Oracle® Workspaces

Application Developer's Guide

10*g* Release 1 (10.1.2)

B25480-01

June 2006

ORACLE®

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

B25480-01

Contents

Preface

This Preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

Oracle Workspaces Application Developer's Guide is intended for any programmers and developers who intend to use the Oracle Workspaces SDK 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 Documents

For more information, see the following manuals in the Oracle Collaboration Suite documentation set:

- *Oracle Workspaces Java API Reference*

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 of Oracle Workspaces SDK

Oracle Workspaces is an extremely flexible framework that organizes and links Oracle Collaboration Suite components and their data.

Capabilities

The Oracle Workspaces SDK is a complete application development framework for integrating custom applications with the Oracle Workspaces environment. The following are the types of functionality the Oracle Workspaces SDK supports:

- Java SDK: A server-side SDK for accessing the Oracle Workspaces environment
  - Workspace management APIs
    - Create new, delete, search for, and access existing workspaces
    - Query and modify contents of workspaces (add, modify, and remove library contents, meetings, tasks, discussion forums and topics, email and discussion messages, announcements, views and links)
    - Modify workspace properties, such as name and description. See Appendix A, "Workspace and Application Properties" for a list of these properties.
    - Add and remove workspace members, modify member's roles, and access member's presence (and initiate chat sessions using Oracle Real-Time Collaboration protocol).
  - Application management APIs
    - Manage application-level properties and roles
    - Manage workspace templates
    - Manage component services (such as Oracle Calendar)
- URI model
  - Oracle Workspaces provides a documented URL-based access method to retrieve any page in the Oracle Workspaces Web client.
- Template development
  - Templates are XML documents that are used to create new workspaces with some predefined settings and content. These settings and content include workspace properties, library structure, library content, discussion forums, tasks, views, and members. Templates can be created based on an existing workspace, or created and edited with standard XML development tools or the Java SDK.

Supported Standards

The following standards are supported by Oracle Workspaces:

- IMAP/SMTP for Workspaces Inbox and Discussions
- WebDAV Access to Library content
- All other protocols supported by Oracle Content Services. See *Oracle Content Services Application Developer's Guide* for more information.

2

Architecture of Oracle Workspaces

This chapter describes the main parts of Oracle Workspaces, and the Java classes that represent these parts in the SDK.

Parts of Oracle Workspaces

Oracle Workspaces consists of the following top-level components:

- Users
- Workspaces
- Templates

Users

A *user* is anyone who is provisioned with an Oracle Collaboration Suite account. Users have access to the Oracle Workspaces application.

An *application role* is a description of the actions that a user may perform in the Oracle Workspaces application. The following application roles exist:

- *Application administrator*: Users who have this role are "super users." They may perform administration tasks, such as grant other users roles, manage templates, or act like the workspace administrators for any workspace.

- *Workspace creator*: Users who have this role may create workspaces.

If users are not granted any of these roles, they are simply *workspaces users*. Any user can be added as a member to any workspace.

When assigning roles to users or adding users as members to a workspace, you could choose to assign workspace roles (reader, writer or administrator) to any user or group. A *group* is a representation of an Oracle Internet Directory group. The group is expanded to its members before performing a particular operation (such as role assignment or the addition of members to a workspace). As a result, operations are not performed on the group but on the individual users of the group.

Workspaces

A *workspace* is a container that groups members, resources, and views. The workspace controls how its members may interact with the resources it contains.

Members

A *member* is a user that belongs to a workspace. Each member may be granted a *workspace role*.

A *workspace role* is a description of the actions that the member may perform within one particular workspace. Three predefined workspace roles exist:

- *Reader*: Members who have this role may view workspace content but may not create new content or edit or delete existing content.

- *Writer*: Members who have this role may read, edit, and delete existing content and add new content.

- *Administrator*: Members who have this role have all the privileges of the writer role, and they may also manage workspace configuration and membership.

Resources

A *resource* is an object that represents an Oracle Collaboration Suite component, or a subsystem that manages a pre-determined set of objects and their functions. The following table describes the resources available in Oracle Workspaces, the name by which they are represented in the Oracle Workspaces Web client, and the Oracle Collaboration Suite components that they represent:

Table 2–1   Resource Class Names and Corresponding Oracle Collaboration Suite Component

Java SDK Resource Class Name	What You See in the Oracle Workspaces Web Client	Oracle Collaboration Suite Component
CwCalendarResource	Meetings & Tasks	Oracle Calendar
CwDiscussionResource	Discussions	Oracle Discussions
CwEmailResource	Inbox	Oracle Mail
CwFilesResource	Library	Oracle Content Services
CwViewResource	View	No Oracle Collaboration Suite component equivalent, a workspace-specific resource

No two resources of the same type may be added to the same workspace. Each workspace has its own instance of each resource.

Resource Items

A *resource item* is an object that represents a data item in a specific resource or workspace. The following table describes the resource items that each resource controls, and the name by which each resource item is represented in the Oracle Workspaces Web client:

Table 2–2   Resource Item Class Names and Corresponding Oracle Workspaces Web Client Components

Resource	Java SDK Resource Item Class Name	What You See in the Oracle Workspaces Web Client
Calendar	CwEvent	Meeting
Calendar	CwTask	Task
Discussion	CwBoard	Forum
Discussion	CwThread	Topic
Discussion	CwMessage	Message
Files	CwFolder	Folder

Table 2–2   (Cont.) Resource Item Class Names and Corresponding Oracle Workspaces Web Client Components

Resource	Java SDK Resource Item Class Name	What You See in the Oracle Workspaces Web Client
Files	CwFilesResourceItem, CwFilesResourceItemWithContent	File
Not managed by resource but by workspace	CwAnnouncement, CwAnnouncementList	Announcements

A resource may have multiple instances of resource items. Resource item classes extend the CwResourceItem class.

Announcements

An *announcement* is a message posted by a member of a workspace that all members of that workspace may see on the workspace home page. Each announcement can have a title, body and an expiration date.

Views

A *view* is a group of resource items that is treated as one resource item. For example, a product development workspace can contain a view for new developers. This view can include functional and design specifications contained in a files resource, and a developer's forum contained in a discussion resource.

Templates

A *template* is a container for general workspace characteristics that can be used as a basis for creating new workspaces. A template is stored as an XML file.

3

Building Applications with Oracle Workspaces Java API

This chapter describes the following requirements and tasks involved in building applications with Oracle Workspaces Java API:

- Requirements for Compilation and Execution of Applications that Use Oracle Workspaces Java API
- Creating a Connection to Oracle Workspaces
- Managing Application Properties
- Working with Workspaces
- Managing Content in a Workspace

Requirements for Compilation and Execution of Applications that Use Oracle Workspaces Java API

This section describes settings you must configure before executing applications that use Oracle Workspaces Java API.

Execution Location of Applications that Use Oracle Workspaces Java API

Run Java applications that use the Oracle Workspaces Java API from a machine on which Oracle Collaboration Suite Applications tier is installed.

