

Oracle Content Services 10g
Web Services
Technical Paper

An Oracle White Paper
September 2005
Author: Matt Shannon

Disclaimer.....	4
Overview	4
Introduction.....	4
Web Services and Standards	4
Core Concepts	6
Everything is an Item.....	6
Service Managers.....	7
ArchiveManager	7
CategoryManager.....	7
CommonManager	8
ContainerManager.....	8
DomainManager.....	8
FileManager.....	8
GroupManager	9
LockManager	10
PagingManager	10
ProgressManager	10
QuotaManager	10
RecordsManager.....	10
RemoteLoginManager	12
RequestManager	12
SearchManager.....	13
SecurityManager	13
ServiceToServiceLoginManager.....	15
SessionManager	15
SortManager.....	16
SystemConfigurationManager	17
TrashManager	17
UserManager.....	17
VersionManager	17
VirusManager.....	18
WorkflowManager	19
WorkspaceManager.....	20
Client Session	22
Content Upload / Download	22
Attribute Chaining	24
Troubleshooting.....	25
Capturing SOAP Packets	25
Node Log Level.....	25
Consuming Web Services.....	27
Java Stubs.....	27
Code Examples.....	28
CreateFolderHierarchy.java	28
TestCategories.java and TestCategories2.java	28
TestFileOperations.java	28
TestRecordManagement.java	28

TestS2S.java	29
TestSearch.java and TestSearch2.java	29
TestSecurity.java.....	29
TestVersioning.java and TestVersioning2.java.....	29
Conclusion.....	30
Appendix A	31
Appendix B	40

Oracle Content Services 10g Web Services

DISCLAIMER

The purpose of this paper is not to teach the reader Web Services standards and concepts; there are many good books and Internet sites devoted to this. In fact, it is strongly encouraged that the reader be well versed in these concepts, as only a very brief overview of these technologies is given in this paper.

OVERVIEW

The Oracle Content Services 10g Web Services provide a means for third-party applications to leverage advanced document and record management capabilities in their own application. The Web Services can be used to simply integrate applications, or to build complete custom applications using the Oracle Content Services 10g framework, in a language and platform-independent way using industry-standard mechanisms. This technical whitepaper provides the reader with the foundations, techniques, and references needed to utilize the Oracle Content Services Web Services.

INTRODUCTION

Oracle Collaboration Suite Release 2 Oracle Files customers have often requested the ability to customize certain aspects of the application, or to provide a supported means to allow integration with other applications in the enterprise. Some Oracle Files customers who were familiar with the product's underlying technology, namely the Oracle Content Management SDK (Oracle CM SDK), wanted supported access to Oracle CM SDK's raw but extremely capable and flexible Java API. Likewise, many Oracle CM SDK customers wanted functionality that was only available in Oracle Files. Fortunately, Oracle Content Services provides the best of both worlds: a feature-rich Web user interface, a powerful Windows-based desktop client, multi-site capabilities, and support for advanced security models, combined with Web Services for integration and customizations.

The depth and completeness you can expect from the Oracle Content Services Web Services is illustrated by the fact that the Web Services use a remotable form of the same façade API that the Web user interface leverages for repository access. Nearly all repository API functionality available to the Web user interface is present and exposed in the Web Services. Creating and managing content (such as documents, records, and folders), assigning metadata to categorize content, searching content, and triggering workflows are just a few of the operations available through Web Services.

WEB SERVICES AND STANDARDS

Web Services are self-describing software components designed to expose functionality of a source application or system in a platform and language-independent way. The self-describing functionality is realized through Web Services Description Language (WSDL), an XML-based enabling technology used to describe the operations, signatures, and communication interfaces supported by the Web Service. Once a client has discovered the service, it uses a WSDL document to decipher the valid operations and the makeup of those operations in terms of request and response parameters. The client then consumes the Web Services by sending structured XML messages back and forth across a transport protocol (typically HTTP) using a strict standard for encoding objects in a language-neutral format known as the Simple Object Access Protocol (SOAP).

With the advent of sophisticated development tools, such as Microsoft's Visual Studio.NET, much of the inner workings of technologies such as Web Services are abstracted away from the developer through multiple layers of system APIs. A developer can generate a Web Services client project in seconds that contains automatically generated helper code (stubs) by simply providing a URL or path to a WSDL file. From a Web Service provider's perspective, this helper code is great in concept, but often fraught with issues should the developer run in to a problem. If the developer does not fully understand the protocols and standards that make Web Services possible, their ability to troubleshoot these problems is greatly diminished.

Appendix A introduces the reader to the technologies that make Web Services possible, and provides Oracle Content Services Web Services low-level examples that show how to develop and troubleshoot Web Services at the XML level.

CORE CONCEPTS

Everything is an Item

There are some 40+ core repository object types used by Oracle Content Services (such as Document, Folder, Record, Link, User, and Group). Each of these repository objects has an associated set of context-specific attributes; for example, the Document object supports a "SIZE" attribute that returns the document's content size in bytes. Rather than expose each of these repository objects as individual complex types in Web Services, a single complex type "Item" is utilized to model/encapsulate the underlying repository object. The Item complex type has three standard attributes, `id`, `name`, and `type`, with a fourth complex attribute called `requestedAttributes` that provides the mechanism to return any other underlying object metadata (for example, "SIZE"). Most Manager operations that return Item instances support an attribute request array parameter as input. Use this parameter to request applicable object attributes that are not implicitly returned with every Item instance (in other words, attributes other than `id`, `name`, or `type`).

```
FileManager fm = ...

AttributeRequest[] ars = new AttributeRequest[] {
    new AttributeRequest(Attributes.DESCRPTION,null),
    new AttributeRequest(Attributes.SIZE,null)
};

Item doc = fm.resolvePath("/acme/Users/Users-M/MATT.SHANNON/resume.doc",ars);

System.out.println(doc.getId() + " - " + doc.getName() + " - " + doc.getType());

NamedValue[] nvs = doc.getRequestedAttributes();
for (int i=0; i<nvs.length; i++)
{
    NamedValue nv = nvs[i];
    System.out.println(nv.getName() + " - " + nv.getValue());
}
```

Output:

```
14207 - resume.doc - DOCUMENT
DESCRIPTION - Resume of Matt Shannon
SIZE - 19968
```

Service Managers

Oracle Content Services exposes a great deal of functionality through Web Services, with approximately 200 supported operations capable of being remotely invoked. Operations that are related are grouped together by a construct known as a manager that has its own WSDL file and SOAP address location. The complete listing of service managers, along with their supported operations, can be found at: <http://host:port/content/wsd1>, with *host* and *port* values replaced to reflect that of your own Oracle Content Services instance.

The following sections provide a brief overview of each manager's capabilities.

ArchiveManager

When a trash folder is emptied, its contents can be moved to a special area in the repository known as an Archive. Each domain (Site) has its own Archive instance, accessible only by content administrators. Archive management operations provided by this manager include facilities to:

- Empty an Archive
- Restore an object from the Archive
- Request for objects to be restored from the Archive (including filters such as date document deleted, name, and so on)
- Alter the configuration of an Archive (enabled/disabled, auto emptying enabled/disabled, and so on)

CategoryManager

Publicly accessible repository objects such as Document and Folder have inbuilt support to capture traditional file system metadata such as description, create_date, created_by, last_modified_date, and last_modified_by. The ability to associate custom metadata to an object instance is realized in the repository by way of category objects.

Consider the example of a technology consulting company that has a requirement to rapidly search and locate documents and folders specific to a particular customer project. Through the Oracle Content Services Web Services (or Web user interface), they can implement a new custom root category object, Project Category, with the following attributes:

Name	Type	Enumerated Values	Required	Default Value	Can Attribute Be Overridden
Project Number	Integer		True		True

A second category object could be introduced, Project Document Category, that subclasses Project Category and introduces the following attributes:

Name	Type	Enumerated Values	Required	Default Value	Can Attribute Be Overridden
Billable Material	Boolean		True	True	True
Document Type	String	Functional Specification, Design Document, Test Specification, Meeting Minutes	False	Design Document	True
Document Reviewer	User		False		True

When an instance of Project Document Category is applied to a document, the user will be prompted for attributes from both the Project Document Category as well as the superclass Project Category. Project Category is a root category, which means that it is a direct descendent from the base system Category object. Project Document Category is a subclass (descendent) of Project Category.

You can use a CategoryConfiguration to configure a workspace (Library) or folder so that a particular category is automatically applied to new files. Should the category require that metadata be supplied by the end user (in other words, should the category contain prompted attributes), then the user will be asked for the missing metadata when they upload content to that particular folder.

Category management operations provided by this manager includes facilities to:

- Create a new category object (providing an array of attribute details)
- Free a category object (only succeeds if no instances of the category are applied to files or folders)
- Add an attribute to an existing category object
- Modify an attribute of an existing category object
- Remove an attribute from an existing category object
- Set category configuration on a particular folder (for example, require that category X be applied to new items in the folder)
- Remove category configuration on a particular folder
- Delete an instance of a category
- Update an instance of a category

Tip – All domain (Site) category objects, and all domain root category objects, can be obtained through an `AttributeRequest` on the domain item for the attributes `Attributes.CATEGORY_CLASSES` and/or `Attributes.ROOT_CATEGORY_CLASSES`.

Direct subclasses for a given category object item can be obtained through an `AttributeRequest` on the category item for the attribute `Attributes.DIRECT_CATEGORY_SUBCLASSES`.

Attribute details for a given category object (such as name, type, required, or default value) can be obtained through an `AttributeRequest` on the category item for the attribute `Attributes.METADATA_ATTRIBUTES`.

Category configuration information (in other words, allowed categories or required categories) for a particular folder item can be obtained through an `AttributeRequest` on the folder item for the attribute `Attributes.CATEGORY_CONFIGURATION`.

CommonManager

Each object instance (such as file, folder, or group) in the system has a unique, numeric object identifier (id). This manager provides facilities to:

- Return an item instance by providing its id
- Return an array of item instances by providing an array of object ids

ContainerManager

Containers are special folders in the system under which workspaces (Libraries) can be created. Containers can be created either directly under the domain (Site), or under a parent container object. This manager provides facilities to:

- Create a new container (can specify whether user requests for workspace creation are allowed for this container)
- Delete a container
- Update a container

DomainManager

An Oracle Content Services instance may have one or more *domains* (also known as Sites). Each domain is an organizational entity comprising a collection of users and their content, metadata, and business rules. This manager provides facilities to:

- Obtain the default domain
- Update settings for a particular domain

FileManager

Tip – Content upload/download is handled by Oracle Content Services using standard HTTP PUT/GET, not through Web Service attachments. See the Content Upload/Download section of this paper for more details.

FileManager provides core document, folder, and link management capabilities, including the ability to:

- Resolve an item based on a supplied path string (the returned Item will be a Document, Link, or Folder)
- Resolve an item based on a supplied relative path string and source folder
- Check to see whether an item exists based on a supplied absolute path
- Check to see whether an item exists based on a supplied relative path string and source folder
- Create and return folders in the supplied destination folders
- Update a folder
- Return (list) items in a given folder, optionally providing sort criteria
- Check to see whether an object with the given name can be created in the given folder
- Copy one or more specified items to a given destination folder (provide conflict resolution criteria, such as overwrite/new version)
- Move one or more specified items to a given destination folder (provide conflict resolution criteria , such as overwrite/new version)
- Delete one or more specified items (deleting folders will trigger deletion of all its items)
- Create one or more document definitions. (Note: These definitions are created with empty content; to load content, perform an HTTP PUT to the WebDAV Server.)
- Create one or more documents, potentially utilizing saved document definitions as source. (Note: These documents are created with empty content; to load content, perform an HTTP PUT to the WebDAV Server.)
- Update a document
- Create a link to a given item in specified destination folder (links can be made to items of type: Domain, Container, Workspace, Folder, Document, and Family)
- Update a link
- Check if a given path string contains any link items
- Get supported languages
- Get supported character sets
- Return most recent documents for connected user, optionally providing sort criteria (the MostRecentDocAgent is responsible for maintaining users' most recent document statistics)
- Uncompress supplied items
- Get name conflict resolution options

See also ContainerManager and WorkspaceManager.

