :" + topic.isHidden());
        }
        //STEP - 2: User Logout
                logout();
            }
          catch(TdWSException ex)
          {
            System.out.println("Error Code :" + ex.getErrorCode());
            System.out.println("Error Message :" + ex.getErrorMessage());
            String[] trace = ex.getServerStackTrace();
            if(trace != null)
            {
               for(int i = 0 ; i < trace.length ; i++)
                   System.out.println("trace[" + i + "]" + trace[i]);
            }
         }
         catch(Exception ex)
         {
           System.out.println("Never went into tdex :" + ex.getMessage());
           ex.printStackTrace();
         }
     }

    public static void logout() throws Exception
    {
        System.out.println("User logout.");
        //Once all the webservices operations are done, invoke logout
        //operation.
        //On logout, user state is destroyed and the cookie is invalidated.
        as.logout();
        System.out.println("Done");
    }
}
```

The `editMessageWithByte()` method edits the topic with the byte array supplied. The byte array should be MIME compatible. The entire message is replaced with the content of the byte array. A new MIME message object is constructed and it overwrites the entire old message.

The sample output for Example 11–4 is as follows:

```
Cookie : JSESSIONID=8c57046a22b88d154290041d4bd0bc48bff029d632b1
Subject :Edited Subject
Plain Text :Edited Body
From Address :Admin <td_superuser@stacx01.us.oracle.com>
Level :0
No of replies :2
Oracle Msg Id :281488
is new since session :false
is new since last visit :false
is new false
Has Attachments :false
X Priority :1
Sent dat :Sun Feb 26 20:13:13 PST 2006
Size :472
Webui URL
:http://stacx01:8888/discussions/app/mainAction.do?fid=15524&tid=281488&mid=281488
#281488
```

```
HTML ? :false
Hidden :false
User logout.
Done
```

12

Understanding the User Service

The User Service interface provides user access operations. It provides the following methods to retrieve user information based on the user attributes.

- Retrieving User Information by Nickname
- Retrieving User Information by E-mail

Retrieving User Information by Nickname

The following is an example for retrieving the user information by nickname:

Example 12–1 Retrieving User Information by Nickname

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.UserServiceServiceLocator;
import oracle.discussions.ws.UserServiceSoapBindingStub;
import oracle.discussions.ws.UserService;

/*
 * Tests the Oracle Discussions Web Services functionality.
 * Invokes various web services methods, to retrieve the results and
 * check if the results are fine.
 */
public class WSClient
{
  private static AuthenticationService as = null;
  public static String getCookie() throws Exception
  {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about
      //the webservices endpoint.
       AuthenticationServiceServiceLocator assl = new
                       AuthenticationServiceServiceLocator();
```

```
        //Get the handle to the actual webservices interface.
         as = assl.getAuthenticationService();
        //Indicate to the server that the client is interested in maintaining
        //session
        //HTTP is stateless and the user state is maintained in the session.
        //Unless the user is willing to participate in a session, user state
        //cannot be preserved.
         ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);

        //Invoke the actual login webservices call.
         as.login("td_superuser","welcome1");
        //Retrieve the cookie. The cookie is set on successful
        //authentication.
         String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                          ._getCall().getMessageContext()
                          .getProperty(HTTPConstants.HEADER_COOKIE);
         System.out.println("Cookie : " + cookie);
         return cookie;
    }

public static void main(String args[])
{
   try
   {
        //STEP - 1: User Login
         String cookie = getCookie();

        //STEP - 2: Locate the web services end point.
        //The Web Services locator contains the webservices end point
        //address. Therefore instantiate the service locator.
         UserServiceServiceLocator ussl = new UserServiceServiceLocator();

        //Get a handle to the actual web service.
         UserService us = ussl.getUserService();

        //Indicate to the server that the user is willing to participate in
        //the session.
        //Since HTTP is stateless, all the client data is maintained in the
        //session
         ((UserServiceSoapBindingStub)us).setMaintainSession(true);

        //Set the cookie retrieved in the call to login webservice.
        //The cookie asserts user's identity to the server.
         ((javax.xml.rpc.Stub)us)._setProperty(HTTPConstants.HEADER_
                                              COOKIE,cookie);

        //Invoke the actual web services call.
         User user = us.getUserByNickname("td_superuser");
         if(user != null)
         {
           System.out.println("User Nickname :" + user.getNickname());
           System.out.println("User Guid :" + user.getGuid());
           System.out.println("User Primary Email :" +
                              user.getPrimaryEmailAddress());
           System.out.println("User DN :" + user.getDN());
           System.out.println("User Domain :" + user.getDomain());
           System.out.println("User Timezone :" + user.getTimeZone());
           System.out.println("User Locale :" + user.getLocale());
           System.out.println("User Creation date :" +
                              user.getMemberSince().getTime());
```

```
            System.out.println("User Last logout time :" +
                               user.getLastLogoutTime().getTime());
            System.out.println("User Last login time :" +
                               user.getLastLoginTime().getTime());
            System.out.println("User Displayname :" + user.getDisplayName());
            System.out.println("User Presence URL :" + user.getPresenceURL());
            System.out.println("User Email id :" + user.getEmail());
            System.out.println("User client chat uri :" + user.getClientChatUri());
            System.out.println("User presence img url :" +
                               user.getPresenceImgUrl());
         }

        //STEP - 2: User Logout
         logout();
      }

      catch(TdWSException ex)
      {
         System.out.println("Error Code :" + ex.getErrorCode());
         System.out.println("Error Message :" + ex.getErrorMessage());
         String[] trace = ex.getServerStackTrace();
         if(trace != null)
         {
           for(int i = 0 ; i < trace.length ; i++)
             System.out.println("trace[" + i + "]" + trace[i]);
          }
      }
      catch(Exception ex)
      {
         System.out.println("Never went into tdex :" + ex.getMessage());
         ex.printStackTrace();
      }
    }

    public static void logout() throws Exception
    {
      System.out.println("User logout.");
     //Once all the webservices operations are done, invoke logout operation
     //On logout, user state is destroyed and the cookie is invalidated.
      as.logout();
      System.out.println("Done");
    }
}
```