Classpath of Oracle Workspaces Java API

Set your CLASSPATH environment variable to include the following files, where the environment variable ORACLE_HOME is set to the directory of your Oracle Collaboration Suite instance:

```
$ORACLE_HOME/discussions/lib/discussions.jar
$ORACLE_HOME/j2ee/home/jazncore.jar
$ORACLE_HOME/j2ee/home/lib/connector.jar
$ORACLE_HOME/j2ee/home/lib/ejb.jar
$ORACLE_HOME/j2ee/home/lib/http_client.jar
$ORACLE_HOME/j2ee/home/lib/jaas.jar
$ORACLE_HOME/j2ee/home/lib/jms.jar
$ORACLE_HOME/j2ee/home/lib/jndi.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/
  applications/workspaces/workspaces/WEB-INF/lib/rtcSDK.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/
  applications/workspaces/workspaces/WEB-INF/lib/files_searchlet.jar
```

```
$ORACLE_HOME/jdbc/lib/classes12.jar
$ORACLE_HOME/jlib/collabsuiteuser.jar
$ORACLE_HOME/jlib/esadmin.jar$ORACLE_HOME/jlib/escommon.jar
$ORACLE_HOME/jlib/esldap.jar
$ORACLE_HOME/jlib/esmail_sdk.jar
$ORACLE_HOME/jlib/javax-ssl-1_1.jar
$ORACLE_HOME/jlib/jssl-1_1.jar
$ORACLE_HOME/jlib/ldap.jar
$ORACLE_HOME/jlib/ldapjclnt10.jar
$ORACLE_HOME/jlib/ocsaddressbook.jar
$ORACLE_HOME/jlib/ocscommon.jar
$ORACLE_HOME/jlib/ojmisc.jar
$ORACLE_HOME/jlib/providerutil.jar
$ORACLE_HOME/jlib/repository.jar
$ORACLE_HOME/lib/activation.jar
$ORACLE_HOME/lib/dms.jar
$ORACLE_HOME/lib/mail.jar
$ORACLE_HOME/lib/servlet.jar
$ORACLE_HOME/lib/xmlparserv2.jar
$ORACLE_HOME/ocal/jlib/ocal_clnt.jar
$ORACLE_HOME/soap/lib/soap.jar
$ORACLE_HOME/workspaces/lib/workspaces_sdk.jar
$ORACLE_HOME/workspaces/lib/workspaces_share.jar
$ORACLE_HOME/ultrasearch/lib/searchlet.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/applications/workspaces/workspaces/WEB-INF/lib/
  rtcSDK.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/applications/workspaces/workspaces/WEB-INF/lib/
  content-ws-client.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/applications/workspaces/workspaces/WEB-INF/lib/
  axis.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/applications/workspaces/workspaces/WEB-INF/lib/
  jaxrpc.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/applications/workspaces/workspaces/WEB-INF/lib/
  commons-logging.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/applications/workspaces/workspaces/WEB-INF/lib/
  commons-discovery.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/applications/workspaces/workspaces/WEB-INF/lib/
  saaj-api.jar
$ORACLE_HOME/j2ee/OC4J_OCSClient/applications/workspaces/workspaces/WEB-INF/lib/
  wsdl4j.jar
$ORACLE_HOME/jlib/orai18n-js.jar
```

Configuration File for Oracle Workspaces Applications

You must provide a configuration file that instructs how your Oracle Workspaces applications connect to Oracle Collaboration Suite Service Registry (in Oracle Internet Directory), which, in turn, provides your application with information about the Oracle Workspaces instance to which it connects.

This file should be deployed on your server and made readable by your applications. (See "Java System Properties" to see how this file is used by applications.)

Create a file with the following contents. (Ensure that you change the Oracle Internet Directory settings to reflect your Oracle Collaboration Suite deployment:

```
# ----------------------------------------
# Logging Settings (OC4J is for debugging)
# ----------------------------------------

# For more info (very verbose), set this log level to debug
oracle.workspaces.log.level              = fatal
```

```
# ---------------------------------------
# Oracle Internet Directory Settings
# ---------------------------------------

oracle.ocs.ldappool.host                = my-company.com
oracle.ocs.ldappool.port                = 3060
oracle.ocs.ldappool.user_dn             = cn=orcladmin
oracle.ocs.ldappool.user_password       = welcome1

# If not in an OC4J container, then skip events
cw.test.use_jms = n
```

Creating a Connection to Oracle Workspaces

This section describes Java system properties you must set and how to log in and initialize Oracle Workspaces.

Java System Properties

Before creating a connection to an Oracle Workspaces instance, your application needs to know which instance to connect to. Oracle Workspaces stores its connection information in the Collaboration Suite Service Registry, which is stored in Oracle Internet Directory. Therefore, you need to provide a configuration file (which has been described previously). Initializing and logging into the Oracle Workspaces system involves the following steps:

Set the following system properties when you execute your Java application:

Table 3–1   Java System Properties

System Property	Description
cw.configFile	The name of your configuration file
oracle.ons.oraclehome	$ORACLE_HOME
java.library.path	$ORACLE_HOME/lib:$ORACLE_HOME/ocal/lib
oracle.home	$ORACLE_HOME
oracle.ons.instancename	The name of the OracleAS Infrastructure instance. You can find the instance name in $ORACLE_HOME/config/ias.properties as follows: IASname=myapps.my-company.com

You may use one of the following options to set these properties:

- Use the -D option to set these properties if you are executing your Java application from the command line.

- Call the System.setProperty method to set these properties within your Java code

Example: Set Java System Properties

The following code extract sets required system properties:

```
public void init(ServletConfig config) throws ServletException
{
  super.init(config);
  String O_H="/home/user1/m12d/apps";
  System.setProperty(
```

```
      "cw.configFile",
      "/home/user1/samples/workspaces/cwConfigFile");
    System.setProperty("oracle.ons.oraclehome", O_H);
    System.setProperty("oracle.home", O_H);
    System.setProperty(
      "oracle.ons.instancename",
      "ocsapps.collabtng15.us.oracle.com");
  }
```

Example: Initialize and Log into Oracle Workspaces, Grant Workspace Application Roles

The following example initializes Oracle Workspaces, grants a specified user workspace creator role, and logs that user into Oracle Workspaces. Note that you will probably not grant workspace roles in your applications; this example is here for informational purposes. However, Oracle Workspaces initialization and logging in as a user will be performed in each of your applications.

This example performs the following steps:

1. Calls the CwService.init() method to initialize Oracle Workspaces. Ensure that you have configured the appropriate Oracle Internet Directory settings in your configuration file.

2. Opens a session with a user who has application administration rights.

3. With this application administrator session, grants workspace creator roles to a specified user.

4. Logs in to the system by opening a new session with the specified user.

```java
import oracle.workspaces.CwService;
import oracle.workspaces.authorization.CwApplicationRole;
import oracle.workspaces.authorization.CwUser;
import oracle.workspaces.session.CwSession;

public class Initialization {

  public static CwService initializeService()
    throws Exception
  {
    try {

      // The method CwService.init() must be called before
      // performing any Oracle Workspaces operations

      CwService.init();
      CwService currentService = CwService.getInstance();
      return currentService;
    } catch (Throwable t) {
      t.printStackTrace();
      System.exit(1);
    }
  }

  public static CwUser createAppAdmin(
    CwService currentService, String appAdminName)
    throws Exception
  {

    // Obtain a user instance of the user from Oracle Internet Directory
```

```
CwUser appAdminUser = currentService.getUserByName(appAdminName);

// Open a session as a user with application administration
// rights. orcladmin has this by default.

CwSession orclAdminSession =
  currentService.openSession(currentService.getUserByName("orcladmin"));

// Grant workspace creator role to the user

orclAdminSession.grant(
  CwApplicationRole.CW_APPLICATION_ROLE_WORKSPACE_CREATORS,
  appAdminUser);

return appAdminUser;
}

public static CwSession loginUser(CwService currentService, CwUser currentUser)
  throws Exception
{
  // Login to the Oracle Workspaces system by opening a session with the user

  return currentService.openSession(currentUser);
}

public static void main(String[] args)
  throws Exception
{
  CwService myService = initializeService();
  CwUser myAppAdminUser = createAppAdmin(
    myService,
    "application.administrator");
  CwSession myAdminSession = loginUser(myService, myAppAdminUser);
}
}
```

CwService Class

The CwService class exposes the following APIs:

- System management: Use the methods init and destroy to initialize the Oracle Workspaces system. Call these methods in the client application only once during the lifetime of the application.

- Session management: Use the openSession and closeSession methods to open new or existing sessions and close existing sessions. Call the getIdentityById method to get a identity (user or group) instance based on its ID. The ID of an identity is the orclGuid attribute of the entry in Oracle Internet Directory. The methods getUserByDn() and getUserByName() respectively return an instance of CwUser given the DN or name of the user.

System management methods are static methods, while session management methods must be invoked on the instance returned by the getInstance() method.

CwSession Class

The CwSession class exposes the following APIs:

1. *Workspace management*: These methods create workspaces, delete workspaces, retrieve a workspace's instance given its ID or path, and lists all accessible workspaces.

