



# Oracle Knowledge Client Library API Implementation Guide

---

*Using Client Library APIs to Create Custom Integrations Within Oracle Knowledge*

---

Version 8.6

Document Number CLAPI86-IG-00

March 2015

.

## COPYRIGHT INFORMATION

Copyright © 2002, 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS:

Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

### Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

---

# Contents

---

<b>Preface</b>	<b>About This Guide</b>	<b>1</b>
	In This Guide	1
	Screen and Text Representations	2
	References to Web Content	2
	Examples of Product Screens and Text	2
	Operating System Variations in Examples and Procedures	2
<b>Chapter 1</b>	<b>Installing the Client Library API</b>	<b>3</b>
	Information Manager Server Side Installation	3
	Intelligent Search Server Side Installation	4
	Client Side Installation	4
	Java Client Installation	4
	C#.Net Client Installation	5
<b>Chapter 2</b>	<b>Client Library Introduction</b>	<b>6</b>
	Native Data Types	6
	Cross Platform Support	7
	Remote Access	8
	Expose Commonly Used Functionality	8
	Consistent Interface Across Products	9
<b>Chapter 3</b>	<b>Architecture Overview</b>	<b>10</b>
	Service Locator Pattern	10
	Specifying a Global Timeout Value for All Connections in Java Implementations	12
	Specifying a Global Timeout Value for All Connections using C#:	13
	Session Facade Design Pattern	13
	Data Transfer Design Pattern	15

Error Handling .....	16
Serialization .....	17
<b>Chapter 4 Information Manager API Overview .....</b>	<b>19</b>
IQRepositoryRequest .....	19
Related ITOs .....	19
getRepositoryxxxByReferenceKey() .....	19
getRepositoryxxxByID() .....	20
getRepositoryXXXforRepositoryKey() .....	20
IQCategoryRequest .....	21
Related ITOs .....	21
getCategory%MODE%ForCategoryKey (FULL, DATA) .....	21
getCategory%MODE%ForID (FULL, DATA, KEY) .....	22
getCategory%MODE%ByReferenceKey (FULL, DATA, KEY) .....	22
category%MODE%ITOCChildrenForParent (FULL, DATA, KEY) .....	22
getCategory%MODE%ListAssignedToView (FULL, DATA, KEY) .....	22
getRequiredCategory%MODE%ListForChannel (FULL, DATA, KEY) .....	22
getCategory%MODE%ListForChannel (FULL, DATA, KEY) .....	22
addCategory .....	22
deleteCategory .....	23
updateCategories .....	23
IQContentChannelRequest .....	23
Related ITOs .....	24
getContentChannel%MODE%ForContentChannelKey (FULL, DATA) .....	24
getContentChannel%MODE%ByReferenceKey (FULL, DATA, KEY) .....	24
Miscellaneous ContentChannel Service Methods .....	24
IQContentRecordRequest .....	25
Related ITOs .....	25
Methods to Retrieve Latest Versions of Documents .....	27
Methods that Return Single Objects .....	27
Methods that Return Lists of Objects .....	28
Methods to Retrieve Published Versions of Documents .....	29
Methods that Return Single Objects .....	30
Methods that Return Lists of Objects .....	30
Content Record Case Link Methods .....	33
Content Record Methods that Create or Modify Data .....	33
IQLocaleRequest .....	34
Related ITOs .....	34
IQSecurityRoleRequest .....	35
Related ITOs .....	35
IQUserGroupRequest .....	36
Related ITOs .....	37
IQUserRequest .....	37
Related ITOs .....	38
Methods that Change User Information .....	38
Methods that Return Information about Users .....	39

IQViewRequest .....	40
Related ITOs .....	40
Methods that Change View Information .....	40
Methods that Return Information about Views .....	40
IQWorkTeamRequest .....	41
Related ITOs .....	41
IQContentRecommendationRequest .....	42
Related ITOs .....	42
IQRatingRequest .....	43
Related ITOs .....	43
<b>Chapter 5 Intelligent Search API Overview .....</b>	<b>45</b>
IQServiceClient .....	45
SessionID, TransactionID .....	46
CCAInfo .....	46
ClientInfo .....	47
Implementing the iRequestModifier Interface to Remove the User Agent Parameters Prefix .....	48
Configuring the Intelligent Search Application to Use the Request Modifier 49	
Implementing a Custom Request Modifier Class .....	49
SearchInfo .....	50
UserInfo .....	50
IQSearchRequest .....	51
GIML .....	51
Question Answering Methods .....	53
Call Center Advisor Methods .....	54
Process Wizard Methods .....	54
<b>Appendix A Error Code Constants .....</b>	<b>56</b>
<b>Appendix B GIML XSD .....</b>	<b>68</b>
<b>Appendix C GIML Response .....</b>	<b>81</b>

# About This Guide

This guide provides instructions and supporting information for implementing the Oracle Knowledge Client Library API for use with an Oracle Knowledge application. This guide is intended for application developers and systems administrators to provide an understanding of the design and architecture of the Oracle Knowledge Client Library to facilitate custom development and integration with Oracle Knowledge technologies.

This preface includes information on:

- **In This Guide** - The general organization of this guide
- **Screen and Text Representations**
- **References to Web Content**
- **Examples of Product Screens and Text**

## In This Guide

The *Oracle Knowledge Client Library API Implementation Guide* is divided into the following sections:

<b>Chapter 1, “Installing the Client Library API”</b>	This chapter describes the Client Library installation for Information Manager and Intelligent Search, as well as the Client-side installation.
<b>Chapter 2, “Client Library Introduction”</b>	This chapter describes the underlying architecture of the Oracle Knowledge client Library API including how the remote interface works, the ITO structure and, the error handling strategy.
<b>Chapter 3, “Architecture Overview”</b>	This chapter provides an overview of how to integrate the client Library into various client side technologies like Java apps, jsp apps, Net apps. and explains the performance implications of various designs.
<b>Chapter 4, “Information Manager API Overview”</b>	This chapter describes the services that are exposed from Information Manager and provides sample code.
<b>Chapter 5, “Intelligent Search API Overview”</b>	This chapter describes the services and methods exposed from Intelligent Search and provides sample code.
<b>Appendix A, Error Code Constants</b>	This appendix provides a list of error code constants used by the client Library. The Oracle Knowledge client Library API uses the error code constants to create localized error messages for each type of error that can be thrown.
<b>Appendix B, GIML XSD</b>	This appendix provides a sample GIML.xsd used by Information Manager to parse and interact with GIML returned and processed by the Information Manager JSP tag library and the Information Manager Management Console.
<b>Appendix C, GIML Response</b>	This appendix provides a sample GIML response to a search request.

## Screen and Text Representations

The product screens, screen text, and file contents depicted in the documentation are examples. This guide attempts to convey the product's appearance and functionality as accurately as possible. Application screen content is compared for overall accuracy with screen shots in the guide prior to release. Updates are made where necessary. However, the actual product contents and displays might differ from the published examples.

## References to Web Content

For your convenience, this guide refers to Uniform Resource Locators (URLs) for resources published on the World Wide Web, when appropriate. We attempt to provide accurate information; however, these resources are controlled by their respective owners and are therefore subject to change at any time.

## Examples of Product Screens and Text

The product screens, screen text, and file contents depicted in the documentation are examples. We attempt to convey the product's appearance and functionality as accurately as possible; however, the actual product contents and displays may differ from the published examples.

## Operating System Variations in Examples and Procedures

We generally use Linux screen displays and naming conventions in our examples and procedures. We include other operating system-specific procedures or steps as noted in section headings, or within topics, as appropriate.

We present command syntax, program output, and screen displays:

- in Linux format first
- in other Unix-specific variants only when necessary for proper operation or to clarify functional differences
- in Windows format only when necessary for clarity

---

# Installing the Client Library API

---

The Oracle Knowledge client library is composed of two primary components for both the Intelligent Search and Information Manager products. The server side component is responsible for processing the incoming requests from the various client side client library based applications.

This chapter discusses:

- **Information Manager Server Side Installation**
- **Intelligent Search Server Side Installation**
- **Client Side Installation**

## Information Manager Server Side Installation

The server side portion of the Information Manager client library software is installed automatically when the IMWS.WAR file is installed on an application server. The IMWS.WAR is a self contained J2EE based web application that is deployed in the `$INQUIRA_ROOT/instances/<instancename>/appserverim/webapps` folder. During the start up IMWS.WAR an environment variable `$IM_HOME` is passed into the JVM specifying the location of the Information Manager configuration information. By default the value of `$IM_HOME` is set to `$INQUIRA_ROOT/InfoManager`.

When the IM server starts up the following actions occur:

- 1 The IMWS .WAR file is expanded by the J2EE server into the webapps folder
- 2 The `WEB-INF/repository.properties` file is read. The value of the property `domain.name` is used to lookup the correct configuration information in the `$IM_HOME/config` directory. By default the value of `domain.name` for the IMWS .WAR is set to `IMWEBSERVICES`.
- 3 Using the value of the `domain.name` property in the `repository.properties` file, the `$IM_HOME/config/<domain.name value>/application.properties` file is located and read. This file contains the settings for the JDBC connection that will be used to connect to the IM database schema
- 4 The configuration settings from the `$IM_HOME/config/SYSTEM/config.properties` are read and cached. These settings are the default settings for all repositories configured in the IM database schema.
- 5 When a user is authenticated, any overridden settings from the `$IM_HOME/config/<domain.name value>/config.properties` are read and used in place of the default SYSTEM settings if they exist.



## Intelligent Search Server Side Installation

The Intelligent Search server side of the client library is installed as part of the Search SOAP gateway deployed as part of the `inquiragw.war`. Configuration of the `inquiragw.war` is documented as part of the standard Oracle Knowledge Intelligent Search installation guide.

## Client Side Installation

The client side connector to the IM client library server is available in two platform specific binary deliverables. Oracle Knowledge provides support for C# on the .Net platform from Microsoft and a Java JAR file for Java based applications. The JAR file that support Java 1.5, 1.6, and 1.7 implementations are provided to you; see “Oracle Knowledge Provided Jars”.

## Java Client Installation

The files for Java-based applications are available in the `$IM_HOME/clientLibrary/Java` folder. The JAR files in this folder are all required to be available on the runtime classpath of an application that will use the Oracle Knowledge client library. There are no client side debug settings available currently. The server side debug is enabled using Log4J root logger. The following property needs to be set in the `$IM_HOME/config/<repository_reference key>/log4j.properties` file:

```
log4j.logger.com.inquirara=DEBUG,EVENTLOGGER
```

Specific levels of logging can be configured by using a more specific class path in the logger parameter. For example:

```
log4j.logger.com.inquirara.im.services=DEBUG,EVENTLOGGER
```

**Note:** For the third party JARS, such as `commons-xxx.jar` or `log4j`, it might be possible to use a newer version of the JAR if the vendor has maintained backward compatibility. However, because Oracle Knowledge tests only the versions of the Jar files deployed with the Client Library, Oracle does not recommend changing the supplied versions. Any other version than the supplied versions may introduce unforeseen issues that can be difficult to detect or troubleshoot. The JARS supplied with the current build may not have the version number as part of the JAR name. The listed versions below are the actual versions deployed by Oracle Knowledge.

### THIRD PARTY JARS REQUIRED

- activation-1.0.2.jar
- axis-1.4.jar
- axis-ant-1.4.jar
- commons-beanutils-1.8.jar
- commons-codec-1.3.jar
- commons-collections-3.1.jar
- commons-discovery-0.2.jar
- commons-io-1.4.jar
- commons-lang-2.3.jar
- commons-logging-1.0.4.jar
- http\_client.jar
- jaxrpc-1.1.jar
- jsonorg-2.0.jar
- mail-1.3.jar
- saaj-1.1.jar
- wsdl4j-1.6.2.jar
- xalan-2.7.0.jar
- xercesImpl-2.0.2.jar

### ORACLE KNOWLEDGE PROVIDED JARS

- infra.jar (from the current build with java 1.7)
- infra-jre1.6.jar (compiled with java 1.6)
- infra-jre1.5.jar (compiled with java 1.5)

- `infra_encryption.jar` (from the current build with java 1.7)
- `infra_encryption1.6.jar` (compiled with java 1.6)
- `infra_encryption1.5.jar` (compiled with java 1.5)
- `javaServiceClient.jar` (from the current build with java 1.7)
- `javaServiceClient-jre1.6.jar` (compiled with java 1.6)
- `javaServiceClient-jre1.5.jar` (compiled with java 1.5)
- `itobjects.jar` (from the current build with java 1.7)
- `itobjects-jre1.6.jar` (compiled with java 1.6)
- `itobjects-jre1.5.jar` (compiled with java 1.5)

## C#/.Net Client Installation

The C#/.Net client library is available as a dynamic linked library (DLL) in both standard and debug builds. These DLLs and files need to be available on the runtime PATH for the client application. The files are available in the `$IM_HOME/clientLibrary/MSFT/` folder.

### ORACLE KNOWLEDGE PROVIDED FILES:

- `IQServiceClientCS.dll`
- `IQServiceClientCS.dll.config`
- `IQServiceClientCS.XmlSerializers.dll`

### CONFIGURATION

The C#/.Net DLL is configured by adjusting the settings in the `IQServiceClientCS.dll.config` file. The primary property to adjust is the `configuration/applicationSettings/IQServiceClientCS.Properties.Settings/setting [name="IQServiceClientCS_com_inquirea_imdb_RequestProcessor"]`. This value should be configured to point to the installation URL of the IM request processor. This is usually in the format:

```
http://<host>:8226/imws/WebObjects/imws.woa/ws/RequestProcessor
```

To turn on DEBUG mode for the C# DLL, you must modify the `IQServiceClientCS.dll.config` file. Add or modify the following XML node to set the `<value>` to True:

```
<IQServiceClientCS.Properties.Settings>
  <setting name="IQServiceClientCS_com_inquirea_imdb_RequestProcessor"
    serializeAs="String">
    <value>http://imdb.inquirea.com:8226/IMWebServicesNG/WebObjects/IMWebServicesNG.woa/
      ws/RequestProcessor</value>
    </setting>
    <setting name="DEBUG_MODE" serializeAs="String">
      <value>True</value>
    </setting>
</IQServiceClientCS.Properties.Settings>
```

You must restart the web application for the setting to take effect. The output should be directed to the same location as your system output.

---

# Client Library Introduction

---

The Oracle Knowledge Client Library API was introduced in version 8.x to make it easier for developers to integrate Oracle Knowledge products into existing applications of all types running on a multitude of client platforms. The design goals of the Oracle Knowledge API are:

- Provide native platform data types
- Support across platforms
- Hide the complexity of connecting to remote services
- Provide API access to commonly used functionality
- Provide a consistent interface to access all Oracle Knowledge products

Prior to the introduction of the Client Library API – Oracle Knowledge supported a variety of mechanisms to integrate external products including SOAP interfaces, WSDL interfaces, and web interfaces. These interfaces all worked in slightly different manners and required different skill sets from developers to learn and use. The Client Library API was designed to streamline the learning required to integrate with Oracle Knowledge products and to provide a foundation for exposing additional value added services to our customers, partners, and professional services personnel.

## Native Data Types

One of the primary design goals of the Oracle Knowledge Client Library API is to provide support for native datatypes on each supported platform. A typical web services based platform tends to rely on XML strings and numeric fields to represent complex data structures. This requires the developer to provide their own abstraction layer between the Oracle Knowledge data structures and the data structures the developer wants to interact with.

The Oracle Knowledge Client Library API operates on the Java platform and .Net. The architecture of the Client Library provides the ability to generate native interfaces for other platforms moving forward as customer needs require.

Providing native data types with the Client Library allows for better coding practices such as type safety, and better integration with native IDE tools like Eclipse and Visual Studio to take advantage of built in code assistants.

Native data types are also easier for developers to interact with by avoiding having to marshall strings into and out of data structures for use in client applications. Less code is better code. The Oracle Knowledge Client Library API is mostly generated from data definitions providing us the ability to provide native data type support in a well-tested and validated fashion.

## Cross Platform Support

The Oracle Knowledge Client Library provides cross platform support with a common APIs. Using Oracle Knowledge service definitions, the Client Library provides native support for Windows .Net, and Java. .Net developers will feel just as comfortable using the Oracle Knowledge Client Library API in Visual Studio as a Java developer will within Eclipse.

The Oracle Knowledge Client Library provides specific deployment mechanisms that are appropriate for each target platform. Java versions of the API are provided as JAR files; .Net versions of the API are distributed as DLL files (in both debug and non-debug versions).

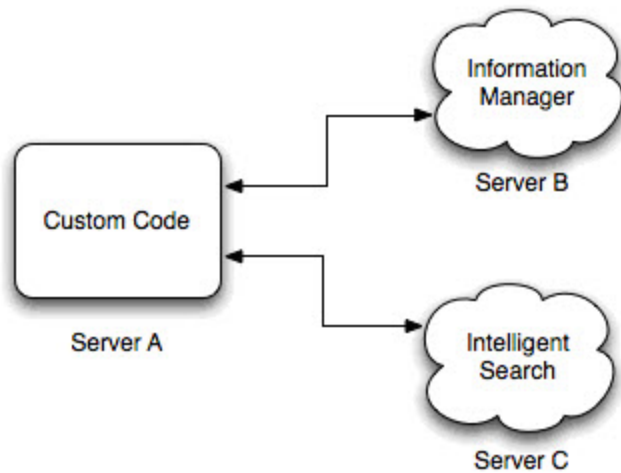
The native data types used in each of the platform specific distributions utilize data types specific to the environment. The following table shows a sampling of the datatypes used in each environment.

Java Datatype	.Net/C# Datatype
• java.lang.String	• System.string
• java.lang.Boolean, boolean	• System.bool
• java.util.ArrayList	• System.Collections.Generic.List
• java.lang.Float, float	• System.Single
• java.lang.Integer, int	• System.Int32
• java.lang.Long, long	• System.Int64
• java.lang.Double, double	• System.Double
• java.util.Date	• System.DateTime

The method for using enumerations in C# is slightly different than in Java. The enumerations in C# are stored in classes in the `IQServiceClientCS.com.inquiraim.enums` namespace. Inside the C# classes there are enumerations matching the Java enumeration classes. Here is the list of classes that contain the enumerations for C#:

- ContentRecordDateRangeFilterMode
- ContentRecordRelationship
- ContentRecordSortField
- EnumAliases
- PriorityEnum
- RoleTypeEnum
- SchemaAttributeTypeEnum
- SchemaTypeEnum

## Remote Access



**Figure 1: Remote access using the Client Library**

The Oracle Knowledge Client Library API is designed to be used with a typical Oracle Knowledge software installation across multiple servers in a networked environment. The Client Library utilizes a configurable transport mechanism between servers (a web service based transport by default) and takes care of the connectivity and data marshalling activities automatically. The developer using the Oracle Knowledge Client Library can write code without regard to the physical location of the services providing the data—greatly simplifying the tasks relating to final deployment of the customized solution.

## Expose Commonly Used Functionality

The Oracle Knowledge Client Library exposes the most commonly utilized methods that interact with Information Manager and Intelligent Search. The methods are designed to use the simplest datatypes possible for the functions being accessed. There is a rich set of Oracle Knowledge specific data types available to provide type safe access to complex data structures representing Oracle Knowledge business objects such as a Content Record.

The services are organized into logical groups based on the general category of usage. The list below represents some of the services available:

- Repository Services
- Security Services
- Content Services
- Category Services
- Search Services

Every attempt is made to replicate the functionality of existing public interfaces such as the Information Manager custom JSP tag library and Search SOAP gateway to provide more flexibility for the developer to choose the platform best suited to their needs. The current Oracle Knowledge Client Library API is a subset of the functionality exposed in the underlying existing product interfaces. Over time, the Client Library will become a superset of the functionality.

## Consistent Interface Across Products

Another guiding principle for the Oracle Knowledge Client Library is to provide a standardized method for accessing all publicly exposed features. This standardization includes authentication and error handling. This allows the developer to create re-usable logic to enable much quicker and robust integrations with legacy technologies.

The method for calling all of the Oracle Knowledge Client Library functions is consistent and standardized. The same type of connection object used for authenticating Information Manager access is used in making Search based procedure calls. The common `ErrorWarningResponse` object provides a standardized mechanism for encapsulating errors across products, remote servers, and processes.

---

# Architecture Overview

---

The Oracle Knowledge client library utilizes a number of J2EE design patterns to access and present data through the Oracle Knowledge client library.

## Service Locator Pattern

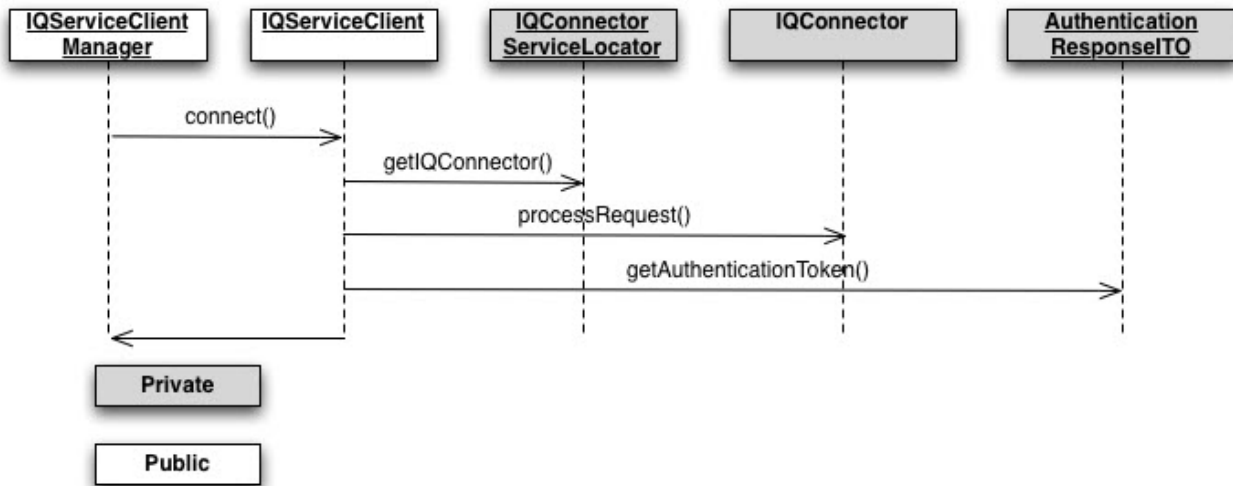
The Oracle Knowledge client library utilizes a modified version of the J2EE service locator design pattern documented at

<http://java.sun.com/blueprints/corej2eepatterns/Patterns/ServiceLocator.html>

This pattern is used to locate services and establish initial contexts that the subsequent requests will utilize. The Oracle Knowledge client library does not utilize JNDI for service location or EJBs for data access, the Oracle Knowledge client library utilizes components that perform similar operations.