The `getUserByNickname()` method returns the user information based on the user nickname. An exception is raised if an invalid user nickname is provided.

The sample output for Example 12–1 is as follows:

```
Cookie : JSESSIONID=8c57046a22b87a5f71b2d03c454fae86e587d7d19393
User Nickname :td_superuser
User Guid :09EC74D0F1A043ADE040578CDE022BB2
User Primary Email :td_superuser@stacx01.us.oracle.com
User DN :cn=td_superuser,cn=users,dc=us,dc=oracle,dc=com
User Domain :stacx01.us.oracle.com
User Timezone :Pacific Standard Time
User Locale :en_US
User Creation date :Sun Feb 26 19:54:18 PST 2006
User Last logout time :Sun Feb 26 19:54:18 PST 2006
User Last login time :Sun Feb 26 19:54:18 PST 2006
User Displayname :Admin
```

```
User Presence URL
:http://stacx01.us.oracle.com:7777/imtapp/Presence?TID=15886&UID=td_
superuser%40us.oracle.com&MODE=JS&K1=AD4F6AD5A4A3D359C9F917DA6D00FEA0CBE79A4B
User Email id :td_superuser@stacx01.us.oracle.com
User client chat uri :rtcmsgr:sendmsg?td_superuser&#64us.oracle.com
User presence img url
:http://stacx01.us.oracle.com:7777/imtapp/Presence?TID=15886&UID=td_
superuser%40us.oracle.com&MODE=IMG&K1=6DDB8CCE52E0A9C70E7E6C78A5A14DAB2D8D9736
User logout.
Done
```

Retrieving User Information by E-mail

Following is an example for retrieving the user information by e-mail:

Example 12–2

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.UserServiceServiceLocator;
import oracle.discussions.ws.UserServiceSoapBindingStub;
import oracle.discussions.ws.UserService;

/**
 * Tests the Oracle Discussions Web Services functionality.
 * Invokes various Web Services methods to retrieve the results and
 * check if the results are fine.
 */
public class WSClient
{
  private static AuthenticationService as = null;
  public static String getCookie() throws Exception
  {
    //STEP - 0: User Login
    //Create the service locator for authentication service. The locator contains
    //information about the Web Services endpoint
     AuthenticationServiceServiceLocator assl = new
                         AuthenticationServiceServiceLocator();
    //Get the handle to the actual webservices interface.
     as = assl.getAuthenticationService();
    //Indicate to the server that the client is interested in maintaining session.
    //HTTP is stateless and the user state is maintained in the session.
    //Unless the user is willing to participate in a session, user state cannot be
    //preserved.
     ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
    //Invoke the actual login webservices call.
     as.login("td_superuser","welcome1");
    //Retrieve the cookie. The cookie is set on successful authentication.
     String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
       ._getCall().getMessageContext().getProperty(HTTPConstants.HEADER_COOKIE);
```

```
 System.out.println("Cookie : " + cookie);
 return cookie;
}

public static void main(String args[])
{
  try
  {
     //STEP - 0: User Login
      String cookie = getCookie();

     //STEP - 1: Locate the Web Services end point.
     //The Web Services locator contains the Web Services end point address.
     //Hence instantiate the service locator.
      UserServiceServiceLocator ussl = new UserServiceServiceLocator();

     //Get a handle to the actual web service.
      UserService us = ussl.getUserService();

     //Indicate to the server that the user is willing to participate in the
     //session.
     //Since HTTP is stateless, all the client data is maintained in the session.
      ((UserServiceSoapBindingStub)us).setMaintainSession(true);

     //Set the cookie retrieved in the call to login webservice.
     //The cookie asserts user's identity to the server.
      ((javax.xml.rpc.Stub)us)._setProperty(HTTPConstants.HEADER_COOKIE,cookie);