GroupManager

A group is a collection of users and groups that can be managed as a single unit. Groups provide a mechanism to conveniently and quickly assign privileges to a collection of users so that you don't need to assign privileges to each user individually. Oracle Content Services groups are locally managed application objects that are distinct from Oracle Internet Directory groups, which are not directly supported in this release. It is possible, however, for Oracle Internet Directory groups to be manually provisioned as Oracle Content Services groups using the GroupManager Web Service. Two distinct lists are maintained within an Oracle Content Services group: the member list, and the manager list.

Managers are implicit group members and have the ability to:

- Add and remove members of the group
- Add and remove managers of the group
- Rename or delete the group

This manager provides facilities to:

- Create a new group, supplying name, description, and member and manager lists
- Update a group, potentially replacing the member and manager lists
- Delete a group

- Find groups based on specific search and sort criteria
- Add members and managers to an existing group
- Remove members and managers from an existing group

LockManager

Lock objects are a mechanism to prevent outside changes occurring on an item that is currently being worked on. The repository will automatically deploy certain types of locks transparently when a caller requests specific types of operations (for example, a check-out call). Certain client applications that have native WebDAV support, such as Microsoft Office, will often request a WebDAV lock when a user edits a supported document type over WebDAV. This manager provides facilities to:

- Acquire a manual lock on one or more items
- Release a manual lock on one or more items
- Return a list of items locked by the current user that match the specified lock types (for example, manual lock, WebDAV lock, and so on)

Lock Type Constants from oracle.ifs.fdk.FdkConstants	Value
LOCKTYPE_MANUAL	1
LOCKTYPE_CHECKOUT	2
LOCKTYPE_FINALIZED	3
LOCKTYPE_WORKFLOW	4
LOCKTYPE_RECORD	5
LOCKTYPE_FAMILYHASRECORD	6
LOCKTYPE_DAV	7

PagingManager

When dealing with large item arrays (such as the result of a search), certain clients may find it is more convenient to deal with just a single "page" of items at a time. This approach is often favored by users with limited network bandwidth. This manager provides facilities to:

- Store a list of items in a paging list
- Retrieve a page of items of a given size from the paging list at the specific start index
- Note: There is only one paging list item array per session!

ProgressManager

Note: This manager is not fully implemented in 10g Release 1 (10.1.1) of Oracle Content Services. It is intended to provide a means to determine state and potentially cancel an in-progress operation occurring against the server-side session. This manager provides facilities to:

- Cancel any in-progress operation occurring server-side
- Return the progress state of an active operation (progress completed vs progress remaining)

QuotaManager

Quota is used to cap the amount of content that can reside in a workspace (Library), including users' personal workspaces. This manager provides facilities to:

- Update the amount of allocated quota for a specified item (requires AdministerQuota privilege on item)
- Request an update to allocated quota for a specified item
- Calculate consumed quota for a specified item


RecordsManager

Recent developments in corporate governance policies have focused heavily on document retention. Most organizations are now required to retain specific types of information for set periods of time. After this retention period elapses, many

organizations require a way to irreversibly destroy such data – or at least to initiate some disposition lifecycle. This manager provides facilities to:

- Create a file plan
- Create a record series under the specified file plan
- Create a record category under the specified record series or file plan
- Create a record folder under a record category
- Delete a file plan (the delete will fail if there are record series or record categories under it)
- Delete a record series (the delete will fail if there are record categories under it)
- Delete a record category (the delete will fail if there are any record folders or records under it)
- Delete a record folder (only empty record folders can be deleted)
- Update a file plan
- Update a record series
- Update a record category
- Update a record folder
- List file plans
- List record series under a specified file plan
- List record categories under a specified record series or file plan
- List record folders under a record category
- Get a file plan that matches a specified name
- Get a record series that matches a specified name under a specified file plan
- Get a record category that matches a specified name under a specified file plan or record series
- Get a record folder that matches a specified name under a specified record category
- Add a record category attribute
- Modify a record category attribute
- Return direct children of a record management object
- Make a record
- Update a record
- Get a recordized object given a record id
- Unrecordize a record from a given record category or record folder
- Set record configuration for a folder
- Delete record configuration for a folder
- Get a required record category for a specified folder

File Plan	A document containing the disposition authority for a set of records. A file plan for electronic records management is the same thing as a paper file plan – a way to organize the taxonomy for records and the associated lifecycle events for records, including retention, disposition, and deletion.
Record Series	A named container for a set of record categories.
Record Category	A description of a set of records within a file plan. Each record category has retention and dispositions data associated with it that is applied to all record folders and records within it.
Record Folder	An extension of a record category used to aggregate records. Record folders may be used to break records into periods that support different retention and disposition instructions than the containing record category.
Record	Information, regardless of medium, controlled by a particular record category.


 **Tip** – A "cutoff" is the close of a period that allows the disposal or transfer of records in larger blocks. The cutoff date keeps records together that relate to the same time frame. Once files reach their cutoff dates, they commence disposition phases (such as retention schedules). The total time you must retain your files is based on the cutoff date, plus the time stipulated on your retention schedule. Cutoff dates add structure and control to file maintenance by establishing

a timeline to move files to various storage mediums.

RemoteLoginManager

This manager provides session creation and logout capabilities, including facilities to:

- Authenticate a user based on user name and password (cleartext)
- Disconnect the current user session

 **Tip** – Cleartext authentication works only through HTTPS unless you change the domain property `IFS.DOMAIN.WS.CleartextAuthenticationRequiresHttps` to false. Use the Oracle Enterprise Manager 10g Application Server Control for Collaboration Suite (Oracle Collaboration Suite Control) to modify domain properties; click **Content** on the Collaboration Suite Home page, then click **Domain Properties**. You must restart the domain once you modify this property. Using `RemoteLoginManager.login()` over HTTP without setting this property to false will likely result in a `FeatureNotEnabled` exception.

RequestManager

Certain operations on a given item (such as copy, move, delete, check in, `createDocument`, or `createWorkspace`) can be configured so that they are controlled by a workflow. For example, the `FileManager.delete()` operation will invoke a `DeleteRequest` if the 'Delete' operation is workflow enabled for the particular item being deleted. This manager provides facilities to:

- Retrieve a list of requests submitted by the given user, matching the given request criteria
- Retrieve a list of requests for which the given user is a responder, matching the given request criteria
- Delete one or more requests with given ids
- Approve a request with the given id for the given user
- Deny a request with the given id for the given user
- Cancel a request with the given id for the given user
- Acknowledge a request with the given id for the given user
- Invoke a "UserRequest" for the given set of target items. If the target item is controlled by a `UserRequestWorkflowConfiguration`, the associated custom workflow will be triggered.
- Return whether the given operation/action is workflow enabled (in other words, request based) for the given item

For a given request item, the following attributes are available:

- `REQUEST_TYPE` – for example, `DELETEREQUEST`
- `REQUEST_WORKFLOW` – the workflow started by the request
- `REQUEST_WORKFLOW_PARAMETERS` – parameters sent to the workflow engine
- `REQUEST_STATUS` – `FdkConstants.REQUEST_STATUS_PENDING` 0, `REQUEST_STATUS_CANCELED` -1, `REQUEST_STATUS_SUCCEEDED` 1, `REQUEST_STATUS_ABORTED` -2, `REQUEST_STATUS_EXPIRED` -3, `REQUEST_STATUS_FAILED` -4
- `REQUEST_CREATED` – request create date
- `REQUEST_RESPONSE` – response to the request (true/false)
- `REQUEST_RESPONSE_DATE` – date of the response
- `REQUEST_TRIGGERED` – was the request triggered (non-blocking) (true/false)
- `REQUEST_FAILURE_CAUSE` – reason for request failure
- `REQUEST_TARGETS` – request target items
- `REQUEST_DEFINITIONS` – definitions that will be utilized to carry out the operation type should the request be approved

SearchManager

To perform a search, you must first build a SearchExpression tree. A SearchExpression is a complex type object comprised of two operands, left operand and right operand, that are associated by a specified operator. Depending on the type of operator, the left and/or right operands may themselves be SearchExpression nodes. This enables you to build a complex SearchExpression tree.

Valid operators include:

Operator	Syntax	Notes
EQUAL	Left operand is the string name of the attribute being compared. Right operand is the value being compared, represented as a String, Integer, Long, or Date.	Wildcard characters * and ? are supported only for the EQUAL comparison operator when comparing an attribute with datatype String. Examples: SIZE GREATER_THAN 1048576 NAME EQUAL *.doc
GREATER_THAN		
GREATER_THAN_EQUAL		
LESS_THAN		
LESS_THAN_EQUAL		
NOT_EQUAL		
IN		
CONTAINS	Left operand must be null, Right operand specifies words or phrases to be found in the content of a document.	Words are specified by spaces, and phrases are enclosed in double quotes. Example: <NULL> CONTAINS "Content Services"
AND	Left and Right operands must themselves be SearchExpressions.	(SIZE GREATER_THAN 1048576) AND (NAME EQUAL *.doc)
OR		
NOT	Left operand must be null, Right operand must be a SearchExpression.	<NULL> NOT (((SIZE GREATER_THAN 1048576) AND (NAME EQUAL *.doc))

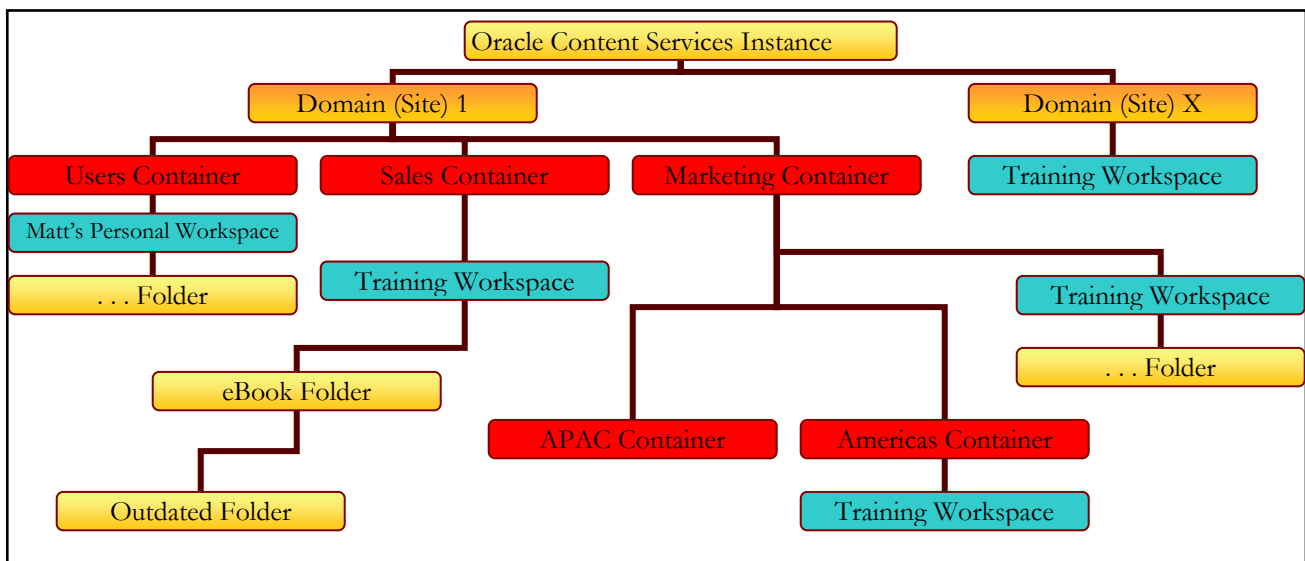
A Search is invoked by supplying SearchManager's search operation with a SearchExpression tree, along with zero or more optional search options.