The service locator pattern is used under the following conditions:

- Lookup and creation of service components could be complex and may be used repeatedly in multiple clients in the application.
- Initial context creation and service object lookups, if frequently required, can be resource-intensive and may impact application performance. This is especially true if the clients and the services are located in different tiers.
- Use a Service Locator object to abstract all JNDI usage and to hide the complexities of initial context creation, EJB home object lookup, and EJB object re-creation. Multiple clients can reuse the Service Locator object to reduce code complexity, provide a single point of control, and improve performance by providing a caching facility.
- This pattern reduces the client complexity that results from the client's dependency on and need to perform lookup and creation processes, which are resource-intensive. To eliminate these problems, this pattern provides a mechanism to abstract all dependencies and network details into the Service Locator.



**Figure 2: Service locator class diagram**

The Oracle Knowledge client library provides 2 classes to facilitate establishing a connection with the back end services and providing access to the business method services exposed in the Oracle Knowledge client library.

- **IQServiceClientManager** – The IQServiceClientManager is the primary entry point used to obtain a reference to the IQServiceClient. The IQServiceClientManager coordinates the authentication of the user and sets up the IQServiceClient for usage.
- **IQServiceClient** – The IQServiceClient object encapsulates the authenticated user credentials and is used to forward requests to the specific client library business service..

A typical usage pattern of these classes is shown below:

```

IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos,
imUrl, searchUrl, throw ExceptionOnError);
  
```

All parameters are required unless specified.

Parameter	Description
<b>User*</b>	The userid of the user connecting to the service. This should be a valid IM management console user (not a web user) to perform content management activities such as creating users, content, etc. This userid can be from the IM repository or a remote authenticated service.
<b>Passwd*</b>	The password for the user as stored in the IM repository in the UserInformation database table or as stored in a remote authentication store such as SSO or LDAP.
<b>Domain*</b>	The search domain that the search requests utilize. Currently this parameter is not used and any string can be passed for this parameter.
<b>Repos*</b>	The reference key for the IM repository that the service will be connecting to.
<b>imUrl*</b>	The URL to the InfoManager client library endpoint. Typically it will be a URL similar to: <a href="http://&lt;host&gt;:8226/imws/WebObjects/imws.woa/ws/RequestProcessor">http://&lt;host&gt;:8226/imws/WebObjects/imws.woa/ws/RequestProcessor</a>
<b>SearchUrl</b>	The URL to the Search client library endpoint. Typically this will be a URL similar to: <a href="http://&lt;host&gt;:8223/inquiragw/services/RequestProcessor">http://&lt;host&gt;:8223/inquiragw/services/RequestProcessor</a>

\*. Required parameter.



You can limit the time that the client will wait for the response from the server using an overloaded `connect()` method, which takes a `clientTimeout` parameter, for example:

```
int clientTimeout = 60000; // 60 seconds in milliseconds.

IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos,
imUrl, searchUrl, throwExceptionOnError, clientTimeout);
```

This method applies the same timeout interval to all requests made thru this client instance. The overloaded methods that take the `clientTimeout` parameter are available in both Java and C#, as described in “Specifying a Global Timeout Value for All Connections in Java Implementations” on page 12 and “Specifying a Global Timeout Value for All Connections using C#.” on page 13.

If you specify the `clientTimeout` parameter explicitly for the `connect` method, it overrides the value specified in the configuration file, as described in “Specifying a Global Timeout Value for All Connections in Java Implementations” on page 12 and “Specifying a Global Timeout Value for All Connections using C#.” on page 13.

If you do not specify an explicit or global timeout interval, the client library will use the default values for Java and C# implementations, as described in “Specifying a Global Timeout Value for All Connections in Java Implementations” on page 12 and “Specifying a Global Timeout Value for All Connections using C#.” on page 13.

When a timeout condition occurs, the application writes an error to `ErrorWarningResponse` (EWR) with the error code `CLIENT_TIMEOUT_ERROR` (103) and the error message `Operation timed out` (“`client.timeout.error`”).

## Specifying a Global Timeout Value for All Connections in Java Implementations

You can specify a global timeout value for all `IQServiceClient` objects within Java implementations by:

- creating a properties file
- placing the properties file in the application’s classpath

To specify a global timeout value:

- create a file named `iqclientlibrary.properties` that specifies the property `iq.clientlibrary.http.client.connection.default.timeout`, for example:

```
#####
# File: iqclientlibrary.properties
#####
iq.clientlibrary.http.client.connection.default.timeout=60000
```

where 60000 is the timeout interval, specified in milliseconds.

- place this properties file in a folder within in the classpath of the application (server) that uses the client library.

If you do not specify an explicit or global timeout interval, the client library will use the default value, which is 10 minutes (600,000 milliseconds). This is the same value used by the standard Axis web service client configuration.

## Specifying a Global Timeout Value for All Connections using C#:

You can specify a global timeout value for all `IQServiceClient` objects within C# implementations by adding a property to either:

- the application configuration file

or, if you are using dynamic load library (dll) -specific configuration files,

- the `IQServiceClientCS.dll.config` file

For example:

```
<applicationSettings>
..
<!-- Inline the App level settings or externalize using configSource. -->
<IQServiceClientCS.Properties.Settings>
<setting name="IQServiceClientCS_ClientTimeout" serializeAs="String">
    <value>60000</value>
</setting>
</IQServiceClientCS.Properties.Settings>
</applicationSettings>
```

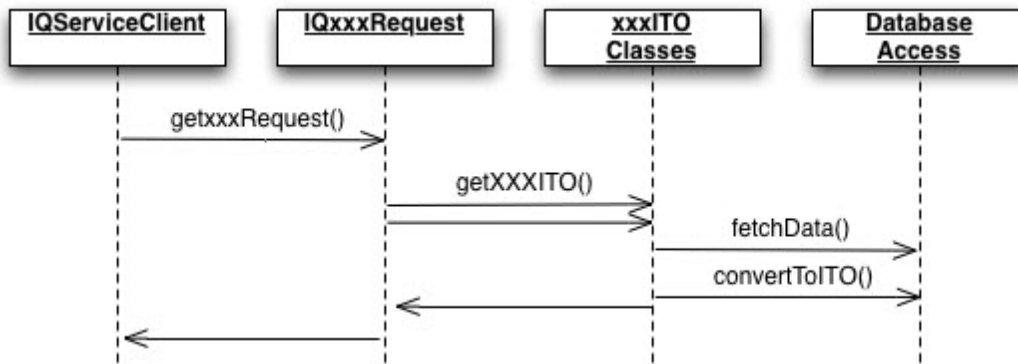
where `<value>60000</value>` specifies the timeout interval in milliseconds.

If you do not specify an explicit or global timeout interval, the client library will use the default value, which is value of 1 minute and 40 seconds (100000 milliseconds) used as default by the .NET framework (`SoapHttpClientProtocol` class).

## Session Facade Design Pattern

The Session Facade design pattern is documented in detail at <http://java.sun.com/blueprints/patterns/SessionFacade.html>. The session Facade design pattern is useful under the following conditions:

- Many business processes involve complex manipulations of business classes. Business classes often participate in multiple business processes or workflows. Complex processes that involve multiple business objects can lead to tight coupling between those classes, with a resulting decrease in flexibility and design clarity. Complex relationships between low-level business components make clients difficult to write.
- The Session Facade pattern defines a higher-level business component that contains and centralizes complex interactions between lower-level business components. A Session Facade is implemented as a session enterprise bean. It provides clients with a single interface for the functionality of an application or application subset. It also decouples lower-level business components from one another, making designs more flexible and comprehensible.
- Fine-grained access through remote interfaces is inadvisable because it increases network traffic and latency. The "before" diagram in Figure 1 below shows a sequence diagram of a client accessing fine-grained business objects through a remote interface. The multiple fine-grained calls create a great deal of network traffic, and performance suffers because of the high latency of the remote calls.



**Figure 3: Sequence diagram applying the session facade pattern**

In Figure 3 the IQServiceClient provides a reference to a IQxxxRequest object. The IQxxxRequest object is designed to aggregate a number of low level ITO methods into a higher level business service such as creating content. A typical client library transaction makes use of many embedded ITO and database access calls to perform the requested operation. The Session Facade object maps to the various IQxxxRequest objects. The Entity Beans map to the ITO objects provided in the client library. The Oracle Knowledge client library provides a number of Session Facade type interfaces that are broken down into a number of request objects that are used to group methods that come from the same functional area. The current list of Session Facade objects is:

- **IQCategoryRequest** – Contains methods to work with Information Manager categories assigned to repositories, views, and documents.
- **IQContentChannelRequest** – Contains methods to work with the definition of Information Manager content channels.
- **IQContentRecommendationRequest** – Contains methods to work with content recommendations made for a content channel.
- **IQContentRecordRequest** – Contains methods to create, modify, and retrieve content records from the Information Manager repository.
- **QLocaleRequest** – Contains methods to work with locales assigned to a repository.
- **QRatingRequest** – Contains methods to rate content records and to retrieve results of the ratings.
- **QRepositoryRequest** – Contains methods to retrieve meta data about an IM repository.
- **QSearchRequest** – Contains methods to submit search requests.
- **QSecurityRoleRequest** – Contains methods to retrieve meta data about security roles assigned to repositories and users.
- **QUserGroupRequest** – Contains methods to retrieve information about user groups assigned to channels and views.
- **QViewRequest** – Contains methods to manage users assigned to views and to retrieve information about users assigned to views and views assigned o channels and repositories.
- **QWorkTeamRequest** – Contains methods to retrieve information of work teams assigned to a repository.

These requests all contain a number of methods that are used to interact with the underlying Oracle Knowledge services at a high level. All of the methods in the Session Facade interfaces use standard data types or specialized Oracle Knowledge data types called ITOs as parameters (see “Data Transfer Design Pattern” on page 15).

## Data Transfer Design Pattern

The data transfer design pattern is used to abstract the retrieval of data from the database for use in the client library. The returned ValueObjects are stateless and completely self-contained, requiring no additional network access in order to provide the requested data.

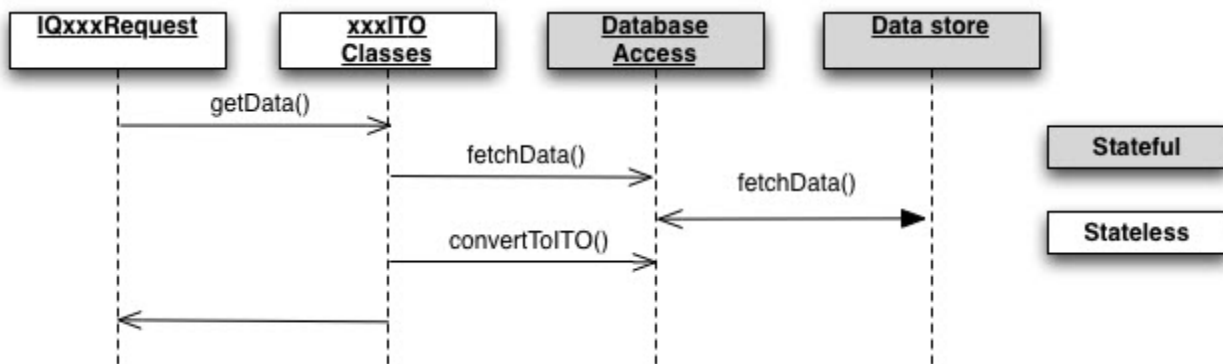
The J2EE data transfer design pattern can be found at <http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html>. Here is a synopsis of the reasons for the use of the data transfer design pattern:

All access to an enterprise bean is performed via remote interfaces to the bean. Every call to an enterprise bean is potentially a remote method call with network overhead.

Typically, applications have a greater frequency of read transactions than update transactions. The client requires the data from the business tier for presentation, display, and other read-only types of processing. The client updates the data in the business tier much less frequently than it reads the data.

The client usually requires values for more than one attribute or dependent object from an enterprise bean. Thus, the client may invoke multiple remote calls to obtain the required data.

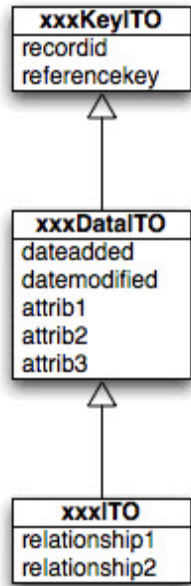
The number of calls made by the client to the enterprise bean impacts network performance. Chatterier applications, those with increased traffic between client and server tiers, often degrade network performance.



**Figure 4: J2EE Data transfer design pattern**

The ValueObjects for the Oracle Knowledge client library are called ITOs - Oracle Knowledge Transfer Objects. The Oracle Knowledge client library provides a number of services that retrieve and format the data into ITOs that are returned to the calling process. The ITO objects are completely stateless and contain all of the required data. An ITO does not need to make additional network calls to retrieve its data.

The Oracle Knowledge ITO structure used by Information Manager provides 3 levels of data based on an increasing level of data.



**Figure 5: ITO hierarchy**

The xxxKeyITO objects typically contain only the **recordid** and perhaps a **referencekey** value. A **referencekey** is a non-localized string variable that can be used to uniquely identify an object in the some IM database tables that contain localizable information. The KeyITO objects are very small and can be used in most API calls as a proxy object. KeyITO objects are typically returned in results that return an array of objects.

xxxDataITO objects extend the xxxKeyITO objects and add the DATEADDED and DATEMODIFIED values from the IM entities. In addition the xxxDataITO objects also include all of the attributes of the corresponding entity that are NOT relationships to other entities. In most cases (but not all) the xxxDataITO corresponds directly to the underlying database access entity. There are some custom ITO objects that combine one or more database entities into a single xxxDataITO (for example ContentRecordITO) to provide a more convenient mechanism for accessing the properties.

The xxxIto objects are sometimes referred to as “full” ITO’s. These ITOs extend the xxxDataITO objects but also include the relationships to other ITO objects. Including the relationships to other ITO objects can result in very large ITO objects being fetched. The full ITO contains arrays of KeyITO objects for all objects in the relationship. For example suppose a RepositoryITO (a full ITO) is retrieved using a method such as getRepositoryForID(). This full ITO will also return an array of KeyITO objects for every User and Content record in the system. Use of the full ITO is discouraged in most cases. It is much more efficient to use the KeyITO and DataITO methods where possible.

## Error Handling

All of the Oracle Knowledge client library methods provide access to a built in ErrorWarningResponse (EWR) object. This object is populated after every call to a client library service request. This object contains a list of all of the errors and warnings that occurred in the scope of the last service method call. It is important to check the EWR object after each call to ensure that the call was successful. If a service method makes multiple client library calls in the scope of a single invocation - all of the errors and warnings from the embedded client library calls are collected and presented in the EWR object returned from the service call.

The EWR contains both errors and warnings. Errors are typically fatal errors indicating that the underlying exception prevented the service call from returning the expected results. Validation and SQL errors are examples of fatal errors that can occur. Warnings are typically non-fatal exceptions that can occur if a method does not perform as expected. An example of a warning could be a retrieve method that does not return any data. This could be a result of invalid parameters or simply no records were found that matched the specified criteria.

The following code sample shows a strategy for checking the EWR object for fatal errors that could be raised in the scope of a service client invocation.

```
public static boolean checkEWR(ErrorWarningResponse ewr) {
    boolean result = true;
    if(ewr != null && ewr.hasErrorsOrWarnings()) {
        // EWR reported a problem, check to see if it is an error
        if(ewr.hasErrors()) {
            // error was reported
            result = true;
            // output the errors
            List<ErrorRecord> errors = ewr.getErrors();
            System.out.println(">>>> Error occurred: >>>>>>>\n");
            for(Iterator<ErrorRecord> iter = errors.iterator() ;
                iter.hasNext();) {
                ErrorRecord rec = iter.next();
                System.out.println(rec);
            }
        } else {
            // warning must have been reported, assume it is safe to go on
            result = false;
        }
    } else {
        result = false;
    }
    return result;
}
```

Every IQxxxRequest object provides a mechanism to retrieve the EWR object through a call to getEWR(). It is possible to interrogate the EWR to provide code for specific types of errors by checking the ErrorRecord.getType() method. The list of error types is defined in the com.inquiria.util.ewr.ErrorTypeEnum class. Each error type reported in the EWR can have many different specific types of error messages depending on the nature of the exception. For example:

deleteContent() in the IQContentRecordRequest can throw an APPLICATION\_ERROR (ErrorTypeEnum)

and one of the following specific messages:

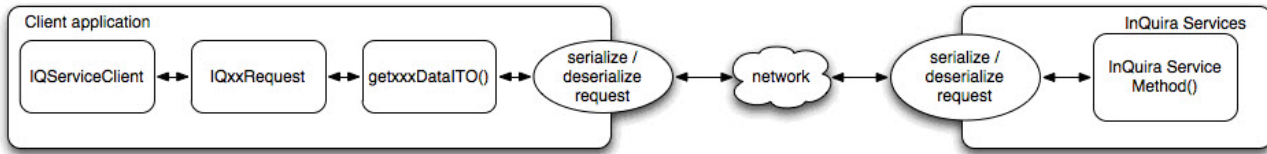
```
CONTENT_SERVICE_IO_DELETE_EXCEPTION, CONTENT_SERVICE_CONTENT_NOT_FOUND, or
CONTENT_SERVICE_INVALID_CONTENTID
```

It is necessary to review each of the specific messages in order to provide the most flexible error handling subsystem.

## Serialization

The Oracle Knowledge client library provides a library of cross platform serialization methods in every ITO to serialize the ITO to an XML format or a JSON format. Developers using the Oracle Knowledge client library can call these serialization methods at any time. These methods are also called automatically by the Oracle

Knowledge client library framework to marshall/unmarshall the parameters and results between the local client application and the remote Oracle Knowledge service.



**Figure 6: Oracle Knowledge client library serialization**

The Oracle Knowledge client library serializes the request parameters in the scope of the IQServiceClient request invocation. The serialized data is transmitted to the Oracle Knowledge service provider via a single WSDL based web service call. The WSDL end point is deployed in both the Information Manager web services deployable (imws.war) and the Intelligent Search runtime gateway (inquirag.war). The URLs to the web service endpoints are the following (default installations):

InfoManager – `http://<host>:8226/imws/WebObjects/imws.woa/ws/RequestProcessor`

Search - `http://<host>:8223/inquiragw/services/RequestProcessor`

The connect() method of the IQServiceManager object utilizes the URLs for the IM and Search services to connect to the remote services and caches the connection information for the duration of the session.

The RequestProcessor web service provides a single WSDL method called processRequest() that takes in an XML formatted string and returns an XML formatted string. The client library serialization/deserialization mechanism on both sides of the web service call performs the necessary work to hide the complexity of the web service invocation. To both the client calling code and the Oracle Knowledge services code the calls all appear to be localized to the current process. It is not recommended to use the WSDL based interface without the Oracle Knowledge supplied serialization mechanism to avoid unpredictable behavior.

# Information Manager API Overview

The Oracle Knowledge client library provides a number of services that expose functionality from Information Manager. The services are broken into groups to simplify the organization of the service methods. The list of available services and the methods available in each service will expand over time with each new release.

## IQRepositoryRequest

The `IQRepositoryRequest` object is obtained from the `IQServiceClient` by executing the following code to obtain a reference to the `IQServiceClient` from the `IQServiceClientManager`. The `IQServiceClientManager` is a static class that can be accessed with the `connect` method. The result of a successful connection is a valid `IQServiceClient` object that can be used to retrieve an `IQRepositoryRequest` object. This same pattern is used to obtain an `IQxxxRequest` in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQRepositoryRequest request = client.getRepositoryRequest();
...
```

## Related ITOs

The `IQRepositoryRequest` utilizes the `RepositoryXXXITO` classes as both parameters and return types. Choosing the correct method is important to ensure that a performance penalty is not unnecessarily incurred.

The `RepositoryKeyITO` is the lightest weight object available. It provides the `recordid` and `reference key` of the repository. It can be used as a parameter or returned by one of the `get()` methods. This object does NOT contain any other data or relationships. Where possible use methods that utilize the `KeyITO`.

The `RepositoryDataITO` inherits from `RepositoryKeyITO`. It adds the additional properties from the `SITE` table as data with `get/set` methods for each property. This is also a lightweight object that only requires a single database fetch to populate.

The `RepositoryITO` (a full ITO) is a VERY HEAVYWEIGHT object and methods that use or return it should be avoided as much as possible. It extends the `RepositoryDataITO` but also adds data from all of the relationships associated with a repository. This object can be very large!! The relationships to other objects are arrays of `xxxKeyITO` objects.

## getRepositoryxxxByReferenceKey()

This family of methods provides access to a `RepositoryITO`, `RepositoryDataITO`, and a `RepositoryKeyITO` depending on the specific method called. The typical usage of these methods is to resolve a repository reference key so that the metadata of a repository can be used in other calls.



**Important!** If the RepositoryITO version of this method is used, the results should be cached and reused. This method should only be called once.

There is a possibility that the metadata of a repository could become out of date if the structure of a repository changes in any way—that is, new channels, new security role, and so on.

These methods DO NOT resolve Views – those methods are available in the IQViewRequest.

IQRepositoryRequest Methods:

```
getRepositoryByReferenceKey() returns RepositoryITO
getRepositoryDataByReferenceKey() returns RepositoryDataITO
getRepositoryKeyByReferenceKey() returns RepositoryKeyITO
```

## getRepositoryxxxByID()

This family of methods uses the GUID of the repository to retrieve Repository ITOs. The GUID is from the IM database schema in the SITE table from the RECORDID column. These methods are typically used when it is necessary to interact with the database directly where the primary keys have been used or retrieved. The GUID is a VARCHAR key the table containing a globally unique identifier generated by Information Manager.

**Important!** Avoid using methods that use or return RepositoryITO objects (full ITO) as much as possible.

IQRepositoryRequest Methods:

```
public RepositoryITO getRepositoryForID(String referencekey);
public RepositoryDataITO getRepositoryDataForID(String referencekey);
public RepositoryKeyITO getRepositoryKeyForID(String referencekey);
```

## getRepositoryXXXforRepositoryKey()

This family of methods takes a RepositoryKeyITO as a parameter and returns either a RepositoryDataITO or a full RepositoryITO object. These methods are typically used as part of an object resolution algorithm. This usually would occur if a RepositoryKeyITO had been retrieved in a previous method and additional data is needed about the repository.

**Important!** Avoid using methods that use or return RepositoryITO objects (full ITO) as much as possible.

IQRepositoryRequest Methods:

```
public RepositoryKeyITO getRepositoryForRepositoryKey(RepositoryKeyITO keyito);
public RepositoryDataITO getRepositoryDataForRepositoryKey(RepositoryKeyITO keyito);
```

## IQCategoryRequest

The IQCategoryRequest object is obtained from the IQServiceClient by executing the following code to obtain a reference to the IQServiceClient from the IQServiceClientManager. The IQServiceClientManager is a static class that can be accessed with the connect() method. The result of a successful connection is a valid IQServiceClient object that can be used to retrieve an IQCategoryRequest object. This same pattern is used to obtain an IQxxxRequest in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQCategoryRequest request = client.getCategoryRequest();
...
```

The IQCategoryRequest is the primary service that allows a programmer to work with the IM category tree. An IM category is similar to a search facet and during the indexing process all of the IM categories can be converted to facets. Categories can be used for a number of different tasks within an IM repository. Categories can be used to categorize content records with products, versions, colors, sizes, business units, etc.

