ORACLE®

AquaLogic® User Interaction

Development Guide

Version 6.x
Revised: January 2009

Contents

4. About Content Service Development

5. About Identity Services (IDS)

6. About AquaLogic Interaction 6.5 UI Customization

7. About the ALI Activity Stream API

8. About the Pathways API

9. API Libraries

10. Additional Development References

About ALUI Development

AquaLogic User Interaction (ALUI) is a powerful framework that combines portal, content management, collaboration, integration and search technologies, and provides key application services that allow you to build integrated solutions across diverse platforms and systems.

Every ALUI component is designed to provide a personalized experience for each organization and for specific groups and users. Using the AquaLogic Interaction Development Kit (IDK), you can create powerful applications to meet the specific needs of your organization.

- *About the ALUI Development Environment* on page 9: If you are developing services for AquaLogic Interaction, you will need to understand the system and prepare your IDE for use with the AquaLogic Interaction Development Kit (IDK).
- *About Pagelets and Portlets* on page 47: Pagelets and portlets are web applications that produce a self-contained, reusable user interface widget. Pagelets can be used for everything from displaying useful information to building integrated applications that combine functionality from multiple systems.
- *About Adaptive Page Layouts (ALI 6.5)* on page 426: Adaptive page layouts allow you to change the look and feel of the portal user interface using adaptive tags in standard XHTML.
- *About Content Service Development* on page 349: Content services allow you to search external repositories through the portal and index external content in the portal Knowledge Directory. These services allow users to access documents and other resources from multiple repositories without leaving the portal workspace.
- *About Identity Services (IDS)* on page 391 : Identity Services allow you to integrate established repositories of user information into your portal, and *About Profile Services (PWS)* on page 410: Profile services are used to import information about existing portal users from external systems. This information is mapped to portal properties and made available to other services.

About the ALUI Development Environment

If you are developing services for AquaLogic Interaction, you will need to understand the system and prepare your IDE for use with the AquaLogic Interaction Development Kit (IDK).

The following topics provide step-by-step instructions for the most common tasks in setting up an IDK development environment. For details on installing the IDK, see the *IDK Product Documentation*. To download the IDK or ALI Logging Utilities, go to the *ALUI Developer Center* on BEA dev2dev.

Java

- *Setting Up a Custom Java IDK Project in Eclipse with WTP* on page 10
- *Setting Up a Custom Java IDK Project in Eclipse Stand-Alone (without WTP)* on page 12
- *Deploying a Custom Java IDK Project in Eclipse with WTP* on page 10
- *Deploying a Custom Java IDK Project in Eclipse Stand-Alone (without WTP)* on page 12
- *Debugging a Custom IDK Project - Java* on page 13

.NET

- *Setting Up a Custom .NET IDK Project* on page 14
- *Deploying a Custom .NET IDK Project in IIS* on page 15

Related Topics

- *About Server Communication and the Gateway* on page 40
- *About the IDK Logging API* on page 16

Setting Up a Custom Java IDK Project in Eclipse with WTP

These steps describe how to set up a custom Java IDK project in Eclipse with WTP installed.

These instructions assume you have installed the Java version of the AquaLogic Interaction Development Kit (IDK).

1. Open Eclipse and click **File ➤ New ➤ Other ➤ Web ➤ Dynamic Web Project** .
2. Type the **Project Name** (for example, "idkproject").
3. Choose a **Target Runtime** from the drop-down list. If you have not previously configured a server runtime, click **New...** to configure your Tomcat setup.
4. Click **Finish** to complete the Dynamic Web Project wizard.
5. Import the IDK web project template:
   a) Right-click the project in the Project Explorer and click **Import ➤ General ➤ File System** .
   b) To define the **From directory** field, navigate to the IDK root directory and select the **\devkit\WEB-INF** folder.
   c) Change the **Into folder** field to **<project name>/WebContent/WEB-INF**.
   d) Click **Finish**.

Note: The Eclipse Web project view hides the imported JARs stored in WEB-INF/lib and puts those files under ./Java Resources/src/Libraries/Web App Libraries.

Deploying a Custom Java IDK Project in Eclipse with WTP

These steps describe how to deploy a custom Java IDK project in Eclipse with WTP installed.

These instructions use Tomcat as an example.