     //Invoke the actual web services call.
      User user = us.getUserByEmail("td_superuser@stacx01.us.oracle.com");
      if(user != null)
      {
         System.out.println("User Nickname :" + user.getNickname());
         System.out.println("User Guid :" + user.getGuid());
         System.out.println("User Primary Email :" +
                            user.getPrimaryEmailAddress());
         System.out.println("User DN :" + user.getDN());
         System.out.println("User Domain :" + user.getDomain());
         System.out.println("User Timezone :" + user.getTimeZone());
         System.out.println("User Locale :" + user.getLocale());
         System.out.println("User Creation date :" +
                            user.getMemberSince().getTime());
         System.out.println("User Last logout time :" +
                            user.getLastLogoutTime().getTime());
         System.out.println("User Last login time :" +
                            user.getLastLoginTime().getTime());
         System.out.println("User Displayname :" + user.getDisplayName());
         System.out.println("User Presence URL :" + user.getPresenceURL());
         System.out.println("User Email id :" + user.getEmail());
         System.out.println("User client chat uri :" + user.getClientChatUri());
         System.out.println("User presence img url :" +
                            user.getPresenceImgUrl());
      }

      //STEP - 2: User Logout
       logout();
    }
    catch(TdWSException ex)
    {
        System.out.println("Error Code :" + ex.getErrorCode());
```

```
                System.out.println("Error Message :" + ex.getErrorMessage());
                String[] trace = ex.getServerStackTrace();
                if(trace != null)
                {
                   for(int i = 0 ; i < trace.length ; i++)
                   System.out.println("trace[" + i + "]" + trace[i]);
                }
             }
          catch(Exception ex)
          {
             System.out.println("Never went into tdex :" + ex.getMessage());
             ex.printStackTrace();
          }
       }

       public static void logout() throws Exception
       {
             System.out.println("User logout.");
            //Once all the webservices operations are done, invoke logout
            //operation
            //On logout, user state is destroyed and the cookie is invalidated
             as.logout();
             System.out.println("Done");
        }
}
```

The `getUserByEmail()` method returns the user based on the e-mail ID of the user. An exception is raised if an invalid user e-mail address is provided. It returns the user bean representing the user. A `TdWSException` is thrown on any error in retrieving the user information or if the user e-mail is invalid.

The sample output of Example 12–2 is as follows:

```
Cookie : JSESSIONID=8c57046a22b87a5f71b2d03c454fae86e587d7d19393
User Nickname :td_superuser
User Guid :09EC74D0F1A043ADE040578CDE022BB2
User Primary Email :td_superuser@stacx01.us.oracle.com
User DN :cn=td_superuser,cn=users,dc=us,dc=oracle,dc=com
User Domain :stacx01.us.oracle.com
User Timezone :Pacific Standard Time
User Locale :en_US
User Creation date :Sun Feb 26 19:54:18 PST 2006
User Last logout time :Sun Feb 26 19:54:18 PST 2006
User Last login time :Sun Feb 26 19:54:18 PST 2006
User Displayname :Admin
User Presence URL
:http://stacx01.us.oracle.com:7777/imtapp/Presence?TID=15886&UID=td_
superuser%40us.oracle.com&MODE=JS&K1=AD4F6AD5A4A3D359C9F917DA6D00FEA0CBE79A4B
User Email id :td_superuser@stacx01.us.oracle.com
User client chat uri :rtcmsgr:sendmsg?td_superuser&#64us.oracle.com
User presence img url
:http://stacx01.us.oracle.com:7777/imtapp/Presence?TID=15886&UID=td_
superuser%40us.oracle.com&MODE=IMG&K1=6DDB8CCE52E0A9C70E7E6C78A5A14DAB2D8D9736
User logout.
Done
```

13

Understanding the Subscription Service

The Subscription Service interface provides methods related to subscriptions on Oracle Discussions elements. Any Oracle Discussions element that is added as a favorite can be subscribed. After subscribing to an element, the subscribing user will receive e-mail notifications, if there is any change in the state of the element or any of its child elements. For example, if a user subscribes to a forum, the user will receive a notification on any change in any of the topics or messages in the forum or if there is any change in the forum state itself.

A subscription is removed under the following circumstances:

- User access is removed on that element
- The element is deleted
- User unsubscribes that element

This chapter addresses the following tasks:

Updating a Container Subscription

The following is an example for updating a container subscription:

Example 13–1 Updating a Subscription

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.SubscriptionServiceSoapBindingStub;
import oracle.discussions.ws.SubscriptionServiceServiceLocator;
import oracle.discussions.ws.SubscriptionService;

public class WSClient
{
  private static AuthenticationService as = null;
```

```
  public static String getCookie() throws Exception
  {
     //STEP - 0: User Login
     //Create the service locator for authentication service. The locator
     //contains information about the Web Services endpoint
      AuthenticationServiceServiceLocator assl = new
                   AuthenticationServiceServiceLocator();
     //Get the handle to the actual webservices interface.
      as = assl.getAuthenticationService();
     //Indicate to the server that the client is interested in maintaining
     //session
     //HTTP is stateless and the user state is maintained in the session.
     //Unless the user is willing to participate in a session, user state
     //cannot be preserved.
      ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
     //Invoke the actual login webservices call.
      as.login("td_superuser","welcome1");
     //Retrieve the cookie. The cookie is set on successful authentication.
      String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                ._getCall().getMessageContext()
                                .getProperty(HTTPConstants.HEADER_COOKIE);
      System.out.println("Cookie : " + cookie);
      return cookie;
  }

  public static void main(String args[])
  {
    try
    {
//STEP - 1: User Login
 String cookie = getCookie();

//STEP - 2: Locate the Web Services end point
//The Web Services locator contains the webservices end point address
//Hence instantiate the service locator.
 SubscriptionServiceServiceLocator sssl = new
                SubscriptionServiceServiceLocator();

//Get a handle to the actual web service.
 SubscriptionService ss = sssl.getSubscriptionService();

//Indicate to the server that the user is willing to participate in the
//session
//Since HTTP is stateless, all the client data is maintained in the session
 ((SubscriptionServiceSoapBindingStub)ss).setMaintainSession(true);

//Set the cookie retrieved in the call to login webservice.
//The cookie asserts user's identity to the server.
 ((javax.xml.rpc.Stub)ss)._setProperty(HTTPConstants.HEADER_COOKIE,cookie);

//Invoke the actual web services call.
 Subscription sub = new Subscription();
//The ID changes accordingly
 sub.setContainerId(85766);
 sub.setType(0);
 sub.setEmailAddress("tduser@stacx01.us.oracle.com");
//The ID changes accordingly
 sub.setTopicId(281487);
 ss.updateSubscription(sub);
```

```
        System.out.println("Subscription successfully updated");
      //The IDs need to be changed accordingly
       sub = ss.getTopicSubscription(85766,281487);
       if(sub != null)
       {
         System.out.println("Author :" + sub.getEmailAddress());
         System.out.println("Container id :" + sub.getContainerId());

         System.out.println("Topic id :" + sub.getTopicId());
       }

       //STEP - 2: User Logout
        logout();
     }
     catch(TdWSException ex)
     {
        System.out.println("Error Code :" + ex.getErrorCode());
        System.out.println("Error Message :" + ex.getErrorMessage());
        String[] trace = ex.getServerStackTrace();
        if(trace != null)
        {
           for(int i = 0 ; i < trace.length ; i++)
           System.out.println("trace[" + i + "]" + trace[i]);
        }
     }
     catch(Exception ex)
      {
          System.out.println("Never went into tdex :" + ex.getMessage());
          ex.printStackTrace();
      }
  }
   public static void logout() throws Exception
   {
        System.out.println("User logout.");
       //Once all the webservices operations are done, invoke logout
       //operation.
       //On logout, user state is destroyed and the cookie is invalidated.
        as.logout();
        System.out.println("Done");
   }
}
```

The `updateSubscription()` method updates the subscription corresponding to the container or the topic represented by the `id` fields in the subscription bean. If the topic ID is `0`, and if the container ID represents a valid container, then the subscription represents a container subscription and is updated. If both topic and container IDs are greater than `0`, and if the IDs represent a valid topic, then the topic subscription is updated. This method returns a a subscription bean corresponding to the subscription with the updated values.

The `getContainerSubscription()` method returns the subscription for the logged-in user on the container represented by `lContainerId`. This method returns `null` if the user does not have any subscription on the container.

The sample output for Example 13–1 is as follows:

```
Cookie : JSESSIONID=8c57046a22b89075a82ab179489c9d53ffc82b40f783
Subscription successfully updated
Author :tduser@stacx01.us.oracle.com
Container id :85766
Topic id :281487
```

```
User logout.
Done
```

Unsubscribing from an Existing Container Subscription

The following is an example for unsubscribing from an existing container subscription:

Example 13–2 Unsubscribing to an Existing Subscription

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.SubscriptionServiceSoapBindingStub;
import oracle.discussions.ws.SubscriptionServiceServiceLocator;
import oracle.discussions.ws.SubscriptionService;

/*
 * Tests the Oracle Discussions Web Services functionality.
 * Invokes various web services methods, to retrieve the results and
 * check if the results are fine.
 */
public class WSClient
{
   private static AuthenticationService as = null;
   public static String getCookie() throws Exception
   {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about the Web Services endpoint
       AuthenticationServiceServiceLocator assl = new
                        AuthenticationServiceServiceLocator();
      //Get the handle to the actual webservices interface.
       as = assl.getAuthenticationService();
      //Indicate to the server that the client is interested in maintaining
      //session
      //HTTP is stateless and the user state is maintained in the session.
      //Unless the user is willing to participate in a session, user state cannot
      //be preserved.
       ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
      //Invoke the actual login webservices call.
       as.login("td_superuser","welcome1");
      //Retrieve the cookie. The cookie is set on successful authentication
       String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                              ._getCall().getMessageContext()
                              .getProperty(HTTPConstants.HEADER_COOKIE);
       System.out.println("Cookie : " + cookie);
       return cookie;
   }

   public static void main(String args[])
```

```
{
  try
  {
      //STEP - 1: User Login
             String cookie = getCookie();