When a category is assigned to a content record, the category applies to all translated versions of that document.

When a category is assigned to a user, it represents the skills that the user has. These skills can be used to influence workflow and task assignments to help automate the process more efficiently.

## Related ITOs

The IQCategoryRequest utilizes the CategoryXXXITO classes as both parameters and return types. Choosing the correct method is important to ensure that a performance penalty is not unnecessarily incurred.

The CategoryKeyITO is the lightest weight object available. It provides the recordid and reference key of the category. It can be used as a parameter or returned by one of the get() methods. This object does NOT contain any other data or relationships. Where possible use methods that utilize the KeyITO.

The CategoryDataITO inherits from CategoryKeyITO. It adds the additional properties from the TAG table (dateadded, date modified, description, sort order, name, object\_id) as data with get/set methods for each property. This is also a lightweight object that only requires a single database fetch to populate.

The CategoryITO (a full ITO) can be a VERY HEAVYWEIGHT object and methods that use or return it should be avoided as much as possible. It extends the CategoryDataITO but also adds data from all of the relationships to the TAG table as well. This object can become very large in repositories with large category trees. Returning a CategoryITO that represents the root of the tree could require a large amount of data to be fetched and formatted. Use this object and methods that create or return it with care.

## getCategory%MODE%ForCategoryKey (FULL, DATA)

This family of methods return either CategoryDataITO, or CategoryITO objects when passing in a CategoryKeyITO. These are convenience methods used to obtain more information for a specific CategoryKeyITO. The typical use case for this family of methods would occur after fetching an array of CategoryKeyITO objects and then needing to display the details of each record as the list is being iterated.

```
public CategoryITO getCategoryITOfForCategoryKey(CategoryKeyITO ito);
public CategoryDataITO getCategoryDataITOfForCategoryKey(CategoryKeyITO ito);
```

## getCategory%MODE%ForID (FULL, DATA, KEY)

This family of methods is used to resolve a RECORDID from the TAG table into a CategoryxxxITO. In general it is better to use the lightest weight object type possible (i.e. a Key or Data object is lighter than a full ITO object). The typical case for this family of methods would be when directly interacting with the database or a legacy system that has stored the primary key of the categories.

## getCategory%MODE%ByReferenceKey (FULL, DATA, KEY)

This family of methods is used to resolve category reference keys. The typical use case for these methods is when a string representing the reference key of the category is passed via a URL parameter in a HTML or JSP page to be processed.

## category%MODE%ITOChildrenForParent (FULL, DATA, KEY)

This family of methods is used to return the child categories of a parent category branch. If the parent Category can't be found or does not have any child categories the returned list will be NULL.

```
public List<CategoryITO> categoryITOChildrenForParent(String parentrefkey);
public List<CategoryDataITO> categoryDataITOChildrenForParent(String parentrefkey);
public List<CategoryKeyITO> categoryKeyITOChildrenForParent(String parentrefkey);
```

## getCategory%MODE%ListAssignedToView (FULL, DATA, KEY)

This family of methods returns the category ITOs assigned to a repository view. If the view does not exist or have any categories assigned to it, NULL is returned. The resultset can be limited by setting the fetchlimit to a non-zero number.

## getRequiredCategory%MODE%ListForChannel (FULL, DATA, KEY)

This family of methods is used to return the list of category branches that have been assigned to a channel that require at least one category to be specified when creating a document in that channel. The typical use case for this method is when creating a custom data entry form that will be used to create content for a channel. A channel may require that the user select one category from a list of choices to assign to the content record. To limit the number of records being returned, pass in a non-zero fetch limit. If the channel does not exist or does not have any categories assigned to it a NULL will be returned.

## getCategory%MODE%ListForChannel (FULL, DATA, KEY)

This family of methods is used to return the list of categories assigned to a channel. These methods are typically used when creating a custom data entry for a content channel to allow the user to choose from a list of categories that have been assigned to the channel from the IM management console. To limit the number of records returned from the call, pass in a non-zero fetch limit. If the channel does not exist or does not have any categories assigned to it, a NULL will be returned.

## addCategory

This method is used to create categories. If a category is added without a parent category, it is considered a root branch category. If a category is created with a parent category, it will be represented as a sub category

in the IM management console. The locale provided is used to localize the category name. It is not possible to translate a category using the `addCategory()` or `updateCategory()` methods

```
public CategoryITO addCategory(CategoryITO category, String locale);
```

## deleteCategory

This method is used to delete a category branch or a single category. If the category being deleted is a branch, all of the children of the branch will also be deleted. If the method returns `TRUE`, the deletion was successful. If the method returns `FALSE`, the deletion failed and the EWR should be checked to determine the root cause of the problem. Only authorized IM management console users are allowed to delete categories.

```
public boolean deleteCategory(CategoryKeyITO ito); updateCategories(List<CategoryITO>,
List<String locale>)
```

## updateCategories

This family of methods is used to update either a single category or multiple categories. When updating a category, it is necessary to pass in the category that is being updated with the newly updated information in the ITO. The locale provided is used to update the specific localized version of the category. If the category doesn't exist, an exception is thrown. When updating multiple categories for multiple locales, it is necessary to have a one to one mapping between the number of categories passed in to the number of locales passed in. The returned `CategoryITO` objects reflect the newly saved changes to the categories that were submitted.

```
public List<CategoryITO> updateCategories(List<CategoryITO>, String localecode);
public CategoryITO updateCategories(List<CategoryITO>, String locale);
```

**Important!** The method `public List<CategoryITO> updateCategories(List<CategoryITO>, List<String> localecodes)` is deprecated. You should use the method `public List<CategoryITO> updateCategories(List<CategoryITO>, String localecode)` instead.

## IQContentChannelRequest

The `IQContentChannelRequest` object is obtained from the `IQServiceClient` by executing the following code to obtain a reference to the `IQServiceClient` from the `IQServiceClientManager`. The `IQServiceClientManager` is a static class that can be accessed with the `connect()` method. The result of a successful connection is a valid `IQServiceClient` object that can be used to retrieve an `IQContentChannelRequest` object. This same pattern is used to obtain an `IQxxxRequest` in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos,
imUrl, searchUrl, throwExceptionOnError);
IQContentChannelRequest request = client.getContentChannelRequest();
...
```

The `IQContentChannelRequest` is the primary gateway to access methods that provide information about IM content channels. Currently this service request only provides some basic meta data about the channel and to resolve content channel reference keys.

## Related ITOs

The `IQContentChannelRequest` utilizes the `ContentChannel%MODE%ITO` classes as both parameters and return types. Choosing the correct method is important to ensure that a performance penalty is not unnecessarily incurred.

The `ContentChannelKeyITO` is the lightest weight object available. It provides the recordid and reference key of the content channel. It can be used as a parameter or returned by one of the `get()` methods. This object does NOT contain any other data or relationships. Where possible use methods that utilize the `KeyITO` object.

The `ContentChannelDataITO` inherits from `ContentChannelKeyITO`. It adds the additional properties from the `CONTENTCHANNEL` table as data with `get/set` methods for each property. This is also a lightweight object that only requires a single database fetch to populate.

The `ContentChannelITO` (a full ITO) can be a VERY HEAVYWEIGHT object and methods that use or return it should be avoided as much as possible. It extends the `ContentChannelDataITO` and adds data from all of the relationships to the `CONTENTCHANNEL` table as well. This object can become very large in repositories with large amounts of content.

The `XMLSchemaAttributeITO` (a full ITO) contains all of the attributes defined for a content channel. These attributes are mapped from the `SCHEMAATTRIBUTE` and `SCHEMAATTRIBUTERESOURCE` tables in the IM database. THE `XMLSCHEMA` table represents the collection of `XMLSCHEMAATTRIBUTES` that make up a content channel.

## `getContentChannel%MODE%ForContentChannelKey` (FULL, DATA)

This family of methods is used to resolve `ContentChannelKeyITO` objects in order to obtain more information about the associated `ContentChannel`.

## `getContentChannel%MODE%ByReferenceKey` (FULL, DATA, KEY)

This family of methods is used to resolve `Content Channel` reference keys into ITO objects.

## Miscellaneous ContentChannel Service Methods

In addition to the standard services provided by the `IQContentChannelRequest` there are some additional methods that provide access to specific information about a content channel

- `public ContentChannelITO getContentChannelByReferenceKeyAndLocale (String refkey, String String locale)` - This method returns the localized information for the specified content channel. If the channel has not been translated to the requested locale, the reference keys for the non-translated resources will be returned instead. This method could return a NULL object if no channel matches the requested reference key.
- `public List<ContentChannelITO> getContentChannelsByLocale (String locale)` - This method returns all of the channels in the repository using the specified locale code. This array can be NULL if there are no channels available in the repository.
- `public boolean hasFiles (String channelrefkey)` - This method is used to determine if the content channel definition associated with the reference key has any attributes that are of type FILE.

- `public List<XMLSchemaAttributeITO> getChannelSchema (String channelrefkey)` - This method returns all of the attributes defined for the specified content channel. The `XMLSchemaAttributeITO` contains the properties for a single attribute of the channel.

## IQContentRecordRequest

The `IQContentRecordRequest` object is obtained from the `IQServiceClient` by executing the following code to obtain a reference to the `IQServiceClient` from the `IQServiceClientManager`. The `IQServiceClientManager` is a static class that can be accessed with the `connect()` method. The result of a successful connection is a valid `IQServiceClient` object that can be used to retrieve an `IQContentRecordRequest` object. This same pattern is used to obtain an `IQxxxRequest` in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQContentRecordRequest request = client.getContentRecordRequest();
...
```

The `IQContentRecordRequest` is the primary gateway for working with content records from the IM repository. An IM content record has a number of relationships to other data that capture the categories, user groups, and other important attributes of the document. The `ContentRecordITO` objects are a denormalized view of all of the relationships embedded in the business logic of an IM content record. In general an IM content record can exist in one of two states: in development or published. The content channel definition will determine the behavior of its children documents.

A document that is under development may be made available to users through the Oracle Knowledge Search engine if the latest version or draft states of the document are crawled and indexed. The IM JSP tag library also provides a mechanism to allow the display of non-published versions of a document. By default only the published versions of the document are available to consumers of the content.

If a content channel has a workflow assigned to it, then each change to the document will potentially move the document through a workflow and increment the minor version number. Initially the document would be created as version 0.1. Once the document has completed the workflow it's major version will be incremented to the next highest integer and its minor version set to 0. The first published version of a document will be version 1.0. It is possible for a document to complete the workflow but not be published. There are properties available in the channel definition to determine if/when a change will affect workflow.

For content channels without a workflow defined the initial version of the document is 1.0. Each version of a document in a channel without workflow only increments the major version number each time the document is saved.

Content records can be secured from non-authenticated users by setting the user group on either a document or an attribute within the document. Once a user group has been assigned to a document only authenticated users will be able to view the document. If no user groups have been assigned to the document, all users will be able to view the document.

## Related ITOs

The `IQContentRecordRequest` utilizes the `ContentRecord%MODE%ITO` classes as both parameters and return types. Choosing the correct method is important to ensure that a performance penalty is not unnecessarily incurred.

The `ContentRecordKeyITO` is the lightest weight object available. It provides the recordid of the content record. It can be used as a parameter or returned by one of the `get()` methods. This object does NOT contain any other data or relationships. Where possible use methods that utilize the `KeyITO` object.

The `ContentRecordDataITO` inherits from `ContentRecordKeyITO`. It adds the additional properties from the `CONTENTTEXT` table as data with `get/set` methods for each property. This is also a lightweight object that only requires a single database fetch to populate.

The `ContentRecordExtendedITO` inherits from `ContentRecordDataITO`. It adds the to-one relationships of a `CONTENTTEXT` database table to the `ContentRecordDataITO`. An example of a to-one relationship is the XML for the content record (stored in the `CONTENTDATA` table). This ITO requires multiple fetches or joins to return the related data.

The `ContentRecordITO` (a full ITO) can be a VERY HEAVYWEIGHT object and methods that use or return it should be avoided as much as possible. It extends the `ContentRecordExtendedITO` but also adds data from all of the relationships to the `CONTENTTEXT` (or `CONTENTTEXT PUB` for published documents) table as well. This object can become very large in repositories with large amounts of content.

Methods that require a `CONTENTID` need either the `RECORDID` from the `CONTENT` table or the `CONTENTID` from the `CONTENTTEXT` or `CONTENTTEXT PUB` tables. This value is a GUID formatted string. Methods that need a `DOCID` can use the numeric value of the document prefaced by the channel-defined prefix - i.e. `SOL1234`. The locale code values are in the form `en_US` (language code\_location code). Methods that specify a locale code will return the localized version of the content record. If the content record has not been translated to the requested locale a `NO_DATA_FOUND` exception will be raised in the EWR.

The master document is typically the original locale that the document was created in. All of the translated documents are based off of the master document. If the master document is changed, it can trigger notifications to the translated document owners to update their version of the document.

Unless specified in the method name, the default behavior for these methods does NOT increment the view count of a document. Typically only methods that return a single document will be able to increment the view count for the document and to affect the underlying user reputation model.

There are methods that return single objects or lists. The single objects or lists can be of one of the defined types: `ContentRecordKeyITO`, `ContentRecordDataITO`, `ContentRecordExtendedITO`, or `ContentRecordITO`. Typically the methods that return lists of objects return lists of `ContentRecordDataITO` objects.

The methods that return lists of records typically take a number of parameters most of which are optional. The more parameters that are provided, the more the result set is filtered. In general, the channel reference key is the only required parameter. If no localecode is specified, the default repository locale is used. The `getXXX` methods do not make any security evaluations on the fetch other than the parameters provided. If a user does not have the correct security roles or user groups associated with their user account, the user will not be able to view the returned documents.

**Note:** Some methods provide a `maxrecords` parameter. If the `maxrecords` parameter is NOT specified, the default value of 100 records will be used to limit the result set.

**Note:** Some methods provide an `activityType` parameter for fetching single documents. The `activityType` is a string that is passed to IM analytics and must be set if the IM Popular Content and Accessed Content reports are going to be used. Failure to set the `activityType` will result in reports that do not provide any detailed usage scenarios.

The methods that use dates as parameters typically use `java.util.Date` typed objects (or their platform specific equivalents). The time portion of the datatype is typically not used when fetching content based on date.

The `com.inqira.im.enums.ContentRecordDateRangeFilterMode` class provides 2 enumerations:

- `VALIDATE_DISPLAY_START_DATE_IN_RANGE` - This filter mode ensures that the content record display start date (only) is in range defined by `startDate` and `endDate` parameters.
- `VALIDATE_DISPLAY_DATES_IN_RANGE` - This filter mode ensures that both the start and end dates are in range defined by `startDate` and `endDate` parameters.

This mode along with the `startDate` and `endDate` parameters is considered separately from parameter `displayDatesValidNow`, which is a boolean flag indicating if the returned documents should be displayed.

The `com.inquiria.im.enums.ContentRecordSortField` class is used to specify how the list of returned records is sorted. The methods that use the `ContentRecordSortField` enumeration are sorted according to the boolean ascending parameter. If ascending is set to `TRUE`, all of the items in the `ContentRecordSortField` list are sorted in ascending order. The same sort direction is applied to all sort fields specified in the list of sort criteria. For example, if ascending is set to `TRUE` using `INDEXMASTERIDENTIFIERS` and `PRIORITY` then the SQL clause will be generated similarly to

```
...
ORDER BY INDEXMASTERIDENTIFIERS ASCENDING, PRIORITY ASCENDING
```

The order of the sort fields in the list will be used to generate the SQL `ORDER BY` clause. The valid choices for the `ContentRecordSortField` enumeration are:

- `INDEXMASTERIDENTIFIERS`
- `PRIORITY`
- `PUBLISHEDDATE`
- `DATEADDED`
- `DATEMODIFIED`
- `CREATEDATE`
- `DISPLAYENDDATE`
- `DISPLAYSTARTDATE`
- `EVENTENDDATE`
- `EVENTSTARTDATE`
- `DOCUMENTID`
- `MOSTPOPULAR`

## Methods to Retrieve Latest Versions of Documents

This family of methods provides convenience methods to retrieve one or more IM documents from the IM repository. The latest version of an IM document may be newer than the version that is currently published. The latest version may not be published yet (still in workflow). The latest version of the document may not have the same category or user groups assigned to it as compared to the currently published version.

### Methods that Return Single Objects

```
public ContentRecordKeyITO
getLatestMasterContentRecordKeyByContentID(String contentid);

public ContentRecordKeyITO
getLatestMasterContentRecordKeyByDocumentID(String docid);

public ContentRecordKeyITO
getLatestContentRecordKeyByContentIDAndLocale(String contentid,
String localecode);

public ContentRecordKeyITO
getLatestContentRecordKeyByDocumentIDAndLocale(String docid, String localecode );

public ContentRecordITO
```



```

getLatestContentRecordByContentIDAndLocale(String contentid, String localecode);

public ContentRecordITO
getLatestContentRecordByContentIDAndLocaleAndIncrementViewCount(String contentid,
String locale, String activityType);

public ContentRecordITO
getLatestContentRecordByDocumentIDAndLocale(String docid, String localecode);

public ContentRecordITO
getLatestContentRecordByDocumentIDAndLocaleAndIncrementViewCount(String docid, String
localecode, String activityType);

```

## Methods that Return Lists of Objects

```

public List<ContentRecordDataITO> getLatestContentRecordDataITOs (
String channelrefkey,
String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
List<String> caseLinkCaseValues,
int maxrecords);

public List<ContentRecordExtendedITO> getLatestContentRecordExtendedITOs (
String channelrefkey,
String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
List<String> caseLinkCaseValues,
int maxrecords);

public List<ContentRecordDataITO> getLatestContentRecordDataITOsByContentIDs
(List<String> contentids, String localecode, int maxrecords);

public List<ContentRecordExtendedITO> getLatestContentRecordExtendedITOsByContentIDs
(List<String> contentids, String localecode, int maxrecords);

public List<ContentRecordDataITO> getLatestSortedContentRecordDataITOs(
String channelrefkey,
String String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
Date updateSinceDate,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
List<String> caseLinkCaseValues,
boolean ascending,
List<ContentRecordSortField> orderings,
int maxrecords);

```

```

public List<ContentRecordExtendedITO> getLatestSortedContentRecordExtendedITOs (
    String channelrefkey,
    String localecode,
    boolean displayDatesValidNow,
    Date startDate,
    Date endDate,
    ContentRecordDateRangeFilterMode mode,
    Date updateSinceDate,
    List<String> viewrefkey,
    Boolean useHierarchicalCategories,
    List<String> categoryrefkey,
    List<String> usergrouprefkey,
    List<String> caseLinkCaseValues,
    boolean ascending,
    List<ContentRecordSortField> orderings,
    int maxrecords);

public List<ContentRecordExtendedITO> getLatestSortedContentRecordExtendedITOs (
    String channelrefkey,
    String localecode,
    boolean displayDatesValidNow,
    Date startDate,
    Date endDate,
    ContentRecordDateRangeFilterMode mode,
    Date updateSinceDate,
    List<String> viewrefkey,
    Boolean useHierarchicalCategories,
    List<String> categoryrefkey,
    List<String> usergrouprefkey,
    List<String> caseLinkCaseValues,
    boolean ascending,
    List<ContentRecordSortField> orderings,
    int maxrecords);

public List<ContentRecordDataITO>
getLatestSortedContentRecordDataITOsByContentIDs(List<String> contentids, String
localecode, int maxrecords, boolean ascending, List<ContentRecordSortField> orderings);

public List<ContentRecordExtendedITO>
getLatestSortedContentRecordExtendedITOsByContentIDs(List<String> contentids, String
localecode, int maxrecords, boolean ascending, List<ContentRecordSortField> orderings);

public List<ContentRecordDataITO>
getLatestContentRecordDataITOsByDocumentIDs (List<String> docids, String localecode, int
maxrecords);

public List<ContentRecordExtendedITO>
getLatestContentRecordExtendedITOsByDocumentIDs (List<String> docids, String
localecode, int maxrecords);

public List<ContentRecordDataITO>
getLatestSortedContentRecordDataITOsByDocumentIDs(List<String> docids, String
localecode, int maxrecords, boolean ascending,
List<ContentRecordSortField> orderings);

public List<ContentRecordExtendedITO>
getLatestSortedContentRecordExtendedITOsByDocumentIDs(List<String> docids, String
localecode, int maxrecords, boolean ascending, List<ContentRecordSortField> orderings);

```

## Methods to Retrieve Published Versions of Documents

This family of methods is used to return only published documents. When a document is published the current set of categories and user groups is captured and stored with the published document. It is possible to

have a master document published but none or some of the translated versions of the document. Currently there can only be a single version of a published document published, i.e. If version 3.0 is published, version 2.0 is removed.

A published document may not be the latest version of the document. The latest version could still be in workflow or it may not yet be published. A document can be published but still may not be viewable by a user who has permission to view the specific document if the start and end dates of a document are being enforced (the default behavior). If a document is published but the start or end dates of the document fall outside of the current date - the user will not be able to view the details of the document.