2. *Template management*: These methods create a template from a workspace, as well as store, delete, and list templates. Once a template is created, it can be used to create a new workspace using the createWorkspace() method.

3. *Session management*: These methods return session information such as last login or logout time.

4. *Privilege management*: These methods grant, revoke, and list application-level roles.

5. *Recovery management*: These methods find information about failed operations that require recovery and recover workspaces and the CW system from these failed operations.

Managing Application Properties

See Appendix A, "Workspace and Application Properties" for a list of application properties that can be retrieved and specified. These properties let an application administrator specify the default template to be used for new templates and the workspace creation mode.

Working with Workspaces

This section describes how to create workspaces, search for existing workspaces, and set workspace properties.

Creating Workspaces

The CwWorkspace and CwWorkspaceDefinition classes are used to manage workspaces in the system. The CwSession class is used to create workspaces.

Workspaces are typically created from an existing template.

CwWorkspace Class

The CwWorkspace class exposes the following API:

- *Resource management*: These methods create, delete, and list resources in the workspace.

- *Workspace property management*: These methods retrieves and updates the workspace's properties. Workspace properties include the workspace's name, description, and configuration options. These methods do not provide information about the properties of its sub-workspaces.

- *Membership management*: These methods add, remove, and list members and subgroups in the workspace.

- *Member role management*: These methods lists and reassigns roles to members of the workspace.

CwWorkspaceDefinition Class

An instance of this class contains all the information needed to create a workspace or subworkspace.

Example: Create New Workspace Based on Default System Template

The following method returns a new workspace based on the default system template:

```
public static CwWorkspace
  createWorkspaceWithAllResources(
    CwSession session,
    String    workspaceName,
    String    workspaceDisplayname,
    String    workspaceDescription)
  throws Exception
  {

    // Create a workspace definition object

    CwWorkspaceDefinition wspcDef = new CwWorkspaceDefinition();

    // Set the workspace name, display name, and description
    // in the workspace definition object.
    //
    // Both name and display name must be unique.

    wspcDef.setName(workspaceName);
    wspcDef.setDisplayName(workspaceDisplayname);
    wspcDef.setDescription(workspaceDescription);

    // Use the default system template (set to "Basic Workspace Template"
    // upon installation), which includes
    // calendar (meetings & tasks),
    // discussion board and inbox

    String defaultTemplateName =
      session.getCwApplicationPropertyValue(
        CwPropertyConstants.DEFAULT_WORKSPACE_TEMPLATE_NAME);

    // Create a workspace template object

    CwWorkspaceTemplate template =
      session.getWorkspaceTemplate(defaultTemplateName);

    // Set the template in the workspace definition object

    wspcDef.setTemplate(template);

    // Create the workspace with the workspace definition object

    CwWorkspace newWspc = session.createWorkspace(wspcDef);

    return (newWspc);
  }
```

Example: Create Workspace

The following code excerpt creates a new workspace:

```
// mySession is an open session

CwWorkspaceDefinition wspcDef = new CwWorkspaceDefinition();

// Both the name and display name must be unique.

wspcDef.setName("Workspace name");
```

```
                    wspcDef.setDisplayName("Workspace display name");
                    wspcDef.setDescription("Worspace definition");

                    CwWorkspace newWorkspace =  mySession.createWorkspace(wspcDef);
```

Setting Workspace Properties

For a list of workspace properties, see Appendix A, "Workspace and Application Properties".

Managing Content in a Workspace

This section describes how to do the following:

- Creating Resources and Resource Items
- Setting Resource and Resource Item Properties
- Manage the following resources, resource items, and other content that may be found in a workspace:
    - Managing Members
    - Managing Calendar
    - Managing Discussion
    - Managing Email
    - Managing Files
    - Managing Views
    - Managing Announcements
    - Managing Attachments

Creating Resources and Resource Items

To create a resource or a resource item, instantiate the corresponding definition class. A *definition class* contains the set of properties that may be used to instantiate the resource or item. Required properties are included in the constructor of the definition class. Optional properties and additional information may be set by using methods on the definition class.

An instance of the definition class is only used at creation time. Once a resource or item is instantiated, the corresponding definition instance is never used again.

The following is an example of creating a calendar resource:

```
CwResourceDefinition calDefn =
  new CwResourceDefinition(
    "CalResource",
    "My calendar resource",
    CwEntityType.CALENDAR_RESOURCE);
CwCalendarResource calResource =
  (CwCalendarResource)workspace.createResource(calDefn);
```

The following is an example of creating an event:

```
CwEventDefinition meetingDefn = new CwEventDefinition(
  CwCalendarConstants.EVENT_TYPE_APPOINTMENT,
    "my meeting summary",
    startTime,
```

```
        endTime);
CwEvent meeting = calResource.createEvent(meetingDefn);
```

Resources are also created when a workspace is created with a template. Define the resources you would like to create in the workspace template. To get a resource object, call one of the CwWorkspace.getResource methods such as getResourceByType.

Setting Resource and Resource Item Properties

To view or make any changes to the properties, retrieve an instance of the corresponding properties class.

The following is an example of retrieving an instance of a calendar resource properties class:

```
CwCalendarResourceProperties calProps =
  (CwCalendarResourceProperties)calResource.getProperties();
```

The following is an example of retrieving an instance of an event Properties class:

```
CwEventProperties eventProps = (CwEventProperties)meeting.getProperties();
```

The methods on the Properties classes are very similar to those on the Definition classes. In addition, additional system generated properties, such as getCreatedTime and getUID, are available from the Properties class.

Properties are not immediately saved when they are set on a Properties instance. To save the changes on a Properties instance, call the storeProperties method. This method looks at all the changes made to the Properties instance since the last call, and saves all those changes to the repository. This enhances performance because the database is accessed only once during a single call to storeProperties, instead of every time a property is changed.

Managing Members

Call the CwWorkspace.createMemberable method to create a member (of class CwMemberable) from a CwUser object.

Example: Add Members to Existing Workspace

The following code excerpt adds two members (one who has a reader role and other who has a writer role) to an existing workspace:

```
// newWorkspace is a workspace that has already been created.
//
// currentSession is a session opened by a user with the
// administrator role in the workspace newWorkspace.

CwUser userReader1, userWriter1;

private static String userReader1Name = "Reader";
private static String userWriter1Name = "Writer";

userReader1 = currentSession.getUserByName(userReader1Name);
userWriter1 = currentSession.getUserByName(userWriter1Name);

CwMemberable[] membersToBeAdded = new CwMemberable[2];

membersToBeAdded[0]  =
  newWorkspace.createMemberable(
```

```
            userReader1,
            CwMemberRoleType.CW_MEMBER_ROLE_READER);

membersToBeAdded[1] =
  newWorkspace.createMemberable(
      userWriter1,
      CwMemberRoleType.CW_MEMBER_ROLE_WRITER);


// Now, add the members to the workspace
CwAddWorkspaceMembersResponse response =
  newWorkspace.addMembers(membersToBeAdded);

CwMember[] addedMembers      = response.getSuccessfulMembers();
CwMemberable[] failedMembers = response.getFailedMemberables();
Throwable[] failedExceptions = response.getFailedExceptions();
```

Managing Calendar

The CwCalendarResource class represents a calendar resource, and the CwEvent class represents an event resource item or a meeting.