Search options include the ability to:

- Restrict search to one or more specified folders (including optionally sub-folders)
- Restrict search to documents and folders (default), or records
- Restrict search to the Site Archive for documents that were deleted from a specified workspace, and optionally in the specified deletion date range
- Include/exclude non-current versions of a versioned document from being searched
- Set the start index for the first item in the server's search result array that should be returned (default is 1, which returns all items from the search result array, starting from the very first search result)
- Limit the number of items returned back to the client from the server's search result array

SecurityManager

Appendix B provides the list of default roles and their explicit permissions that ship with Oracle Content Services. A role, which is comprised of a set of access permissions, is granted as a single unit to a user or group to provide them privileges to perform certain operations. Two types of roles exist, namely, core administration roles and standard (non-administration) roles.



Administration roles can be granted to a user or group at the domain (Site) level. Many of these roles can also be granted at the container or workspace level. Using the sample hierarchy shown in the previous figure as an example, user "Larry" is granted the propagating ContentAdministrator role for Domain1, with user "Matt" being granted the same role in the security configuration for the Marketing Container. Roles that are propagating in nature mean that the initial grant applies to the target object along with all of its children. Using our example scenario again, Larry has ContentAdministrator permissions across the entire domain (Site), including its containers, workspaces, and folders, whereas Matt can only exercise these permissions on objects contained in the Marketing Container hierarchy.

The SecurityManager provides facilities to:

- Fetch a role based on its id
- Fetch a role based on its common name
- Fetch a default role based on its registered role key
- Return the available roles in the domain (Site) that apply to a specified target object
- Return the available roles in the domain that apply to a specified item type
- Update security configuration on a specified target item (for example, a workspace)
- Add grants to the security configuration on a specified target item
- Remove security configuration on a specified target item
- Check whether the specified user/group has the specified permission (not role!) on the specified target item
- Return the set of users that are granted the specified role on the specified security configuration item (includes users granted the role by way of group membership, as well as those users who were granted the role on a parent object should the role be propagating)
- Return true/false if the specified user has any grants on the specified security configuration that were propagated from a parent
- Return true/false if the specified user has permissions on the specified target object only by way of a single explicit grant to that user on the target

Tip – In order to be able to leverage an administration role (such as QuotaAdministrator), the user's session must be switched into administration mode by using the SessionManager's setSessionMode method and providing the option `FdkConstants.SESSION_MODE_DOMAIN_ADMINISTRATION`.

ServiceToServiceLoginManager

IMPORTANT NOTICE: The AXIS Java client stubs shipped with Oracle Content Services do not support digest authentication (see <http://ws.apache.org/axis/java/security.html>). Clients that use the Java stubs can authenticate through basic authentication, or alternatively by using an HTTP client and the S2S servlet.

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 partner application instead supplies a digest credential with each user session login request (or potentially, a basic credential over HTTPS) that is used to validate/authenticate the partner as being trusted to the particular provider service.


Oracle Content Services operates as the trusting provider service, with the partner service being potentially any application (registered/configured with Oracle Internet Directory) that is capable of establishing a client SOAP over HTTP Web Service connection with digest authentication headers. For this manager to function, the server must be configured for S2S authentication, and the domain property `IFS.DOMAIN.CREDENTIALMANAGER`. `ServiceToServiceAuthenticationEnabled` must be set to true. Use the Oracle Collaboration Suite Control to set domain properties.


This manager provides facilities to:

- Authenticate a user with given user name. The S2S header `ORA_S2S_PROXY_USER` must also be set to the value of the supplied user's Oracle Internet Directory nickname attribute.

As an example, consider a custom portlet within Oracle Portal that renders information pertaining to the user's most recently accessed documents in Oracle Content Services. The runtime flow would be similar to the following:

1. User "matt" authenticates to Oracle Portal (through the Oracle Login Server).
2. A default portal user home page is requested to be displayed that contains the "most recent documents" portlet (referred to as "portlet").
3. The portlet checks to see if an existing Oracle Content Services session cookie exists for the user in the user's HttpSession (or equivalent) object store. If no existing session exists, proceed with service to service authentication (step 4). Otherwise, obtain a FileManager (step 7) and make the appropriate call.
4. The portlet obtains the portal service's credentials from a credential store (such as Oracle Internet Directory).
5. The portlet initiates a Web Service request to the Oracle Content Services ServiceToServiceLoginManager with digest authentication HTTP headers present that identify the partner service (Oracle Portal), and HTTP header `ORA_S2S_PROXY_USER` set to a non-null value (for example, "matt"). The ServiceToServiceLoginManager login method is called, supplying the user name of the portal authenticated user ("matt"[@non-default realm]).
6. A session cookie is returned that identifies the newly created Oracle Content Services user session. This, in turn, is stored in an appropriate location, such as the user's HttpSession object.
7. Using the session cookie, obtain a FileManager, and then call the `getMostRecentDocuments` method.
8. Process the returned item array and render portlet results.

 **Tip** – Remember to call `RemoteLoginManager.logout()` to disconnect the user's session when it is no longer needed.

 **Tip** – A number of Web Services clients provide no published means to access the underlying HTTP transport to allow setting of the HTTP headers and so forth. These clients are thus are incapable of utilizing the `ServiceToServiceLoginManager`. As a workaround, Oracle Content Services ships an S2S Servlet that can be accessed through a regular HTTP client library, such as `HttpClient`, part of the Apache Jakarta Commons project. The session cookie returned through authentication to the S2S servlet can then be used by the Web Services client.

SessionManager

When a client performs an operation in Oracle Content Services using Web Services, such as creating, modifying, or deleting an object, the changes are immediately committed to the repository. However, in some cases, the client may

need to commit a set of dependent operations together. That is, the client needs to ensure that if all operations cannot be successfully performed, none of the operations are performed; in other words, the operations must be atomically committed or rolled back. SessionManager provides methods to enable clients to wrap operations in a transaction block that can be used to control when the changes are committed to the repository. This manager provides facilities to:

- Return the current connected user as an item
- Toggle the current user’s session into or out of administration mode (if applicable)
- Begin a transaction on this session, returning a transaction id
- Commit a transaction on this session with a specified transaction id
- Roll back a transaction on this session with a specified transaction id (id <= 0 implies to roll back the entire transaction stack)
- Keep a session alive to prevent it timing out (NO-OP)
- Get supplied session properties (such as WebServicesConstants.LOGIN_USER, WebServicesConstants.SESSION_TIMEOUT, WebServicesConstants.TRANSACTION_TIMEOUT)


 **Tip** – Oracle Content Services supports nested transactions. Consider the following example:

```

1      Begin Outer-Transaction
2          Begin Inner-Transaction
3              ; perform some operation
4          Begin Inner-Transaction
5              ; perform some operation
6          Abort (Rollback) Inner-Transaction
7      Commit Inner-Transaction
8      Abort (Rollback) Outer-Transaction

```

The innermost transaction (lines 4-6) is rolled back. Hence, any changes made in that particular transaction will never be applied. The encapsulating inner transaction (lines 2-7) is set to be committed, though its changes are at the mercy of the outer transaction. Because the outer transaction (line 8) is rolled back, any changes set to be committed in the inner transactions will not be applied. Thus, inner transactions are essentially just save-points. They can be rolled back individually, but in order for them to be committed, the outer transaction must be committed.

 **Tip** – Operations that result in Data Definition Language (DDL) operations in the database (such as creating tables) will be immediately committed and cannot be reversed by a transaction. Creating a category object is one such example.

SortManager

Oracle Content Services allows users to set default sort preferences for various tables used by the application. Applicable tables are defined in the FdkConstants class that ships with the Web Service Java stubs library. Currently, these tables include:

Table	Description	Table	Description
fileListingTable	File listing	recentFilesTable	Recent Files
versionHistoryTable	Version history	groupMemberList	Group Members list
joinableWorkspacesTable	Joinable Workspaces	groupMembers	Group Members
lockedFilesTable	Locked Files	memberListingTable	Member Listing
userSearchResultsTable	User Search Results	rolesTable	Roles
checkedOutFilesTable	Checked Out Files	securityConfigTable	Security Config
workflowReportTable	Workflow Reports	selectedUsersOrGroupsTable	Selected Users or Groups
virusNamesTable	Virus Report	usersOrGroupsResultsTable	Users or Groups Results

Additionally, many Web Services operations that return item arrays support defining the sort sequence as part of the operation parameters. This manager provides facilities to:

- Set the user's sort preference for the specified table
- Obtain the user's primary sort attribute for the specified table
- Obtain the user's primary sort direction for the specified table (true=ascending, false=descending)
- Obtain the user's secondary sort attribute for the specified table
- Obtain the user's secondary sort direction for the specified table (true=ascending, false=descending)
- Sort an array/list of items (where the array of items is an attribute of the specified item)

SystemConfigurationManager

This manager provides facilities to:

- Return values for requested system configuration properties

Applicable properties are defined in the SystemConfigurationProperties class that ships with the Web Service Java stubs library. Properties include:

SYSTEM_ANTIVIRUS_ENABLED: When requested, returns a Boolean indicating whether the antivirus feature is enabled.

LAST_ANTIVIRUS_DEFINITION_UPDATE: When requested, returns a Date indicating the last date the antivirus definition file was updated.

WORKFLOW_ENABLED: When requested, returns whether the Workflow feature is enabled.

TrashManager

Each workspace (including personal workspaces) comes with a trash folder that, if enabled, stores deleted items for that workspace. This manager provides facilities to:

- Empty the specified trash folder
- Set trash configuration on the specified trash folder (enabled/disabled, auto-empty enabled/disabled, minimum holding period in seconds before auto purge when using auto-empty feature)

UserManager

This manager provides facilities to:

- Get a user with a specified name
- Obtain one or more user preferences for the current user with the specified preference keys
- Set one or more user preferences for the current user
- Obtain one or more user preferences for the specified user for the specified preference keys
- Set one or more user preferences for the specified user
- Set one or more domain-level (Site-level) default user preference values
- Obtain one or more domain-level default user preference values
- Find provisioned users in the domain that matches the specified search criteria
- Find users in the current user's Oracle Internet Directory that match the specified search criteria
- Synchronize the preferences of the current user with those found in Oracle Internet Directory
- Synchronize the preferences of the specified user with those found in Oracle Internet Directory

VersionManager

Versioning provides the ability to track changes made to an object's content and metadata throughout its lifecycle.