## Methods that Return Single Objects

```
public ContentRecordKeyITO
getPublishedContentRecordKeyByContentIDAndLocale(String contentid, String localeCode)

public ContentRecordKeyITO
getPublishedContentRecordKeyByDocumentIDAndLocale(String docid, String localecode);

public ContentRecordKeyITO
getPublishedContentRecordKeyByDocumentIDAndLocale(String docid, String localecode,
String activityType);

public ContentRecordITO getPublishedContentRecordByContentIDAndLocale(String contentid,
String localecode);

public ContentRecordITO
getPublishedContentRecordByContentIDAndLocaleAndIncrementViewCount(String contentid,
String localecode, String activitytype);

public ContentRecordITO
getPublishedContentRecordByDocumentIDAndLocale(String docid, String localecode);

public ContentRecordITO
getPublishedContentRecordByDocumentIDAndLocaleAndIncrementViewCount(String docid,
String localecode, String activitytype);
```

## Methods that Return Lists of Objects

```
public List<ContentRecordDataITO> getPublishedContentRecordDataITOs (
String channelrefkey,
String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
Date, updateSinceDate,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
List<String> caseLinkCaseValues,
int maxrecords);

public List<ContentRecordExtendedITO> getPublishedContentRecordExtendedITOs ( String
channelrefkey,
String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
Date, updateSinceDate,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
```

```
List<String> caseLinkCaseValues,
int maxrecords);

public List<ContentRecordDataITO>
getPublishedContentRecordDataITOsByContentIDs (List<String> contentids,
String localecode, int maxrecords);

public List<ContentRecordExtendedITO>
getPublishedContentRecordExtendedITOsByContentIDs (List<String> contentids, String
localecode, int maxrecords);

public List<ContentRecordDataITO>
getPublishedSortedContentRecordDataITOs (
String channelrefkey,
String String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
Date updateSinceDate,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
List<String> caseLinkCaseValues,
boolean ascending,
List<ContentRecordSortField> orderings,
int maxrecords);

public List<ContentRecordExtendedITO> getPublishedSortedContentRecordExtendedITOs (
String channelrefkey,
String String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
Date updateSinceDate,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
List<String> caseLinkCaseValues,
boolean ascending,
List<ContentRecordSortField> orderings,
int maxrecords);

public List<ContentRecordExtendedITO> getPublishedSortedContentRecordExtendedITOs (
String channelrefkey,
String String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
Date updateSinceDate,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
List<String> caseLinkCaseValues,
boolean ascending,
List<ContentRecordSortField> orderings,
int maxrecords);

public List<ContentRecordDataITO>
```

```
getPublishedSortedContentRecordDataITOsForFilterTerm(
String channelrefkey,
String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
Date updateSinceDate,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryRefKey,
List<String> usergroupRefKey,
List<String> caselinkCaseValues,
boolean ascending,
List<ContentRecordSortField> orderings,
int maxrecords,
String filter Term);

public List<ContentRecordExtendedITO>
getPublishedSortedContentRecordExtendedITOsForFilterTerm(
String channelrefkey,
String String localecode,
boolean displayDatesValidNow,
Date startDate,
Date endDate,
ContentRecordDateRangeFilterMode mode,
Date updateSinceDate,
List<String> viewrefkey,
Boolean useHierarchicalCategories,
List<String> categoryrefkey,
List<String> usergrouprefkey,
List<String> caseLinkCaseValues,
boolean ascending,
List<ContentRecordSortField> orderings,
int maxrecords);

public list<ContentRecordDataITO>
getLatestSortedContentRecordDataITOsByContentIDs(List<String> contentIDs, String
localeCode, int maxRecords, boolean ascending, List<ContentRecordSortField> orderings);

public List<ContentRecordExtendedITO>
getPublishedSortedContentRecordExtendedITOsByContentIDs(List<String> contentids, String
localecode, int maxrecords, boolean ascending, List<ContentRecordSortField> orderings);

public List<ContentRecordDataITO>
getPublishedSortedContentRecordDataITOsByContentIDs(List<String> contentIDs, String
localecode, int maxrecords, boolean ascending, List<ContentRecordSortField> orderings);

public List<ContentRecordExtendedITO>
getPublishedContentRecordExtendedITOsByDocumentIDs(List<String> docids, String
localecode, int maxrecords);

public List<ContentRecordDataITO>
getPublishedContentRecordDataITOsByDocumentIDs(List<String> docids, String localecode,
int maxrecords);

public List<ContentRecordDataITO>>
getPublishedSortedContentRecordDataITOsByDocumentIDs(List<String> docids, String
localecode, int maxrecords, boolean ascending, List<ContentRecordSortField> orderings);

public List<ContentRecordExtendedITO>>
getPublishedSortedContentRecordExtendedITOsByDocumentIDs(List<String> docids, String
localecode, int maxrecords, boolean ascending, List<ContentRecordSortField> orderings);
```

## Content Record Case Link Methods

This family of methods returns lists of CaseLinkxxxITO objects. The FULL mode returns CaseLinkITO objects, the DATA mode returns CaseLinkDataITO objects, and the KEY mode returns CaseLinkKeyITO objects. These methods are used to get more information about the case links associated with a content record.

## Content Record Methods that Create or Modify Data

This family of methods performs some type of action on a content record. These actions are checked for security based on the security roles assigned to the user executing the methods. These methods can cause inbox tasks to be generated and notifications to be sent.

### WORKFLOW METHODS

The following methods are used to push content records through the workflow process defined for the content channel. The approve and reject methods can only approve or reject to the next sequential step in the workflow (i.e. these methods can't skip steps either forward or backward). There is no feedback from the approve or reject methods that indicates whether the workflow has been completed. Adding a comment to a document in workflow does not advance the workflow step. There currently is no provision to specify which step the workflow is being approved to or rejected back to. The next step or previous step will be set based on any defined workflow step conditions (if any). If none are defined, then the next sequential step defined in the workflow is used.

```
public boolean approve(ContentRecordKeyITO content, String comments);
public boolean reject(ContentRecordKeyITO content, String comments);
public boolean addComment(ContentRecordKeyITO content, String comments);
```

### CONTENT METHODS

These methods are used to create or modify content in the IM repository. Some methods take the input parameter ContentRecordKeyITO; a light weight object intended to identify an existing content record. Others take the input parameter ContentRecordITO which contains the raw data that will be used to create or modify the content records. A ContentRecordITO has numerous values such as dateadded, datemodified, documentid, contentid, etc. that are ignored if they are provided while creating content. In a similar manner, changing attributes such as documentid and locale is not permitted while modifying content. In most cases the methods will simply ignore values that are provided and should not be changed; no warnings or errors will be raised by the methods.

The ContentRecordITO is a very heavy weight object that could contain a large amount of data (case links, categories, user groups, etc). The returned ContentRecordITO object from createContent(), modifyContent(), translateContent() methods represent the changed data after the change has occurred.

**Note:** Methods that create, modify, or delete content are executed atomically. If deleteContent() is called on a document, all localized versions and point releases of that document are deleted—including any of the attached resources for that document.

The owner of a content record is the user to whom notifications are sent for workflow events.

```
public ContentRecordITO createContent(ContentRecordITO content, boolean publish);
public ContentRecordITO modifyContent(ContentRecordITO content, boolean publish);
public boolean deleteContent(String contentid);
public ContentRecordITO translateContent(ContentTranslationITO translationITO, Boolean
```

```
publish);

public ContentRecordITO saveAsNew(ContentRecordKeyITO content, boolean publish);

public boolean changeOwnerForDocumentID(String documentID, UserKeyITO owner, String
localecode);

public boolean changeOwnerForContentRecord(ContentRecordKeyITO content, UserKeyITO
owner);
```

## CASE LINKING METHODS

These methods are used to add and remove case links from content records. A case link is a reference to a CRM case where the underlying IM document was linked to a CRM case. This is usually done from the CRM UI where Oracle Knowledge products are integrated into the UI. The typical workflow is that the CRM agent creates a new support case while talking to a customer. While searching the knowledgebase to resolve a customer issue, the CRM agent has the ability to link a specific knowledgebase article to the CRM case. This linkage is important in understanding the value of a knowledgebase article—the more links an article accrues, the more relevant and valuable the article becomes.

```
public boolean addCaseLinkList (ContentRecordKeyITO content, List<CaseLinkDataITO>
links, List<Integer> incidents);

public CaseLinkITO createCaseLink(CaseLinkITO caselink, CaseLinkContentITO
caseLinkContent, ContentRecordKeyITO contentRecord);

public CaseLinkITO deleteCaseLink(CaseLinkITO caselink, ContentRecordKeyITO
contentrecord);
```

## IQLocaleRequest

The IQLocaleRequest object is obtained from the IQServiceClient by executing the following code to obtain a reference to the IQServiceClient from the IQServiceClientManager. The IQServiceClientManager is a static class that can be accessed with the connect() method. The result of a successful connection is a valid IQServiceClient object that can be used to retrieve an IQContentRecordRequest object. This same pattern is used to obtain an IQxxxRequest in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQLocaleRecordRequest request = client.getLocaleRecordRequest();
...
```

An IM repository has a default locale assigned to it. If a locale is not provided as part of a method call, in many cases the default locale for the repository is used. A repository can support multiple locales concurrently, allowing users to create content in any supported locale.

Each IM locale includes information about the locale including the time and date format used to display information, the required character encoding, and a flag that indicates which locale would be used as the default locale if there is more than one locale defined for a given language (i.e. en\_US, en\_GB the group default might be set as en\_US for U.S customers).

## Related ITOs

The IQLocaleRequest utilizes the LocaleKeyITO and LocaleITO classes as return types.

The LocaleKeyITO is the lightest weight object available. It provides the recordid of the LOCALE table record. It can be used as a parameter or returned by one of the get() methods. This object does NOT contain any other data or relationships. Where possible use methods that utilize the KeyITO object.

The LocaleITO inherits from LocaleKeyITO . It adds the additional properties from the LOCALE table as data with get/set methods for each property. This is also a lightweight object that only requires a single database fetch to populate. There are no embedded relationships included with this ITO.

```
public LocaleKeyITO getLocaleKeyByLocaleCode(String localecode);
public LocaleKeyITO availableLocaleListForRepository(String repositoryrefkey);
public LocaleITO getLocaleByLocaleCode(String localecode);
public LocaleITO getRepositoryDefaultLocale(String repositoryrefkey);
public LocaleITO getSystemDefaultLocale();
public LocaleITO availableLocaleListForRepository(String repositoryReferenceKey);
```

## IQSecurityRoleRequest

The IQSecurityRoleRequest object is obtained from the IQServiceClient by executing the following code to obtain a reference to the IQServiceClient from the IQServiceClientManager. The IQServiceClientManager is a static class that can be accessed with the connect() method. The result of a successful connection is a valid IQServiceClient object that can be used to retrieve an IQSecurityRoleRequest object. This same pattern is used to obtain an IQxxxRequest in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQSecurityRoleRequest request = client.getSecurityRoleRequest();
...
```

The security roles in an IM repository are broadly split into 2 groups: Console roles and Web roles. Console roles should only be assigned to users that need to access the IM management console for creating or managing content or to administrators that are responsible for overall system maintenance activities.

Console based security roles are composed of a number of different components:

- A list of permissions that enable access to the various menus within the IM management console
- A list of privileges assigned to the role for each content channel (add, edit, delete, translate, etc)
- A list of data form results that the security role can access
- A list of workflow steps that the security role can approve
- A list of user groups that the security role can access

Web-based security roles are composed only of the user groups to which the security role is allowed access.

A user can be assigned one or more security roles. The net security realm for a user is the sum total collection of all permissions and privileges associated with all of the security roles assigned to the user.

## Related ITOs

The IQSecurityRoleRequest utilizes the SecurityRoleKeyITO, SecurityRoleDataITO, and SecurityRoleITO classes as return types. Some of the methods in this request also use UserKeyITO objects as input parameters.



The SecurityRoleKeyITO is the lightest weight object available. It provides the recordid and reference key from the SECURITYROLE table.

The SecurityRoleDataITO inherits from SecurityRoleKeyITO . It adds the additional properties from the SECURITYROLE table as data and has get/set methods for each property. This is also a lightweight object that only requires a single database fetch to populate. There are no embedded relationships included with this ITO.

The SecurityRoleITO inherits from SecurityRoleDataITO . It adds the additional properties and relationships from the SECURITYROLE table as data with get/set methods for each property. This ITO can return a large amount of data if there are a lot of users in the system assigned to a specific security role , such as when all users in a system are assigned to the 'guest' user role.

The UserKeyITO contains the recordid from the USERINFORMATION table. This is a very lightweight object representing a user in the IM repository. This user can be either a console user or a web user.

```
public List<SecurityRoleKeyITO>
getSecurityRoleKeysForRepository(String repositoryrefkey, int limit);

public List<SecurityRoleKeyITO>
getSecurityRoleKeysForUser(UserKeyITO user);

public List<SecurityRoleDataITO>
getSecurityRoleDatasForRepository(String repositoryrefkey, int limit);

public List<SecurityRoleDataITO>
getSecurityRoleDatasForUser(UserKeyITO user);

public List<SecurityRoleITO>
getSecurityRolesForRepository(String repositoryrefkey, int limit);

public List<SecurityRoleITO>
getSecurityRolesForUser(UserKeyITO user);
```

## IQUserGroupRequest

The IQUserGroupRequest object is obtained from the IQServiceClient by executing the following code to obtain a reference to the IQServiceClient from the IQServiceClientManager. The IQServiceClientManager is a static class that can be accessed with the connect() method. The result of a successful connection is a valid IQServiceClient object that can be used to retrieve an IQUserGroupRequest object. This same pattern is used to obtain an IQxxxRequest in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQUserGroupRequest request = client.getUserGroupRequest();
...
```

By default, user groups are defined at the Repository level. User groups can be assigned to one or more Repository Views. Assigning a user group to a view is typically done when Views represent a department or business unit and the content authors need to create content targeted for a specific group of users to utilize. The list of User Groups assigned to a View is used to filter the choices available to the content author when creating or editing a record.

User groups can be assigned to channels. Assigning a user group to a channel is used to filter the list of user groups that can be assigned to a content record. When user groups are assigned to both channels and views, the list of available user groups that can be assigned to a content record is filtered first by list assigned to the View and then by the list of user groups assigned to the Channel.

User groups can be assigned to security roles. Assigning a user group to a security role indicates the user group security that users will be granted if they are assigned that role. The list of user groups assigned to a security role is used to filter the list of documents the user is allowed access.

## Related ITOs

The `IUserGroupRequest` utilizes the `UserGroupKeyITO`, `UserGroupDataITO`, and `UserGroupITO` classes as return types.

The `UserGroupKeyITO` is the lightest weight object available. It provides the recordid and reference key from the TAG table. The TAG table contains both categories and user group information. The `UserGroupxxx` objects represent the security required to access the document in the runtime environment.

The `UserGroupDataITO` inherits from `UserGroupKeyITO`. It adds the additional properties from the TAG table as data with get/set methods for each property. This is also a lightweight object that only requires a single database fetch to populate. There are no embedded relationships included with this ITO.

The `UserGroupITO` inherits from `UserGroupDataITO`. It adds the additional properties and relationships from the TAG table as data with get/set methods for each property. This ITO can return a large amount of data if there are a lot of user groups in the system.

Additional ITOs used as input parameters:

\*`SecurityRoleKeyITO` - This ITO represents the security role assigned to a user.

The primary difference between the available methods is the type of objects they return. Where possible it is usually a better choice to utilize the `xxxKeyITO` objects.

```
public List<UserGroupKeyITO>
getUserGroupKeyListForView(String viewrefkey, int limit);

public List<UserGroupKeyITO>
getAllowedUserGroupKeyListForChannel(String channelrefkey, int limit);

public List<UserGroupKeyITO>
getUserGroupKeyListForSecurityRole (SecurityRoleKeyITO securityRole, int limit) ;

public List<UserGroupDataITO>
getAllowedUserGroupDataListForChannel(String channelrefkey, int limit);

public List<UserGroupDataITO>
getUserGroupDataListForView(String viewrefkey, int limit);

public List<UserGroupDataITO>
getUserGroupDataListForSecurityRole (SecurityRoleKeyITO securityRole, int limit) ;

public List<UserGroupITO>
getUserGroupListForView(String viewrefkey, int limit);

public List<UserGroupITO>
getUserGroupListForSecurityRole (SecurityRoleKeyITO securityRole, int limit) ;
```

## IUserRequest

The `IUserRequest` object is obtained from the `IQServiceClient` by executing the following code to obtain a reference to the `IQServiceClient` from the `IQServiceClientManager`. The `IQServiceClientManager` is a static class that can be accessed with the `connect()` method. The result of a successful connection is a valid `IQServiceClient` object that can be used to retrieve an `IUserRequest` object. This same pattern is used to obtain an `IQxxxRequest` in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IUserRequest request = client.getUserRequest();
...
```

The `IUserRequest` is the primary interface to methods that allow access to the IM user records. Users in IM are divided into two broad categories: Console users and Web Users. Console users are users that have permission to access the IM management console. Console users typically include content authors, administrators, and system support personnel. Web users can access content created in the IM repository but do not have any permissions to access the IM management console or to maintain the IM repository. Web users are typically customers, employees, or partners that access the knowledge base.

Console users can be assigned one or more IM security roles.

Console users can be grouped into workteams. Workteams are used to simplify task assignment.

An IM repository can be configured to support a user reputation model. The reputation model is used to reward console users for performing activities that are beneficial to the company, such as creating knowledge articles that are highly rated by users or used to resolve customer support cases.

## Related ITOs

The `IUserRequest` utilizes the `UserKeyITO`, `UserDataITO`, and `UserITO` classes as return types. This request also utilizes a number of other ITO objects as input parameters for some of its methods.

The `UserKeyITO` is the lightest weight object available. It provides the `RECORDID` from the `USERINFORMATION` table. The `USERINFORMATION` table contains all of the relevant user information and demographic data that IM collects.

The `UserDataITO` inherits from `UserKeyITO`. It adds the additional properties from the `USERINFORMATION` table as data and has `get/set` methods for each property. This is also a lightweight object that only requires a single database fetch to populate. There are no embedded relationships included with this ITO.

The `UserITO` inherits from `UserDataITO`. It adds the additional properties and relationships from the `USERINFORMATION` table as data with `get/set` methods for each property. This ITO can return a large amount of data if there is a lot of user related data in the system.

Additional ITOs used as input parameters:

- `SecurityRoleKeyITO` - This ITO represents the security role assigned to a user.
- `ViewKeyITO` - This ITO represents the view to which the user is assigned
- `WorkTeamKeyITO` - This ITO represents the workteam the user is assigned to
- `ReputationLevelITO`, `ReputationLevelDataITO`, `ReputationLevelKeyITO` - This family of ITO objects represents the reputation level information for the specified user.

## Methods that Change User Information

This family of methods can be used to redefine a user or to manage a user. Methods that use a `UserITO` as an input parameter (`createUser()` and `updateUser()`) will take the values passed in and either create or modify the existing attributes of the user with the new data. The returned `UserITO` object should reflect the new changes to the user.

```
public UserITO createUser(UserITO user);
public UserITO updateUser(UserITO user);
```

```
public boolean deleteUser(String login);
public boolean lockUser(String login);
public boolean unlockUser(String login);
public boolean addUserReputationPoints(String login, int points);
public boolean addUsersToRole(SecurityRoleKeyITO role, List<UserKeyITO> users);
public boolean addUsersToView(ViewKeyITO view, List<UserKeyITO>);
public boolean addUsersToWorkTeam(WorkTeamKeyITO workteam, List<UserKeyITO>);
public boolean removeUsersFromRole(SecurityRoleKeyITO role, List<UserKeyITO> users);
public boolean removeUsersFromWorkTeam(WorkTeamITO workteam, List<UserKeyITO> users);
```

## Methods that Return Information about Users

This family of methods is used to return information about one or more users. These methods are typically used when searching for a specific user or a group of users matching a set of criteria.

```
public boolean isEmailDuplicate(String email, String repositoryrefkey);
public boolean isUserIDTaken(String login, String repositoryrefkey);
public UserKeyITO getUserKeyByLogin(String login);
public UserDataITO getUserDataByLogin(String login);
public UserDataITO getUserDataForID(String userid);
public UserITO getUserByLogin(String login);
public UserITO getUserForID(String userid);
public UserITO getUserForLoginAndPassword(String login, String password);
public ReputationLevelITO getUserReputation(String login);
public ReputationLevelDataITO getUserReputationData(String login);
public ReputationLevelKeyITO getUserReputationKey(String login);
public List<UserKeyITO> userKeyListByEmail(String email, int limit);
public List<UserKeyITO> userKeyListBySkills(String[] categoryrefkey, int limit);
public List<UserKeyITO> userListKeyByLoginArray(String[] logins);
public List<UserDataITO> userListDataByLoginArray(String[] logins);
public List<UserDataITO> userDataListByEmail(String email, int limit);
public List<UserDataITO> userDataListBySkills(String[] categoryrefkey, int limit);
public List<UserITO> userListByEmail(String email, int limit);
public List<UserITO> userListByLoginArray(String[] logins);
public List<UserITO> userListBySkills(String[] categoryrefkeys, int limit);
public List<UserITO> usersForRole(String repositoryrefkey, String rolerefkey, int limit);
public List<UserITO> usersForView(String viewrefkey, int limit);
public List<UserITO> usersForWorkTeam(String workteamrefkey, int limit);
```

## IQViewRequest

The IQViewRequest object is obtained from the IQServiceClient by executing the following code to obtain a reference to the IQServiceClient from the IQServiceClientManager. The IQServiceClientManager is a static class that can be accessed with the connect() method. The result of a successful connection is a valid IQServiceClient object that can be used to retrieve an IQViewRequest object. This same pattern is used to obtain an IQxxxRequest in general for all requests.

```
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQViewRequest request = client.getViewRequest();
...
```

The IQViewRequest is the primary service entry point for programmers to work with IM repository views. A repository view is a subset of an IM repository. The data for a View is stored in the SITE table in the IM database schema. The primary difference between a repository and a view in the SITE table is that a view has a parent whereas a repository object does not.

## Related ITOs

The IQViewRequest utilizes the ViewKeyITO, ViewDataITO, and ViewITO classes as return types. This request also utilizes a number of other ITO objects as input parameters to some of its methods.

The ViewKeyITO is the lightest weight object available. It provides the RECORDID and REFERENCEKEY from the SITE table.

The ViewDataITO inherits from ViewKeyITO. It adds the additional properties from the SITE table as data and has get/set methods for each property. This is also a lightweight object that only requires a single database fetch to populate. There are no embedded relationships included with this ITO.

The ViewITO inherits from ViewDataITO. It adds the additional properties and relationships from the SITE table as data and has get/set methods for each property. This ITO can return a large amount of data if there is a lot of user or content related data in the system for the specified view.