Example: Creating Meeting in Existing Workspace

The following code excerpt creates one meeting with a duration of one hour that will occur one hour after the current date and time:

```
// The workspace myWorkspace has already been created. It
// has a calendar resource created in it.

long m_referenceTime = Calendar.getInstance().getTimeInMillis();

// Startime: Create a meeting that starts one hour
// after the current time

java.util.Calendar start = new GregorianCalendar();
start.setTimeInMillis(m_referenceTime);
start.set(Calendar.SECOND, 0);  // ignore the seconds
start.set(Calendar.MILLISECOND, 0);  // ignore the milliseconds
start.add(Calendar.DAY_OF_MONTH, 0);
start.add(Calendar.HOUR_OF_DAY, 1);

// Endtime: The meeting will be one hour long

java.util.Calendar end = new GregorianCalendar();
end.setTimeInMillis(m_referenceTime);
end.add(Calendar.DAY_OF_MONTH, 0);
end.add(Calendar.HOUR_OF_DAY, 2);
end.set(Calendar.SECOND, 0);  // ignore the seconds
end.set(Calendar.MILLISECOND, 0);  // ignore the milliseconds

// Create an event definition object

CwEventDefinition evtDef = new
  CwEventDefinition(
    CwCalendarConstants.EVENT_TYPE_APPOINTMENT,
    "New Meeting Title",
    start.getTime(),
    end.getTime());
```

```
// Set attending type: ALL GROUP MEETING (default)
// Note: Since this is an all group meeting, it will
// automatically invite all workspace members.
//
// To create a meeting with a subset of workspace members as
// attendees, set the attending type to
// CwCalendarConstants.ATTENDING_TYPE_SUBGROUP and use the
// CwEventDefinition.setAttendees method to set the list of attendees

evtDef.setAttendingType(CwCalendarConstants.ATTENDING_TYPE_ALL);

// Set event description
evtDef.setDescription("Create an event through OCW API");

// Set event priority
evtDef.setPriority(CwCalendarConstants.PRIORITY_NORMAL);

// Create a reminder
CwCalendarAlarm alm = new CwCalendarAlarm();
alm.setAction(CwCalendarAlarm.ACTION_EMAIL);
alm.setEmail("testUser@oracle.com");
alm.setMinutes(10);
evtDef.addAlarm(alm);

// Create event
//
// First, get the calendar resource
CwCalendarResource m_calResource =
  (CwCalendarResource) myWorkspace.
  getResourceByType(CwResourceType.CALENDAR_RESOURCE);
CwEvent event = m_calResource.createEvent(evtDef);
CwEventProperties resProps =
  (CwEventProperties)event.getProperties();

System.out.println("Done Creating Meeting:" + resProps.getSummary());
```

Managing Discussion

The CwDiscussionResource class represents a discussion resource; CwBoard, a
discussion board; and CwMessage, a message.

Example: Create Discussion Board, and Post New Message and Reply

The following code excerpt creates a discussion board in an existing workspace, and
posts a message and a reply to that message on the new board:

```
// The workspace myWorkspace has already been created. It
// has a discussion resource created in it.

// Retrieve discussion resource
CwDiscussionResource m_discResource =
 (CwDiscussionResource) myWorkspace.getResourceByType(
   CwResourceType.DISCUSSION_RESOURCE));

// Create a discussion board definition object
CwBoardDefinition maxBoardDef =
  new CwBoardDefinition(
    "New Board",
    "A test board",
    "board.email@my.company.com");
```

```
maxBoardDef.setDescription("Description of new board");
maxBoardDef.setSubjectReplyPrefix("Re: ");
maxBoardDef.setOriginalQuoted(true);
maxBoardDef.setBeginQuotePrefix("> ");

// Create a discussion board
CwBoard maxBoard = m_discResource.createBoard(maxBoardDef);

// Post a message on the new board
CwMessageDefinition msgDef = null;

msgDef = new CwMessageDefinition();
msgDef.setSubject("Subject of new message");
msgDef.setText("Text of new message");

// Create message
CwMessage createdMessage = maxBoard.createMessage(msgDef);

// Post a reply to the message

// Create message definition for the reply.
CwMessageDefinition msgDefReply = null;
msgDefReply = new CwMessageDefinition(createdMessage);

msgDefReply.setSubject("Subject of reply message");
msgDefReply.setText("Text of reply message");

// Create reply
CwMessage replyMessage = maxBoard.createMessage(msgDefReply);
```

Managing Email

The email resource, represented by the CwEmailResource class, uses a discussion board resource item to manage its messages. This discussion board is the workspace's inbox.

Therefore, in order to manage an email resource's messages, first retrieve the resource's discussion board by calling the CwEmailResource.getBoard method. Manage the email resource's messages by calling methods on the discussion board object.

Example: Create Email Resource

The following code excerpt creates an email resource in an existing worspace and retrieves its discussion board:

```
// myWorkspace is an existing workspace.
// emailDomain is a string representing
// the email domain name such as my-company.com.

// Create an email resource in myWorkspace
CwEmailResourceDefinition rsrcDef = new CwEmailResourceDefinition(
  "em_rsrc",
  "email resource description",
  "emailrsrc1" + emailDomain // emailrsrc1@my-company.com
);

CwEmailResource emailResource =
  (CwEmailResource)(myWorkspace.createResource(rsrcDef));
```

```
// Get the discussion board underlying the email resource
CwBoard rsrcBoard = emailResource.getBoard();


// Once the board object is obtained, all operations that can be performed
// on a board can be performed on the email resource
```

Managing Files

Call CwFilesResource.getWorkspaceFolder to retrieve the root folder of the workspace. Add new folders and files to this folder.

Example: Add File to Workspace

The following code excerpt adds a file to an existing workspace.

```
// myWorkspace is a workspace that has a files
// resource.


// Get the files resource from the workspace
CwFilesResource files = (CwFilesResource)
  myWorkspace.getResourceByType(CwResourceType.FILES_RESOURCE);


// Create the new file
CwFileDefinition docDef =
  new CwFileDefinition("My_New_Document.txt", "Contents of new file");

docDef.setDescription("Description of new file");


// Retrieve the workspace folder
CwFolder myWorkspaceFolder = files.getWorkspaceFolder();


// Add the document to the workspace folder
CwFile rfpDoc = myWorkspaceFolder.createFile(docDef);
```

Example: Create and Manage Folder

The following code excerpt performs the following operations:

- Retrieve the files resoruce from an existing workspace

- Create a folder in the root folder

- Retrieve and update the properties of a folder

- List contents of a folder

- Delete items in a folder

```
// myWorkspace is an exisitng workspace with
// a files resource


// Retrieve files resource
CwFilesResource m_filesRsrc =
  (CwFilesResource) myWorkspace.getResourceByType(CwResourceType.FILES_RESOURCE);


// Create a folder in the workspace
CwFolder wspcFolder = m_filesRsrc.getWorkspaceFolder();
CwFolderDefinition subfolderDef =
  new CwFolderDefinition("TEST_SUBFOLDER_NAME");
subfolderDef.setDescription("TEST_SUBFOLDER_DESC");
CwFolder subfolder = wspcFolder.createFolder(subfolderDef);
```

```
// Get the properties of a folder
CwFolderProperties props =
  (CwFolderProperties) subfolder.getProperties();
System.out.println("name: " + props.getName());
System.out.println("description: " + props.getDescription());

// Update properties of a folder
props.setName("props_test2");
props.setDescription("description2");
subfolder.storeProperties();

// List contents of a folder
CwFilesResourceItem[] contents = wspcFolder.listContents();

// fldr1, fldr2, and file1 two folders and a file
// that have been previously initialized and added to the root folder.

// Delete items in a folder
CwUid[] uids = new CwUid[] {fldr1.getUid(), fldr2.getUid(), file1.getUid()};
wspcFolder.deleteItems(uids);
```

Managing Announcements

The class CwAnnouncementList manages announcements. Announcements are of type CwAnnouncement. However, CwAnnouncementList is not a resource; it is managed directly by the CwWorkspace class.

