# Oracle® Fusion Middleware

Content Management REST Service Developer's Guide

11*g* Release 1 (11.1.1)

**E15813-05**

January 2011

## 1 Introduction

The OASIS CMIS (Content Management Interoperability Services) Technical Committee works to standardize a web services interface specification that will enable greater interoperability of Enterprise Content Management (ECM) systems. For more information, see the Oasis CMIS site:
`http://www.oasis-open.org/committees/cmis/`

The Content Management REST Service provides a server that uses the CMIS RESTful AtomPub server binding to provide access to Oracle Content Server repositories configured in your application.

This guide is intended to be a supplement to the OASIS CMIS specification, and provides details on the specific implementation of the Content Management REST Service. Before continuing, all users should review the OASIS CMIS specification. This guide references the Content Management Interoperability Services (CMIS) Version 1.0, which can be viewed at the following URL:
`http://docs.oasis-open.org/cmis/CMIS/v1.0/cmis-spec-v1.0.html`.

The specification includes the domain model and two server bindings. As mentioned above, only the RESTful AtomPub binding is currently implemented by the Content Management REST Service. Users should be familiar with Atom and AtomPub, as these are the default formats for responses.

**Note:** CMIS provides a lowest common denominator for a wide range of different content systems; it is not aligned directly with the Oracle Content Server functionality. Refer to the CMIS service document to identify the available functionality.

This guide includes the following sections:

- Section 2, "CMIS Part I - Domain Model"
- Section 3, "CMIS Part II: RESTful AtomPub Binding"
- Section 4, "Content Management REST Service Best Practices and Examples"

ORACLE®

# 2 CMIS Part I - Domain Model

The Domain Model part of the CMIS specification defines a domain model that can be used by applications to work with one or more Content Management repositories/systems.

This section is organized according to the sections in the CMIS Domain Model documentation.

- Section 2.1, "Data Model"
- Section 2.2, "Services"

## 2.1 Data Model

The Content Management REST Service service document consists of AtomPub workspaces. Each workspace maps to a content connection (only UCM repositories are supported by the Content Management REST Service). For details on the service document, see the next section, Section 3, "CMIS Part II: RESTful AtomPub Binding".

### 2.1.1 Repository

For this release, some of the optional capabilities listed in section 2.1.1 have not been implemented. Versioning, ACL, Policies, Relationships, Change Log, Folder Descendants/Tree, and Renditions will be considered for future releases.

Specifically, the Content Management REST Service implementation has the following optional capabilities:

```
capabilityGetDescendants = true
capabilityGetFolderTree = false
capabilityContentStreamUpdatability = anytime
capabilityChanges = none
capabilityRenditions = none
capabilityMultifiling = false
capabilityUnfiling = false
capabilityVersionSpecificFiling = false
capabilityPWCUpdateable = false
capabilityPWCSearchable = false
capabilityAllVersionsSearchable = false
capabilityJoin = none
capabilityACL = none
capabilityQuery = none, metadataonly, or both combined
```

### 2.1.2 Object

Content Management REST Service supports document and folder objects. In CMIS the `cmis:baseTypeId` for a Node will be `cmis:folder` or `cmis:document`. Also, the `cmis:baseId` for a Type will be `cmis:folder` or `cmis:document`.

### 2.1.3 Object-Type

A CMIS Object-Type contains fields mapped from the UCM Content Server metadata field definitions and UCM Content Server SiteStudio region definitions.

The mapping from UCM Content Server metadata fields to CMIS property definitions is as follows:

- TEXT metadata field with option list configured with select list validated and YesNoView or TrueFalseView view: cmis:propertyBoolean

- All other TEXT metadata fields: cmis:propertyString
- LONG TEXT metadata field: cmis:propertyString
- MEMO metadata field: cmis:propertyString
- INTEGER metadata field: cmis:propertyInteger
- DATE metadata field: cmis:propertyDateTime
- DECIMAL metadata field: cmis:propertyDecimal

The mapping from UCM Content Server SiteStudio Region Definition fields to CMIS property definitions is as follows:

- Image Element Definition fields: cmis:propertyString
- WYSIWYG Element Definition fields: cmis:propertyString
- Plain Text Element Definition fields: cmis:propertyString
- Static List Element Definition fields: cmis:propertyString

### 2.1.4 Document Object

Document Objects are the elementary information entities managed by the repository. As defined by the CMIS specification, Document Objects may be version-able, file-able, query-able, control-able and ACLControl-able. As stated earlier, the Content Management REST Service does not support versioning, multi-filing, Policies or ACL for this release.