Although it is possible to track every revision of every document in the system, doing so would be quite expensive from a system resources perspective. Oracle Content Services is extremely flexible with regards to versioning policies, enabling the administrator to choose the right balance between user requirements and system resources/performance. Versioning configuration can be set on a folder to specify:

- Whether manual or automatic versioning should be applied to items in the folder
- The maximum number of version items to retain in family for version items that have DO_NOT_PURGE set to FALSE (automatic purging of the oldest revision will occur when required)

- Whether manual or automatic version labeling should be applied
- What label type to use (such as roman numerals), if automatic versioning labeling is enabled. Label types are found in FdkConstants (for example, FdkConstants.VERSION_LABELING_ROMAN).
- Whether the versioning configuration is final (in other words, whether subfolders have the ability to override the versioning configuration)
- Whether the versioning configuration is enabled

A serial versioning model is employed by Oracle Content Services – in other words, the server maintains a single version series per versioned file. To manually create a new version, the author must first check out the file, which results in the server making a working copy of the latest version (including both content and metadata). This server-resident working copy is accessible only to the user who checked out the file. A lock is also issued to prevent other authors from checking out the versioned file. When ready, the author then checks in the file, which results in a new version being created. The new version becomes the latest version of the file, which is immutable and thus cannot be further updated. The lock acquired at check out is then released, allowing other users to check out the file, and the working copy object is destroyed. This manager provides facilities to:

- Set/replace versioning configuration on a folder
- Remove versioning configuration from a folder
- Return the versions of a specified item and the requested attributes that apply to each item
- Update a version-controlled file's attributes (version comment, version label, do_not_purge flag)
- Make one or more non-version-controlled files versioned
- Check out a specified set of items (optionally providing reservation comments)
- Cancel check out for a specified set of items
- Check in a specified set of items, providing version comments and version labels
- Copy a file version with a specified id to the specified destination folder
- Move a file version with a specified id to the specified destination folder
- Delete a file version with a specified id (so that it is no longer part of version history)
- Copy a file version with a specified id as the new latest version
- Copy a file version with a specified id as the working copy

VirusManager

The ability to scan and potentially repair documents in an unobtrusive and efficient manner is a key feature of Oracle Content Services. A default virus scanning adapter is shipped with Oracle Content Services that allows the system to interface with Symantec AntiVirus Scan Engine (SAVSE) using ICAP 1.0. An asynchronous background agent, called the VirusScanAgent, regularly polls the virus scanning adapter to determine if new virus definitions are available. When a new virus definition is detected, the domain property `IFS.DOMAIN.ANTIVIRUS.LastDefinitionUpdate` is modified accordingly.

The scanning process is on-demand, which means that when a file's contents are requested, the system will use metadata associated with that file to determine whether a scan needs to be performed. In particular, a `LAST_SCANNED_DEFINITION_DATE` attribute is associated with each file, which tracks the virus definition timestamp that was last used to scan the file. If this attribute is null, or older than the `LastDefinitionUpdate` domain property, then the file's contents are streamed through the scanning adapter; otherwise, the file's contents are immediately returned. Depending on the result of the scan, the file's contents are either returned to the user should no virus have been detected (or if a repair was successful), or else an error is returned and the file is quarantined. The `LAST_SCANNED_DEFINITION_DATE` attribute is also updated to reflect the new virus definitions that were used.

When a file is quarantined, the `IS_QUARANTINED` and `QUARANTINED_DATE` attributes for that file are updated accordingly. The VirusScanAgent will also receive an event that notifies it of the infected file. The agent attempts to repair quarantined files that have a `REPAIR_ATTEMPTS` value less than the domain property `IFS.DOMAIN.ANTIVIRUS.MaxRepairAttempts`, and with a `LAST_SCAN_DEFINITION_DATE` older than the


LastDefinitionUpdate domain property. The agent is also responsible for instantiating and associating a VirusReport (Category Instance) on the quarantined file after the repair attempt.

Files under quarantine have the following properties and behaviors:

- File objects themselves will be unaffected by quarantine status (in other words, metadata can still be viewed/modified).
- Contents cannot be opened for read access under any circumstances. Attempts will result in an exception. Contents will remain unreadable even if the antivirus option is disabled.
- Contents may be overwritten.
- Files and their contents may be deleted.
- A file will not have specific infection information available until the VirusScanAgent attempts to repair it.

This manager provides facilities to:

- Scan specified items
- Attempt repair of specified items
- Retrieve a virus report for specified item that was sent for repair

 **Tip** – The virus definition timestamp can be obtained from the SystemConfigurationManager.

DOCUMENT Attributes	Data Type
IS_QUARANTINED	Boolean
QUARANTINED_DATE	Date
LAST_SCANNED_DEFINITION_DATE	Date
REPAIR_ATTEMPTS	Integer

VIRUSREPORT Attributes	Data Type
VIRUS_ID	Long
VIRUS_NAME	String
LAST_ATTEMPTED_REPAIR_DATE	Date
VIRUS_IS_REPAIRED	Boolean

WorkflowManager

For a given item (such as an instance of a file or folder), a number of operations appropriate for the item's type can be configured so that they are workflow controlled. The list of available operations that can be workflow controlled includes:

- Upload document, read document, delete, check in, copy, move, user request, create workspace (Library), join workspace, increase quota

 **Tip** – Most items support the property WORKFLOW_OPERATIONS, which returns a string array containing names of operations that can be workflow-enabled (and configured) for the particular item.

A workflow configuration is used to associate a particular item instance and operation type with a workflow item. A workflow configuration can be set to triggered (non-blocking) or approval-based (blocking). A triggered configuration means that the registered operation will occur right away (as if it were not workflow-driven), yet a workflow process will still be started. An approval-based configuration means that the registered operation will not occur until the workflow process is approved. For example, a triggered workflow could capture all createDocument (upload) requests on a folder item and then, using logic in the workflow, respond to these events (such as calling a Web Service, or logging the createDocument requests to a file). Alternatively, an approval-based workflow could capture all delete requests and then require that a particular responder approve the deletion of the particular items.

This manager provides facilities to:

- Return all registered workflow instances
- Set a workflow configuration on a given item for the given operation type
- Remove a workflow configuration from a given item for the given operation type
- Retrieve all workflow configurations that exists for the given item
- Retrieve workflow configuration for the given item that matches the given operation type
- Validate that the value of given workflow parameter is valid for the given workflow
- Return the request item corresponding to the supplied workflow process id

A workflow configuration item supports the following attributes:

- WORKFLOW_CONFIGURATION_OPERATION – operation on which the workflow configuration is based, such as:
 - CREATE_WORKSPACE_WORKFLOW_CONFIGURATION_OPTION
 - JOIN_WORKSPACE_WORKFLOW_CONFIGURATION_OPTION
 - INCREASE_QUOTA_WORKFLOW_CONFIGURATION_OPTION
- WORKFLOW_CONFIGURATION_WORKFLOW – the workflow specified for a workflow configuration
- WORKFLOW_CONFIGURATION_PARAMETERS – parameters for a workflow configuration – Item array
- WORKFLOW_CONFIGURATION_RESPONDERS – list of responders attached to workflow configuration
- WORKFLOW_CONFIGURATION_RESPONDERS_BYPASS – whether responders that start a request should bypass the workflow (true/false)
- WORKFLOW_CONFIGURATION_IS_TRIGGERED – whether the request is non-blocking (triggered), or approval-based (blocking)

A workflow item supports the following attributes:

- WORKFLOW_TYPE – (WORKFLOW_TYPE_ORACLE, WORKFLOW_TYPE_BPEL)
- WORKFLOW_IS_TRIGGERED – whether the workflow is triggered (non-blocking) (true/false)
- WORKFLOW_REQUIRES_RESPONDERS – whether the workflow requires responders (true/false)
- WORKFLOW_PARAMETERS – parameters for a workflow - Item array of type WorkflowParameter

A workflow parameter item supports the following attributes:

- WORKFLOW_PARAMETER_NAME
- WORKFLOW_PARAMETER_DESCRIPTION
- WORKFLOW_PARAMETER_DEFAULT_VALUE
- WORKFLOW_PARAMETER_PROMPTED
- WORKFLOW_PARAMETER_REQUIRED
- WORKFLOW_PARAMETER_HIDDEN
- WORKFLOW_PARAMETER_FINAL

WorkspaceManager


Workspaces (Libraries) are special folders that can be created directly under a domain (Site) or container to store content, including files and regular folders. Workspaces, unlike containers and regular folders, have a Trash folder and can have quota associated with them. There are two types of workspaces in Oracle Content Services: personal workspaces and shared workspaces. Personal workspaces are a special type of workspace for exclusive use by a user, whereas shared workspaces can be utilized by any user who is granted the appropriate privileges. Workspaces must be uniquely named within the direct container where they are located. This means that workspaces in different containers can have the same name. The correct way to look up a workspace is thus by its path or its unique id. This manager provides facilities to:


- Create a new workspace, or submit a request for the workspace to be created (if workflow has been enabled for the CreateWorkspace operation)
- Create a new personal workspace for the current connected user
- Create a new personal workspace for the specified user

- Update a workspace (name, description, joinable flag)
- Delete a workspace with a specified id
- List joinable workspaces that match the specified name search criteria
- Request to join a workspace with the specified id, requesting specified roles

Client Session

Session Management in a Web Service (utilizing HTTP for the SOAP transport binding) is typically achieved through either SOAP headers, or by HTTP and its cookie support. The Oracle Content Services Web Services support only the latter option. Thus, in order to utilize the Oracle Content Services Web Services and maintain session state, **the client must support HTTP cookies**. Session authentication/establishment for the Oracle Content Services Web Services is performed by either the RemoteLoginManager, or the ServiceToServiceLoginManager. When a client successfully authenticates using the appropriate login operation, a set-cookie directive is returned with the response (the cookie name is JSESSIONID – the standard OC4J Session Cookie). Any subsequent Web Service requests by the client must include this cookie. Failure to include this cookie will result in a SOAP Fault being returned, containing an FdkException with the error code "ORACLE.FDK.AccessDenied." SOAP Headers are not currently used, because not all clients support these. Additionally, the session cookie is required in order to upload/download content through our WebDAV server (which does not support SOAP headers).

 **Tip** – There is a limit imposed on the number of concurrent sessions a user can maintain against Oracle Content Services. This is done for a number of reasons, including as a preventative measure against Denial of Service attacks whereby if the user concurrent session limit was unbounded, the middle-tier computer's memory could potentially be exhausted. Because HTTP is a connection-less protocol, the server has no way of knowing if a client disconnects unless explicitly alerted to by the client by way of a request (such as a logout click/call). Many servers, including Oracle Content Services, impose a session inactivity timeout, whereby if the user has not made a request in a certain period of time, the server frees (and disconnects) the client's inactive session. There is potential that should the client not explicitly log out, yet regularly be creating new connections to the application (and thus new user sessions), the rate at which new sessions are being created will exceed the rate at which the server is disposing of the user's inactive sessions. This results in the concurrent session limit being reached, and in the case of a Web Services request, the server would respond with a SOAP Fault and FdkException accordingly. Thus, it is imperative that the developer not forget to invoke the RemoteLoginManager logout operation when appropriate.

 **Tip** – Microsoft's SOAP Toolkit 3.0, which enables Web Services for pre-.NET applications, contains cookie support (through HttpConnector30). However, adjusting the EndPointURL of a connector causes its stored cookies to be lost. Because of this, when using this toolkit with the Oracle Content Services Web Services, you must manually track the OC4J session cookie on a login operation, and include it in the HTTP headers on any subsequent client requests. .NET and Java clients are also faced with the problem of maintaining session state across the various Web Service managers. Example code is provided in C# and Java that illustrates how managers can share cookie state (see FdkSession and Managers classes in the appropriate programming language framework download bundle).

Content Upload / Download

During the course of Oracle Content Services development, we looked at some of the major Web Services client development platforms and their support for various Web Services standards, including support for attachments. A number of clients either did not support attachments, or had key functionality missing such as the ability to support MIME attachments. Because our Web Services are based on HTTP and require client-side cookie support, we are able to provide content upload and download through the Oracle Content Services WebDAV Server. The WebDAV server must be presented with the same user session cookie that is established through the RemoteLoginManager or ServiceToServiceLoginManager login operation. The client, once identified by the WebDAV server, only needs to perform a standard HTTP GET or PUT to download or upload content. HTTP PUT calls can also participate in Web Service-initiated transactions; this means that a rollback on the transaction will also reverse any content changes.