Example: Create Announcement

The following code excerpt creates an announcement in an existing workspace with an expiration date of one week from today's date:

```
// workspaceUid is a UID of an exisitng workspace

CwWorkspace myWorkspace = mySession.getWorkspaceByUid(workspaceUid);
CwAnnouncementList anncList = myWorkspace.getAnnouncementList();

// Create an announcement

// Create the expiration date of the announcement:
// one week from today
long m_referenceTime = java.util.Calendar.getInstance().getTimeInMillis();
java.util.Calendar myCalendar = new java.util.GregorianCalendar();
myCalendar.setTimeInMillis(m_referenceTime);
myCalendar.add(java.util.Calendar.DAY_OF_YEAR, 7);
java.util.Date oneWeekLater = myCalendar.getTime();

// Create an announcement definition object
CwAnnouncementDefinition anncDef1 = new CwAnnouncementDefinition(
  "announcement test1",
  "Welcome user wspcUser1 to the workspace.",
  "text/plain",
  oneWeekLater);

// Create the announcement on the announcement list
// and retrieve the UID of the announcement
CwUid anncUid1 = anncList.createAnnouncement(anncDef1).getUid();
```

Example: Manage Announcements

The following code excerpt performs several operations on announcements:

```
// anncUid1, anncUid2, anncUid3, and anncUid4
// are UIDs of announcements.
// anncList is an announcement list of a workspace.

// List an announcement by its UID
CwAnnouncement annc1Got = (CwAnnouncement)anncList.getItemByUid(anncUid1);

// List unexpired announcements in the workspace
CwAnnouncement anncs[] = anncList.listUnexpiredAnnouncements();

// Update an announcement
String newDescription = "This Friday team lunch is rescheduled to Monday.";
CwAnnouncement annc2Got = (CwAnnouncement)anncList.getItemByUid(anncUid2);
annc2Got.getProperties().setDescription(newDescription);
annc2Got.storeProperties();

// Delete an announcement
anncList.deleteAnnouncement(anncUid2);

// Delete multiple announcements
anncList.deleteAnnouncements(new CwUid[]{anncUid3, anncUid4});
```

Managing Attachments

An *attachment* (which is called a link in the Oracle Workspace Web client) is a mechanism that associates resource items with each other. Because the CwResourceItem class implements the CwAttachmentEnabled interface, all resource items have the ability to link to other resource items.

Use the createAttachment method to link resource items. Once an attachment is created, a uni-directional link is formed with metadata retained to remember which resource item is the source and which is the destination. The CwAttachmentEnabled.listAttachments method finds all items that the current item points to. The CwAttachmentEnabled.listSourceAttachments method finds all items that point to the current item.

The CwAttachedEntity class represents an attachment. CwAttachmentDefinition represents a resource item to be attached. No resource class exists for attachments; they are directly managed by workspaces. Call CwWorkspaces.addAttachments to create an attachment in a workspace.

Example: Manage Attachments

The following code excerpt performs the following operations:

- Attach an item to another item
- Attach multiple attachments
- List all attachments
- List attachments by type
- Update descriptions of specified attachments
- Remove specified attachments

```
// workspaceUid is a UID of an existing workspace
```

```
CwWorkspace myWorkspace =
  mySession.getWorkspaceByUid(workspaceUid);

// Attach a single item to a given item

// childEntityUid1 is a UID of resource item in the current
// workspace that will be attached to parentEntityUid, a UID
// of another resource item in the workspace.

CwAttachmentDefinition attchDef1 = new CwAttachmentDefinition(
  childEntityUid1,
  "Attachment description");
myWorkspace.addAttachments(parentEntityUid, attchDef1);

// Add multiple attachments
CwAttachmentDefinition attchDef2 = new CwAttachmentDefinition(
  new CwUid[] {childEntityUid2,childEntityUid3},
  "Multiple attachments with the same description");

myWorkspace.addAttachments(parentEntityUid, attchDef2);

// List all attachments

CwAttachedEntity[] myAttachments =
  myWorkspace.listAttachments( parentEntityUid );

System.out.println( "listing attachment uids:" );

for (int i = 0; i < myAttachments.length; i++) {
  System.out.println( "\tattachment # " + i );
  System.out.println( "\t\tattachment uid = " +
    myAttachments[i].getEntityUid().toString() );
  System.out.println( "\t\tattachment description = " +
    myAttachments[i].getDescription() );
  System.out.println( "\t\tattachment role = " +
    myAttachments[i].getEntityRole() );
}

// Listing attachments by type
Map myAttachmentsMap = myWorkspace.listAttachmentsByType( parentEntityUid );
Set myAttachmentsSet = myAttachmentsMap.entrySet();
Object[] myAttachmentsArray = myAttachmentsSet.toArray();

for (int i = 0; i < myAttachmentsArray.length; i++) {
  System.out.println( "\tattachment type = " +
    ((CwEntityType)(((Map.Entry)myAttachmentsArray[i]).getKey()))).getName() );
  CwAttachedEntity[] myAttachmentsSameType =
((CwAttachedEntity[])(((Map.Entry)myAttachmentsArray[i]).getValue()));

  for (int j = 0; j < myAttachmentsSameType.length; j++) {
    System.out.println( "\t\tattachment # " + j );
    System.out.println( "\t\t\tattachment uid = " +
      myAttachmentsSameType[j].getEntityUid().toString() );
    System.out.println( "\t\t\tattachment description = " +
      myAttachmentsSameType[j].getDescription() );
    System.out.println( "\t\t\tattachment role = " +
      myAttachmentsSameType[j].getEntityRole() );
  }
}
```

```
                  // Update attachment descriptions
                  CwAttachedEntity[] attachments =
                    myWorkspace.listAttachments( parentEntityUid );

                  // Update entities 1 and 2
                  attachments[0].setDescription(
                    attachments[0].getDescription() + " - Updated");
                  attachments[1].setDescription(
                    attachments[3].getDescription() + " - Updated");

                  myWorkspace.updateAttachments( parentEntityUid, attachments);

                  // Remove attachments
                  attachments = myWorkspace.listAttachments( parentEntityUid );

                  // Remove entities 1 and 2
                  CwAttachedEntity[] toBeRemovedEntities = new CwAttachedEntity[2];
                  toBeRemovedEntities[0] = attachments[0];
                  toBeRemovedEntities[1] = attachments[3];

                  myWorkspace.removeAttachments( parentEntityUid, toBeRemovedEntities);
```

Managing Views

The CwViewResource class represents a View resource, and the CwView class represents a view.

Example: Manage Views

The following excerpt performs the following operations:

- Create a new view in an existing workspace

- Update properties of that new view

- Retrieve a view by UID

- List the views of a workspace

- List the items in a view

- Delete views

- Delete a view resource

```
// myWorkspace is an exisitng workspace

// Add the view resource to an existing workspace
CwResourceDefinition resDef = new CwResourceDefinition(
  "Views",
  "View resource",
  CwResourceType.VIEW_RESOURCE );

CwViewResource viewRes =
  (CwViewResource)myWorkspace.createResource( resDef );

// Create a new view
CwViewDefinition viewDef = new CwViewDefinition(
  "TestView",
  "TestView description" );

CwView newView = viewRes.createView( viewDef );
```

```
// Retrieve and update the properties of the view
CwResourceItemProperties viewProps = newView.getProperties();
viewProps.setName( "updatedTestView" );
viewProps.setDescription( "updated TestView description" );
newView.storeProperties();

// Retrieve a view by UID
// (this example retrieves the newly created view)
CwView fetchedView = (CwView)viewRes.getItemByUid( newView.getUid() );

// List views in the workspace
CwView[] views = viewRes.listViews();

// List contents of view (more specifically, the items in the view)
CwUid[] items = newView.listItems();

// Delete views
CwUid viewUids[] = new CwUid[views.length];
for (int i = 0; i < views.length; i++) {
  viewUids[i] = views[i].getUid();
}
viewRes.deleteViews( viewUids );

// Delete the resource from the workspace
// (this deletes all views in the workspace)
myWorkspace.deleteResource( viewRes.getUid() );
```

4

Developing Workspace Templates with XML

Templates are XML documents that are used to create new workspaces with some predefined settings and content. These settings and content include workspace properties, library structure, library content, discussion forums, tasks, views, and members.

Templates can be created by using any standard XML editing tool or the Java SDK. An XSD for the workspace template is provided.

Templates can also be generated from an existing workspace by a member who is an administrator of that workspace. These templates can be further edited for more granular control of the template.

Templates are stored in the Oracle Workspaces application by an Oracle Workspaces application administrator. The template can be stored either through the Oracle Workspaces Web client or the Java SDK.

This chapter describes the following topics related to developing workspace templates:

- Generating Templates from Existing Workspaces
- Creating Templates
- Sample Template XML File
- Specifying File Locations in Template
- Storing Templates

Generating Templates from Existing Workspaces

You must have the role of administrator in the workspace from which you want to generate a template. You may generate a template by one of the following ways:

- Oracle Workspaces Web client: Click the name of the workspace > Overview > Settings button > Generate Template button
- Java SDK: Call the CwSession.getWorkspaceTemplate method.