If a Node is determined to be a Document (not a Folder) then any children it has will not be exposed through CMIS. In CMIS, each Document Object is associated with a single content stream, and for WebCenter CMIS REST, this stream is the Oracle Content Server binary associated with the document.

### 2.1.5 Folder Object

The CMIS specification states that Folder Objects do not have a content-stream and are not version-able. If a Node is determined to be a Folder, then the Content Management REST Services exposes it in this manner. (In UCM, folders do not have a content stream and are not versionable).

### 2.1.6 Relationship Object

The Relationship Object section does not apply, since the Content Management REST Service does not support Relationships for this release.

### 2.1.7 Policy Object

The Policy Object section does not apply, since the Content Management REST Service does not support Policies for this release.

### 2.1.8 Access Control

Most of the Access Control section does not apply, since the Content Management REST Service does not support ACL for this release. See below for details on allowable actions.

**2.1.8.1 AllowableActions Mapping** This section lists allowable actions that will be defined for Objects. Because of how this release is implemented, some of these are hard-coded for all objects. Other allowable actions will be set based on the repository configuration.

- canGetObjectRelationships = false
- canCreateRelationship = false
- canGetDescendants = false
- canGetFolderTree = false
- canCheckOut = false (versioning)
- canCancelCheckOut = false (versioning)
- canCheckIn = false (versioning)
- canAddObjectToFolder = false (multi-filing)
- canRemoveObjectFromFolder = false (unfiling/multi-filing)
- canApplyPolicy = false
- canGetAppliedPolicies = false
- canRemovePolicy = false
- canCreatePolicy = false
- canApplyACL = false
- canGetACL = false
- canGetRenditions = false
- canDeleteTree = true
- canGetAllVersions = false (versioning)

### 2.1.9 Versioning

Section 2.10 does not apply, since the Content Management REST Service does not support versioning for this release.

### 2.1.10 Query

CMIS queries return a Result Set where each Entry object will contain only the properties that were specified in the query. As the Content Management REST Service does not support JOINs in queries, each result entry will represent properties from a single node.  Common searches use a query like "SELECT * FROM …".

- The FROM clause specifies a content-type to be searched.
  - FROM cmis:document ==> any UCM document (for example, IDC:GlobalProfile)
  - FROM cmis:folder ==> any UCM folder (for example, IDC:Folder)
  - FROM typeQueryName ==> type's cmis queryName, as long as the type is queryable (for example, ora:t:IDC!;GlobalProfile)
- The cmis:document and cmis:folder types are always queryable. Other types will be queryable if they are searchable in the repository.
- The IN_FOLDER predicate is implemented as the folder ID specified, being the parent of the results.
- The IN_TREE predicate is implemented as the folder ID specified, being a parent in the folder structure of the results.
- The CONTAINS() predicate is a full-text query expression operator.

- Properties of cmis:document and cmis:folder will be queryable and orderable if their corresponding UCM system property is searchable and sortable. The system property mappings are:
  - cmis:createdBy ==> dDocAuthor
  - cmis:lastModifiedBy ==> dDocCreator
  - cmis:creationDate ==> dCreateDate
  - cmis:lastModificationDate ==> dLastModifiedDate (for 10g, folders map to dLastModifiedDate and documents map to dCreateDate)
  - cmis:name ==> dOriginalName (for a document) or dCollectionName (for a folder)
  - cmis:contentStreamFileName ==> dOriginalName
  - cmis:contentStreamLength ==> VaultFileSize
  - cmis:contentStreamMimeType ==> dFormat
  - cmis:objectId ==> dDocName
  - cmis:objectTypeId ==> UCM profile name or SiteStudio Region Definition name

  **Note:** cmis:objectTypeId is never orderable.

  - cmis:path ==> use IN_FOLDER or IN_TREE predicate

  **Note:** Some repositories may have capabilities that are not representable in a CMIS query, and some repositories may have restrictions which will limit the CMIS-query predicates (or combinations of predicates) that can be used in a query. The above mappings should assist you in translating repository capabilities and restrictions into corresponding considerations for CMIS queries.