Core Item objects, such as Document and Folder objects, provide support for URL and ID_URL attributes, which can be requested as part of the attribute request array parameter. In order to download a document from Oracle Content Services, you initiate an HTTP GET call, supplying the appropriate resource locator value to the WebDAV Server. The URL parameter for a document is derived from the document's folder path and name, whereas the ID_URL parameter for a document is based on the document's unique identifier (ID value). Should a document move location (or change name), a bookmarked hyperlink based on the ID_URL will continue to function, whereas a URL (folder path) based hyperlink will no longer be valid. Although the ID_URL may seem most suitable to use when providing hyperlinks to

documents, keep in mind that this value contains no contextual information that could help users identify the particular file it represents.

Although there are a number of different approaches to uploading a document to Oracle Content Services, all rely on an eventual HTTP PUT operation being initiated against the WebDAV Server. A PUT call that supplies the WebDAV server with a resource locator that identifies an existing document in the system will result in an attempt to overwrite the destination document's content. This attempt may or may not succeed, depending on the user's security permissions, document lock status, and destination folder policies (such as versioning policy or enforced recordization). The following steps show the preferred approach to uploading a document to Oracle Content Services:

1. Create a DocumentDefinition object using the FileManager createDocumentDefinition operation, specifying a destination filename, along with an Attribute Request for the new DocumentDefinition's URL. Store the ID and URL of the returned DocumentDefinition item.
2. Upload the source document's content by performing an HTTP PUT operation against the WebDAV Server, using the destination URL obtained in the previous step. This will result in the DocumentDefinition's null content being set.
3. Invoke the FileManager createDocument operation, specifying the DESTINATION_FOLDER and USE_SAVED_DEFINITION options. The value for the USE_SAVED_DEFINITION option should be the ID value obtained from the initial createDocumentDefinition call. Additional options can also be specified, including CATEGORY_DEFINITION, RECORD_DEFINITION, and conflict resolution options such as OVERWRITE or NEWVERSION.

```
FdkSession session = .... // Webservice client session
FileManager fileM = .... // FileManager instance
Item folder = .... // Destination Folder


// Request the URL attribute be returned
AttributeRequest[] requestedAttributes =
    new AttributeRequest[] { new AttributeRequest(Attributes.URL,null) };

// Create DocumentDefinition supplying AttributeRequest
Item docDef = fileM.createDocumentDefinition(
    new NamedValue[] { new NamedValue(Attributes.NAME, "sample.doc") } ,
    requestedAttributes
);

// Extract the DocumentDefinition URL from the returned DocumentDefinition
String destURL = (String) FdkUtils.getAttribute(docDef,Attributes.URL);

// Performs HTTP PUT of source document to destination URL
uploadSampleFile(session,"sample1.doc",destURL);

NamedValue[] documentDef = new NamedValue[] {
    new NamedValue(Options.USE_SAVED_DEFINITION,new Long(docDef.getId())),
    new NamedValue(Options.DESTFOLDER, new Long(folder.getId())),
};
// Create a Document using saved definition item ...
Item doc = fileM.createDocument(documentDef,null,null);
```

 **Tip** – DocumentDefinition items are temporary objects in the repository that get automatically destroyed/cleaned up. DocumentDefinitions are extremely useful – for example, if a particular createDocument call fails due to some type of recoverable exception (such as missing metadata), the document does not need to be re-uploaded; rather, the saved document definition can be retried in the corrected createDocument call.

Attribute Chaining

Extract 1 shows the SOAP body contents of a sample SOAP message depicting a login operation. In the userAttributes section, you can see that the caller is making an AttributeRequest for the PERSONAL_WORKSPACE item of the authenticating user to be returned along with the PATH attribute of that PERSONAL_WORKSPACE item. This is a form of attribute chaining. This particular attribute request entry results in the server returning the following data:

1. personal workspace id
2. personal workspace name
3. personal workspace type
4. personal workspace path

But what if the caller was really only interested in the personal workspace path? The attribute request that we provided resulted in unnecessary information being returned, and consequently wasted network bandwidth and increased client and server processing time. The solution to this problem is an alternative form of attribute chaining, presented in Extract 2. Using the attribute separator ":", we can chain together the two attributes (PERSONAL_WORKSPACE and PATH), which means that the server only sends back the path to the personal workspace, not the actual personal workspace item itself. Bandwidth and response time savings using our login scenario are only minimal, but when dealing with attribute requests for large item result arrays, these savings become much more noticeable.

With either form of attribute chaining, there is no restriction on how deep the chaining of the attributes can be. The only requirement is that in order for the syntax to be accepted, all the elements up to the last one of the composed attribute name must convert to Item.

Extract 1 – Example SOAP body contents for a login operation, depicting one form of attribute chaining.

```
<ns1:login
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:ns1="http://xmlns.oracle.com/content/ws">
>
  <username xsi:type="xsd:string">matt</username>
  <password xsi:type="xsd:string">welcome1</password>
  <options xsi:type="ns1:ArrayOfNamedValue" xsi:nil="true"/>
  <userAttributes soapenc:arrayType="ns1:AttributeRequest[2]"
    xsi:type="soapenc:Array">
  >
    <userAttributes xsi:type="ns1:AttributeRequest"/>
    <attributeName xsi:type="xsd:string">PERSONAL_WORKSPACE</attributeName>
    <requestedAttributes soapenc:arrayType="ns1:AttributeRequest[1]"
      xsi:type="soapenc:Array">
    >
      <requestedAttributes xsi:type="ns1:AttributeRequest">
        <attributeName xsi:type="xsd:string">PATH</attributeName>
        <requestedAttributes xsi:type="ns1:AttributeRequest" xsi:nil="true"/>
      </requestedAttributes>
    </requestedAttributes>
  </userAttributes>
  <userAttributes xsi:type="ns1:AttributeRequest"/>
  <attributeName xsi:type="xsd:string">EMAIL_ADDRESS</attributeName>
  <requestedAttributes xsi:type="ns1:AttributeRequest" xsi:nil="true"/>
  </userAttributes>
</userAttributes>
</ns1:login>
```

Extract 2 – Alternative form of attribute chaining using attribute separator ":".

```
<userAttributes xsi:type="ns1:AttributeRequest"/>
  <attributeName xsi:type="xsd:string">PERSONAL_WORKSPACE:PATH</attributeName>
  <requestedAttributes xsi:type="ns1:AttributeRequest" xsi:nil="true"/>
```

</userAttributes>

Quiz – What will the following Attribute Request return? "PERSONAL_WORKSPACE:PARENT_FOLDER:ID" (Hint: read the attributes right to left).

Answer: This attribute request returns the ID of the container folder in which the user's personal workspace is stored.

Tip – Applicable attributes for the various item types are defined in the Attributes class that ships with the Web Services Java stubs library.

Troubleshooting

Capturing SOAP Packets

A number of software tools exist that make it possible to capture SOAP packets being transferred over HTTP between a Web Services client and server. The first class of tool is a low-level network traffic analyzer, such as the free Ethereal software (<http://www.ethereal.com>). These tools work by analyzing and filtering network traffic passing "over the wire" through a chosen network connection. Tools such as Ethereal are typically harder to set up, but have the advantage of being completely transparent to the client/server application, thus making them ideal for production environment troubleshooting.

The next class of tool works by acting as an intermediary between client and server. The client (application) must be configured to connect to this intermediary (instead of to the server directly), and the intermediary in turn "tunnels" the traffic between the client and server. Free tools include tcpTrace (<http://www.pocketsoap.com/tcpTrace/>) and Trace Utility, which is bundled with the Microsoft SOAP Toolkit Version 3. These tools will always request three main parameters:

1. local port
2. destination server
3. destination port

The local port is the TCP port on the intermediary server to which the client will connect. The client requires the hostname (or IP address) and local port details of the intermediary server. The destination server and destination port is the true Web Services server location to which the intermediary will tunnel traffic.

Consider the following scenario:

- The Web Services server is located at host "linux1.oraclebox.com" on port 7777
- The client is running tcpTrace *locally* on a workstation (hostname: pc.oraclebox.com), and port 8080 on the client machine is available

A valid configuration for this scenario would be:

- tcpTrace is configured with local port=8080, destination server=linux1.oraclebox.com, destination port=7777
- Client application (such as JDeveloper project or C# application) is configured to use server=pc.oraclebox.com (or localhost), server port=8080

Hence, any traffic sent from the client application to pc.oraclebox.com:8080 will be tunneled to linux1.oraclebox.com:7777.

Node Log Level

The Oracle Collaboration Suite Control provides the ability to administer, monitor, and manage Oracle Content Services Applications tiers. You can access Oracle Content Services administrative functions by clicking **Content** in the system components list on the Collaboration Suite Home page. Use the Node Configurations link, accessible from the Content Services Home page, to access the various nodes that comprise the Oracle Content Services instance. Each node has an associated set of servers that run within its associated JVM – for example, the Content HTTP Node contains the EcmHttpServer, which is used by the Oracle Content Services Web UI, WebDAV, and Axis Servlets.

The node configuration for each node also exposes Logging settings that allow you to choose how detailed or verbose logging output should be for a particular component. Setting the "All" option to "Finest" is the simplest way to capture every possible logging statement for the node, but this will obviously cause the log file to grow at a large rate and add significant load onto the server. The log file would also be hard to follow if multiple users or processes were accessing the repository at the same time. Setting fine-grained log options for specific components is definitely a more suitable approach. For troubleshooting Web Services, you should set the following loggers to Finest:

- AllProtocols/DAV
- ClientServicesLayer
- Repository

Tip – If you make any changes to log settings, you must restart the node for your changes to take effect.

Tip – The node type determines where logging information will be recorded. For example, a node of type "HTTP Node" will output its logging details to the associated OC4J container's application.log file. For OC4J_Content, the application.log file is located under the *ORACLE_HOME*/j2ee/OC4J_Content/application-deployments/content/OC4J_Content_default_island_1 directory. A regular node, on the other hand, records its information in a node log file located under the *ORACLE_HOME*/content/log directory. Oracle Content Services logging information is also recorded in the *ORACLE_HOME*/opmn/logs directory.


CONSUMING WEB SERVICES

Java Stubs


As Web Services are self-describing in nature, most advanced development platforms are capable of automatically processing the WSDL to generate the language-specific client-side stubs. In an ideal world, this would work as expected. To save our Java developers from tackling these treacherous development tool wizards (particularly given the number of WSDL files we expose), we have made available prepackaged client-side Java stubs that implement our manager interfaces.

Our stubs, contained in the Java archive "content-ws-client.jar" under the "oracle.ifs.fdk" package, require certain libraries to be present on the client. The most basic of clients would need the following libraries in CLASSPATH:

- Content Services Apache Axis Java Stubbs (content-ws-client.jar)
- Apache Axis libraries
 - axis.jar, commons-discovery-0.2.jar, commons-logging-1.0.3.jar, jaxrpc.jar, saaj.jar, wsdl4j-1.5.1.jar
- JavaMail API and JavaBeans Activation Framework (mail.jar, and activation.jar)
 - These jars are optional. Failure to include them in the classpath will result in the following message: WARNING: Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled.

 **Tip** – content-ws-client.jar also bundles useful Constant classes:

- oracle.ifs.fdk.Attributes – Defines supported attributes and their datatypes for the various item types
- oracle.ifs.fdk.ItemTypes – Constants that represent the String name of each item type
- oracle.ifs.fdk.FdkConstants – Generic Façade Constants, such as Search Operators or Lock Types
- oracle.ifs.fdk.FdkErrorCodes – Constants that represent possible error codes received through an FdkException
- oracle.ifs.fdk.Options – Defines various option name constants
- oracle.ifs.fdk.SystemConfigurationProperties – Defines applicable system properties that can be requested

 **Tip** – A HTTP client is required in order to upload and download content from Oracle Content Services. Oracle ships an HTTP client with Oracle Application Server, based on the Innovation HTTP client. This client (http_client.jar) can be found under the `ORACLE_HOME/j2ee/home/lib` directory.