Creating Templates

Templates can be created using a standard XML editing tool or using the Java SDK. An XSD for the Template is provided.

Retrieve the template XML schema definition (XSD) file by using one of the following options as an application administrator:

- Oracle Workspaces Web client: From the Administration > Templates tab, click the button View Template Schema.

- Java SDK: Call the CwSession.getCwTemplatesSchema method

Use any XML tool to create a new template based on this XSD file.

Sample Template XML File

The following is a sample template that will create a workspace with the following entities and properties:

- The new workspace will have the following properties:

  - Membership Notification = All members

  - Member Access = Enabled

  - Default Member Role = Reader

  - Workspace Listing = False

- Calendar resource with two predefined tasks

- Files resource with several predefined folder and file names

- Discussion resource with two predefined discussion boards

- Email resource

- Empty member list (the member list will contain only the workspace creator)

```
<workspaceTemplate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.oracle.com/cw/wstemplate"
  xsi:schemaLocation=
"http://www.oracle.com/cw/wstemplate http://www.oracle.com/cw/wstemplate.xsd">
  <templateProperties>
    <name>New Product Launch Marketing Template</name>
    <description>
Workspace template for coordinating marketing activities for launching a new
product
    </description>
    <category>system.marketing</category>
    <createdBy>System</createdBy>
    <createdOn>2004-10-01T18:43:10.000000</createdOn>
  </templateProperties>
  <workspaceDetails>
    <properties>
      <MEMBERSHIP_NOTIFICATION>ALL_MEMBERS</MEMBERSHIP_NOTIFICATION>
      <MEMBER_ACCESS>ENABLED</MEMBER_ACCESS>
      <MEMBER_DEFAULT_ROLE>READER</MEMBER_DEFAULT_ROLE>
      <PUBLICLY_LISTED>false</PUBLICLY_LISTED>
    </properties>
    <resources>
      <resource xsi:type="CalendarType">
        <name>Meetings</name>
        <description>Tracks project meetings and tasks</description>
        <vtodos>
          <vtodo id="cust_ref_task">
            <summary>Prepare Reference Customer List</summary>
            <description>Prepare a list of reference customers</description>
            <priority>5</priority>
          </vtodo>
          <vtodo id="comp_anal_task">
```

```
                            <summary>Complete competitive analysis</summary>
                            <description>
Gather information about features supported by the competitors
and perform competitive analysis
                            </description>
                            <priority>5</priority>
                        </vtodo>
                    </vtodos>
                </resource>
                <resource xsi:type="FilesType">
                    <name>Library</name>
                    <description>Workspace document library</description>
                    <libraryDetails>
                        <folders>
                            <folder>
                                <name>Advertising</name>
                                <description>
Folder for information related to advertising campaigns
                                </description>
                                <folderDetails>
                                    <files>
                                        <file>
                                            <name>AnalystDayPresentation.ppt</name>
                                            <description>Presentation for analyst day</description>
                                        </file>
                                        <file>
                                            <name>Budget.xls</name>
                                            <description>Budget for advertising campaigns</description>
                                        </file>
                                        <file>
                                            <name>Options.xls</name>
                                            <description>Advertising options</description>
                                        </file>
                                        <file>
                                            <name>RoadShows.xls</name>
                                            <description>Road show details</description>
                                        </file>
                                    </files>
                                </folderDetails>
                            </folder>
                            <folder>
                                <name>Customers</name>
                                <description>
Folder for storing information related to customers
                                </description>
                                <folderDetails>
                                    <files>
                                        <file id="2042#542919">
                                            <name>CustomerFAQ.doc</name>
                                            <description>
Answers for frequently asked questions
                                            </description>
                                        </file>
                                        <file id="2042#542933">
                                            <name>CustomerFeedback.xls</name>
                                            <description>
Customer feedback on the product
                                            </description>
                                        </file>
                                        <file id="2042#542924">
```

```
                          <name>CustomerReferences.xls</name>
                          <description>List of reference customers</description>
                        </file>
                        <file id="2042#543125">
                          <name>Issues.xls</name>
                          <description>For tracking customer issues</description>
                        </file>
                      </files>
                  </folderDetails>
                </folder>
                <folder>
                  <name>Market Intelligence</name>
                  <description>
Folder for storing market intelligence information
                  </description>
                  <folderDetails>
                    <files>
                      <file>
                        <name>CompetitorFeatures.doc</name>
                        <description>
Summary of features supported by product competitors
                        </description>
                      </file>
                      <file>
                        <name>MarketOverview.doc</name>
                        <description>Market share overview</description>
                      </file>
                    </files>
                  </folderDetails>
                </folder>
                <folder>
                  <name>Partners</name>
                  <description>
Folder for information about internal and external partners
                  </description>
                  <folderDetails>
                    <files>
                      <file id="2042#543071">
                        <name>PartnersList.xls</name>
                        <description>List of product partners</description>
                      </file>
                    </files>
                  </folderDetails>
                </folder>
                <folder>
                  <name>Pricing</name>
                  <description>
Folder for information needed for analyzing pricing options
                  </description>
                  <folderDetails>
                    <files>
                      <file id="2042#543025">
                        <name>PricingOptions.xls</name>
                        <description>List of pricing options</description>
                      </file>
                    </files>
                  </folderDetails>
                </folder>
                <folder>
                  <name>Product Collaterals</name>
```

```
                     <description>
Folder for storing product collaterals
                     </description>
                     <folderDetails>
                       <files>
                         <file id="2042#542833">
                           <name>ProductDataSheet.doc</name>
                           <description>Data sheet for the product</description>
                         </file>
                       </files>
                       <folders>
                         <folder id="2042#542839">
                           <name>White Papers</name>
                           <description>
Folder for storing product white papers
                           </description>
                         </folder>
                       </folders>
                     </folderDetails>
                   </folder>
                 </folders>
               </libraryDetails>
            </resource>
            <resource xsi:type="DiscussionsType">
              <name>TD</name>
              <description>Workspace discussion forums</description>
              <boards>
                <board id="2044#TD_B%9944">
                  <name>Advertising Campaign Ideas</name>
                  <description>
A board for discussing new ideas for advertising campaigns
                  </description>
                </board>
                <board>
                  <name>Marketing Strategies</name>
                  <description>
A board for discussing new marketing strategies
                  </description>
                </board>
              </boards>
            </resource>
            <resource xsi:type="ViewsType">
              <name>Views</name>
              <description>Logical Grouping of workspace contents</description>
              <views>
                <view>
                  <name>Customer reference program</name>
                  <description>
A view for information related to the customer reference program
                  </description>
                  <viewItems>
                    <itemRef refid="cust_ref_task"/>
                    <itemRef refid="2042#542839"/>
                    <itemRef refid="2042#542919"/>
                    <itemRef refid="2042#542933"/>
                    <itemRef refid="2042#542924"/>
                    <itemRef refid="2042#543125"/>
                    <itemRef refid="2042#543025"/>
                    <itemRef refid="2042#542833"/>
                  </viewItems>
```

```
            </view>
            <view>
              <name>Sales Support</name>
              <description>
A view for information needed by the sales team
              </description>
              <viewItems>
                <itemRef refid="comp_anal_task"/>
                <itemRef refid="2044#TD_B%9944"/>
                <itemRef refid="2042#542839"/>
                <itemRef refid="2042#543071"/>
                <itemRef refid="2042#543025"/>
                <itemRef refid="2042#542919"/>
                <itemRef refid="2042#542933"/>
                <itemRef refid="2042#542924"/>
                <itemRef refid="2042#542833"/>
              </viewItems>
            </view>
          </views>
        </resource>
        <resource xsi:type="EmailType">
          <name>Inbox</name>
          <description>Project email inbox</description>
        </resource>
      </resources>
    </workspaceDetails>
</workspaceTemplate>
```

Specifying File Locations in Template

Suppose you have a template file that has several files defined in it. If you create a workspace with that template, it will create empty files with names as specified in the template. If you want the template to copy a file instead of creating an empty one, specify the following in the template file:

1. In a <contentLocation> element, specify the full Oracle Content Services path name of the file to be copied. The user creating a new workspace using this template must have read access to this file in Oracle Content Services.