- Nested properties are not queryable or orderable.
- The Content Management REST Service implementation reports as orderable any properties which UCM specifies as sortable. This list can in some cases include properties which UCM cannot actually sort on. If you wish to allow ordering on a field for which UCM is reporting a sort error, follow the steps below to make the specified UCM field sortable:
  1. Go to Administration and open Admin Applets.
  2. Open the Configuration Manager applet and click Advanced Search Design...
  3. Edit the field you wish to make orderable and select 'Is sortable'.
  4. Save your changes and exit Administration.

***Table 1 Search Considerations and Recommendations***

| Consideration | Recommendation |
|---|---|
| UCM provides limited support for querying on null or non-null values. | Be aware of the differences in search behavior and do not write search expressions that depend on unsupported criteria. |
| Recursive search for folders is not supported by UCM, but is supported for documents if configured on the UCM server as described to the right. | Scope the search to only include documents (add a select clause like "select * from cmis:document").<br><br>Set the search path on the Search object (add a where clause like "where IN_ TREE('/StellentRepository/IDC:Folder/2').<br><br>Configure the folders_g CollectiveSearchRecursiveContent and related settings like CollectionMaxBranch in the UCM config.cfg file. |
| Multivalued property operators perform substring matches. This is true for ANY <multiValuedQueryName> IN ( <literal>, ... ) or <literal> = ANY < multiValuedQueryName>. In UCM, a field with an option list stores values in a comma-delimited manner. For example, if you have values "A", "B", and "C", these will be represented as "A, B, C". Using an ANY or ANY IN search for 'A, B' will find this item. | Be aware of the differences in search behavior and consider changing the UCM option list delimiter character in the Configuration Manager applet to reduce the potential for finding extra matches. |
| When searching folders (FROM cmis:folder), at most one value can be specified per criteria. Each criteria is logically ANDed with the others to make a more selective query. There is no support for OR or NOT when searching folders. | There is no support for `OR` and `NOT` in UCM folder search. |
| Not all properties are searchable, and if the search encounters a property that is not searchable, it will return a ParseException (400 error). | Understand which properties are searchable for a given content type by examining the UCM Configuration Manager information fields section, or by reviewing the ContentType definition.<br><br>Example URLS:<br><br>`http://myContentServer/idc/idcplg?IdcService=VCR_GET_CONTENT_TYPE&vcrContentType=IDC:Folder&IsSoap=1`<br><br>then look for the isSearchable field setting.<br>Or, examine the specific type through CMIS. |
| Not all ContentTypes are searchable. An attempt to search for a non-searchable ContentTypes will throw an exception. For example, the IDC:FileReference ContentType is not searchable. | Be aware that not all ContentTypes are searchable. |
| Only String multi-valued properties can be searched. | Do not specify search for multi-valued property types other than String. |
| The `not` operator cannot be used for LONG properties. | Try to restructure the query using supported syntax. |