Additional ITOs used as input parameters:

- UserKeyITO, UserITO - These objects represent the users associated with the specified ViewITO,

## Methods that Change View Information

This family of methods is used to modify information contained in a view or relationships associated with a view. If a view is removed from a repository, all of the content that was assigned to the view will be re-assigned to the root repository.

```
public boolean addUsersToView(ViewKeyITO view, List<UserITO> users);
public boolean addViewsToParent(ViewKeyITO parentview, List<ViewITO> views);
public boolean removeUsersFromView(ViewKeyITO view, List<UserKeyITO> users);
public boolean removeViewsFromParent(ViewKeyITO parentview, List<ViewKeyITO> views);
```

## Methods that Return Information about Views

This family of methods is used to return information about a view or a list of views based on filter criteria.

```
public ViewKeyITO getViewKeyByReferenceKey(String ReferenceKey);
```

```

public ViewDataITO getViewDataByReferenceKey(String ReferenceKey);
public ViewITO getViewByReferenceKey(String ReferenceKey);
public List<ViewKeyITO> getViewKeysForParentView(String parentviewrefkey, int limit);
public List<ViewKeyITO> getViewKeysForRepository(String repositoryrefkey, int limit);
public List<ViewKeyITO> getViewKeysForUser(UserKeyITO user, int limit);
public List<ViewDataITO> getViewDatasForParentView(String parentviewrefkey, int limit);
public List<ViewDataITO> getViewDatasForRepository(String repositoryrefkey, int limit);
public List<ViewDataITO> getViewDatasForUser(UserKeyITO user, int limit);
public List<ViewITO> getViewsForParentView(String parentviewrefkey, int limit);
public List<ViewITO> getViewsForRepository(String repositoryrefkey, int limit);
public List<ViewITO> getViewsForUser(UserKeyITO user, int limit);

```

## IQWorkTeamRequest

The IQWorkTeamRequest object is obtained from the IQServiceClient by executing the following code to obtain a reference to the IQServiceClient from the IQServiceClientManager. The IQServiceClientManager is a static class that can be accessed with the connect() method. The result of a successful connection is a valid IQServiceClient object that can be used to retrieve an IQWorkTeamRequest object. This same pattern is used to obtain an IQxxxRequest in general for all requests.

```

...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQWorkTeamRequest request = client.getWorkTeamRequest();
...

```

Workteams are used when there are multiple content authors that can share similar responsibilities in terms of completing tasks in the inbox of the IM management console. If workteams are used, the assignment of tasks can be greatly simplified and reduce the amount of manual intervention required to assign tasks to users.

Workteams can be used as part of the conditional workflow evaluation process. Users can be part of zero or more workteams.

## Related ITOs

The IQWorkTeamRequest utilizes the WorkTeamKeyITO, WorkTeamDataITO, and WorkTeamITO classes as return types.

The WorkTeamKeyITO is the lightest weight object available. It provides the RECORDID and REFERENCEKEY from the WORKTEAM table.

The WorkTeamDataITO inherits from WorkTeamKeyITO. It adds the additional properties from the WORKTEAM table as data and has get/set methods for each property. This is also a lightweight object that only requires a single database fetch to populate. There are no embedded relationships included with this ITO.

The WorkTeamITO inherits from WorkTeamDataITO. It adds the additional properties and relationships from the WORKTEAM table as data and has get/set methods for each property. This ITO can return a large amount of data if there is a lot of user or content related data in the system for the specified workteam.

```

public List<WorkTeamKeyITO> workTeamListKeyForRepository(String repositoryrefkey, int
limit);

public List<WorkTeamDataITO> workTeamListDataForRepository(String repositoryrefkey, int
limit);

public List<WorkTeamITO> workTeamListForRepository(String repositoryrefkey, int
limit);

```

## IQContentRecommendationRequest

The `IQContentRecommendationRequest` object is obtained from the `IQServiceClient` by executing the following code to obtain a reference to the `IQServiceClient` from the `IQServiceClientManager`. The `IQServiceClientManager` is a static class that can be accessed with the `connect()` method. The result of a successful connection is a valid `IQServiceClient` object that can be used to retrieve an `IQContentRecommendationRequest` object. This same pattern is used to obtain an `IQxxxRequest` in general for all requests.

```

...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQContentRecommendationRequest request = client.getContentRecommendationRequest();
...

```

IM allows users to submit a recommendation to improve gaps in existing content articles or to suggest new topics for inclusion in the knowledgebase. Once a content recommendation is submitted, workflow tasks are generated and placed into the queue for someone to address. A content recommendation can be accepted, and a new article can be generated (or an existing article can be updated) based on the recommendation.

The current client library service request only allows the retrieval of submitted recommendations.

## Related ITOs

The `IQContentRecommendationRequest` utilizes the `ContentRecommendationKeyITO`, `ContentRecommendationDataITO`, and `ContentRecommendationITO` classes as return types.

The `ContentRecommendationKeyITO` is the lightest weight object available. It provides the `RECORDID` from the `CONTENTRECOMMENDATION` table.

The `ContentRecommendationDataITO` inherits from `ContentRecommendationKeyITO`. It adds the additional properties from the `CONTENTRECOMMENDATION` table as data and has `get/set` methods for each property. This is also a lightweight object that only requires a single database fetch to populate. There are no embedded relationships included with this ITO.

The `ContentRecommendationITO` inherits from `ContentRecommendationDataITO`. It adds the additional properties and relationships from the `CONTENTRECOMMENDATION` table as data and has `get/set` methods for each property.

This service utilizes additional ITO objects as input parameters to several methods:

- `CategoryKeyITO` - This object represents a category associated with the content recommendation.
- `RepositoryKeyITO` - This object represents the repository associated with the content recommendation.
- `ContentChannelKeyITO` - This object represents the channel associated with the content recommendation.

```

public List<ContentRecommendationKeyITO>
getContentRecommendationKeyListForChannel(String channelrefkey, int limit);

```

```

public List<ContentRecommendationDataITO>
getContentRecommendationDataListForChannel(String channelrefkey, int limit);

public List<ContentRecommendationITO> getContentRecommendationListForChannel(String
channelrefkey, int limit);

```

## IQRatingRequest

The IQRatingRequest object is obtained from the IQServiceClient by executing the following code to obtain a reference to the IQServiceClient from the IQServiceClientManager. The IQServiceClientManager is a static class that can be accessed with the connect() method. The result of a successful connection is a valid IQServiceClient object that can be used to retrieve an IQRatingRequest object. This same pattern is used to obtain an IQxxxRequest in general for all requests.

```

...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos, imUrl,
searchUrl, throwExceptionOnError);
IQRatingRequest request = client.getRatingRequest();
...

```

IM provides the ability to rate content and discussion forums. Each rating can be composed of one or more questions. Each question can have one or more answers (i.e. a multi-select listbox, checkboxes, radio buttons, textboxes) associated with each question. The definition of the survey/rating is stored in the SURVEY, SURVEYQUESTION, and SURVEYANSWER tables in the IM database schema.

When a user submits a rating against a discussion topic or a content record, the results are stored in the SURVEYRESULTS and SURVEYRESULTSDETAIL tables. Each question that is answered is recorded in the SURVEYRESULTSDETAIL table. The SURVEYRESULT table captures who submitted the survey and when.

Most implementations try to avoid allowing users to submit a survey response more than time. There is a method available to validate whether the user has already submitted a response.

In some UI implementations, there may be a link provided to the user to see the average rating or number of ratings submitted. This information is aggregated and can be returned in the AggregateFormResultsITO object. The list of individual responses can also be returned in the FormResultsITO object.

**Note:** It is quite possible for a busy site to have a large number of ratings submitted for a specific document. Where possible use the fetchlimit argument to limit the amount of data being returned from the query.

## Related ITOs

The IQRatingRequest utilizes the RatingITO, FormResultsITO, and AggregateFormResultsITO classes as return types.

The RatingITO represents the definition of a rating. This information is stored in the SURVEY table in the IM database schema.

The FormResultsITO represents an answer submitted for a specific rating/survey by a single user. The FormResultsITO contains references to the answers for each question in the survey.

The AggregateFormResultsITO represents a summarized view of the submitted responses for a specific survey or rating.

```

public boolean hasUserRatedContent(String docid, String userlogin);
public boolean rateContent(String docid, FormResultsITO rating);
public RatingITO getRatingDefinition(String surveyrefkey);

```



```
public RatingITO getRatingDefinition(String surveyrefkey, String localecode);  
public RatingITO getContentRatingDefinition(String docid);  
public List<FormResultsITO> getDetailContentRatingResults(String docid, int limit);
```

---

# Intelligent Search API Overview

---

The Oracle Knowledge Intelligent Search Client Library provides a wrapper around the existing SOAP based interfaces deployed as part of the Search runtime software. The Search Client Library has been designed to abstract the remote procedure calls necessary to use the SOAP interfaces and to simplify the management of state based data between calls to the services.

At a high level the Search API is broken into 3 broad categories of methods: Question answering/processing methods, Contact Center interaction methods, and Process Wizard interactions.

**Note:** The Process Wizard feature is deprecated and replaced by the AnswerFlow component. See the AnswerFlow Implementation and User's Guide, located in the Oracle Knowledge Intelligent Search User Documentation Library, for more information.

Most of the Search API methods return a GIML object. The GIML (Graphical Interface XML) object describes the result set that is returned. The contents of the GIML will vary based on the API method being called.

The first method that needs to be called after the login() call is the initialScreen(). This method initializes the search session and provides a context for subsequent calls to the Search runtime.

In order to maintain the correct order of events within a session a transaction ID is generated with each response GIML, it is necessary to pass the transaction ID from the previous method call as a parameter (priorTransactionid) to subsequent method calls to ensure that the correct order of activity is properly recorded for the Analytic reports.

## IQServiceClient

The IQServiceClient object has several methods available in it that pertain to the Search API and are ignored with the Information Manager API calls. These xxxInfo objects are used to set parameters that can be accessed by the search runtime engine while evaluating the search behavior and language rules configured for a customer's implementation.

- getCCAInfo() - com.inquirasearch.CCAInfo
- getClientInfo() - com.inquirasearch.ClientInfo
- getSearchInfo() - com.inquirasearch.SearchInfo
- getUserInfo() - com.inquirasearch.UserInfo

To use the specific xxxInfo classes to set parameters that can be added to the search request the following code snippet illustrates the general procedure

```
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos,
imUrl, searchUrl, throw ExceptionOnError);
client.getSearchInfo().setResultLanguages("en-US,de-DE,fr-FR");
client.getSearchInfo().setLanguage("en-US");
```

After setting the required parameters using the IQServiceClient - subsequent calls to the IQSearchRequest service will utilize those parameters during the Search session to process each incoming request. The values are persistent for the life of the IQServiceClient or until they are reset.

## SessionID, TransactionID

The Search Client Library was designed to simplify the management of the search session and transaction IDs. Prior to the Client Library framework, it was necessary to pass the transaction and session information as part of the request. The Search Library framework stores this information in the SearchInfo object and passes it along with each request without requiring any additional interaction.

The IQServiceClient provides a convenience method that can be used to retrieve the transaction ID from the GIML after the response has been received.

The following code sample shows how to access the transaction ID and pass it to other search API calls:

```
IQSearchRequest iqSR = client.getSearchRequest();
//submit query
String query = "How do I play the piano?";
GIML answers = iqSR.askQuestion(query, true);
System.out.println(answers.toString());
int transactionID = answers.transactionID;
//Define input variables for feedback - would normally get from the UI
//transactionID gets put into the URL as the priorTransactionID
int priorTransactionID= transactionID;
```

**Note:** There are some differences between the Information Manager and Search API in terms of passing parameters into the methods. IM typically uses arrays to pass in multiple values of the same parameter while the Search API typically uses a comma separated string such as "en-US,de-DE,fr-FR". In addition, the locale codes are formatted slightly differently between Search and Information Manager. Search uses a format such as en-US, while Information Manager uses a format en\_US.

## CCAInfo

The CCAInfo object contains parameters relating to the Call Center Adaptor functionality.

Parameter	Description
<b>Name</b>	Request type name, which defines the chain of handlers, used to process the specified request. The value should be one of the request types specified in the #.xml. For example: <code>getCCAInfo().setTypes("CCAGetPage");</code>
<b>Connected</b>	This property is no longer in use as of 8.1.3.
<b>Case Description</b>	Case summary description of the CRM case being linked. By default this is the question used to return answers for the case.
<b>ExtSolutionList</b>	Not used since 8.1.3. Kept for backward compatibility.
<b>SRKey</b>	Service request case number.
<b>System</b>	Configurable property that specifies the type of CRM system. This is configured in System Manager.
<b>Content IDs</b>	List of IM document IDs that would be linked to IM side for case links.
<b>Answer solution list</b>	Used to replace ExtSolutionList after 8.1.3. Each solution ID is in the format of "answerid:docid" that identifies a specific answer document ( answer id or docid are both search terms ). The list of solution ids is separated by comma.
<b>Types</b>	This property is no longer in use as of 8.1.3

## ClientInfo

The ClientInfo object contains parameters passed in from the client that include HTTP header information. These values are initialized with each request via a helper class on the Client Library (not the server side). They are not typically overridden. Context parameters that are passed into the search runtime are usually sent via the request parameter map.

Parameter	Description
<b>agentAddress</b>	This is the original request URL as returned from <code>HttpRequest.getRequestURL()</code> method.
<b>processorAddress</b>	This is the URL to the search runtime answering the search request. This is typically passed in during the initial <code>connect()</code> call. This is the address of the <code>inquiragw</code> process.
<b>extSessionID</b>	This parameter contains the external session identifier - i.e. the <code>JSESSIONID</code> .
<b>referrer</b>	The HTTP request header field that allows the client to specify the address of the resource from which the URI was obtained.
<b>cookies</b>	The map of the available cookies for the current HTTP request.
<b>address</b>	The remote address from which the request originated.
<b>host</b>	The remote host name from which the request originated.
<b>request header map</b>	A map containing all of the request header values that were present in the original HTTP request.
<b>request parameter map</b>	A map containing all of the available URI parameters that are passed in the HTTP request. This parameter map can include security restrictions such as <code>agent.parameters.restriction.level.foo0=COLLECTIONS.inquiragw</code> or <code>agent.parameters.restriction.level.foo1=DOC_TYPES.HTML</code> .  The <code>setRequestParameterMap()</code> method prefixes all request parameters that you add to this map with the string <code>'user-agent.parameters.'</code> . See xref for information on implementing the <code>iRequestModifier</code> interface, which enables you to modify the request parameters and remove the prefix string so that the application can process the requests correctly.
<b>request attribute map</b>	A map containing all of the available HTTP request attributes.

The ClientInfo object passes information that you can use with custom security plugins to enforce IM user group restrictions. For example, the following code snippet sets some custom parameters, (`IM.Restrict.View`, `IM.RestrictUserGroup`, `myCustomAttribute`) in the ClientInfo object. The custom ClientInfo object is automatically passed with each search request (`askQuestion`) from the Client Library framework.

```
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos,
imUrl, searchUrl, throwExceptionOnError);

Map myMap = new Map();
myMap.put("IM.Restrict.View", "SAMPLE.MYCOMPANY.ALL_OTHER");
myMap.put("IM.RestrictUserGroup", "MYCOMPANY.EXTERNAL");
myMap.put("myCustomAttribute", "A_B_C");
client.getClientInfo().setRequestParameterMap(myMap);

client.getSearchRequest().askQuestion("How do I configure my router?", false);
```

**Important!** The `setRequestParameterMap()` method prefixes all arguments with the string `user-agent.parameters.`; however, the application cannot recognize and process the arguments using the key as defined in the previous example code. You must modify the request arguments, or remove the prefix string `'user-agent.parameters.'` from the search request, you must:

- implement custom code using the provided `com.inquiria.infra.requestprocessor.IRequestModifier` interface
- configure the application to use the custom interface

If you do not implement the custom code using the `IRequestModifier` interface, the application will not recognize the keys of the custom arguments, such as `user-agent.parameters.IM.Restrict.View` and `user-agent.parameters.IM.Restrict.UserGroup`; the application will recognize only arguments in the non-prefixed form, such as `IM.Restrict.View` and `IM.Restrict.UserGroup`.

## Implementing the `IRequestModifier` Interface to Remove the User Agent Parameters Prefix

This section provides an example of implementing the `IRequestModifier` interface. This implementation generates duplicate attributes of the arguments that have the `user-agent.parameters.` prefix, and ensures that they are in the Search Request.

```
package com.inquiria.request;

import java.util.Enumeration;
import java.util.Properties;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import com.inquiria.infra.requestprocessor.IRequestModifier;

public class RequestModifierExample implements IRequestModifier {

    /*
    * PATTERN of "^user-agent\\.parameters\\.(.*)" defines which arguments will be modified
    */
    private static final Pattern PATTERN = Pattern.compile("^user-
agent\\.parameters\\.(.*)");

    /**
    * Adds additional arguments to the request where the PATTERN matches their key
    * Each new argument has the same value, but a different key to the one matched in the
    pattern
    * @param request
    */
    @Override
    @SuppressWarnings("rawtypes")
    public void process(Request request) {
        Properties params = request.getParameters(PATTERN);
        Properties newParams = processParameters(params);
        Enumeration propertyNames = newParams.propertyNames();
        while (propertyNames.hasMoreElements()) {
            String key = (String) propertyNames.nextElement();
            String value = newParams.getProperty(key);
            request.addArgument(key, value);
        }
    }

    /**
    * Creates a list of parameters based on the PATTERN
    * The content outside of the parenthesis is removed
    * E.g., "user-agent.parameters.RequestSource" becomes "RequestSource"
    * @param params
    * @return
    */
}
```

```

*/
@SuppressWarnings("rawtypes")
private static Properties processParameters(Properties params) {
    Properties newParams = new Properties();
    Enumeration paramsEnumerator = params.propertyNames();
    while (paramsEnumerator.hasMoreElements()) {
        String key = (String) paramsEnumerator.nextElement();
        Matcher m = PATTERN.matcher(key);
        if (m.matches()) {
            newParams.put(m.group(1), params.getProperty(key));
        }
    }
    return newParams;
}
}

```

**Note:** This implementation automatically adds all request parameters in the Address bar as user-agent.parameters; these parameters are converted to normal attributes when using the example code. For example, you could add `&IM.Restrict.View=ORACLE` in the URL to access all content in the ORACLE view.

## Configuring the Intelligent Search Application to Use the Request Modifier

You configure the application to use the `iRequestModifier` implementation by adding the following configuration code to the `autoimport.xml` or `custom.xml` files in your environment's content processing instance, then making the custom configuration available to the production instance:

```

<root>
  <requestprocessor>
    <requestModifier>com.inquirira.request.RequestModifierExample</requestModifier>
  </requestprocessor>
</root>

```

Or, you can add the `<requestModifier>` configuration directly to the `<number>.xml` file; add the configuration anywhere within the `<requestprocessor>` node.

You can add the configuration as follows:

- Add the configuration to the file `autoimport.xml`, located in the *instance* folder, then re-start the instance. The application copies the configuration to the latest `<number>.xml` file.
- Add the configuration to the file `custom.xml`, located in the *instance* folder, then re-start the instance. The configuration will always be in the configuration even if it does not exist in the latest number XML file.
- Manually update the latest number XML file to include the configuration.

You can make the custom configuration available to the production environment by synchronizing the content processing and runtime instances.

## Implementing a Custom Request Modifier Class

When implementing a custom Request Modifier class, you should consider the following points:

- The `getParameters()` method of the Request object takes a compiled Pattern object, and will only return Properties that match that pattern.
- The `addArgument()` method of the Request object takes a String key and value. It is not enough to add entries to the Properties object that returns from `getParameters()`. That object only contains a sub-set of the arguments and adding to it will not add them to the Request.

- You can add any arguments to the Request; however, including security arguments within the Request increases the risk of exposing secure content to unauthorized users.

## SearchInfo

The SearchInfo object contains parameters that are used by search runtime. The following values are available:

Parameter	Description
<b>baseURL</b>	This parameter is used to replace the base URL on a per request basis. This parameter is rarely overridden using the SearchInfo class.
<b>language</b>	This parameter is used to specify the language the question is being asked in. This parameter can be changed for each request by resetting the SearchInfo.setLanguage().
<b>resultLanguages</b>	This parameter is used to specify which languages are valid to return results in for the specified question.
<b>domainGroup</b>	This parameter allows the domain group to be specified per Search request. A domain group is a set of rules and concepts that are logically grouped together in the Dictionary. If not specified a predefined domain group resolution protocol is used to determine the correct domain group to use.
<b>navigationApplicationId</b>	This parameter specifies the collection of facets that will be used for this request.
<b>segment</b>	This parameter is passed thru to the analytics log. This parameter is only used to specify the user segment that the request should be associated with.
<b>querySource</b>	This parameter is only used in analytics. If used to group activities in the analytics reports. This string usually represents some action the user took. i.e. SearchClickthru.
<b>uiMode</b>	The UI mode is not typically set by programmers. This parameter indicates whether this is an initial request response or a search within a previous result set.
<b>requestSource</b>	This parameter is not typically used or changed by programmers. Represents the URL where the request came from.
<b>subject</b>	If the domainGroup is not specified then the subject and language are looked up in a map that helps resolve the required domain group to use for the search request. If there is not a current mapping the default domain group is used instead.
<b>Session</b>	Contains the session information. This parameter is not typically set manually—the Client Library framework manages it automatically.
<b>transactionId</b>	Contains the transaction ID for the last request. This parameter is not typically set manually—the client Library framework maintains it automatically.

## UserInfo

The UserInfo object contains parameters that contain user specific related parameters. These parameters are typically set after the IQServiceClient.connect() is called and the user has been authenticated.

Parameter	Description
<b>displayName</b>	The display name (combination of first and last name).
<b>id</b>	The GUID from the USERINFORMATION table if the user has been authenticated from Information Manager.
<b>login</b>	The user login ID.
<b>repositoryRefKey</b>	The Information Manager repository reference key the user is associated with.
<b>defaultViewRefKey</b>	The default Information Manager repository view the user is associated with. If the user is not assigned to a view the root repository is assumed.

Parameter ( <i>continued</i> )	Description ( <i>continued</i> )
<b>defaultLocale</b>	This is the default locale the user is associated with. If the user does not have a default locale assigned, the default locale of the repository is assumed.
<b>Principal</b>	This parameter represents the internal search user that has been authenticated. This is a Java security principal class. Information Manager does not use it.
<b>Domain</b>	This is the authentication realm that is being used to handle the user authentication tasks. This is defined in System Manager currently.
<b>Email</b>	This is the user's email address as provided by the user information data store. In the case of an IM managed user this information is stored in the USERINFORMATION table of the IM schema. For remote authentication schemes this information must be provided by the remote data store.
<b>attributeMap</b>	This map contains a listing of additional properties defined for a user. These can be user preference variables or any other name value pairs.
<b>SSOFieldMap</b>	This map contains a listing of the single sign on parameters returned from the SSO authenticated page.

## IQSearchRequest

The IQSearchRequest object is obtained from the IQServiceClient by executing the following code to obtain a reference to the IQServiceClient from the IQServiceClientManager. The IQServiceClientManager is a static class that can be accessed with the connect() method. The result of a successful connection is a valid IQServiceClient object that can be used to retrieve an IQSearchRequest object. This same pattern is used to obtain an IQxxxRequest in general for all requests.

The searchURL parameter is the location of where a Search runtime client SOAP gateway is deployed. This is typically in the format:

```
http://<host>:8223/inquiragw/services/RequestProcessor
...
IQServiceClient client = IQServiceClientManager.connect(user, passwd, domain, repos,
imUrl, searchUrl, throwExceptionOnError);
IQSearchRequest request = client.getSearchRequest();
...
```

The IQSearchRequest is the primary entry point for accessing the search client Library methods. The IQSearchRequest provides a thin wrapper over the existing Search SOAP gateway methods. Currently, not all of the methods available in the SOAP gateway are available through the Search client Library.