2. In a <contentID> element, specify an ID for the file to be copied. This ID is a string that is unique to the template file.

3. Put the <contentLocation> and <contentID> elements in a <contentLocation> element. Put all <contentLocation> elements in a <contentLocations> element.

4. In the file definition element (the <file> element) add a <content> element. In this <content> element, add the corresponding <contentID> of the file you would like copied.

The following XML extract copies the contents of `/workspaces/files/AnalystDayPresentation.ppt` to `Advertising/AnalystDayPresentation.ppt` in the new workspace:

```
<templateProperties>
  <!-- Template properties -->
</templateProperties>

<contentLocations>
  <contentLocation>
    <contentID>0</contentID>
    <location>/workspaces/files/AnalystDayPresentation.ppt</location>
```

```
          </contentLocation>
      </contentLocations>

      <workspaceDetails>
        <properties><!-- Workspace properties --></properties>
        <resources>
          <resource xsi:type="FilesType">
            <name>Library</name>
            <description>Workspace document library</description>
            <libraryDetails>
              <folders>
                <folder>
                  <name>Advertising</name>
                  <description>
                    Folder for information related to advertising campaigns
                  </description>
                  <folderDetails>
                    <files>
                      <file>
                        <name>AnalystDayPresentation.ppt</name>
                        <content>
                          <contentID>0</contentID>
                        </content>
                        <description>Presentation for analyst day</description>
                      </file>
                    </files>
                  </folderDetails>
                </folder>
              </folders>
            </libraryDetails>
          </resource>
        </resources>
      </workspaceDetails>
```

Storing Templates

Templates are stored in the Oracle Workspaces application by an Oracle Workspaces application administrator. They can be stored either through the Oracle Workspaces Web client or the Java SDK.

- Oracle Workspaces Web client: Click Administration Tab > Templates tab > Store New Template button. You can cut-and-paste the template XML from here.

- Java SDK: Call the CwSession.storeWorkspaceTemplate method, passing a string representing the template content

5

Oracle Workspaces URI Access

The Oracle Workspaces Web client uses an MVC (Model-View-Controller) design pattern for displaying its pages. This chapter describes the URLs that are exposed and the unique and persistent parameters used to access specific pages in the UI.

Oracle Workspace URLs

This section describes end-user Oracle Workspaces URLs, including the following views and functions in Oracle Workspaces:

- Overview Page

- Browse Library, Meetings, Tasks, Discussions, Announcements, Views, Members

- Create New Meeting, New Task, Announcement, View

Using these URLs require that a developer use the Java SDK to access workspace UIDs and IDs of objects (resources) within a workspace. The CwUID class in the Java SDK can represent the UID of an object in a workspace as a string with following format:

`<workspaceId>#<resourceId>#<resourceItemId>`

> **Note:** Once you have retrieved a CwUID string with the Java API, you must escape some of its characters before using it in an Oracle Workspace URL. See "Creating Oracle Workspace URLs" for more information about retrieving CwUid strings with the Java API and using them in these URLs.

In the following table, <fileUID>, <folderUID>, <meetingUID>, <taskUID>, <announcementUID>, <forumUID>, <threadUID>, and <viewUID> are the string representations of the CwUid of the corresponding object. See the CwUid class in *Oracle Workspaces Java API Reference* for methods to retrieve this information.

Table 5–1   Oracle Workspace URLs

UI View	URI <host:port/workspaces/URI>	Parameter Formats
My Workspaces	/cwHomeAction.do?event=<event>	<event> ■ goHome ■ goHomeAll
Workspace Overview Page	/workspaceAction.do? event=goHome& workspaceUid=<workspaceUID>	
Workspace Properties	/workspaceAction.do? event=showProperties &workspaceUid=<workspaceUID>	
Workspace Services	/workspaceAction.do? event=showResources &workspaceUid=<workspaceUID>	
Library Overview	/filesAction.do?event=filesFolderList &workspaceUid=<workspaceUID>	
Folder	/filesAction.do? event=filesFolderList &uid=<folderUID>	
File Properties	/filesAction.do?event=filesFolderPropsForm &uid=<fileUID>	
File Links	/filesAction.do? event=filesFolderPropsFormLinks &uid=<fileUID>	
Meetings Overview	/calendarAction.do? event=meetingsView &workspaceUid=<workspaceUID> &date=<date> &viewType=<viewType> &meetingsScope=<meetingScope>	<date>: Format yyyy/mm/dd <viewType>: ■ weekView ■ weeklyList ■ dayView ■ dailyList <meetingScope>: ■ myMeetings ■ allMeetings
Create New Meeting	/calendarAction.do? event=meetingCreateForm &workspaceUid=<workspaceUID> &date=<date> &timeHour=<timeHour>	<date>: Format yyyy/mm/dd <timeHour>: Format hh
Meeting Properties	/calendarAction.do? event=meetingModifyForm &workspaceUid=<workspaceUID> &eventUid=<meetingUID>	

Table 5–1 (Cont.) Oracle Workspace URLs

UI View	URI <host:port/workspaces/URI>	Parameter Formats
Tasks Overview	/calendarAction.do? event=tasksView &workspaceUid=<workspaceUID> &taskType=<taskType> &tasksScope=<taskScope> &event=sort &source=tasksViewTable &value=<value> &state=	<taskType>: ■ active ■ complete ■ all ■ incomplete <taskScope>: ■ myTasks ■ allTasks <taskScope>: ■ myTasks ■ allTasks <value>: ■ PRIORITY ■ TITLE ■ PERCENT_COMPLETE ■ START_DATE ■ DUE_DATE ■ TASK_PRIMARY_ ASSIGNEE ■ TASK_ASSIGNED_BY ■ allTasks
New Task	/calendarAction.do? event=taskCreateFormNew &workspaceUid=<workspaceUID> &date=<date>	<date>: Format yyyy/mm/dd
Edit Task	/calendarAction.do? event=taskModifyForm &workspaceUid=<workspaceUID> &taskUid=<taskUID>	
Announcements	/workspaceAction.do? event=showAnnouncements &workspaceUid=<workspaceUID>	
New Announcement	/workspaceAction.do? event=addAnnouncementForm &workspaceUid=<workspaceUID>	
View Announcement	/workspaceAction.do? event=viewAnnouncement &uid=<announcementUID>	
Discussions	/discussionAction.do? event=goHome &workspaceUid=<workspaceUID>	
A Forum	/discussionAction.do? event=oracle.discussions.ui.event.board.browse &oracle.discussions.ui.fid=<forumUID>	

Table 5–1   (Cont.) Oracle Workspace URLs

UI View	URI <host:port/workspaces/URI>	Parameter Formats
Browse Topic	/discussionAction.do? event=oracle.discussions.ui.event.thread.browse_message <messageView> &oracle.discussions.ui.fid=<forumUID> &oracle.discussions.ui.tid=<threadUID>	
Inbox	/emailAction.do? event=goHome &workspaceUid=<workspaceUID>	
Views	/viewAction.do? event=eventViewListViews& workspaceUid=<workspaceUID>	
New View	/viewAction.do?event=eventViewCreateViewForm &workspaceUid=<workspaceUID>	
A View	/viewAction.do? event=eventViewGetContents &uid=<viewUID>	
Members	/workspaceAction.do? event=showPeople& workspaceUid=<workspaceUID>	
Add Members	/workspaceAction.do? event=addMembersForm &workspaceUid=<workspaceUID>	
Logout	/cwHomeAction.do?event=logout	

Creating Oracle Workspace URLs

Once you have retrieved a CwUID string with the Java API, you must escape some of its characters before using it in an Oracle Workspace URL. For example, you must escape the # character to %23 before using it in a URL.