The Apache Jakarta Commons project also offers a flexible and powerful HTTP client library as part of the HttpClient component. This component expands core functionality bundled in the JDK's java.net package. The following libraries are required in order to use the Apache HttpClient component (<http://jakarta.apache.org/commons/httpclient/>):

- commons-httpclient-3.0[-rc3 or newer].jar
- commons-logging-1.0.3.jar
- commons-codec-1.3.jar
- junit.jar

CODE EXAMPLES

The Web Services Development Kit shipped with Oracle Content Services provides a number of useful examples and guides, including:

- Java documentation (javadoc) for the Oracle Content Services Axis Java Stubs
- Oracle Content Services Application Developer's Guide
- Basic Web Services Java samples, demonstrating core features and functionality
- BPEL examples that leverage Oracle Content Services Web Services
- Command-line tools for bulk loading of Libraries and Groups

In addition to this kit, product management has released a Web Services accelerator that includes JDeveloper support, along with a framework for rapidly building and prototyping Oracle Content Services Web Services applications. This kit is more optimized for production/real-world use, although it is more complex from a readability perspective. It contains:

- Session management code similar to what is available for in-process clients (such as the Web UI)
- Methods to quickly establish sessions by supplying either user credentials, or trusted application credentials along with a proxy user
- Example code and an associated readme that demonstrates how to create a trusted application to leverage service to service authentication
- A number of utility methods that simplify processing of Web Services requests/responses from Oracle Content Services
- A number of additional Web Services samples (see below)

Web Services samples shipped with the PM Web Services accelerator (and in addition to those found in the out-of-box Development Kit) include:

CreateFolderHierarchy.java

Demonstrates:

- Authentication
- Enabling/disabling administration mode
- Obtaining Item properties
- Resolving Items based on path
- Looking up security roles
- Applying security roles through security configuration
- Creating containers, Libraries, and folders

TestCategories.java and TestCategories2.java

Demonstrates:

- Creating category class objects
- Enforcing/restricting specific category objects on folders using category configurations
- Verifying category configuration information
- Uploading a document and providing category instance metadata details
- Overriding category configurations along with attribute overrides on subfolders

TestFileOperations.java

Demonstrates:

- Uploading documents and specifying language/character set information for text-based documents
- Obtaining folder listings
- Extracting ZIP files

TestRecordManagement.java

Demonstrates:

- Creating a file plan
- Creating a record series
- Creating a record category and setting disposition phases/stages for the record
- Creating a record category that introduces custom metadata
- Enforcing recordization on a folder using a record configuration and mandatory record category
- Creating a document with supplied custom record metadata under an enforced recordization folder

TestS2S.java

Demonstrates:

- Service-to-Service authentication for a proxied user, using a trusted application entity in Oracle Internet Directory

TestSearch.java and TestSearch2.java

Demonstrates:

- Obtaining root category information from domain item and determining the make-up of the category object
- Creating search expressions
- Searching based on item description
- Searching based on item name and folder restriction
- Searching for items with a custom category instance applied
- Searching for items with a custom category instance that has specific attribute values
- Deleting items and emptying the trash folder

TestSecurity.java

Demonstrates:

- Looking up users
- Looking up roles
- Applying grants to items using security configurations
- Verifying security configuration

TestVersioning.java and TestVersioning2.java

Demonstrates:

- Setting versioning policies on folders using versioning configurations
- Automatic versioning of documents with auto-version labels using roman numerals
- Manual versioning of documents
- Verifying versioning configuration
- Uploading multiple versions of a file in a folder managed by a versioning configuration policy
- Explicit check-out/check-in
- Determining conflict resolution functionality for recoverable error conditions
- Specifying conflict resolution options for a file management operation
- Moving documents

CONCLUSION

The Oracle Content Services Web Services provide any application or business process the ability to store, retrieve, and manage terabytes of unstructured content and associated metadata in a robust, scalable, and highly available content and record management repository contained within a single Oracle 10g database instance. As this paper has shown, the Web Services are both complete and highly feature-rich, enabling Oracle Content Services to be used for all manner of customer, partner, and internal purposes.

Testimony to this is Oracle Workspaces, which, by leveraging the Oracle Content Services Web Services and Service-to-Service authentication, provides a value-added application that has all the same functionality available to it as that of the Oracle Content Services Web UI, but with the ability to contextually integrate this content with mail, calendar, discussions, presence, web conferencing, and instant messaging.

APPENDIX A


This appendix aims to provide the reader with a brief introduction to the technologies (WSDL & SOAP) that make Web Services possible, using actual Oracle Content Services Web Services examples. The first technology outlined is the Web Service Definition Language (WSDL), which describes a Web Service in terms of its operations, signatures, and communication interfaces.

Each WSDL Document is a valid XML file, comprised of five main components:

- `<service>` element – defines the name of the Web Service and its supported bindings. Each binding is referenced through a child `<port>` element that specifies the binding implementation name, along with an address to access the binding.
- `<binding>` elements – a binding element denotes an available concrete implementation of an abstract portType definition. It specifies the supported operations for a chosen transport protocol and binding style. For example, the service may support RPC binding using HTTP for operations X and Y that are abstractly defined by the portType ZZZ element.
- `<portType>` elements – a portType element defines a set of operations supported by the service, each captured in a separate `<operation>` child element. Each `<operation>` element, in turn, can have up to three child elements of its own: `<input>`, `<output>`, and `<fault>`. The input, output, and fault elements specify, by way of a message attribute, the message structure that will be communicated between client and server for the particular operation.
- `<message>` elements – a message element defines a specific logical structure that can be communicated. It is comprised of zero or more child `<part>` elements that specify the makeup of the message. Each `<part>` element defines a name and type, for example, name = "username" and type = "xsd:string". The type of the message `<part>` can either be one of the standard XSD types, such as string, integer, or boolean, or a custom complex type that is defined in the WSDL document through a `<complexType>` element.
- `<types>` element – a container for custom data type structures introduced by the service. The `<complexType>` child element is used to define a specific custom type. Each custom type typically describes its composition thorough a sequence of child elements that specify name, type, and whether the attribute can be null. Array structures are also supported by `complexType` through a `complexContent` restriction element.

On the next few pages, you will find extracts taken from the Oracle Content Services Web Services RemoteLoginManager.wsdl file relevant to a login operation. Descriptions provided below document these extracts.

The RemoteLoginManagerService shown in Extract A1 supports a single binding RemoteLoginManagerSoapBinding that has an end-point URL of `http://host:port/content/ws/RemoteLoginManager` (note that *host* and *port* contain real values in a production instance scenario). The binding uses the HTTP protocol for transport, and has implementation style `rpc`, which indicates that the operations in this binding are Remote Procedure Call (RPC) type operations. Note that HTTP is the only transport mechanism supported by the Oracle Content Services Web Services at this time. The binding exposes the "login" operation, and the input and output elements specify only `<body>` child elements (no `<header>` element). This indicates that the SOAP request/response envelope contains only a SOAP body. The attribute value `use="encoded"` of the body and fault elements indicates that in serializing the SOAP message, the parameters should be encoded using the style specified by the `encodingStyle` attribute (note that `use="literal"` would require no encoding).

 **Tip** – Oracle Content Services currently only supports a SOAP encoding style of **RPC** binding with **encoded** use. This combination is known as Section 5 encoding. Future updates of Oracle Content Services are likely to add support for the **Document** binding style with **literal** use (document/literal).

The `type` attribute set in the binding element indicates the particular portType to which the binding is specific, in this case RemoteLoginManager. The RemoteLoginManager portType describes the login operation in terms of what messages will be communicated (input=request, output=valid response, and fault=exception state). The

parameterOrder attribute of the login operation specifies the order in which the message parameters are to be received (for the request).

The <input> child element of the login operation for the RemoteLoginManager portType specifies that the message to be communicated is defined by the loginRequest message structure. The message element loginRequest supports four parts: username, password, options, and userAttributes. The username and password parts are of standard xsd type "string." The options and userAttributes parts use complex types, defined in the types section of the WSDL file (see Extract A2).

ArrayOfAttributeRequest is a complex type that is comprised of an array of AttributeRequest complex type objects. An AttributeRequest complex type is in turn comprised of two elements: attributeName, which is a standard xsd type "string," and requestedAttributes, which has type ArrayOfAttributeRequest. Due to the recursive nature of this structure, it is capable of growing exponentially and should allow the client to submit some very complex attribute requests.

The <output> child element of the login operation for the RemoteLoginManager portType specifies that the message to be communicated is defined by the loginResponse message structure. The message element loginResponse contains just one part: loginReturn. The loginReturn part is of type ArrayOfNamedValue, which is a complex type comprised of an array of NamedValue complex type objects. A NamedValue complex type is comprised of two elements: name, which is a standard xsd type "string," and value, which has type xsd:anyType.

Tip - If a Web Service defines an operation as capable of requesting or responding with a parameter of type xsd:anyType, at message delivery time, the client or server must specify the true data type of the parameter through an xsi:type attribute.

For our NamedValue complex type, the value attribute could represent a simple standard type, such as the following:

```
<value xsi:type="xsd:string">Matt.Shannon</value>
```

Alternatively, it could represent a complex type, such as the following:

```
<value xsi:type="files:Item" xmlns:files="http://xmlns.oracle.com/content/ws">
  <id>123</id>
  <name>matt</name>
  <type>USER</type>
  <requestedAttributes>...
</value>
```

The <fault> child element of the login operation for the RemoteLoginManager portType specifies that the message to be communicated in case of an exception is defined by the FdkException message structure. The message element FdkException contains just one part: fault. The fault part is of type FdkException, which is a complex type comprised of an errorCode, detailedErrorCode, exceptionEntries, info, and serverStackTraceId. errorCode, detailedErrorCode, and serverStackTraceId are of standard xsd type "string." exceptionEntries and serverStackTraceId both use complex types, with the exceptionEntries type being an array of FdkExceptionEntry. An FdkException message is thus capable of encapsulating the complete exception stack trace returned from a failed server operation.

The complex types listed in Extract A2 make up a number of the core structures used by the Oracle Content Services Web Services. Their flexible and recursive nature make them capable of storing almost any repository object structure. Because of this, you will see these same types used over and over again in the various service managers provided.

Tip - For a complete listing of available service managers along with their supported operations, visit the URL <http://host:port/content/wsdl>, with host and port values replaced with those of your own Oracle Content Services instance.