*Table 1 (Cont.) Search Considerations and Recommendations*

| Consideration | Recommendation |
|---|---|
| Sorting on non-indexed fields results in an exception.<br><br>Searching on a non-indexed field throws an exception, with the embedded exception code of, for example, "DRG-10837: section dStatus does not exist". | Understand which fields have been indexed before using them as sort criteria<br><br>Example: URL: `http://myContentServer/idc/idcplg?IdcService=GET_ADVANCED_SEARCH_OPTIONS&IsSoap=1`, then look for "IsSortable". Or, examine the type through CMIS to see if the property definition is queryable. |
| Empty values are not allowed in a search query. | Do not use a criteria such as `cmis:name != ''`.. |
| `Notequals` operator is not supported for non-String properties | Be aware of the differences in search behavior and do not write search expressions that depend on unsupported criteria. |
| Multiple search paths on the same UCM repository are not supported. | Be aware of the differences in search behavior and do not write search expressions that depend on unsupported criteria. |
| When searching for documents, recursive search (folder tree search) is supported if UCM is configured properly.<br>If search path is not set, then all documents in the repository will be searched (both filed and unfiled). | Configure the content server folders_g CollectionSearchRecursiveContent and related settings like CollectionMaxBranch in the UCM config.cfg file. These are described in UCM documentation. To perform a document search scoped to a folder tree, use the IN_TREE predicate. |
| When searching for documents and using the LIKE operator, wildcards (%) are only supported in the last path element. | Be aware of the differences in search behavior and do not write search expressions that depend upon unsupported criteria. |
| When searching documents (select * from cmis:document), it is not possible to limit the search to more than just a single content type; for example, this is not supported: "select * from IDC:MyProfile, IDC:AnotherProfile" because it has multiple explicit content types and JOINS are not supported. | If you need to limit the search to more than just a single content type, issue multiple queries to achieve the same behavior. If you want to search across all types, use cmis:document in the select statement. |
| UCM search does not support OR when `cmis:objectTypeId` is specified in a query; other parameters can be ANDed with this criteria, but OR is not supported.<br><br>For example, it is not supported to do this: `'cmis:objectTypeId = 'IDC:GlobalProfile' \|\|`<br><br>`myField='bar'` | If this functionality is necessary, issue multiple queries to achieve the same behavior. |
| `cmis:objectTypeId` criteria only supports `==`, `!=`, and `like`. Use of `!=` is restricted to the case of excluding folders (which behaves the same as adding `select * from cmis:document`). | Be aware of the valid operators when using `cmis:objectTypeId` criteria. |
| Queries may be case-sensitive depending on the selected UCM search engine. | Be aware that the UCM search engine selection can affect case-sensitivity. Metadata searches using OracleTextSearch as the search engine are generally case-insensitive. Metadata searches using `DATABASE.FULLTEXT` as the search engine are generally case-sensitive. The exact behavior sometimes varies by metadata field. |

For example queries, see Section 4, "Content Management REST Service Best Practices and Examples".

## 2.2 Services

The methods described in the Services section are implemented by the Content Management REST Service; specific implementation details are covered in the next section, Section 3, "CMIS Part II: RESTful AtomPub Binding".

# 3 CMIS Part II: RESTful AtomPub Binding

The RESTful AtomPub Binding part of the CMIS specification defines a specification based on AtomPub that can be used by applications to work with one or more Content Management Repositories. REST services are available through a WebCenter Spaces instance; for details, see the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter*.

## 3.1 Service Document

All navigation of a repository begins with the AtomPub Service Document. From this document, all accessible content in a repository can be discovered through the collections, links, and templates.

The URI to the service document, relative to the CMIS web application's context-root, is /rest/cmis/repository.

Therefore, if an application is deployed with a library-context-root-override as in the example above, the service document would be accessed through the following URL:

http://hostname:port/rest/cmis/repository

(The REST application is only available with the Spaces application and not with any custom portal application development.)

By default, this document will contain a workspace for each configured UCM repository (only UCM repositories are supported by CMIS REST in Oracle WebCenter). A service document for a single repository can be obtained by using the repositoryId query parameter, as described in section 5 of the CMIS AtomPub binding specification.

As noted in the previous section, the service document consists of AtomPub workspaces. Each workspace maps to a WebCenter Oracle Content Server connection.

Specific URIs beyond the service document are not published; it is assumed that users will start at the service document and navigate the collections and links down expected paths. The relationships of the links and the titles and types of the collections are all defined in the CMIS specification, and thus can be commonly navigated by a client implementation. There are also templates defined for each repository, for easier access of objects by path, object by ID, type by ID, and queries. The format of the variables for the path and ID templates can be discovered by viewing the Entries of Folders and Documents.

## 3.2 Response Formats

*Section 2.2: Response Formats* indicates that Atom/AtomPub style formats will be returned by default unless overridden by a supported media type expressed in the Accept header.

A generic AtomPub feed reader can walk through any of the feeds returned by the CMIS REST server. It will not see all the CMIS specifics, but will be able to navigate through links. In general, to set up a feed reader, you need to know the URI of a particular feed, which can be discovered by navigating through the service document, for example, the workspace link for "typesdescendants".

For details on query syntax, see the CMIS specification. For Content Management REST Service best practices and examples, see the next section, Section 4, "Content Management REST Service Best Practices and Examples".