For example, suppose that the CwUID for a Library Folder is the following string:

```
FFA70A6283EC980BE030018A19B8665A#9#2613#7775486
```

In a URL, this CwUID would be represented as follows:

```
http://<host>/workspaces/filesAction.do?
  event=filesFolderList
  &uid=FFA70A6283EC980BE030018A19B8665A%239%232613%237775486
```

The following Java method getWorkspaceFolder takes a Workspace as an argument and outputs to System.out the URLs to access the Workspace's Library overview and its Folders:

```
private void getWorkspaceFolder(CwWorkspace workspace)
  throws CwException
{
  CwFolder wFolder = getFilesResource(workspace).getWorkspaceFolder();

  if (wFolder != null)
  {
    CwFilesResourceItem[] contents = wFolder.listContents();

    String libraryUrl =
       "/workspaces/filesAction.do?event=filesFolderList&workspaceUid=" +
```

```
            escapeURL(workspace.getUid().toString());

      System.out.println("Library URL: " + libraryURL);

      for (int i=0; i<contents.length; i++)
      {
        CwFilesResourceItem item = contents[i];
        CwResourceItemProperties props = item.getProperties();
        CwUid itemId = props.getUid();

        boolean isFolder = (itemId.getEntityType() == CwEntityType.FILES_FOLDER);

        if (isFolder)
        {
          String itemName = props.getName();
          System.out.println("Folder Name: "+itemName);

          String folderUrl =
            "/workspaces/filesAction.do" +
            "?event=filesFolderList" +
            "&uid=" +
            escapeURL(item.getUid().toString());

          System.out.println("Folder URL: "+folderUrl);
        }
      }
    }
  }
}

private String escapeURL(String url)
{
  if (url == null)
    return null;

  url = url.replaceAll("\\x25", "%25"); // replaces %
  url = url.replaceAll("\\x23", "%23"); // replaces #
  url = url.replaceAll("\\x2B", "%2B"); // replaces +
  url = url.replaceAll("\\x2F", "%2F"); // replaces /

  return url;
}
```

Oracle Workspaces Application Administration URLs

This section describes application administration URLs, which includes the following functions:

- Oracle Workspaces Application Management (Properties, Roles, List of Workspaces that Require Recovery, Re-Ownership)

- Template Management

- Proxy as Admin to any Workspace

Table 5–2   Oracle Workspaces Application Administration URLs

UI View	URI <host:port/workspaces/URI>	Parameter Formats
Application Properties	/cwHomeAction.do?event=appPropsList	
Application Role Management	/cwHomeAction.do?event=appRolesHome	
Workspaces that Need Recovery	/cwHomeAction.do?event=showRecoveryInfo	
Unowned Workspaces	/cwHomeAction.do?event=showOrphanedWorkspaces	
All Workspaces	/cwHomeAction.do?event=listAllWorkspaces	
Access Workspace as Application Administrator	/workspaceAction.do?event=adminEnterWorkspace&sysAdminMode=y&workspaceUid=	
Templates	/templatesAction.do?event=goHome	
View a Template	/templatesAction.do?event=previewTemplate&explorerMode=0&wsCreationWzMode=FALSE&templateID=<Template_Name>	
View XML of a Template	/templatesAction.do?event=downloadTemplate&templateID=<Template_Name>	
View XSD	/templatesAction.do?event=getTemplateSchema	
Upload New Template	/templatesAction.do?event=uploadTemplateForm	
System Services (Resources)	/cwHomeAction.do?event=showResourceTypeStatus	

A

Workspace and Application Properties

The following tables describe the following workspace and application properties:

- Workspace Properties:
    - User Supplied Properties
    - Workspace Configuration Properties
    - System Generated Properties
    - Resource Specific Properties
    - Resource Specific System Generated Properties
- Application Properties

User Supplied Properties

These properties are set by the user.

Table A–1   User Supplied Properties

Property Name	Description	Changed By	Stored in Template?	Default Value
Description	Contains whatever text the user desires to describe the functionality of the workspace	Workspace Administrator	No	None
Display Name	This name shows at the top of the workspace home page. It can be changed at any time	Workspace Administrator	No	None
Name	This value is set by the user at creation time and cannot be changed. It is used to identify the workspace and is used for naming workspace-mapped containers in resources (for example, Content Services shared folder).	None	No	None

Workspace Configuration Properties

These properties define certain aspects of workspace behavior, and have default values for easy configuration.

Table A–2   Workspace Configuration Properties

Property Name	Description	Changed By	Stored in Template?	Default Value
Contact_email_ address	Email address for issues regarding this workspace, such as request for membership	Workspace Administrator	No	None
Contact_person_ name	The contact person for this workspace	Workspace Administrator	No	Workspace owner
member_access	If it is set to disabled, only workspace administrators can access the workspace. Possible values are ENABLED and DISABLED.	Workspace Administrator	Yes	DISABLED
Member_default_role	Role applied to new members if none is specified	Workspace Administrator	Yes	Reader
Membership_ notification	Determines which members are notified by email about membership changes, such as adding or dropping a member. Possible values are NONE, ALL_MEMBERS, and USER (the affected user)	Workspace administrator	Yes	NONE
Owner	The owner of the workspace	Workspace Administrator	No	Creator of workspace
publicly_listed	If true, non-members, in searches, can see the workspace's name, description, and email address of the contact person. In order for non-members to access this workspace, they must request membership from the contact person of the workspace.	Workspace Administrator	false	false
read_only	When true, restricts access to the workspace content to read-only	Workspace Administrator	No	false
welcome_message	Welcome messages, such as instructions for where to start or tips for finding information	Workspace Administrator	No	NULL

System Generated Properties

These properties are defined and maintained by Oracle Workspaces.

Table A–3   System Generated Properties

Property Name	Description	Changed By	Stored in Template?	Default Value
Creation time	The date on which the workspace was created	None	No	Not applicable
Last modified time	The last time that a modification was made to the workspace	None	No	Not applicable
Parent UID	The unique identifier of the parent workspace, or CwWorkspaceProperties.UID_NO_ PARENT if this is a root workspace	None	No	CwWorkspaceP roperties.UID_ NO_PARENT

Table A–3   (Cont.) System Generated Properties

Property Name	Description	Changed By	Stored in Template?	Default Value
Path	The path name, including the names of ancestor workspaces. For example, /wspc1/wspc2/wspc3.	None	No	None
UID	The unique identifier for this workspace	None	No	None
Workspace template	The template used in creating this workspace	None	No	Template used

Resource Specific Properties

These properties provide detail and customization options for the underlying resources.

Table A–4   Resource Specific Properties

Property Name	Description	Changed By	Stored in Template?	Default Value
Description	Contains the text the user wishes to describe the functionality of the resource	Workspace administrator	Yes	None
Name	This value is set by the user at creation time and cannot be changed. It is used to identify the resource	Workspace administrator	Yes	Resource type-specific

Resource Specific System Generated Properties

These properties are maintained by Oracle Workspaces.

Table A–5   Resource Specific System Generated Properties

Property Name	Description	Changed By	Stored in Template?	Default Value
Creation Time	The date on which the resource was create.	None	No	Current time
Creator	The creator of the resource	None	No	None
Last Modified Time	The last time the resource properties were changed	None	No	None
Resource Type	The type of resource (such as files or calendar)	None	Yes	None
UID	The unique identifier for this resource	None	No	None

Application Properties

These properties can be retrieved and set with the methods CwSession.getCwApplicationPropertyValue and CwSession.setCwApplicationPropertyValue.

All property names are constants defined in the class CwPropertyConstants.

All application properties can be changed only by the application administrator. However, any user can look up any application property value.

Table A–6   Application Properties

Property Name	Description	Default Value
DEFAULT_WORKSPACE_ TEMPLATE_NAME	The default template to use for new workspaces	Basic Workspace Template
WORKSPACE_CREATION_MODE	Determines if the WORKSPACE_ CREATOR_ROLE is needed for workspace creation. If the mode is WORKSPACE_ CREATION_MODE_PUBLIC, any user can create a workspace If it is WORKSPACE_CREATION_ MODE_RESTRICTED, only users with WORKSPACE_CREATOR_ ROLE can create workspaces.	WORKSPACE_CREATION_MODE_ PUBLIC

Index