            //STEP - 2: Locate the Web Services end point.
            //The Web Services locator contains the webservices end point
            //address.
            //Hence instantiate the service locator
             SubscriptionServiceServiceLocator sssl = new
                           SubscriptionServiceServiceLocator();

            //Get a handle to the actual web service.
             SubscriptionService ss = sssl.getSubscriptionService();

            //Indicate to the server that the user is willing to participate in
            //the session.
            //Since HTTP is stateless, all the client data is maintained in the
            //session.
             ((SubscriptionServiceSoapBindingStub)ss).setMaintainSession(true);

            //Set the cookie retrieved in the call to login webservice.
            //The cookie asserts user's identity to the server.
             ((javax.xml.rpc.Stub)ss)._setProperty(HTTPConstants.HEADER_
                                                   COOKIE,cookie);

            //Invoke the actual web services call.
             Subscription sub = new Subscription();
            //The ID needs to be changed accordingly
             sub.setContainerId(85764);
             sub.setType(0);
            //The e-mail address needs to be changed accordingly
             sub.setEmailAddress("td_superuser@stacx01.us.oracle.com");
             sub.setTopicId(0);
             ss.unsubscribe(sub);

            //STEP - 2: User Logout
             logout();
         }
        catch(TdWSException ex)
        {
             System.out.println("Error Code :" + ex.getErrorCode());
             System.out.println("Error Message :" + ex.getErrorMessage());
             String[] trace = ex.getServerStackTrace();
             if(trace != null)
             {
                for(int i = 0 ; i < trace.length ; i++)
                System.out.println("trace[" + i + "]" + trace[i]);
              }
         }
        catch(Exception ex)
        {
             System.out.println("Never went into tdex :" + ex.getMessage());
             ex.printStackTrace();
        }
    }

    public static void logout() throws Exception
    {
```

```
                System.out.println("User logout.");
               //Once all the webservices operations are done, invoke logout
               //operation.
               //On logout, user state is destroyed and the cookie is invalidated.
                as.logout();
                System.out.println("Done");
   }
}
```

The `unsubscribe()` method removes the user subscription on the subscription element represented by the ID fields in the subscription bean. If the topic ID is 0, and if container ID represents a valid container, then the subscription represents a container subscription, and is unsubscribed. On the other hand, if both topic and container IDs are greater than 0, and if the IDs represent a valid topic, then the topic subscription is removed. The e-mail address of the subscription should correspond to the subscriber.

The sample output for Example 13–2 is as follows:

```
Cookie : JSESSIONID=8c57046a22b86759fc2215c0485fa5a1a7d6f381d9dc
Subscription successfully removed
User logout.
Done
```

The following is an example unsubscribing from a topic subscription:

Example 13–3 Unsubscribing from a Topic Subscription

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.SubscriptionServiceSoapBindingStub;
import oracle.discussions.ws.SubscriptionServiceServiceLocator;
import oracle.discussions.ws.SubscriptionService;

/*
 * Tests the Oracle Discussions Web Services functionality.
 * Invokes various web services methods, to retrieve the results and
 * check if the results are fine.
 */
public class WSClient
{
   private static AuthenticationService as = null;
   public static String getCookie() throws Exception
   {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
      //contains information about the Web Services endpoint
       AuthenticationServiceServiceLocator assl = new
                       AuthenticationServiceServiceLocator();
      //Get the handle to the actual webservices interface.
       as = assl.getAuthenticationService();
      //Indicate to the server that the client is interested in maintaining
```

```
//session
//HTTP is stateless and the user state is maintained in the session.
//Unless the user is willing to participate in a session, user state cannot
//be preserved.
 ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
//Invoke the actual login webservices call.
       as.login("td_superuser","welcome1");
      //Retrieve the cookie. The cookie is set on successful authentication
       String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                                ._getCall().getMessageContext()
                                .getProperty(HTTPConstants.HEADER_COOKIE);
       System.out.println("Cookie : " + cookie);
       return cookie;
    }

   public static void main(String args[])
   {
     try
     {
         //STEP - 1: User Login
          String cookie = getCookie();

         //STEP - 2: Locate the Web Services end point.
         //The Web Services locator contains the webservices end point
         //address.
         //Hence instantiate the service locator
          SubscriptionServiceServiceLocator sssl = new
                         SubscriptionServiceServiceLocator();

         //Get a handle to the actual web service.
          SubscriptionService ss = sssl.getSubscriptionService();

         //Indicate to the server that the user is willing to participate in
         //the session.
         //Since HTTP is stateless, all the client data is maintained in the
         //session.
          ((SubscriptionServiceSoapBindingStub)ss).setMaintainSession(true);

         //Set the cookie retrieved in the call to login webservice.
         //The cookie asserts user's identity to the server.
          ((javax.xml.rpc.Stub)ss)._setProperty(HTTPConstants.HEADER_
                                                  COOKIE,cookie);

         //Invoke the actual web services call.
          Subscription sub = new Subscription();
         //The ID needs to be changed accordingly
          sub.setContainerId(85764);
          sub.setType(0);
         //The e-mail address needs to be changed accordingly
          sub.setEmailAddress("td_superuser@stacx01.us.oracle.com");
         //The ID needs to be changed accordingly
          sub.setTopicId(281784);
          ss.unsubscribe(sub);

         //STEP - 2: User Logout
          logout();
     }
     catch(TdWSException ex)
     {
          System.out.println("Error Code :" + ex.getErrorCode());
```

```
                System.out.println("Error Message :" + ex.getErrorMessage());
                String[] trace = ex.getServerStackTrace();
                if(trace != null)
                {
                   for(int i = 0 ; i < trace.length ; i++)
                   System.out.println("trace[" + i + "]" + trace[i]);
                }
          }
          catch(Exception ex)
      {
                System.out.println("Never went into tdex :" + ex.getMessage());
                ex.printStackTrace();
      }
    }

    public static void logout() throws Exception
    {
                System.out.println("User logout.");
               //Once all the webservices operations are done, invoke logout
               //operation.
               //On logout, user state is destroyed and the cookie is invalidated.
                as.logout();
                System.out.println("Done");
    }
}
```