## 3.3 Additional Functionality

The Content Management REST Service provides the following additional functionality beyond the CMIS specification.

- Section 3.3.1, "Folder Children Collection"
- Section 3.3.2, "Document Entry"
- Section 3.3.3, "Content Stream"

### 3.3.1 Folder Children Collection

The specification defines the following CMIS services:

- GET:  getChildren
- POST: createDocument
  or createFolder
  or createPolicy
  or moveObject
  or addObjectToFolder

The Content Management REST Service also provides the following:

- POST: create
  This new service has five query parameters: uid, fileName, contentId, comments, simpleResponse, and one header parameter: Slug. This new service is meant to be used as a simple binary request upload.  A new document is created with this service. Slug and fileName (optional, though only one needs to be defined and fileName is checked first) are used to name the binary that is attached to the request. The comments parameter is optional and contentId is optional if UCM is set up to auto-generate the dDocName.
- POST: create  Content-Type: multipart/form-data
  This new service has a single query parameter: uid, which is the uid of the folder in which the document is to be created. The boolean query parameter simpleResponse will return a response of media type: application/atom+xml;type=entry, if set to false.  If set to true, a  response of media type: text/html will be returned with a URI pointing to the newly created document.  The comments and simpleResponse parameters are both optional, contentId is optional if UCM is set up to auto-generate the dDocName, and the name "fileUpload" is required.

```
<html>
<head>
    <title>simple post</title>
</head>
<body>
<form
```

```
action="http://<host>:<port>/rest/api/cmis/children/StellentRepository?uid=IDC:
Folder/2"
      method="POST"
      enctype="multipart/form-data">
    Select a document to upload: <input type="file" name="fileUpload"/><br>
     <input type="hidden" name="comments" value="this is just a comment"/>
     <input type="hidden" name="contentId" value="uniqueID1"/>
     <input type="hidden" name="simpleResponse" value="true"/>
    <input type="submit" value="Submit"/>
</form>
</body>
</html>
```

### 3.3.2 Document Entry

The specification defines the following CMIS services:

- GET: getObject, getObjectOfLatestVersion (getObject)
- PUT: updateProperties
- DELETE: deleteObject

The Content Management REST Service also provides the following:

- POST: postToDelete
  This new service has two query parameters: uid and _method, and allows a document to be deleted through POST.

  ```
  http://<host>:<port>/rest/api/cmis/document/repoName?uid=ABC&_method="delete"
  ```

### 3.3.3 Content Stream

The specification defines the following CMIS services:

- GET: getContentStream
- PUT: setContentStream
- DELETE: deleteContentStream

The Content Management REST Service also provides the following:

- POST: postTunnelContentStream
  This new service has five query parameters: uid, overwriteFlag, fileName, comments, _method, and one header parameter: Slug. This new service is meant to be used as a simple binary request upload or delete through POST. A document must already exist for this service. Slug and fileName (optional, though only one needs to be defined and fileName is checked first) are used to name the binary that is attached to the request. The overwriteFlag parameter defaults to true, the comments parameter is optional and _method can be "delete" or "put" (not case sensitive).

  ```
  http://<host>:<port>/rest/api/cmis/stream/repoName?uid=ABC&_method="delete"
  ```

- POST: postTunnelContentStream
  Content-Type: multipart/form-data
  This new service has a single query parameter: uid and is meant to be used as a simple html multipart/form-data upload or delete through POST. A document must already exist for this service. The attribute name="fileUpload" is required, "comments" is optional, and valid values for "_method" are "delete" or "put" (not case sensitive).

```
<form
action="http://<host>:<port>/rest/api/cmis/stream/repoName?uid=WDOC019113"
      method="POST"
      enctype="multipart/form-data">
    Select a document to upload: <input type="file" name="fileUpload"/><br>
     <input type="hidden" name="comments" value="this is just a comment"/>
     <input type="hidden" name="_method" value="PUT"/>
    <input type="submit" value="Submit"/>
</form>
```

# 4 Content Management REST Service Best Practices and Examples

This section provides best practices and examples using the Content Management REST Service. For details on query syntax, see the CMIS specification.