## GIML

The GIML object that is returned from the Search client Library calls contains the same information as the SOAP response from a Search query using tools like the IM JSP tag library or the default ui.jsp page supplied with the Search runtime.

The GIML is an XML formatted string that contains nodes that represent the various portions of the result set broken into the Search components defined in the Dictionary. Appendix 10 contains the XML schema used by Information Manager to parse the GIML returned by the various search requests. The search results returned from a search request are broken into sections called <PURPOSE> that indicate where the answer should be displayed in the UI. Answer purposes are categories to which you assign answer actions within Dictionary rules. Answer purposes correspond to display characteristics defined in the user interface, enabling you to establish consistent, focused, and targeted presentation for various types of application content, such as



general site information, online glossaries, promotional material, and site features, such as calculators and other tools. The list of available PURPOSE types is configured in the #.xml file. Additional PURPOSE types can be added if required.

### Default Answer Purposes

Purpose	Description	Default Response Template	Default Presentation
Answer	Displays responses that directly address the user's question.	Answer template	In the Answer area of the response page
Act	Displays links that provide actions that the user can take on the web site	Act template	In the Act Now portlet
Promote	Displays cross-sell or up-sell advertisements for products related to the intent of the question	Promote template	In the Promotion template
Related Topic	Displays links to major topic categories defined for the web site	Link To Category template	In the Related Topics portlet
Define	Displays links to terms used in the question as well as similar content	n/a	
Jump to Page	Displays content configured in the Dictionary for use with the direct page display feature	n/a	
Converse	Displays conversational response intended for use with a virtual representative on the response page	Converse template	
Feature Content	Displays specific featured content from the web site that supplements the answers	Feature Content template	In the Featured Content area of the page
Contact	For use with the Contact Deflection feature	n/a	

The Oracle Knowledge UI documentation refers to these regions on the page as portlets (not to be confused with JSR-168 portlets). The following table describes the default answer portlets/templates provided with the initial installation of Oracle Knowledge Search.

Portlets	Description
<b>Promotion</b>	Use this portlet to display promotional information, such as cross-sell or up-sell advertisements for products related to the intent of the question. You can configure responses to include graphics as links to pages that contain more detailed information.
<b>Act Now</b>	Use this portlet to display information about relevant activities that users can perform immediately on the site. This portlet favors concise, imperative messages that compel users to access beneficial features.
<b>Learn More</b>	Use this portlet to display brief summaries of content areas that are relevant to the user's question, such as tools and calculators.
<b>Definition</b>	Use this portlet to display definitions of terms related to the user's question. This portal is ideal for displaying existing glossary information adapted from various formats.
<b>Feature Content</b>	Use this portlet to display more detailed information about relevant content areas and site features, such as tools and calculators. The Feature Content portlet displays responses in the lower portion of the answer area and not in a segregated box, which provides space for more detailed information, such as graphical tools.

The general structure of the GIML response includes the list of request parameters listed at the top in the <params> node. The <params> node is followed by a <responses> node which contains a <purpose> node

for each returned section of the search result page. The <responses> node is followed by a <facets> node that contains the list of each facet identified for the search results.

There is a <purpose> node for each configured portlet that will be displayed in the UI. Each <purpose> has one or more <answer> nodes that contain each specific answer returned along with some meta data about the answer (how to display it, highlighting info, URLs, etc).

Appendix C shows a sample GIML response to a search request. The GIML response will vary based on the type of search request being made, the parameters passed in, and the portlets configured in the search configuration.

## Question Answering Methods

This family of methods is used to answer questions and to process subsequent actions within the search session. Each session MUST start with `initialScreen()` to initialize the Search session.

Methods that require a `priorTransactionID` field should obtain the transaction from the `IQServiceClient` after each call so it can be passed along to Analytics. This is important to be able to track the order of things that each user does in a single search session.

- `public boolean login(Map<String, String> loginFieldMap)`  
This request is typically only used by the Search admin tools. It should not be used by the non-Oracle Knowledge tools or programs. Use the `IQServiceClient.connect()` to authenticate.
- `public GIML initialScreen()`  
This method should be called before any other search method to initialize the Search session.
- `public GIML askQuestion(String question, boolean startOver)`  
This method is used to ask a question to the Search runtime. If the `startOver` parameter is set to `TRUE` then the previous Search results are ignored and not used as a filter within the scope of the subsequent question results.
- `public GIML askWithinDocument(int priorTransactionID, String question, int SearchWithinDocid, String searchWithinDocEncoding, String searchWithinDocUrl)`  
This method is used to search for a question within a specific document. The `INT` value representing the `searchWithinDocid` and `priorTransactionID` are obtained from the previous GIML response. The `searchWithinDocEncoding` is used to limit the request set to a specified language. The `searchWithinDocUrl` string should use the URL where the document is located from the previous GIML.
- `public void clickThru(int priorTransactionID, int answered, String trackedURL, int searchWithinDocid, boolean isUnstructured)`  
This method is used to record a `clickThru` event if a user selects a link in the search results to view more information on. The `clickThru()` action is recorded and becomes an important component of the user activity analytics reports.
- `public GIML echo()`  
This method is used to perform a PING type of test against the Search runtime. It does not return any search results.
- `public String getSessionState(int priorTransactionId)`  
This method is used to return a string containing the session ID and other session information recorded for the current session.
- `public String highlightAnswer(int priorTransactionID, int answerID, String trackedURL, int searchWithinDocID, boolean isPDF, boolean trackClickthru, String highlightInfo)`

This method is used to highlight the snippet of an answer that is most relevant to the question being asked. This method should be called after the `askQuestion()`. The `answerID` is the available from the response GIML. This method can also be used to highlight answers in a PDF document.

- `public GIML navigate(int priorTransactionID, String facet, boolean facetShowAll)`  
This method is used when navigating the facets displayed in a UI. The default UI provided by Oracle Knowledge contains the facet list on the left hand side. The returned GIML should reflect the filtered result set of answers.
- `public GIML pageResults(int priorTransactionID, String purposeName, PageDirectionEnum pagedirection, int newPageSize, int pageNumber)`  
This method is used when the initial number of search results for a particular <PURPOSE> exceed the space available to display the answers. The returned GIML object contains the answers that should be displayed passed on the specified page number.
- `public void recordFeedback(int priorTransactionID, String userFeedback, int userrating)`  
This method is used to rate a page of search results. This is usually displayed in the Ratings portlet. This method is different that the `rateContent()` provided by the Information Manager `IQRatingRequest`. This feedback is rates the overall quality of the search results - not the specific document. This information is used to help tune the Search results.
- `public GIML showSimilarAnswers(int priorTransactionID, int answerID, String trackedURL, string relatedIDs)`  
This method is used to show results that are similar to a result in the current result set. The `answerID` parameter should be retrieved from the current GIML response (same as the `priorTransactionID`).

## Call Center Advisor Methods

The family of Call Center Advisor methods is used to support contact deflection and to provide the ability to link/unlink CRM system cases with a search result.

- `public GIML addCCASolutions(int priorTransactionID)`  
This method is used to add a link from a CRM system service request to a document listed in the Oracle Knowledge search results. This method would effectively add all of the solutions found in the current GIML as a solution to the CRM case. This action is primarily used for analytic tracking and is not recorded in Information Manager.
- `public GIML removeCCASolutions(int priorTransactionID)`  
This method is used to remove a previously linked CRM solution from a search result. This action is primarily used for analytic tracking and is not recorded in Information Manager.
- `public GIML refreshCCA(int priorTransactionID)`  
This method is used to refresh the CRM data prior to either linking or unlinking a request.
- `public void contactDeflectionResponse(int priorTransactionID, boolean deflected)`  
This method should be called if a contact has been successfully deflected by an action in the UI. This information is primarily logged for analytics purposes.

## Process Wizard Methods

This family of methods is used to interact with the Search Process Wizard functionality.

**Note:** The Process Wizard feature is deprecated and replaced by the AnswerFlow component. See the AnswerFlow Implementation and User's Guide, located in the Oracle Knowledge Intelligent Search User Documentation Library, for more information.

A Process Wizard is basically a conditional branching algorithm that is defined in the Dictionary and can be assigned to an intent response. Depending on the option the user selects the Process Wizard will route the user down a list of choices.

- `public void startWizard( int priorTransactionID, int answerID, String trackedURL, string wizardID)`  
This method should be called prior to starting the first step of a process wizard. The answerID represents the answer that is actually a process wizard reference in the search results screen.
- `public void updateWizard(int priorTransactionID, String wizardID, String stepUUID, String nextStepID, String nextStepUUID, Map<String, String> stepFieldValueMap, boolean back)`  
This method should be called after performing each step of the process wizard. This information is primarily used for analytics purposes.
- `public GIML askWizardQuestion(int priorTransactionID, String question, int wizardID, String stepID, String stepUUID, Map <String, String> stepFieldValueMap, Map<String, String> parameterValueMap)`  
This method is used to ask a question as part of a process wizard. This method needs to know where in the process wizard the question was asked. The stepFieldValueMap contains a list of key value pairs that represent the choices presented to the user for them to select an option from. The parameterValueMap is used to pass along any search parameters in addition to those in the ClientInfo and SearchInfo data structures.
- `public void finishWizard(int priorTransactionID, String wizardID, String stepID, String stepUUID, Map<String, String> fieldValueMap, boolean complete)`  
This method should be called when the final step of the process wizard is complete. This information is primarily used by analytics for tracking purposes.

# Error Code Constants

The Oracle Knowledge client Library API uses the error code constants to create localized error messages for each type of error that can be thrown. The constants are stored in the `com.inquiria.im.util.ErrorCodeConstants` class in the `imservices.jar` that is deployed with Information Manager. The string values for each constant below are the keys in the localized error message file that is used to return localized error messages. These property files are stored in the `imservices.jar` (`errormessages.properties`). Oracle Knowledge provides localized messages in the following languages: English (default), German, Spanish, French, Italian, Japanese, Korean, Dutch, Portuguese, Russian, Ukrainian, Slovak, and Chinese.