The sample output for Example 13–3 is as follows:

```
Cookie : JSESSIONID=8c57046a22b89bdff0080cd141e8940be470e31c2c8b
Subscription successfully removed
User logout.
Done
```

14

Understanding the MyDiscussions Service

The MyDiscussions Service interface provides methods to access content of user interest. This interface has methods for the following operations:

- Bookmarking Oracle Discussions elements
- Selectively accessing content of user interest
- Selectively accessing most popular or recent Oracle Discussions content
- Searching the favorite container of a user

This interface also provides bulk methods to add or remove containers and topics to favorites container.

This chapter focusses on Searching the Favorite Container of a User.

Searching the Favorite Container of a User

The following is an example for searching the favorite container of a user:

Example 14–1 Searching the favorite container of a user

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.MyDiscussionsServiceSoapBindingStub;
import oracle.discussions.ws.MyDiscussionsServiceServiceLocator;
import oracle.discussions.ws.MyDiscussionsService;

public class WSClient
{
   private static AuthenticationService as = null;
   public static String getCookie() throws Exception
   {
      //STEP - 1: User Login
      //Create the service locator for authentication service. The locator
      //contains information about the webservices endpoint.
       AuthenticationServiceServiceLocator assl = new
```

```
                        AuthenticationServiceServiceLocator();
 //Get the handle to the actual webservices interface.
  as = assl.getAuthenticationService();
 //Indicate to the server that the client is interested in maintaining
 //session.
 //HTTP is stateless and the user state is maintained in the session.
 //Unless the user is willing to participate in a session, user state
 //cannot be preserved.
  ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
 //Invoke the actual login webservices call.
  as.login("td_superuser","welcome1");
 //Retrieve the cookie. The cookie is set on successful authentication.
  String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                             ._getCall().getMessageContext()
                             .getProperty(HTTPConstants.HEADER_COOKIE);
  System.out.println("Cookie : " + cookie);
  return cookie;
}

public static void main(String args[])
{
  try
  {
   //STEP - 1: User Login
    String cookie = getCookie();

   //STEP - 2: Locate the web services end point.
   //The web services locator contains the webservices end point
   //address.
   //Hence instantiate the service locator.
    MyDiscussionsServiceServiceLocator mdssl = new
               MyDiscussionsServiceServiceLocator();

   //Get a handle to the actual web service.
    MyDiscussionsService mdsl = mdssl.getMyDiscussionsService();
   //Indicate to the server that the user is willing to participate in
   //the session.
   //Since HTTP is stateless, all the client data is maintained in the
   //session.
    ((MyDiscussionsServiceSoapBindingStub)mdsl)
                      .setMaintainSession(true);

   //Set the cookie retrieved in the call to login webservice.
   //The cookie asserts user's identity to the server.
    ((javax.xml.rpc.Stub)mdsl)._setProperty(HTTPConstants.HEADER_
                                          COOKIE,cookie);

   //Invoke the actual web services call.
    SearchTerm st = new SearchTerm();
    st.setAuthor("td_superuser");
    ForumMessage[] topics = mdsl.search(st);
    if(topics != null)
    {
     for(int i = 0 ; i < topics.length ; i++)
     {
       System.out.println("Got Something");
       System.out.println("Subject :" + topics[i].getSubject());
       System.out.println("Plain Text :" +
                      topics[i].getBodyPlainText());
       String[] frmAdd = topics[i].getFromAddresses();
```