## 4.1 Best Practices

The following list provides suggested best practices for repositories that will use the Content Management REST Service.

- To determine the types that can be used in the "FROM" portion of a query, see the types collection from the AtomPub service document. The type must be queryable and the query name of that type must be used.

  For example, IDC:GlobalProfile might have type information similar to the following:

  ```
  <cmis:localName>IDC:GlobalProfile</cmis:localName>
  <cmis:displayName>IDC:GlobalProfile</cmis:displayName>
  <cmis:queryName>ora:t:IDC!;GlobalProfile</cmis:queryName>
  <cmis:queryable>true</cmis:queryable>
  ```

  An example query for the type information above could be: "SELECT * FROM ora:t:IDC!;GlobalProfile".

- To determine the properties that can be used in the "SELECT" and "WHERE" portions of a query, see the entry for the associated type. Each property definition of that type will be listed and will have a setting for queryable and orderable. The cmis:queryname will be the value to be used in the query.

  For example, IDC:GlobalProfile might have property definition information similar to the following:

  ```
  <cmis:propertyStringDefinition>

  <cmis:id>/stanl18-ucm11g/IDC:GlobalProfile.ora:p:dDocName</cmis:id>
              <cmis:localName>dDocName</cmis:localName>
              <cmis:displayName>dDocName</cmis:displayName>
              <cmis:queryName>ora:p:dDocName</cmis:queryName>
              <cmis:description>Content ID</cmis:description>
              <cmis:propertyType>string</cmis:propertyType>
              <cmis:cardinality>single</cmis:cardinality>
              <cmis:updatability>readwrite</cmis:updatability>
              <cmis:inherited>false</cmis:inherited>
              <cmis:required>false</cmis:required>
              <cmis:queryable>true</cmis:queryable>
              <cmis:orderable>true</cmis:orderable>
            </cmis:propertyStringDefinition>
  ```

An example query for the property definition information above could be: "SELECT ora:p:dDocName FROM ora:t:IDC!;GlobalProfile"

- To keep queries more readable, avoid non-alphanumeric characters in ContentType and PropertyDefinition names.

## 4.2 Content Management REST Service Examples

This section provides some example queries. For details on query syntax, see the CMIS specification. (To get the full URI for a query, see the query URI template in the service document.)

- SELECT * from cmis:folder
- SELECT cmis:name, cmis:contentStreamFileName, cmis:contentStreamMimeType, cmis:contentStreamLength FROM cmis:document WHERE cmis:contentStreamFileName = 'BinaryName' AND cmis:contentStreamMimeType = 'text/html' AND cmis:contentStreamLength > 1
- SELECT cmis:name, cmis:creationDate, cmis:lastModificationDate FROM cmis:folder WHERE cmis:name = 'Trash' AND cmis:lastModificationDate > TIMESTAMP '2008-05-18T10:32:44.703-06:00'
- SELECT * FROM cmis:document WHERE cmis:name LIKE 'baker%'
- SELECT * FROM cmis:document WHERE cmis:name NOT IN ('nodeBoolean', 'nodeLong')
- SELECT cmis:name from cmis:document where IN_TREE('/StellentRepository')
- SELECT * FROM ora:t:IDC:GlobalProfile WHERE ora:p:dRevClassID > 1 ORDER BY ora:p:dDocTitle,ora:p:dInDate DESC
- SELECT * FROM ora:t:IDC:GlobalProfile WHERE ora:p:xBooleanTestField = FALSE ORDER BY ora:p:dDocTitle ASC
- SELECT ora:p:xMultiValuedDelimiterTest FROM ora:t:IDC:GlobalProfile WHERE ANY ora:p:xMultiValuedDelimiterTest NOT IN ('four')
- SELECT cmis:name FROM ora:t:IDC:GlobalProfile WHERE CONTAINS('test') ORDER BY ora:p:dInDate DESC
- SELECT * FROM cmis:document where IN_TREE('/StellentRepository/IDC:Folder/2')

## 5 Documentation Accessibility

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

## Accessibility of Code Examples in Documentation

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

## Accessibility of Links to External Web Sites in Documentation

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

## Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Oracle Fusion Middleware Content Management REST Service Developer's Guide, 11*g* Release 1 (11.1.1)
E15813-05