**Note:** In most cases, the Oracle Knowledge provided error messages should not be displayed to end users, a more specific, user-friendly error message should be provided instead.

```
public class ErrorCodeConstants {  
    /** Error Property for resource = Unable to find answer */  
    public final static String ANSWER_NOT_FOUND = "answer.not.found";  
  
    /** Error Property for resource = {0} is null or has 0 elements */  
    public final static String ARRAY_NOT_NULL = "array.not.null";  
  
    /** Error Property for resource = Authentication Failed */  
    public final static String AUTHENTICATION_FAILED = "authentication.failed";  
  
    /** Error Property for resource = Authentication Token has expired */  
    public final static String AUTHENTICATION_TOKEN_EXPIRED = "authentication.token.expired";  
  
    /** Error Property for resource = Authentication Token not found */  
    public final static String AUTHENTICATION_TOKEN_NOT_FOUND = "authentication.token.not.found";  
  
    /** Error Property for resource = Authentication Token is not valid */  
    public final static String AUTHENTICATION_TOKEN_NOT_VALID = "authentication.token.not.valid";  
  
    /** Error Property for resource = Could not authenticate user with the supplied information. Information supplied may be incorrect or user account may be locked or inactive */  
    public final static String AUTHENTICATION_USER_NOT_AUTHENTICATED = "authentication.user.not.authenticated";  
  
    /** Error Property for resource = Unable to find Case Link */  
    public final static String CASELINK_NOT_FOUND = "caselink.not.found";  
  
    /** Error Property for resource = ContentId not found */  
    public final static String CASELINK_NOT_FOUND_CONTENTID = "caselink.not.found.contentid";  
}
```

```
/** Error Property for resource = Error adding caselink data */
public final static String CASELINK_SERVICE_ADD_ERROR = "caselink.ser-
vice.add.error";

/** Error Property for resource = Error deleting caselink data */
public final static String CASELINK_SERVICE_DELETE_ERROR = "caselink.ser-
vice.delete.error";

/** Error Property for resource = Case Number not found */
public final static String CASELINK_SERVICE_INVALID_CASENUMBER = "caselink.ser-
vice.invalid.caseNumber";

/** Error Property for resource = Neither a Content Id or a Document Id has
been passed into service */
public final static String CASELINK_SERVICE_INVALID_ID = "caselink.service.in-
valid.id";

/** Error Property for resource = Incident Value is not valid */
public final static String CASELINK_SERVICE_INVALID_INCIDENT = "caselink.ser-
vice.invalid.incident";

/** Error Property for resource = Error updating user reputation points */
public final static String CASELINK_SERVICE_REPUTATION_ERROR = "caselink.ser-
vice.reputation.error";

/** Error Property for resource = Error constructing response */
public final static String CASELINK_SERVICE_RESPONSE_ERROR = "caselink.ser-
vice.response.error";

/** Error Property for resource = Unable to find Case Link Content */
public final static String CASELINKCONTENT_NOT_FOUND = "caselinkcon-
tent.not.found";

/** Error Property for resource = Input not valid. Make sure you are passing a
guid, objectid or reference key. */
public final static String CATEGORY_CATEGORY_IS_INVALID = "category.catego-
ry.is.invalid";

/** Error Property for resource = No content channel found for reference key =
{0} */
public final static String CATEGORY_CHANNEL_NOT_FOUND = "category.chan-
nel.not.found";

/** Error Property for resource = Duplicate Reference Key - Reference Keys must
be unique within the repository. Value = {0} */
public final static String CATEGORY_DUPLICATE_REFKEY = "category.duplicate.re-
fkey";

/** Error Property for resource = Locale not found for code = {0} */
public final static String CATEGORY_LOCALE_NOT_FOUND = "category.lo-
cale.not.found";

/** Error Property for resource = No category display name has been passed into
service */
public final static String CATEGORY_NO_DISPLAY_NAME = "category.no.dis-
play.name";
```

```
/** Error Property for resource = No category reference key has been passed
into service */
public final static String CATEGORY_NO_REFKEY_PASSED_IN = "category.no.refkey.passed.in";

/** Error Property for resource = Category not found */
public final static String CATEGORY_NOT_FOUND = "category.not.found";

/** Error Property for resource = Parent Category not found for parent reference
key = {0} */
public final static String CATEGORY_PARENT_REFERENCEKEY_NOT_FOUND = "category.parent.referencekey.not.found";

/** Error Property for resource = Invalid category reference key */
public final static String CATEGORY_REFERENCEKEY_NOT_FOUND = "category.referencekey.not.found";

/** Error Property for resource = Content Channel not found for reference key */
public final static String CHANNEL_NOT_FOUND_REFKEY = "channel.not.found.refkey";

/** Error Property for resource = Active locale is null */
public final static String CONTENT_SERVICE_ACTIVE_LOCALE_VALIDATION_EXCEPTION = "content.service.active.locale.validation.exception";

/** Error Property for resource = Active user is null */
public final static String CONTENT_SERVICE_ACTIVE_USER_VALIDATION_EXCEPTION = "content.service.active.user.validation.exception";

/** Error Property for resource = An error has occurred trying to change the
owner of a document. ID = '{0}' */
public final static String CONTENT_SERVICE_CHANGE_OWNER_ERROR = "content.service.change.owner.error";

/** Error Property for resource = Content Channel is null or is invalid */
public final static String CONTENT_SERVICE_CHANNEL_VALIDATION_EXCEPTION = "content.service.channel.validation.exception";

/** Error Property for resource = A class cast exception has occurred */
public final static String CONTENT_SERVICE_CLASSCAST_EXCEPTION = "content.service.classcast.exception";

/** Error Property for resource = Locale for content record is null */
public final static String CONTENT_SERVICE_CONTENT_LOCALE_VALIDATION_EXCEPTION = "content.service.content.locale.validation.exception";

/** Error Property for resource = Unable to find {0} content record for supplied
parameters */
public final static String CONTENT_SERVICE_CONTENT_NOT_FOUND = "content.service.content.not.found";

/** Error Property for resource = Attempt to edit a non master record. Non
master records can be edited using the translateContent service only. Content id =
'{0}' Master locale = '{1}' Intended locale = '{2}' */
public final static String CONTENT_SERVICE_EDIT_NON_MASTER_RECORD = "con-
```

```
tent.service.edit.non.master.record";

    /** Error Property for resource = A jdbc error has occurred */
    public final static String CONTENT_SERVICE_ERROR_CREATING_JDBC_CONNECTION =
"content.service.error.creating.jdbc.connection";

    /** Error Property for resource = A database exception has occurred */
    public final static String CONTENT_SERVICE_GENERAL_DATABASE_EXCEPTION = "con-
tent.service.general.database.exception";

    /** Error Property for resource = A general exception has occurred */
    public final static String CONTENT_SERVICE_GENERAL_EXCEPTION = "content.ser-
vice.general.exception";

    /** Error Property for resource = Case Link object not found */
    public final static String CONTENT_SERVICE_INVALID_CASEVALUE = "content.ser-
vice.invalid.casevalue";

    /** Error Property for resource = Category not found with reference key {0} */
    public final static String CONTENT_SERVICE_INVALID_CATEGORY_REF_KEY = "con-
tent.service.invalid.category_ref_key";

    /** Error Property for resource = Invalid content channel supplied '{0}' */
    public final static String CONTENT_SERVICE_INVALID_CHANNEL_EXT = "content.ser-
vice.invalid.channel_ext";

    /** Error Property for resource = Invalid content channel reference key supplied
'{0}' */
    public final static String CONTENT_SERVICE_INVALID_CHANNEL_REF_KEY = "con-
tent.service.invalid.channel_ref_key";

    /** Error Property for resource = Invalid contentID supplied '{0}' */
    public final static String CONTENT_SERVICE_INVALID_CONTENTID = "content.ser-
vice.invalid.contentid";

    /** Error Property for resource = No valid contentIDs supplied */
    public final static String CONTENT_SERVICE_INVALID_CONTENTIDS = "content.ser-
vice.invalid.contentids";

    /** Error Property for resource = Invalid documentID supplied '{0}' */
    public final static String CONTENT_SERVICE_INVALID_DOCUMENTID = "content.ser-
vice.invalid.documentid";

    /** Error Property for resource = Invalid documentID and version combination
supplied. DocumentID = '{0}' Version = '{1}.{2}' */
    public final static String CONTENT_SERVICE_INVALID_DOCUMENTID_AND_VERSION =
"content.service.invalid.documentid.and.version";

    /** Error Property for resource = No valid documentIDs supplied */
    public final static String CONTENT_SERVICE_INVALID_DOCUMENTIDS = "content.ser-
vice.invalid.documentids";

    /** Error Property for resource = Invalid locale supplied {0} */
    public final static String CONTENT_SERVICE_INVALID_LOCALE = "content.ser-
vice.invalid.locale";
```



```
/** Error Property for resource = Invalid locale code supplied '{0}' */
public final static String CONTENT_SERVICE_INVALID_LOCALE_CODE = "content.ser-
vice.invalid.locale_code";

/** Error Property for resource = Invalid locale supplied '{0}' */
public final static String CONTENT_SERVICE_INVALID_LOCALE_EXT = "content.ser-
vice.invalid.locale_ext";

/** Error Property for resource = Null ordering supplied */
public final static String CONTENT_SERVICE_INVALID_ORDERING = "content.ser-
vice.invalid.ordering";

/** Error Property for resource = No valid orderings supplied */
public final static String CONTENT_SERVICE_INVALID_ORDERINGS = "content.ser-
vice.invalid.orderings";

/** Error Property for resource = Invalid repository supplied */
public final static String CONTENT_SERVICE_INVALID_REPOSITORY = "content.ser-
vice.invalid.repository";

/** Error Property for resource = Invalid repository supplied '{0}' */
public final static String CONTENT_SERVICE_INVALID_REPOSITORY_EXT = "con-
tent.service.invalid.repository_ext";

/** Error Property for resource = Invalid repository reference key supplied
'{0}' */
public final static String CONTENT_SERVICE_INVALID_REPOSITORY_REF_KEY = "con-
tent.service.invalid.repository_ref_key";

/** Error Property for resource = Invalid User or Locale */
public final static String CONTENT_SERVICE_INVALID_USER_OR_LOCALE = "con-
tent.service.invalid.user.or.locale";

/** Error Property for resource = Invalid user group reference key supplied
'{0}' */
public final static String CONTENT_SERVICE_INVALID_USERGROUP_REF_KEY = "con-
tent.service.invalid.usergroup_ref_key";

/** Error Property for resource = Could not delete content resource folder */
public final static String CONTENT_SERVICE_IO_DELETE_EXCEPTION = "content.ser-
vice.io.delete.exception";

/** Error Property for resource = Unable to find latest content record for
supplied parameters */
public final static String CONTENT_SERVICE_LATEST_CONTENT_NOT_FOUND = "con-
tent.service.latest_content_not_found";

/** Error Property for resource = Unable to find any latest content records for
supplied parameters */
public final static String CONTENT_SERVICE_LATEST_CONTENTS_NOT_FOUND = "con-
tent.service.latest_contents_not_found";

/** Error Property for resource = Unable to find master content record for
supplied parameters */
public final static String CONTENT_SERVICE_MASTER_CONTENT_NOT_FOUND = "con-
tent.service.master_content_not_found";
```

```
/** Error Property for resource = At least one filter parameter must be set */
public final static String CONTENT_SERVICE_NO_FILTER_SUPPLIED = "content.ser-
vice.no_filter_supplied";

/** Error Property for resource = Unable to find published content record for
supplied parameters */
public final static String CONTENT_SERVICE_PUBLISHED_CONTENT_NOT_FOUND = "con-
tent.service.published_content_not_found";

/** Error Property for resource = Unable to find any published content records
for supplied parameters */
public final static String CONTENT_SERVICE_PUBLISHED_CONTENTS_NOT_FOUND = "con-
tent.service.published_contents_not_found";

/** Error Property for resource = Content Channel must belong to the passed in
repository */
public final static String
CONTENT_SERVICE_REPOSITORY_CHANNEL_VALIDATION_EXCEPTION = "content.service.repos-
itory.channel.validation.exception";

/** Error Property for resource = Repository is null */
public final static String CONTENT_SERVICE_REPOSITORY_VALIDATION_EXCEPTION =
"content.service.repository.validation.exception";

/** Error Property for resource = Attempt to translate a content record using
the record's master locale. Please provide a different locale. Master locale for
record = {0} */
public final static String CONTENT_SERVICE_SAME_LOCALE_TRANSLATION_ERROR =
"content.service.same.locale.translation.error";

/** Error Property for resource = Content Channel not found */
public final static String CONTENTCHANNEL_NOT_FOUND = "contentchan-
nel.not.found";

/** Error Property for resource = Content Channel Privileges not found */
public final static String CONTENTCHANNELPRIVILEGE_NOT_FOUND = "contentchan-
nelprivilege.not.found";

/** Error Property for resource = Content History Record Found */
public final static String CONTENTHISTORY_NOT_FOUND = "contenthisto-
ry.not.found";

/** Error Property for resource = Content Locale Request Not found */
public final static String CONTENTLOCALEREQUEST_NOT_FOUND = "contentlocalere-
quest.not.found";

/** Error Property for resource = Content Metrics not found */
public final static String CONTENTMETRICS_NOT_FOUND = "contentmet-
rics.not.found";

/** Error Property for resource = Content Preview Url not found */
public final static String CONTENTPREVIEWURL_NOT_FOUND = "contentpre-
viewurl.not.found";
```

```
/** Error Property for resource = Content Recommendation not Found */
public final static String CONTENTRECOMMENDATION_NOT_FOUND = "contentrecommen-
dation.not.found";

/** Error Property for resource = Can't find content record with the supplied
document id */
public final static String CONTENTRECORD_NOT_FOUND = "contentrecord.not.found";

/** Error Property for resource = Content User Visit not found */
public final static String CONTENTUSERVISIT_NOT_FOUND = "contentuservis-
it.not.found";

/** Error Property for resource = Locale is not allowed to be null in context */
public final static String CONTEXT_LOCALE_NULL = "context.locale.null";

/** Error Property for resource = Context is not allowed to be null */
public final static String CONTEXT_NULL = "context.null";

/** Error Property for resource = Repository is not allowed to be null in context
*/
public final static String CONTEXT_REPOSITORY_NULL = "context.repository.null";

/** Error Property for resource = User is not allowed to be null in context */
public final static String CONTEXT_USER_NULL = "context.user.null";

/** Error Property for resource = Data Form not found */
public final static String DATAFORM_NOT_FOUND = "dataform.not.found";

/** Error Property for resource = Data Form Privilege not found */
public final static String DATAFORMPRIVILEGE_NOT_FOUND = "dataformprivi-
lege.not.found";

/** Error Property for resource = Data List not found */
public final static String DATALIST_NOT_FOUND = "datalist.not.found";

/** Error Property for resource = Data List filter not found */
public final static String DATALISTFILTER_NOT_FOUND = "datalistfil-
ter.not.found";

/** Error Property for resource = Data List Item not Found */
public final static String DATALISTITEM_NOT_FOUND = "datalistitem.not.found";

/** Error Property for resource = Discussion Board Abuse Item not found */
public final static String DBABUSE_NOT_FOUND = "dbabuse.not.found";

/** Error Property for resource = Discussion Board Forum not found */
public final static String DBFORUM_NOT_FOUND = "dbforum.not.found";

/** Error Property for resource = Discussion Board Forum Metrics not found */
public final static String DBFORUMMETRICS_NOT_FOUND = "dbforummet-
rics.not.found";

/** Error Property for resource = Discussion Board Message not found */
public final static String DBMESSAGE_NOT_FOUND = "dbmessage.not.found";

/** Error Property for resource = Discussion Board Message Filter not found */
```

```
public final static String DBMESSAGEFILTER_NOT_FOUND = "dbmessagefilter.not.found";

/** Error Property for resource = Discussion Board Metrics not found */
public final static String DBMETRICS_NOT_FOUND = "dbmetrics.not.found";

/** Error Property for resource = Discussion Board Privilege not Found */
public final static String DBPRIVILEGE_NOT_FOUND = "dbprivilege.not.found";

/** Error Property for resource = Discussion Board Privilege Value not found */
public final static String DBPRIVILEGEVALUE_NOT_FOUND = "dbprivilegevalue.not.found";

/** Error Property for resource = Discussion Board Topic not found */
public final static String DBTOPIC_NOT_FOUND = "dbtopic.not.found";

/** Error Property for resource = Discussion Board Topic Log not found */
public final static String DBTOPICLOG_NOT_FOUND = "dbtopiclog.not.found";

/** Error Property for resource = Discussion Board Topic Metrics not found */
public final static String DBTOPICMETRICS_NOT_FOUND = "dbtopicmetrics.not.found";

/** Error Property for resource = Discussion Board User Visit not found */
public final static String DBUSERVISIT_NOT_FOUND = "dbuservisit.not.found";

/** Error Property for resource = Discussion Board not found */
public final static String DISCUSSIONBOARD_NOT_FOUND = "discussionboard.not.found";

/** Error Property for resource = Editor Group not Found */
public final static String EDITORGROUP_NOT_FOUND = "editorgroup.not.found";

/** Error Property for resource = Extended View Attributes not found */
public final static String EXTENDEDVIEWATTRIBUTES_NOT_FOUND = "extendedviewattributes.not.found";

/** Error Property for resource = {0} is not allowed to be empty */
public final static String FIELD_NOT_NULL = "field.not.null";

/** Error Property for resource = Data Form Result not found */
public final static String FORMRESULTS_NOT_FOUND = "formresults.not.found";

/** Error Property for resource = Illegal Argument Exception */
public final static String ILLEGAL_ARGUMENT_EXCEPTION = "illegal.argument.exception";

/** Error Property for resource = Internal service call response is invalid */
public final static String INTERNAL_CALL_RESPONSE_INVALID = "internal.call.response.invalid";

/** Error Property for resource = The List objects passed in are not of the same size */
public final static String LISTS_NOT_SAME_SIZE = "lists.not.same.size";
```

```
/** Error Property for resource = Could not find a matching locale */
public final static String LOCALE_NOT_FOUND = "locale.not.found";

/** Error Property for resource = Locale not found for code = {0} */
public final static String LOCALE_NOT_FOUND_WITH_CODE = "lo-
cale.not.found.with.code";

/** Error Property for resource = Localized Token not found */
public final static String LOCALIZEDTOKENS_NOT_FOUND = "localizedto-
kens.not.found";

/** Error Property for resource = There is no default locale set to the repos-
itory */
public final static String NO_DEFAULT_LOCALE_FOR_REPOSITORY = "no.default.lo-
cale.for.repository";

/** Error Property for resource = The Content Channel does not have a rating
assigned to it */
public final static String NO_RATING_FOR_CHANNEL = "no.rating.for.channel";

/** Error Property for resource = Authentication Token is not valid */
public final static String NO_XML_REQUEST_FOUND = "no.xml.request.found";

/** Error Property for resource = XML Request parameters are not valid */
public final static String NO_XML_REQUEST_PARAMETERS_NOT_VALID = "no.xml.re-
quest.parameters.not.valid";

/** Error Property for resource = The system could not find an object with
supplied information : {0} */
public final static String OBJECT_NOT_FOUND_ERROR = "object.not.found.error";

/** Error Property for resource = Privilege not found */
public final static String PRIVILEGE_NOT_FOUND = "privilege.not.found";

/** Error Property for resource = Question not Found */
public final static String QUESTION_NOT_FOUND = "question.not.found";

/** Error Property for resource = Rating not found */
public final static String RATING_NOT_FOUND = "rating.not.found";

/** Error Property for resource = The supplied rating is out of sync with its
rating definition */
public final static String RATING_OUT_OF_SYNC = "rating.out.of.sync";

/** Error Property for resource = Replacement Token not found */
public final static String REPLACEMENTTOKENS_NOT_FOUND = "replacementto-
kens.not.found";

/** Error Property for resource = Repository not found with the supplied in-
formation */
public final static String REPOSITORY_NOT_FOUND = "repository.not.found";

/** Error Property for resource = Repository not found with id = {0} */
public final static String REPOSITORY_NOT_FOUND_WITH_ID = "reposito-
ry.not.found.with.id";
```

```
/** Error Property for resource = Repository not found with reference key = {0}
 */
public final static String REPOSITORY_NOT_FOUND_WITH_REFKEY = "reposito-
ry.not.found.with.refkey";

/** Error Property for resource = Reputation Configuration not found */
public final static String REPUTATIONCONFIG_NOT_FOUND = "reputationcon-
fig.not.found";

/** Error Property for resource = Reputation Level not found */
public final static String REPUTATIONLEVEL_NOT_FOUND = "reputationlev-
el.not.found";

/** Error Property for resource = Reputation Reward not found */
public final static String REPUTATIONREWARD_NOT_FOUND = "reputationre-
ward.not.found";

/** Error Property for resource = Request XML is invalid; XML = {0} */
public final static String REQUEST_XML_ERROR = "request.xml.error";

/** Error Property for resource = Role not found with the supplied information */
public final static String ROLE_NOT_FOUND = "role.not.found";

/** Error Property for resource = Validation failed for saving object */
public final static String SAVE_VALIDATION_FAILED = "save.validation.failed";

/** Error Property for resource = Secured Activity not found */
public final static String SECUREDACTIVITY_NOT_FOUND = "securedactivi-
ty.not.found";

/** Error Property for resource = Secured Application Item not found */
public final static String SECUREDAPPITEM_NOT_FOUND = "securedap-
pitem.not.found";

/** Error Property for resource = Security Role not found */
public final static String SECURITYROLE_NOT_FOUND = "securityrole.not.found";

/** Error Property for resource = Skill not found with the supplied information
 */
public final static String SKILL_NOT_FOUND = "skill.not.found";

/** Error Property for resource = Stylesheet not found */
public final static String STYLESHEET_NOT_FOUND = "stylesheet.not.found";

/** Error Property for resource = Subscription not found */
public final static String SUBSCRIPTION_NOT_FOUND = "subscription.not.found";

/** Error Property for resource = Task Status not found */
public final static String TASKSTATUS_NOT_FOUND = "taskstatus.not.found";

/** Error Property for resource = An unexpected error has occurred that prevented
this operation from completing */
public final static String UNEXPECTED_ERROR = "unexpected.error";
/** Error Property for resource = An unknown error has occurred that prevented
this operation from completing */
public final static String UNKNOWN_ERROR = "unknown.error";
```

```
/** Error Property for resource = Email is already used by another user */
public final static String USER_EMAIL_DUPLICATE = "user.email.duplicate";

/** Error Property for resource = Email is invalid */
public final static String USER_INVALID_EMAIL = "user.invalid.email";

/** Error Property for resource = First Name is invalid */
public final static String USER_INVALID_FIRST_NAME = "user.invalid.first.name";

/** Error Property for resource = Last Name is invalid */
public final static String USER_INVALID_LAST_NAME = "user.invalid.last.name";

/** Error Property for resource = Locale is invalid */
public final static String USER_INVALID_LOCALE = "user.invalid.locale";

/** Error Property for resource = Login is invalid */
public final static String USER_INVALID_LOGIN = "user.invalid.login";

/** Error Property for resource = Password is invalid */
public final static String USER_INVALID_PASSWORD = "user.invalid.password";

/** Error Property for resource = User type is invalid (ADMIN/WEB) */
public final static String USER_INVALID_USERTYPE = "user.invalid.usertype";

/** Error Property for resource = User is not authorized to perform : {0} */
public final static String USER_NOT_AUTHORIZED = "user.not.authorized";

/** Error Property for resource = User not found with the supplied information */
public final static String USER_NOT_FOUND = "user.not.found";

/** Error Property for resource = User not found with id = {0} */
public final static String USER_NOT_FOUND_WITH_ID = "user.not.found.with.id";

/** Error Property for resource = Reputation Level not found for user */
public final static String USER_REPUTATION_NOT_FOUND = "user.reputa-
tion.not.found";

/** Error Property for resource = Invalid email format. */
public final static String USER_VALIDATION_INVALID_EMAIL_FORMAT = "user.vali-
dation.invalid.email.format";

/** Error Property for resource = User Login can't be the same as the password */
public final static String USER_VALIDATION_LOGIN_PASSWORD_SAME = "user.valida-
tion.login.password.same";

/** Error Property for resource = User Group not found */
public final static String USERGROUP_NOT_FOUND = "usergroup.not.found";

/** Error Property for resource = User login is already used by another user */
public final static String USERID_DUPLICATE = "userid.duplicate";

/** Error Property for resource = User Key Value not found */
public final static String USERKEYVALUES_NOT_FOUND = "userkeyvalues.not.found";

/** Error Property for resource = View not found with the supplied information */
public final static String VIEW_NOT_FOUND = "view.not.found";
```

```
/** Error Property for resource = Workflow not found */
public final static String WORKFLOW_NOT_FOUND = "workflow.not.found";

/** Error Property for resource = Workflow Condition not found */
public final static String WORKFLOWCONDITION_NOT_FOUND = "workflowcondi-
tion.not.found";

/** Error Property for resource = Workflow Step not found */
public final static String WORKFLOWSTEP_NOT_FOUND = "workflowstep.not.found";

/** Error Property for resource = Work Team not found */
public final static String WORKTEAM_NOT_FOUND = "workteam.not.found";

/** Error Property for resource = XML request type is not valid */
public final static String XML_REQUEST_TYPE_NOT_VALID = "xml.re-
quest.type.not.valid";

/** Error Property for resource = XML response type is not valid */
public final static String XML_RESPONSE_TYPE_NOT_VALID = "xml.re-
sponse.type.not.valid";

/** Error Property for resource = XML Attribute User Group not found */
public final static String XMLATTRIBUTEUSERGROUP_NOT_FOUND = "xmlattributeuser-
group.not.found";

/** Error Property for resource = XML Schema not found */
public final static String XMLSCHEMA_NOT_FOUND = "xmlschema.not.found";

/** Error Property for resource = XML Schema Attribute not found */
public final static String XMLSCHEMAATTRIBUTE_NOT_FOUND = "xmlschemaattrib-
ute.not.found";
}
```



# GIML XSD

The GIML that can be returned from a Search request will vary based on the type of request being made and the configuration of the rules and Dictionary configured for the specific customer installation. The GIML.xsd below is used by Information Manager to parse and interact with GIML returned and processed by the IM JSP tag library and IM management console.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="message">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="params" minOccurs="0"/>
        <xsd:element ref="responses" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="facets" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="query" minOccurs="0"/>
        <xsd:element ref="config" minOccurs="0"/>
        <xsd:element ref="constraint" minOccurs="0"/>
        <xsd:element ref="find" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="session" minOccurs="0"/>
        <xsd:element ref="Message" minOccurs="0"/>
        <xsd:element ref="MessageCode" minOccurs="0"/>
        <xsd:element ref="StackTrace" minOccurs="0"/>
        <xsd:element ref="satisfied" minOccurs="0"/>
        <xsd:element ref="Redirect" minOccurs="0"/>
        <xsd:element ref="ListOfInquieraSrLinkedAnswersIo" min-
Occurs="0"/>
      </xsd:all>
      <xsd:attribute name="type" use="required" type="xsd:string"/>
      <xsd:attribute name="XSL_MODE" use="optional" type="xsd:string"/>
      <xsd:attribute name="language" use="optional" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Message" type="xsd:string"/>
  <xsd:element name="MessageCode" type="xsd:string"/>
  <xsd:element name="StackTrace" type="xsd:string"/>
  <xsd:element name="satisfied" type="xsd:string"/>
  <!--responses-->
  <xsd:element name="Redirect">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:attribute name="type" use="optional" type="xsd:string"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element> <!--responses-->
  <xsd:element name="responses">
    <xsd:complexType>
      <xsd:choice maxOccurs="unbounded">
        <xsd:element ref="primaryAnswer" maxOccurs="unbounded" minOccurs=
"0"/>
        <xsd:element ref="purpose" maxOccurs="unbounded" minOccurs="0"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

        </xsd:choice>
        <xsd:attribute name="type" use="required" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<!--params-->
<xsd:element name="params">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="param" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="param">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="name" use="required" type="xsd:string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<!--session-->
<xsd:element name="session">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="binary" use="required" type="xsd:string"/>
                <xsd:attribute name="id" use="optional" type="xsd:string"/>
                <xsd:attribute name="extId" use="optional" type="xsd:string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<!--constaint-->
<xsd:element name="constraint">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="user"/>
            <xsd:element ref="host"/>
            <xsd:element ref="address"/>
            <xsd:element ref="application"/>
            <xsd:element ref="version"/>
            <xsd:element ref="language"/>
            <xsd:element ref="domainGroup"/>
            <xsd:element ref="QuestionType"/>
            <xsd:element ref="result_language"/>
            <xsd:element name="segment" maxOccurs="unbounded" minOccurs="0"
type="xsd:string"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<xsd:element name="user">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element ref="principal"/>
        </xsd:choice>

```

```

    </xsd:complexType>
</xsd:element>
<xsd:element name="principal">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="binary" use="required" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="host" type="xsd:string"/>
<xsd:element name="address" type="xsd:string"/>
<xsd:element name="application" type="xsd:string"/>
<xsd:element name="version" type="xsd:string"/>
<xsd:element name="language" type="xsd:string"/>
<xsd:element name="domainGroup" type="xsd:string"/>
<xsd:element name="QuestionType" type="xsd:string"/>
<xsd:element name="result_language" type="xsd:string"/>
<!--query-->
<xsd:element name="query">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="question" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="question">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="3">
      <xsd:element ref="original"/>
      <xsd:element ref="paraphrase"/>
      <xsd:element ref="spellchecked"/>
    </xsd:choice>
    <xsd:attribute name="transactionId" use="required" type="xsd:string"/>
    <xsd:attribute name="language" use="optional" type="xsd:string"/>
    <xsd:attribute name="type" use="optional" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="original" type="xsd:string"/>
<xsd:element name="paraphrase" type="xsd:string"/>
<xsd:element name="spellchecked">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="correction"/>
      <xsd:element ref="original"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<xsd:element name="correction">
  <xsd:complexType mixed="true">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="suggestion"/>
    </xsd:choice>
    <xsd:attribute name="word" use="required" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>

```

```

<xsd:element name="suggestion">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="confidence" use="required" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<!--config-->
<xsd:element name="config">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="param" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!--facets-->
<xsd:element name="facets">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="result-facet" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="facet" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<xsd:element name="result-facet">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="id"/>
      <xsd:element ref="description"/>
      <xsd:element ref="data" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="count" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="result-facet" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:all>
    <xsd:attribute name="inEffect" use="optional" type="xsd:string"/>
    <xsd:attribute name="incomplete" use="optional" type="xsd:string"/>
    <xsd:attribute name="showlink" use="optional" type="xsd:string"/>
    <xsd:attribute name="tempSelect" use="optional" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="id" type="xsd:string"/>
<xsd:element name="description" type="xsd:string"/>
<xsd:element name="data" type="xsd:string"/>
<xsd:element name="facet">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="item" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="hidden" use="optional" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="count">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">

```

```

        <xsd:attribute name="atLeast" use="optional" type="xsd:string"/>
    </xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
<xsd:element name="primaryAnswer">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="answer" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="purpose">
    <xsd:complexType>
        <xsd:all minOccurs="0">
            <xsd:element ref="wizard" maxOccurs="unbounded" minOccurs="0"/>
            <xsd:element ref="answer" maxOccurs="unbounded" minOccurs="0"/>
        </xsd:all>
        <xsd:attribute name="page_more" use="optional" type="xsd:string"/>
        <xsd:attribute name="page_number" use="optional" type="xsd:string"/>
        <xsd:attribute name="page_start" use="optional" type="xsd:string"/>
        <xsd:attribute name="page_size" use="optional" type="xsd:string"/>
        <xsd:attribute name="total_results" use="optional" type="xsd:string"/>
        <xsd:attribute name="unshown_results" use="optional" type="xsd:string"/>
        <xsd:attribute name="score" use="required" type="xsd:string"/>
        <xsd:attribute name="name" use="required" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="answer">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="link" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="sentence" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="title" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="section" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="highlighted_link" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="click_through_link" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="similar_response_link" minOccurs="0" maxOccurs="
"1"/>
            <xsd:element ref="summary" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="chart" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="chartTypes" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="unavailable_fields" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="changed_fields" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="sortCol" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="table" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="timestamp" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="facets" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="cca" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="name" minOccurs="0" maxOccurs="1" />
            <xsd:element ref="element" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="text" minOccurs="0" maxOccurs="1"/>
        </xsd:choice>
        <xsd:attribute name="type" use="required" type="xsd:string" />
        <xsd:attribute name="answer_id" use="optional" type="xsd:string"/>
        <xsd:attribute name="score" use="optional" type="xsd:string"/>
        <xsd:attribute name="docType" use="optional" type="xsd:string"/>

```

```

        <xsd:attribute name="collectionId" use="optional" type="xsd:string"/>
        <xsd:attribute name="docId" use="optional" type="xsd:string"/>
    <xsd:attribute name="collectionName" use="optional" type="xsd:string"/>
    <xsd:attribute name="charset" use="optional" type="xsd:string"/>
    <xsd:attribute name="highlight_version" use="optional" type=
"xsd:string"/>
    <xsd:attribute name="language" use="optional" type="xsd:string"/>
    <xsd:attribute name="table_summary_rows" use="optional" type=
"xsd:string"/>
    <xsd:attribute name="similar_count" use="optional" type="xsd:string"/>
    <xsd:attribute name="rule" use="optional" type="xsd:string"/>
    <xsd:attribute name="imDocId" use="optional" type=
"xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="sortCol" type="xsd:string"/>

<xsd:element name="element">
    <xsd:complexType >
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="type" use="optional" type="xsd:string" />
                <xsd:attribute name="id" use="optional" type="xsd:string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="sentence">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="type" use="required" type="xsd:string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<xsd:element name="title">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="snippet" maxOccurs="unbounded"/>
        </xsd:choice>
        <xsd:attribute name="idx" use="optional" type="xsd:string"/>
        <xsd:attribute name="url" use="optional" type="xsd:string"/>
        <xsd:attribute name="type" use="optional" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="link">
    <xsd:complexType mixed="true">
        <xsd:all minOccurs="0" maxOccurs="1">
            <xsd:element ref="protocol"/>
            <xsd:element ref="host"/>
            <xsd:element ref="port"/>
            <xsd:element ref="path"/>
            <xsd:element ref="file"/>
            <xsd:element ref="params"/>
            <xsd:element ref="anchor" minOccurs="0"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>

```

```

        </xsd:all>
        <xsd:attribute name="type" use="required" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="protocol" type="xsd:string"/>
<xsd:element name="port" type="xsd:string"/>
<xsd:element name="path" type="xsd:string"/>
<xsd:element name="file" type="xsd:string"/>
<xsd:element name="anchor" type="xsd:string"/>
<xsd:element name="summary">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="title"/>
            <xsd:element ref="link"/>
            <xsd:element ref="highlighted_link"/>
            <xsd:element ref="click_through_link"/>
            <xsd:element ref="timestamp"/>
            <xsd:element ref="similar_response_link"/>
            <xsd:element ref="excerpt"/>
            <xsd:element ref="description"/>
            <xsd:element ref="paraphrase"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<xsd:element name="excerpt">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element ref="snippet" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="type" use="required" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="snippet">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="name" use="optional" type="xsd:string"/>
                <xsd:attribute name="score" use="optional" type="xsd:string"/>
                <xsd:attribute name="lvl" use="optional" type="xsd:string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<xsd:element name="chart">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="headers"/>
            <xsd:element ref="item"/>
        </xsd:choice>
        <xsd:attribute name="description" use="required" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="headers">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="header" type="xsd:string"/>
        </xsd:choice>
    </xsd:complexType>

```