```
                if(frmAdd != null)
                {
                    for(int j = 0 ; j < frmAdd.length ; j++)
                        System.out.println("From Address :" + frmAdd[j]);
                }
                System.out.println("Level :" + topics[i].getLevel());
                System.out.println("No of replies :" +
                                   topics[i].getNumberOfReplies());
                System.out.println("Oracle Msg Id :" + topics[i].getMessageId());
                System.out.println("is new since session :" +
                                   topics[i].isNewSinceSession());
                System.out.println("is new since last visit :" +
                                   topics[i].isNewSinceLastVisit());
                System.out.println("is new " + topics[i].isNewMessage());
                System.out.println("Has Attachments :" +
                                   topics[i].isHasAttachments());
                System.out.println("X Priority :" + topics[i].getXPriority());
                System.out.println("Sent date :" + topics[i].getSentDate());
                System.out.println("Size :" + topics[i].getSize());
                System.out.println("HTML ?  :" + topics[i].isHTMLContentType());
                System.out.println("WebUI URL :" + topics[i].getWebUIUrl());
             }
          }

          //STEP - 3: User Logout
           logout();
       }
       catch(TdWSException ex)
       {
          System.out.println("Error Code :" + ex.getErrorCode());
          System.out.println("Error Message :" + ex.getErrorMessage());
          String[] trace = ex.getServerStackTrace();
          if(trace != null)
          {
              for(int i = 0 ; i < trace.length ; i++)
                  System.out.println("trace[" + i + "]" + trace[i]);
          }
       }
       catch(Exception ex)
       {
          System.out.println("Never went into tdex :" + ex.getMessage());
          ex.printStackTrace();
       }
   }

   public static void logout() throws Exception
   {
       System.out.println("User logout.");
       //Once all the webservices operations are done, invoke logout
       //operation.
       //On logout, user state is destroyed and the cookie is invalidated.
       as.logout();
       System.out.println("Done");
   }
}
```

The `search()` method searches favorite containers of the user for the search criterion specified in the search term. It returns `null` if there are no favorite containers for the user. The search criterion can be supplied in the search term. Wild-card searches are also possible. The search is limited to searching message subject and content. To search

on a custom list of containers, use the search method in the Container Service interface.

The sample output for Example 14–1 is as follows:

```
Subject :To be Edited
Plain Text :a<table border="1" cellpadding="1" cellspacing="1"
width="500"><tbody><tr><td>Krishna<br></td><td><br></td><td><br></td></tr><tr><td>
<br></td><td><br></td><td><br></td></tr><tr><td><br></td><td><br></td><td>Chaitany
a<br></td></tr></tbody></table><br>
<br>
<a href="http://www.oracle.com" target="_blank">Oracle</a><br>

From Address :Admin <td_superuser@stacx01.us.oracle.com>
Level :0
No of replies :0
Oracle Msg Id :81131
is new since session :false
is new since last visit :false
is new false
Has Attachments :true
X Priority :1
Sent date
:java.util.GregorianCalendar[time=1137756640000,areFieldsSet=true,areAllFieldsSet=
true,lenient=true,zone=sun.util.calendar.ZoneInfo[id="GMT",offset=0,dstSavings=0,u
seDaylight=false,transitions=0,lastRule=null],firstDayOfWeek=1,minimalDaysInFirstW
eek=1,ERA=1,YEAR=2006,MONTH=0,WEEK_OF_YEAR=3,WEEK_OF_MONTH=3,DAY_OF_MONTH=20,DAY_
OF_YEAR=20,DAY_OF_WEEK=6,DAY_OF_WEEK_IN_MONTH=3,AM_PM=0,HOUR=11,HOUR_OF_
DAY=11,MINUTE=30,SECOND=40,MILLISECOND=0,ZONE_OFFSET=0,DST_OFFSET=0]
Size :63580
HTML ?  :true
WebUI URL
:http://stacx01:8888/discussions/app/mainAction.do?fid=6826&tid=81131&mid=81131#81
131
```

15

Understanding the RSS URL Service

RSS (Rich Site Summary) protocol RSS is a Web content syndication format. RSS aggregators reduce the time and effort needed to regularly check the Web sites of interest for updates, creating a unique information space. An aggregator is able to subscribe to a feed, check for new content at user-determined intervals, and retrieve the content. The content is sometimes described as being pulled to the subscriber.

Oracle Discussions provides RSS feeds for containers, topics and other elements of Oracle Discussions. An aggregator can be used to subscribe to the feed published by Oracle Discussions. The aggregator polls Oracle Discussions server and pulls the latest feed. These new updates are shown as a pop-up window on the system tray on the desktop.

RSS URL Service returns the URLs of various RSS feeds provided by Oracle Discussions.

Retrieving a RSS URL

The following is an example for retrieving a RSS URL for a category and a forum:

Example 15–1 Retrieving a RSS URL