Extract A1 from RemoteLoginManager.wsdl showing service, binding, portType, and message elements. Note: the ordering of sections has been modified from that of the original WSDL document for better readability. In addition, information pertaining to the logout operation has been removed.

```

<wsdl:definitions targetNamespace="http://xmlns.oracle.com/content/ws"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:impl="http://xmlns.oracle.com/content/ws"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
>

  <wsdl:service name="RemoteLoginManagerService">
    <wsdl:port name="RemoteLoginManager" binding="impl:RemoteLoginManagerSoapBinding">
      <wsdlsoap:address location="http://host:port/content/ws/RemoteLoginManager"/>
    </wsdl:port>
  </wsdl:service>

  <wsdl:binding name="RemoteLoginManagerSoapBinding" type="impl:RemoteLoginManager">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="login">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="loginRequest">
        <wsdlsoap:body use="encoded" namespace="http://xmlns.oracle.com/content/ws"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
      </wsdl:input>
      <wsdl:output name="loginResponse">
        <wsdlsoap:body use="encoded" namespace="http://xmlns.oracle.com/content/ws"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
      </wsdl:output>
      <wsdl:fault name="FdkException">
        <wsdlsoap:fault name="FdkException" use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://xmlns.oracle.com/content/ws"/>
      </wsdl:fault>
    </wsdl:operation> . . .
  </wsdl:binding>

  <wsdl:portType name="RemoteLoginManager">
    <wsdl:operation name="login" parameterOrder="username password options userAttributes">
      <wsdl:input name="loginRequest" message="impl:loginRequest"/>
      <wsdl:output name="loginResponse" message="impl:loginResponse"/>
      <wsdl:fault name="FdkException" message="impl:FdkException"/>
    </wsdl:operation> . . .
  </wsdl:portType>

  <wsdl:message name="loginRequest">
    <wsdl:part name="username" type="soapenc:string"/>
    <wsdl:part name="password" type="soapenc:string"/>
    <wsdl:part name="options" type="impl:ArrayOfNamedValue"/>
    <wsdl:part name="userAttributes" type="impl:ArrayOfAttributeRequest"/>
  </wsdl:message>
  <wsdl:message name="loginResponse">
    <wsdl:part name="loginReturn" type="impl:ArrayOfNamedValue"/>
  </wsdl:message>
  <wsdl:message name="FdkException">
    <wsdl:part name="fault" type="impl:FdkException"/>
  </wsdl:message> . . .

```

Extract A2 from RemoteLoginManager.wsdl showing a subset of complex types used by the service.

```
<wsdl:types> <schema . . .
  <complexType name="NamedValue">
    <sequence>
      <element name="name" nillable="true" type="soapenc:string"/>
      <element name="value" nillable="true" type="xsd:anyType"/>
    </sequence>
  </complexType>
  <complexType name="ArrayOfNamedValue">
    <complexContent> <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="impl:NamedValue[]"/>
    </restriction> </complexContent>
  </complexType>
  <complexType name="AttributeRequest">
    <sequence>
      <element name="attributeName" nillable="true" type="soapenc:string"/>
      <element name="requestedAttributes" nillable="true"
        type="impl:ArrayOfAttributeRequest"/>
    </sequence>
  </complexType>
  <complexType name="ArrayOfAttributeRequest">
    <complexContent> <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="impl:AttributeRequest[]"/>
    </restriction> </complexContent>
  </complexType>
  <complexType name="FdkExceptionEntry">
    <sequence>
      <element name="detailedErrorCode" nillable="true" type="soapenc:string"/>
      <element name="errorCode" nillable="true" type="soapenc:string"/>
      <element name="id" type="xsd:long"/>
      <element name="info" nillable="true" type="impl:ArrayOfNamedValue"/>
      <element name="serverStackTraceId" nillable="true" type="soapenc:string"/>
    </sequence>
  </complexType>
  <complexType name="ArrayOfFdkExceptionEntry">
    <complexContent> <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="impl:FdkExceptionEntry[]"/>
    </restriction> </complexContent>
  </complexType>
  <complexType name="FdkException">
    <sequence>
      <element name="detailedErrorCode" nillable="true" type="soapenc:string"/>
      <element name="errorCode" nillable="true" type="soapenc:string"/>
      <element name="exceptionEntries" nillable="true"
        type="impl:ArrayOfFdkExceptionEntry"/>
      <element name="info" nillable="true" type="impl:ArrayOfNamedValue"/>
      <element name="serverStackTraceId" nillable="true" type="soapenc:string"/>
    </sequence>
  </complexType>
  <complexType name="Item">
    <sequence>
      <element name="id" type="xsd:long"/>
      <element name="name" nillable="true" type="soapenc:string"/>
      <element name="requestedAttributes" nillable="true" type="impl:ArrayOfNamedValue"/>
      <element name="type" nillable="true" type="soapenc:string"/>
    </sequence>
  </complexType>
  . . .
</schema> </wsdl:types>
```

SOAP

Once the client has discovered the service and has processed the WSDL document, it is now equipped with the information necessary to determine, for a particular transport protocol and binding style, valid operations that are offered by the service along with the message structures that will be communicated. The message payloads are encoded in XML using the language-neutral format SOAP. WC3 defines SOAP as providing a "simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML." (See <http://www.w3.org/2000/xmlsoap/Group/1/08/29/soap12-part1.html>)

SOAP essentially specifies a mechanism for serializing an application-defined datatype, packaging instances of these datatypes along with RPC information in a SOAP envelope, and how to transport the SOAP envelope using a specific protocol binding.

Tip – HTTP is the only protocol supported by Oracle Content Services for transporting SOAP messages. SOAP over HTTP is by far the most common mechanism for transporting message payloads, and has the added benefit of being able to pass through corporate firewalls and proxies that allow HTTP data.

Extract A3 shows an example SOAP message depicting a login operation request with associated attribute requests for the user's personal workspace and associated path, and the URL to the user's domain object.

The outermost element of a SOAP Message is an Envelope element qualified to the SOAP namespace <http://schemas.xmlsoap.org/soap/envelope/> (SOAP 1.1). There is only one SOAP Envelope element per SOAP Message. A SOAP Envelope must contain a Body element and optionally, a Header element. These Body and Header elements are also qualified to the SOAP namespace. The SOAP Body is used to convey either details of the requested operation and the server response, or alternatively SOAP fault information.

Working through the example listed in Extract A3, we see that our SOAP Body contains a `login` element qualified to the Oracle Content Services Web Services namespace <http://xmlns.oracle.com/content/ws>. It is no coincidence that the child elements of this `login` element match those defined by the `loginRequest` message structure present in the `RemoteLoginManager.wsdl` document. The `username` and `password` elements are both defined as having standard type XSD string with values "matt" and "welcome1," respectively. The `options` element sets the attribute `xsi:nil="True"` which indicates to the server that it should use a value of null for this message parameter. The `userAttributes` element specifies that it is an array of type `AttributeRequest` qualified to the Oracle Content Services Web Services namespace, and that its length is two. Note: the complex type `AttributeRequest` is defined in the types section of the `RemoteLoginManager.wsdl`. We are requesting that the server return the URL of the user's `DOMAIN` object, along with the user's `PERSONAL_WORKSPACE`, and the `PATH` to the `PERSONAL_WORKSPACE`. `PATH` is essentially a chained attribute of user by way of the `PERSONAL_WORKSPACE` attribute.

Tip – Users, Documents, Folders, and other objects are modeled by Oracle Content Service Web Services using the complex type `Item`. As seen in the WSDL, any returned instance of `Item` will include by default the `id`, `name`, and `type` of the underlying object concerned. Thus, it is not necessary for an operation that returns `Items` to explicitly request `ID`, `NAME`, or `TYPE` attributes. The only time you may wish to request these attributes explicitly is when using a specific type of chaining that is discussed in the attribute chaining section of this paper. A complete listing of valid item types can be found in the `oracle.ifs.fdk.ItemTypes` class that ships with the Web Service Java stubs library.

Extract A3 – example SOAP message for a login operation. To minimize network bandwidth, request only attributes needed by the client application. If none are required, set `xsi:nil="true"` on the `requestedAttributes` element.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>
  <soapenv:Body>
    <ns1:login
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="http://xmlns.oracle.com/content/ws">
      >
        <username xsi:type="soapenc:string">matt</username>
        <password xsi:type="soapenc:string">welcome1</password>
        <options xsi:type="ns1:ArrayOfNamedValue" xsi:nil="true"/>
        <userAttributes soapenc:arrayType="ns1:AttributeRequest[2]"
          xsi:type="soapenc:Array"
        >
          <userAttributes href="#id0"/>
          <userAttributes xsi:type="ns1:AttributeRequest"/>
            <attributeName xsi:type="soapenc:string">DOMAIN:URL</attributeName>
            <requestedAttributes xsi:type="ns1:AttributeRequest" xsi:nil="true"/>
          </userAttributes>
        </userAttributes>
      </ns1:login>
      <multiRef id="id0" soapenc:root="0"
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xsi:type="ns2:AttributeRequest" xmlns:ns2="http://xmlns.oracle.com/content/ws"
      >
        <attributeName xsi:type="soapenc:string">PERSONAL_WORKSPACE</attributeName>
        <requestedAttributes soapenc:arrayType="ns2:AttributeRequest[1]"
          xsi:type="soapenc:Array">
          <requestedAttributes href="#id1"/>
        </requestedAttributes>
      </multiRef>
      <multiRef id="id1" soapenc:root="0"
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xsi:type="ns3:AttributeRequest"
        xmlns:ns3="http://xmlns.oracle.com/content/ws"
      >
        <attributeName xsi:type="soapenc:string">PATH</attributeName>
        <requestedAttributes xsi:type="ns3:AttributeRequest" xsi:nil="true"/>
      </multiRef>
    </soapenv:Body>
  </soapenv:Envelope>
```

The actual HTTP request would look something along the lines of the following:

```
POST /files/ws/RemoteLoginManager HTTP/1.1
Content-Type: text/xml
Host: linux1.oraclebox.com:7777
Content-Length: XXX
<soapenv:Envelope ... >
  <soapenv:Body ... </soapenv:Body>
</soapenv:Envelope>
```

The result of the preceding SOAP login request (in other words, the server's SOAP response) is listed in Extract A4. As is the case with all SOAP messages, the response begins with Envelope and Body elements, respectively. The client on receipt of this message would first process the SOAP Body to check for the existence of a Fault element qualified to the SOAP namespace. If none exists, it means the server was happy with the original SOAP request, and thus the SOAP Body must contain a message structure as defined by the output element in the WSDL document for the particular operation concerned. In our example, the SOAP Body must contain a loginReponse element qualified to the Oracle Content Services Web Services namespace, containing a child element loginReturn that is of complex type ArrayOfNamedValue. The loginReturn element shown in Extract A4 specifies that it is an array of type NamedValue, and that its length is three. The child elements of loginReturn (which should be NamedValue instances) have been encoded as multi-ref accessors. These multi-ref accessors have a single attribute "href" that contains a fragment identifier to an independent element containing the serialized NamedValue instance. The SOAP response, once processed, conveys the following information:

```

LOGIN_USER (Item)
  id (long)          8808
  name (string)      matt
  type (string)      USER
  requestedAttributes (NamedValue[])
    DOMAIN:URL (string) http://acme.com:7777/content/dav/oracle
    PERSONAL_WORKSPACE (Item)
      id (long)          13573
      name (string)      matt
      type (string)      WORKSPACE
      requestedAttributes (NamedValue[])
        PATH (string)    /oracle/users/Users-M/matt
SESSION_TIMEOUT (int) 1800
TRANSACTION_TIMEOUT (int) 120

```

The SOAP response contains two Item instances: the first one is the actual USER who connected, and the second one is the requested attribute PERSONAL_WORKSPACE of the USER instance. As you can see, the id, name, and type values are automatically supplied for Item instances.

Extract A4 – example SOAP response message resulting from the login operation request that was shown in extract A3.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>
  <soapenv:Body>
    <ns1:loginResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="http://xmlns.oracle.com/content/ws"
    >
      <loginReturn soapenc:arrayType="ns1:NamedValue[3]" xsi:type="soapenc:Array">
        <loginReturn href="#id0"/>
        <loginReturn href="#id1"/>
        <loginReturn href="#id2"/>
      </loginReturn>
    </ns1:loginResponse>
    <multiRef id="id1" soapenc:root="0"
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xsi:type="ns2:NamedValue" xmlns:ns2="http://xmlns.oracle.com/content/ws"
    >
      <name xsi:type="xsd:string">SESSION_TIMEOUT</name>
      <value xsi:type="soapenc:int">1800</value>
    </multiRef>
    <multiRef id="id2" soapenc:root="0"
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xsi:type="ns3:NamedValue" xmlns:ns3="http://xmlns.oracle.com/content/ws"
    >
      <name xsi:type="xsd:string">TRANSACTION_TIMEOUT</name>
      <value xsi:type="soapenc:int">120</value>
    </multiRef>
    <multiRef id="id0" soapenc:root="0"
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xsi:type="ns4:NamedValue" xmlns:ns4="http://xmlns.oracle.com/content/ws"
    >
      <name xsi:type="xsd:string">LOGIN_USER</name>
      <value href="#id3"/>
    </multiRef>
    <multiRef id="id3" soapenc:root="0"
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xsi:type="ns5:Item" xmlns:ns5="http://xmlns.oracle.com/content/ws"
    >
      <id xsi:type="xsd:long">8808</id>
      <name xsi:type="xsd:string"> matt</name>
      <requestedAttributes soapenc:arrayType="ns5:NamedValue[2]" xsi:type="soapenc:Array">
        <requestedAttributes href="#id4"/>
        <requestedAttributes href="#id5"/>
      </requestedAttributes>
      <type xsi:type="xsd:string">USER</type>
    </multiRef>
    <multiRef id="id5" soapenc:root="0"
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xsi:type="ns6:NamedValue" xmlns:ns6="http://xmlns.oracle.com/content/ws"
    >
      <name xsi:type="xsd:string">PERSONAL_WORKSPACE</name>
      <value href="#id6"/>
    </multiRef>
    <multiRef id="id4" soapenc:root="0"
```

```

    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xsi:type="ns7:NamedValue" xmlns:ns7="http://xmlns.oracle.com/content/ws"
  >
    <name xsi:type="xsd:string">DOMAIN:URL</name>
    <value xsi:type="soapenc:string">http://acme.com:7777/content/dav/oracle</value>
  </multiRef>
  <multiRef id="id6" soapenc:root="0"
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xsi:type="ns8:Item" xmlns:ns8="http://xmlns.oracle.com/content/ws"
  >
    <id xsi:type="xsd:long">13573</id>
    <name xsi:type="xsd:string">matt</name>
    <requestedAttributes soapenc:arrayType="ns8:NamedValue[1]" xsi:type="soapenc:Array">
      <requestedAttributes href="#id7"/>
    </requestedAttributes>
    <type xsi:type="xsd:string">WORKSPACE</type>
  </multiRef>
  <multiRef id="id7" soapenc:root="0"
    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xsi:type="ns9:NamedValue" xmlns:ns9="http://xmlns.oracle.com/content/ws"
  >
    <name xsi:type="xsd:string">PATH</name>
    <value xsi:type="soapenc:string">/oracle/users/Users-M/matt</value>
  </multiRef>
</soapenv:Body>
</soapenv:Envelope>

```

APPENDIX B

Out-Of-The-Box Roles

Administration Role	Permissions	Applicable to (Domain, Container, Workspace)			Propogating
CategoryAdministrator	Discover, AdministerCategory	D			False
ConfigurationAdministrator	Discover, AdministerConfiguration	D	C	W	True
ContainerAdministrator	Discover, AdministerContainer, CreateContainer	D	C		True
ContentAdministrator	Discover, AddItem, AddVersion, Copy, CreateFolder, Delete, GetContent, GetMetadata, Lock, Move, SetAttribute, SetContent, SetMetadata	D	C	W	True
DomainAdministrator	Discover, AdministerDomain	D			False
QuotaAdministrator	Discover, AdministerQuota	D	C	W?	True
RecordsAdministrator	Discover, AdministerRecord	D			False
RoleAdministrator	Discover, AdministerRole	D			False
SecurityAdministrator	Discover, AdministerSecurity	D	C	W	True
UserAdministrator	Discover, AdministerUser	D			False
WorkspaceAdministrator	Discover, AdministerWorkspace, CreateWorkspace	D	C	W	True

Standard (non-administration) Role	Permissions	Applicable to (Domain, Container, Workspace)			Propogating
None	NONE				False
AdministrativeAssistant	Discover, AddItem, AdministerConfiguration, AdministerSecurity, CreateFolder			W	False
Administrator	Discover, AddItem, AddVersion, AdministerConfiguration, AdministerSecurity, AdministerWorkspace, Copy, CreateFolder, Delete, GetContent, GetMetadata, Lock, Move, SetAttribute, SetContent, SetMetadata			W	True
Approver	Discover, Copy, GetContent, GetMetadata, Lock, SetAttribute, SetContent, SetMetadata			W	False
Author	Discover, AddItem, AddVersion, Copy, CreateFolder, Delete, GetContent, GetMetadata, Lock, Move, SetAttribute, SetContent, SetMetadata			W	False
Commentator	Discover, Copy, GetContent, GetMetadata, Lock, SetAttribute, SetContent, SetMetadata			W	False
ContainerViewer	Discover	D	C		False
ContentEditor	Discover, AddItem, AddVersion, Copy, CreateFolder, GetContent, GetMetadata, Lock, SetAttribute, SetContent, SetMetadata			W	False
Custodian	Discover, AddItem, Copy, CreateFolder, Delete, GetMetadata, Lock, Move, SetAttribute, SetMetadata			W	False
Discoverer	Discover			W	False

Standard (non-administration) Role	Permissions	Applicable to (Domain, Container, Workspace)			Propogating
Limited Author	Discover, AddItem, AddVersion, Copy, CreateFolder, GetContent, GetMetadata, Lock, SetAttribute, SetContent, SetMetadata			W	False
Manager	Discover, AddItem, AdministerSecurity, CreateFolder			W	False
Organizer	Discover, Copy, Delete, GetMetadata, Lock, Move, SetAttribute, SetMetadata			W	False
Reader	Discover, Copy, GetContent, GetMetadata			W	False
Reviewer	Discover, Copy, GetContent, GetMetadata			W	False
WorkspaceCreator	Discover, CreateWorkspace	D	C		True

Role	FdkConstant
CategoryAdministrator	ECM_ROLEKEY_CATEGORYADMIN
ConfigurationAdministrator	ECM_ROLEKEY_CONFIGURATIONADMIN
ContainerAdministrator	ECM_ROLEKEY_CONTAINERADMIN
ContentAdministrator	ECM_ROLEKEY_CONTENTADMIN
DomainAdministrator	ECM_ROLEKEY_DOMAINADMIN
QuotaAdministrator	ECM_ROLEKEY_QUOTAADMIN
RecordsAdministrator	ECM_ROLEKEY_RECORDSADMIN
RoleAdministrator	ECM_ROLEKEY_ROLEADMIN
SecurityAdministrator	ECM_ROLEKEY_SECURITYADMIN
UserAdministrator	ECM_ROLEKEY_USERADMIN
WorkspaceAdministrator	ECM_ROLEKEY_WORKSPACEADMIN
ContainerViewer	ECM_ROLEKEY_CONTAINERVERIEWER
WorkspaceCreator	ECM_ROLEKEY_WORKSPACECREATOR
Administrator	ECM_ROLEKEY_ADMIN

Permission	Capability to	FdkConstant
Discover	discover an item and view its basic metadata (such as name, description, and createdate). Permission is implicit should the user be granted any other permission on the item.	CAPABILITY_DISCOVER
SetAttribute	set basic attributes of an item (like description). Permission is required to rename Document, Folder, Family, and Link items. For link items, this permission also provides the ability to change the object to which the link refers.	CAPABILITY_SET_ATTR
GetContent	get the content of a document item	CAPABILITY_GET_CONTENT
SetContent	set the content of a non-version-controlled document item	CAPABILITY_SET_CONTENT
Delete	delete an item	CAPABILITY_DELETE
Lock	lock a document item	CAPABILITY_LOCK
AddItem	add an item to a folder (by a create or move operation)	CAPABILITY_ADDITEM
AddVersion	add a new version to a version-controlled document item	CAPABILITY_ADDVERSION
CreateFolder	create a folder	CAPABILITY_CREATEFOLDER

Permission	Capability to	FdkConstant
AdministerDomain	modify a domain's (Site's) properties (such as ECM_DOMAIN_SECURITY_WORLDGRANTSEENABLED)	CAPABILITY_ADMINISTER_DOMAIN
AdministerUser	modify or delete a domain's users and groups. Additionally, this permission enables you to get/set user preferences, including domain defaults.	CAPABILITY_ADMINISTER_USER
AdministerCategory	create, modify, or delete an ad-hoc category class object	CAPABILITY_ADMINISTER_CATEGORY
AdministerRecord	create, modify, or delete a record file plan. Additionally, this permission enables you to unrecordize an existing record item and perform a variety of other RM administration activities.	CAPABILITY_ADMINISTER_RECORD
AdministerRole	update out-of-box role descriptions. (custom role functionality will be officially supported in a later release)	CAPABILITY_ADMINISTER_ROLE
AdministerContainer	modify or delete a container. (permission is required on the parent item of the container that is to be modified/deleted, in other words, on the domain or a parent container)	CAPABILITY_ADMINISTER_CONTAINER
CreateContainer	create a container	CAPABILITY_CREATECONTAINER
AdministerQuota	modify a workspace item's quota configuration	CAPABILITY_ADMINISTER_QUOTA
AdministerWorkspace	modify or delete a workspace	CAPABILITY_ADMINISTER_WORKSPACE
CreateWorkspace	create a workspace (not needed for creation of a personal workspace)	CAPABILITY_CREATEWORKSPACE
AdministerSecurity	create, modify, or delete security configuration of an item	CAPABILITY_ADMINISTER_SECURITY
AdministerConfiguration	create, modify, or delete configuration categories on an item (with exception of SecurityConfiguration and QuotaConfiguration)	CAPABILITY_ADMINISTER_CONFIGURATION
Move	Move an item. Note: also requires AddItem permission on the destination folder.	CAPABILITY_MOVE
GetMetadata	get the metadata of an item (in other words, view category information)	CAPABILITY_GETMETADATA
SetMetadata	set the metadata of an item (in other words, create, modify, or delete category information)	CAPABILITY_SETMETADATA
Copy	copy an item	CAPABILITY_COPY

Tips –

- Within a single grant, the same role may not appear more than once.
- Within a single grant, the "NONE" role may not be combined with any other role.
- If the grantee is the "World" group, the domain must be enabled for "world group grants."
- SetAttribute permission is required to rename a Document, Folder, Family, or Link.
- AdministerWorkspace permission is required to rename a Workspace.
- AdministerContainer permission is required to rename a container.

- System Admin privileges are required to rename a domain.
- The SecurityAdministrator role is the most powerful, because users with this role can grant themselves or anybody else all available access.
- In order to delete a container, the user must have AdministerContainer permission on the parent of the container being deleted, for example, by having the ContainerAdministrator role and switching into administration mode.
- When deleting a Container, all recursively contained sub-containers and sub-workspaces are also deleted. The user has to have permission to delete the sub-containers according to the rule stated above. The user must also have AdministerWorkspace permission on all of the sub-workspaces that are to be deleted. If the user does not have these required permissions, the originating container delete will fail with an ACCESS_DENIED exception. Once a container is deleted, it is deleted permanently; deleted workspaces have the workspace's contents moved to the Archive.



Oracle Content Services 10g Web Services Technical Paper

September 2005

Author: Matt Shannon – Oracle Australia

Contributing Author: Marla Azriel

Special Thanks: Simon Azriel, Dragos Harabor, Vasant Kumar, Dave Long, Luis Saenz, Sean Trabosh

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Copyright © 2005, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.