```

    </xsd:complexType>
</xsd:element>
<xsd:element name="item">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="field"/>
      <xsd:element ref="description"/>
      <xsd:element ref="data"/>
    </xsd:choice>
    <xsd:attribute name="shape" use="optional" type="xsd:string"/>
    <xsd:attribute name="coords" use="optional" type="xsd:string"/>
    <xsd:attribute name="id" use="optional" type="xsd:string"/>
    <xsd:attribute name="selected" use="optional" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="field">
  <xsd:complexType mixed="true">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="option"/>
      <xsd:element ref="map"/>
    </xsd:choice>
    <xsd:attribute name="id" use="optional" type="xsd:string"/>
    <xsd:attribute name="type" use="optional" type="xsd:string"/>
    <xsd:attribute name="isSharedType" use="optional" type="xsd:string"/>
    <xsd:attribute name="description" use="optional" type="xsd:string"/>
    <xsd:attribute name="orig" use="optional" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="option">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="value" use="optional" type="xsd:string"/>
        <xsd:attribute name="id" use="optional" type="xsd:string"/>
        <xsd:attribute name="selected" use="optional" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="map">
  <xsd:complexType>
    <xsd:attribute name="field" use="optional" type="xsd:string"/>
    <xsd:attribute name="variable" use="optional" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="transition">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="condition"/>
    </xsd:choice>
    <xsd:attribute name="step" use="optional" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="condition">
  <xsd:complexType>
    <xsd:attribute name="field" use="optional" type="xsd:string"/>
    <xsd:attribute name="op" use="optional" type="xsd:string"/>
  </xsd:complexType>

```



```

        <xsd:attribute name="value" use="optional" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="chartTypes">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="chart"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<xsd:element name="table">
    <xsd:complexType>
        <xsd:all>
            <xsd:element ref="header"/>
            <xsd:element ref="row" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:all>
        <xsd:attribute name="total_possible_results" use="required" type=
"xsd:string"/>
        <xsd:attribute name="total_results" use="required" type="xsd:string"/>
        <xsd:attribute name="complete" use="required" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="header">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="field" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="row">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="field"/>
            <xsd:element ref="facets"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<xsd:element name="unavailable_fields">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="field" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="changed_fields">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="old_field"/>
            <xsd:element ref="new_field"/>
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<xsd:element name="old_field" type="xsd:string"/>
<xsd:element name="new_field" type="xsd:string"/>
<xsd:element name="highlighted_link">
    <xsd:complexType>
        <xsd:simpleContent>

```

```

        <xsd:extension base="xsd:string">
            <xsd:attribute name="type" use="optional" type="xsd:string"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
</xsd:element>
<xsd:element name="click_through_link">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="type" use="optional" type="xsd:string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<xsd:element name="similar_response_link">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="type" use="optional" type="xsd:string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<xsd:element name="timestamp">
    <xsd:complexType>
        <xsd:all>
            <xsd:element ref="date"/>
            <xsd:element ref="month"/>
            <xsd:element ref="year"/>
            <xsd:element ref="hour"/>
            <xsd:element ref="minute"/>
            <xsd:element ref="second"/>
            <xsd:element ref="millisecond"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>
<xsd:element name="year" type="xsd:string"/>
<xsd:element name="date" type="xsd:string"/>
<xsd:element name="month" type="xsd:string"/>
<xsd:element name="second" type="xsd:string"/>
<xsd:element name="minute" type="xsd:string"/>
<xsd:element name="hour" type="xsd:string"/>
<xsd:element name="millisecond" type="xsd:string"/>
<xsd:element name="section">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="section"/>
            <xsd:element ref="title"/>
            <xsd:element ref="text"/>
            <xsd:element ref="field"/>
            <xsd:element ref="transition"/>
        </xsd:choice>
        <xsd:attribute name="type" use="optional" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="text">

```

```

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="snippet" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="idx" use="optional" type="xsd:string"/>
      <xsd:attribute name="url" use="optional" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="wizard">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="step" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" use="optional" type="xsd:string"/>
      <xsd:attribute name="version" use="optional" type="xsd:string"/>
      <xsd:attribute name="label" use="optional" type="xsd:string"/>
      <xsd:attribute name="description" use="optional" type="xsd:string"/>
      <xsd:attribute name="first_step" use="optional" type="xsd:string"/>
      <xsd:attribute name="default_step" use="optional" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="step">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="section" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" use="optional" type="xsd:string"/>
      <xsd:attribute name="label" use="optional" type="xsd:string"/>
      <xsd:attribute name="uuid" use="optional" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="cca">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="solution"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="solution" type="xsd:string"/>
  <xsd:element name="find">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" ref="docu-
mentDetail" />
      </xsd:sequence>
      <xsd:attribute name="total" type="xsd:int" default=
"0"/>
      <xsd:attribute name="size" use="optional" type=
"xsd:int" default="0" />
      <xsd:attribute name="startIndex" use="optional"
type="xsd:int" default="0"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="documentDetail">
    <xsd:complexType mixed="true">
      <xsd:all maxOccurs="1">

```

```

        <xsd:element name="title" minOccurs="0"
            type="xsd:string" />
        <xsd:element name="facet" minOccurs="0"
            type="xsd:string" />
        <xsd:element name="excerpt" minOccurs="0"
            type="xsd:string" />
        <xsd:element name="url" minOccurs="0" type=
"xsd:string" />
        <xsd:element name="fmdt" minOccurs="0"
            type="xsd:string" />
    </xsd:all>
    <xsd:attribute name="colId" use="optional"
        type="xsd:string" />
    <xsd:attribute name="colName" use="optional"
        type="xsd:NCName" />
    <xsd:attribute name="docId" use="optional"
        type="xsd:string" />
    <xsd:attribute name="enc" use="optional" type=
"xsd:NCName" />
    <xsd:attribute name="extId" type="xsd:NCName" />
    <xsd:attribute name="guid" type="xsd:NMTOKEN" />
    <xsd:attribute name="lang" use="optional" type=
"xsd:NCName" />
    <xsd:attribute name="mdt" use="optional" type=
"xsd:long" default="0"/>
    <xsd:attribute name="status" type="xsd:NCName" />
    <xsd:attribute name="uniqueId" use="optional"
        type="xsd:string" />
    </xsd:complexType>
</xsd:element>

<!-- ccaLinkedAnswers -->
<xsd:element name="ListOfInquirasrLinkedAnswersIo">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="ServiceRequest" minOccurs="0"
maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

    <xsd:element name="ServiceRequest">
    <xsd:complexType>
        <xsd:all>
            <xsd:element ref="SRNumber" minOccurs="0" maxOc-
curs="1"/>
            <xsd:element ref="ListOfInquirasAnswersEai" minOccurs="0" maxOccurs=
"1"/>
        </xsd:all>
    </xsd:complexType>
</xsd:element>

    <xsd:element name="SRNumber" type="xsd:string"/>

    <xsd:element name="ListOfInquirasAnswersEai">
    <xsd:complexType>
        <xsd:choice>

```

```

        <xsd:element ref="InquirasAnswersEai" minOccurs="0" maxOccurs=
"unbounded"/>
    </xsd:choice>
</xsd:complexType>
</xsd:element>

        <xsd:element name="InquirasAnswersEai">
<xsd:complexType mixed="true">
    <xsd:all minOccurs="0" maxOccurs="1">
        <xsd:element ref="Key"/>
            <xsd:element ref="URL"/>
            <xsd:element ref="CaseLinkKey"/>
            <xsd:element ref="Excerpt"/>
            <xsd:element ref="DocumentType"/>
        <xsd:element ref="Title"/>
            <xsd:element ref="UserName"/>
            <xsd:element ref="VersionNumber"/>
            <xsd:element ref="DocumentId"/>
            <xsd:element ref="EntityType"/>
            <xsd:element ref="Comments"/>
            <xsd:element ref="LinkedDate"/>
            <xsd:element ref="Status"/>
            <xsd:element ref="LinkStatus"/>
    </xsd:all>
</xsd:complexType>
</xsd:element>

        <xsd:element name="Key" type="xsd:string"/>
        <xsd:element name="URL" type="xsd:string"/>
        <xsd:element name="CaseLinkKey" type="xsd:string"/>
        <xsd:element name="Excerpt" type="xsd:string"/>
        <xsd:element name="DocumentType" type="xsd:string"/>
        <xsd:element name="Title" type="xsd:string"/>
        <xsd:element name="UserName" type="xsd:string"/>
        <xsd:element name="VersionNumber" type="xsd:string"/>
        <xsd:element name="DocumentId" type="xsd:string"/>
        <xsd:element name="EntityType" type="xsd:string"/>
        <xsd:element name="Comments" type="xsd:string"/>
        <xsd:element name="LinkedDate" type="xsd:string"/>
        <xsd:element name="Status" type="xsd:string"/>
        <xsd:element name="LinkStatus" type="xsd:string"/>
</xsd:schema>

```

# GIML Response

The following GIML.xml shows a sample GIML response to a search request. The GIML response varies based on the type of search request being made, the parameters passed in, and the portlets configured in the search configuration.

```
- <message type="response">
  - <params>
    <param name="type">Navigate</param>
    <param name="charset">UTF-8</param>
    <param name="Question" />
    <param name="structured_chart" />
    <param name="TransactionId">583954092</param>
    <param name="ui_mode">navigate</param>
    <param name="Facet">CMS-CATEGORY.CAT1</param>
    <param name="SearchWithin" />
    <param name="FacetShowAll" />
    <param name="PriorTransactionId">583954091</param>
    <param name="FacetPriorTransactionId">583954091</param>
    <param name="user-agent.parameters.debug">>true</param>
    <param name="user-agent.parameters.facet">CMS%2dCATEGORY%2eCAT1</param>
    <param name="user-agent.parameters.charset">UTF-8</param>
    <param name="user-agent.parameters.facetCollectionID" />
    <param name="user-agent.parameters.language">en-US</param>
    <param name="user-agent.parameters.rated">>false</param>
    <param name="user-agent.parameters.prior_transaction_id">583954091</param>
    <param name="user-agent.parameters.result_language">en-US</param>
    <param name="user-agent.parameters.question_box" />
    <param name="user-agent.parameters.ui_mode">navigate</param>
    <param name="user-agent.parameters.structured_chart" />
    <param name="user-agent.headers.host">whoiam:8223</param>
    <param name="user-agent.headers.user-agent">Mozilla/5.0 (Windows; U; Windows NT
    5.1; en-US; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7</param>
    <param name="user-agent.headers.accept">text/xml,application/
    xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5</param>
    <param name="user-agent.headers.accept-language">en,it;q=0.5</param>
    <param name="user-agent.headers.accept-encoding">gzip,deflate</param>
    <param name="user-agent.headers.accept-charset">ISO-8859-1,utf-8;q=0.7,*;q=
    0.7</
    param>
    <param name="user-agent.headers.keep-alive">300</param>
    <param name="user-agent.headers.connection">keep-alive</param>
    <param name="user-agent.headers.referer">http://whoiam:8223/inquiragw/ui.jsp</
    param>
    <param name="user-agent.headers.cookie">JSESSIONID=
    81C6E5DAB05224BC2CB1371FB805D0FA</param>
    <param name="agentAddress">http://whoiam:8223/inquiragw/ui.jsp</param>
    <param name="processorAddress">transport: local</param>
    <param name="baseURL">http://whoiam:8223/inquiragw/ui.jsp</param>
    <param name="ExternalSessionId">81C6E5DAB05224BC2CB1371FB805D0FA</param>
    <param name="processorVersion">8.2.2 (6) </param>
  </params>
  - <responses type="response">
    - <purpose page_number="0" page_more="0" page_start="0" score=
      "0.9778846056057693" page_size="2" unshown_results="0" total_results="2"
```

```

name="ANSWER">
- <answer type="unstructured" score="0.9778846056057693" docType="CMS-XML"
  language="en-US" charset="UTF-8" collectionId="2" collectionName=
  "IM_GLEN_GLEN" answer_id="16777216" docId="4194305" imDocId="GL1"
  highlight_version="true">
- <section>
  - <title url="http://whoiam:8226/{instanceContext}/index?page=
    content&id=GL1&actp=search&viewlocale=en_US">
    <snippet lvl="0">this document is in category 1 test</snippet>
    </title>
  - <text url="http://whoiam:8226/{instanceContext}/index?page=
    content&id=GL1&actp=search&viewlocale=en_US">
    <snippet lvl="1">this document is in category 1</snippet>
    <snippet lvl="3">test</snippet>
    </text>
</section>
<highlighted_link type="text">http://whoiam:8223/inquiragw/
  ui.jsp?ui_mode=answer&prior_transaction_id=
  583954092&iq_action=5&answer_id=16777216&highlight_info=
  4194305,7,13&turl=
  http%3A%2F%2Fwhoiam%3A8226%2F%7BinstanceContext%7D%2Findex%3Fpage%3
  Dcontent%26id%3DGL1%26actp%3Dsearch%26viewlocale%3Den_US#__highligh
  t</
  highlighted_link>
- <link type="highlight">
  <protocol>http</protocol>
  <host>whoiam</host>
  <port>8223</port>
  <path>/inquiragw/</path>
  <file>ui.jsp</file>
- <params>
  <param name="ui_mode">answer</param>
  <param name="prior_transaction_id">583954092</param>
  <param name="iq_action">5</param>
  <param name="answer_id">16777216</param>
  <param name="highlight_info">4194305,7,13</param>
  <param name="turl">http://whoiam:8226/{instanceContext}/
    index?page=content&id=GL1&actp=search&viewlocale=
    en_US</param>
  </params>
  <anchor>__highlight</anchor>
</link>
<click_through_link type="text">http://whoiam:8223/inquiragw/
  ui.jsp?ui_mode=answer&prior_transaction_id=
  583954092&iq_action=4&answer_id=16777216&turl=
  http%3A%2F%2Fwhoiam%3A8226%2F%7BinstanceContext%7D%2Findex%3Fpage%
  3Dcontent%26id%3DGL1%26actp%3Dsearch%26viewlocale%3Den_US</
  click_through_link>
- <link type="click">
  <protocol>http</protocol>
  <host>whoiam</host>
  <port>8223</port>
  <path>/inquiragw/</path>
  <file>ui.jsp</file>
- <params>
  <param name="ui_mode">answer</param>
  <param name="prior_transaction_id">583954092</param>
  <param name="iq_action">4</param>
  <param name="answer_id">16777216</param>
  <param name="turl">http://whoiam:8226/{instanceContext}/
    index?page=content&id=GL1&actp=search&viewlocale=
    en_US</param>
  </params>

```

```

</link>
<similar_response_link type="text">http://whoiam:8223/inquiragw/
  ui.jsp?ui_mode=answer&prior_transaction_id=
  583954092&iq_action=12&answer_id=16777216&debug=
  true&related_ids=</similar_response_link>
- <link type="similar">
  <protocol>http</protocol>
  <host>whoiam</host>
  <port>8223</port>
  <path>/inquiragw/</path>
  <file>ui.jsp</file>
  - <params>
    <param name="ui_mode">answer</param>
    <param name="prior_transaction_id">583954092</param>
    <param name="iq_action">12</param>
    <param name="answer_id">16777216</param>
    <param name="debug">true</param>
    <param name="related_ids" />
  </params>
</link>
- <timestamp>
  <date>18</date>
  <month>11</month>
  <year>2009</year>
  <hour>4</hour>
  <minute>17</minute>
  <second>26</second>
  <millisecond>0</millisecond>
</timestamp>
- <facets>
  - <facet hidden="true">
    - <item id="CMS-SOURCE-TYPE" selected="false">
      <description>Content Source Type</description>
    </item>
    - <item id="CMS-SOURCE-TYPE.IM" selected="false">
      <description>IM</description>
    </item>
  </facet>
</facets>
</answer>
- <answer type="unstructured" score="0.9682692210865386" docType="CMS-XML"
  language="en-US" charset="UTF-8" collectionId="2" collectionName=
  "IM_GLEN_GLEN" answer_id="16777217" docId="4194307" imDocId="GL3"
  highlight_version="true">
  - <section>
    - <title url="http://whoiam:8226/{instanceContext}/index?page=
      content&id=GL3&actp=search&viewlocale=en_US">
    <snippet lvl="0">this document is in category 1 and 2</snippet>
    </title>
  - <text>
    <snippet lvl="0">To display a list of all available content records
      click List</snippet>
    </text>
  - <text url="http://whoiam:8226/{instanceContext}/index?page=
    content&id=GL3&actp=search&viewlocale=en_US">
    <snippet lvl="1">.</snippet>
    <snippet lvl="3">test</snippet>
    </text>
  - <text>
    <snippet lvl="0">Note:</snippet>
    </text>

```



```

</section>
<highlighted_link type="text">http://whoiam:8223/inquiragw/
  ui.jsp?ui_mode=answer&amp;prior_transaction_id=
    583954092&amp;iq_action=5&amp;answer_id=
    16777217&amp;highlight_info=4194307,54,55&amp;turl=
    http%3A%2F%2Fwhoiam%3A8226%2F%7BinstanceContext%7D%2Findex%3Fpage%
3Dcontent%26id%3DGL3%26actp%3Dsearch%26viewlocale%3Den_US#__highli      ght</
highlighted_link>
- <link type="highlight">
  <protocol>http</protocol>
  <host>whoiam</host>
  <port>8223</port>
  <path>/inquiragw/</path>
  <file>ui.jsp</file>
- <params>
  <param name="ui_mode">answer</param>
  <param name="prior_transaction_id">583954092</param>
  <param name="iq_action">5</param>
  <param name="answer_id">16777217</param>
  <param name="highlight_info">4194307,54,55</param>
  <param name="turl">http://whoiam:8226/{instanceContext}/
    index?page=content&amp;id=GL3&amp;actp=
search&amp;viewlocale=          en_US</param>
  </params>
  <anchor>__highlight</anchor>
</link>
<click_through_link type="text">http://whoiam:8223/inquiragw/
  ui.jsp?ui_mode=answer&amp;prior_transaction_id=
    583954092&amp;iq_action=4&amp;answer_id=16777217&amp;turl=
    http%3A%2F%2Fwhoiam%3A8226%2F%7BinstanceContext%7D%2Findex%3Fpage%3      D
content%26id%3DGL3%26actp%3Dsearch%26viewlocale%3Den_US</
  click_through_link>
- <link type="click">
  <protocol>http</protocol>
  <host>whoiam</host>
  <port>8223</port>
  <path>/inquiragw/</path>
  <file>ui.jsp</file>
- <params>
  <param name="ui_mode">answer</param>
  <param name="prior_transaction_id">583954092</param>
  <param name="iq_action">4</param>
  <param name="answer_id">16777217</param>
  <param name="turl">http://whoiam:8226/{instanceContext}/
index?page=          content&amp;id=GL3&amp;actp=search&amp;viewlocale=
en_US</param>
  </params>
</link>
<similar_response_link type="text">http://whoiam:8223/inquiragw/
  ui.jsp?ui_mode=answer&amp;prior_transaction_id=          583954092&amp;iq_action=
12&amp;answer_id=16777217&amp;debug=          true&amp;related_ids=</
similar_response_link>
- <link type="similar">
  <protocol>http</protocol>
  <host>whoiam</host>
  <port>8223</port>
  <path>/inquiragw/</path>
  <file>ui.jsp</file>
- <params>

```

```

        <param name="ui_mode">answer</param>
        <param name="prior_transaction_id">583954092</param>
        <param name="iq_action">12</param>
        <param name="answer_id">16777217</param>
        <param name="debug">true</param>
        <param name="related_ids" />
    </params>
</link>
- <timestamp>
    <date>18</date>
    <month>11</month>
    <year>2009</year>
    <hour>4</hour>
    <minute>18</minute>
    <second>22</second>
    <millisecond>0</millisecond>
</timestamp>
- <facets>
    - <facet hidden="true">
        - <item id="CMS-SOURCE-TYPE" selected="false">
            <description>Content Source Type</description>
        </item>
        - <item id="CMS-SOURCE-TYPE.IM" selected="false">
            <description>IM</description>
        </item>
    </facet>
</facets>
</answer>
</purpose>
</responses>
- <facets>
    - <result-facet>
        <id>DOC_TYPES</id>
        <description>DocTypes</description>
        <count>2</count>
    - <result-facet>
        <id>DOC_TYPES%2eCMS%2dXML</id>
        <description>CMS-XML</description>
        <count>2</count>
    </result-facet>
</result-facet>
- <result-facet>
    <id>COLLECTIONS</id>
    <description>Collections</description>
    <count>2</count>
    - <result-facet>
        <id>COLLECTIONS%2eIM_GLEN_GLEN</id>
        <description>IM_GLEN_GLEN</description>
        <count>2</count>
    </result-facet>
</result-facet>
- <result-facet>
    <id>CMS%2dVIEW</id>
    <description>Information Manager View</description>
    <count>2</count>
    - <result-facet>
        <id>CMS%2dVIEW%2eGLEN</id>
        <description>glen</description>
        <count>2</count>
    </result-facet>
</result-facet>

```

```

    </result-facet>
  - <result-facet>
    <id>CMS%2dCATEGORY</id>
    <description>Information Manager Category</description>
    <count>2</count>
    - <result-facet inEffect="true">
      <id>CMS%2dCATEGORY%2eCAT1</id>
      <description>cat1</description>
      <count>2</count>
    </result-facet>
  </result-facet>
  - <result-facet>
    <id>CMS%2dCHANNEL</id>
    <description>Information Manager Channel</description>
    <count>2</count>
    - <result-facet>
      <id>CMS%2dCHANNEL%2eGLEN</id>
      <description>glen</description>
      <count>2</count>
    </result-facet>
  </result-facet>
</facets>
- <constraint>
  <language>en-US</language>
  <result_language>en-US</result_language>
  <host>10.0.9.48</host>
  <address>10.0.9.48</address>
</constraint>
- <query>
  - <question referenceId="583954091" language="en-US" transactionId="583954092">
    <original>test</original>
    <paraphrase>test</paraphrase>
  </question>
  - <question language="en-US" transactionId="583954091">
    <original>test</original>
    <paraphrase>test</paraphrase>
  </question>
  - <question language="en-US" transactionId="583954089">
    <original>test</original>
    <paraphrase>test</paraphrase>
  </question>
</query>
- <config>
  <param name="searchWithin">false</param>
</config>
</message>

```