```
import java.util.Date;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

import org.apache.axis.transport.http.HTTPConstants;
import oracle.discussions.ws.exceptions.TdWSException;
import oracle.discussions.ws.beans.*;
import oracle.discussions.ws.AuthenticationServiceServiceLocator;
import oracle.discussions.ws.AuthenticationServiceSoapBindingStub;
import oracle.discussions.ws.AuthenticationService;
import oracle.discussions.ws.RssUrlServiceServiceLocator;
import oracle.discussions.ws.RssUrlServiceSoapBindingStub;
import oracle.discussions.ws.RssUrlService;

public class WSClient
{
  private static AuthenticationService as = null;

  public static String getCookie() throws Exception
  {
      //STEP - 0: User Login
      //Create the service locator for authentication service. The locator
```

```
//contains information about the web Services endpoint
      AuthenticationServiceServiceLocator assl = new
                      AuthenticationServiceServiceLocator();
     //Get the handle to the actual webservices interface.
      as = assl.getAuthenticationService();
     //Indicate to the server that the client is interested in maintaining
     //session.
     //HTTP is stateless and the user state is maintained in the session.
     //Unless the user is willing to participate in a session, user state
     //cannot be preserved.
      ((AuthenticationServiceSoapBindingStub)as).setMaintainSession(true);
     //Invoke the actual login webservices call.
      as.login("td_superuser","welcome1");
     //Retrieve the cookie. The cookie is set on successful authentication
      String cookie = (String)((AuthenticationServiceSoapBindingStub)as)
                           ._getCall().getMessageContext()
                           .getProperty(HTTPConstants.HEADER_COOKIE);
      System.out.println("Cookie : " + cookie);
      return cookie;
   }

  public static void main(String args[])
  {
    try
    {
      //STEP - 0: User Login
       String cookie = getCookie();

      //STEP - 1: Locate the web services end point
      //The web services locator contains the webservices end point address
      //Hence instantiate the service locator
       RssUrlServiceServiceLocator russl = new
                         RssUrlServiceServiceLocator();
      //Get a handle to the actual web service.
       RssUrlService rus = russl.getRssUrlService();
      //Indicate to the server that the user is willing to participate in
      //the session.
      //Since HTTP is stateless, all the client data is maintained in the
      //session.
       ((RssUrlServiceSoapBindingStub)rus).setMaintainSession(true);
      //Set the cookie retrieved in the call to login webservice.
      //The cookie asserts user's identity to the server.
       ((javax.xml.rpc.Stub)rus)._setProperty(HTTPConstants.HEADER_
                                              COOKIE,cookie);
      //Invoke the actual web services call. Change the ID accordingly
       System.out.println("Url :" + rus.getContainerRssUrl(11212));

      //STEP - 2: User Logout
       logout();
     }
     catch(TdWSException ex)
     {
      System.out.println("Error Code :" + ex.getErrorCode());
      System.out.println("Error Message :" + ex.getErrorMessage());
      String[] trace = ex.getServerStackTrace();
      if(trace != null)
      {
        for(int i = 0 ; i < trace.length ; i++)
           System.out.println("trace[" + i + "]" + trace[i]);
       }
```

```
            }
      catch(Exception ex)
      {
          System.out.println("Never went into tdex :" + ex.getMessage());
          ex.printStackTrace();
      }
   }

   public static void logout() throws Exception
   {
       System.out.println("User logout.");
       //Once all the webservices operations are done, invoke logout
       //operation.
       //On logout, user state is destroyed and the cookie is invalidated.
       as.logout();
       System.out.println("Done");
   }
}
```

The `getContainerRssUrl()` method returns the RSS URL of the container depicted by the supplied container ID. RSS URL depicting a category or forum is returned, depending on if the supplied container id corresponds to a category or forum. On copying the category URL into a browser, RSS feed with details about all the containers in the category is printed on the browser screen. On copying the forum URL, a feed with all the topics in the forum is printed.

The sample output for Example 15–1 is as follows:

```
Cookie : JSESSIONID=8c57046a22b8492acb2e79ef4e6db6d9a0ee83eeb356
The Rss Url : http://stacx01.us.oracle.com:7777/discussions/rss/fid=11212
```

A

Performance Tips and Tricks

The following are some of the factors that affect the performance of Oracle Discussions Web Services:

- All the bulk operations in the Web Service interfaces take more time because they have to create or retrieve bulk data.
- Search is slower if the number of topics in a container is extremely large. It also takes more time if the wild card search criterion is given or if the search scope is for entire Oracle Discussions.
- All the listing processes of the containers, categories and forums will take more time if the call is invoked on the root of Oracle Discussions (`container id` is `-1` and `control` parameter as `*`).
- Listing the topics of a forum might take more time if the number of topics is very large in a forum.
- Listing of messages in a topic takes more time if the topic has large number of replies.
- In invoking bulk operations, users might experience client socket timeout exception. In the case of all create/update/delete operations (data definition, data manipulation) statements, though the socket times out, the operations will go through, unless there is an error in user input data. But in case of list/get operations (all retrieval operations), the user will not have a way of retrieving the data and he will have to narrow down/refine the query and fire it again.

  In create or update or delete operations, if there is an exception and the socket times out, the user will not be notified. It is planned to include notifications for bulk operations in next version.

Index