1. Define the server in Eclipse:
   a) Click **File ➤ New ➤ Other ➤ Server ➤ Server** and click **Next**.
   b) Select the server type (Tomcat v5.0) and click **Next**.
   c) Select the Tomcat v5.0 installation directory and click **Next**.
   d) Add your custom project to the list of configured projects and click **Finish**.
2. Run and debug the application:

a) In Project Explorer, right-click your custom project and click **Debug As ➤ Debug On Server** .
b) Select the existing server and click **Finish**.

3. Content services, identity services and SCI pages require additional configuration. You must add the custom class to the appropriate *Impl keys in the web.xml file in the WEB-INF directory. For details, see *Web Service Class Names (*Impl)* on page 421.
4. When Tomcat starts in a new **Servers** tab, hit http://localhost:8080/<projectname>/servlet/AxisServlet to ensure that Axis has deployed correctly and the web service APIs are correctly configured.

Debugging a Custom IDK Project - Java

After you create a custom IDK project, you must deploy it in your Java application server.

These instructions use Tomcat as an example.

1. Define the server in Eclipse:
   a) Click **File ➤ New ➤ Other ➤ Server ➤ Server** and click **Next**.
   b) Select the server type as Tomcat v5.0 and click **Next**.
   c) Select the Tomcat v5.0 installation directory and click **Next**.
   d) Add your project to the list of configured Tomcat projects and click **Finish**.
2. Content services, identity services and SCI pages require additional configuration. You must add the custom class to the appropriate *Impl keys in the web.xml file in the WEB-INF directory. For details on Impl keys, see *Web Service Class Names (*Impl)* on page 421.
3. Run and debug the application:
   a) In Eclipse Project Explorer, right-click your project and click **Debug As ➤ Debug On Server** .
   b) Select the existing server and click **Finish**.
4. When Tomcat starts in a new **Servers** tab, hit `http://localhost:8080/<project name>/servlet/AxisServlet` to ensure that Axis has deployed correctly and the web service APIs are correctly configured.

Setting Up a Custom Java IDK Project in Eclipse Stand-Alone (without WTP)

These steps describe how to set up a custom Java IDK project in Eclipse stand-alone (without WTP installed).

These instructions assume you have installed the Java version of the AquaLogic Interaction Development Kit (IDK).

1. Open Eclipse and click **File ➤ New ➤ Project** .
2. Type the **Project Name** (for example, "idkproject"). Click **Next** and **Finish**.
3. In the Package Explorer in Eclipse, right-click on the new project and click **Properties ➤ Java Build Path ➤ Libraries ➤ Add External Jars** .
4. Select the *.jar files from the IDK installation directory under the idk\6.0\devkit\java\WEB-INF\lib directory. Click **OK**.

Deploying a Custom Java IDK Project in Eclipse Stand-Alone (without WTP)

These steps describe how to deploy a custom Java IDK project in Eclipse stand-alone (without WTP installed).

The instructions below are for Tomcat or WebLogic. For IBM WebSphere, you must create a .war or .ear file that is compatible with WebSphere. You must first create an appropriate server-config.wsdd using the IDK DeployServlet or the supplied service wsdd files. See the WebSphere documentation for detailed instructions

1. Deploy the IDK in your application server:
   a) Create a folder for the custom project in the application server's **\webapps** directory. (For example, if Tomcat is installed in C:\tomcat and the project name is "idkproject", the path would be C:\tomcat\webapps\idkproject.)
   b) Navigate to the IDK installation directory and copy the **WEB-INF** and its **\LIB** subfolder to the directory you created in the previous step. This loads Apache AXIS into the application server.

c) Confirm that Apache AXIS is available by opening the following page in a browser: http://<hostname:port>/<projectname>/servlet/AxisServlet. (Change <hostname:port> to fit your application server, for example, localhost:8080 for Tomcat. Change <projectname> to the name of the folder you created in step 1a.) The browser should display the message "And now... Some Services" and a list of installed services.

2. Compile the class that implements the IDK interface(s) and copy the entire package structure to the appropriate location in your web application, usually the \WEB-INF\classes directory.
3. Content services, identity services and SCI pages require additional configuration. You must add the custom class to the appropriate *Impl keys in the web.xml file in the WEB-INF directory. For details, see *Web Service Class Names (*Impl)* on page 421.
4. Start your application server. In most cases, you must restart your application server after copying a file.

Debugging a Custom IDK Project - Java

After you create a custom IDK project, you must deploy it in your Java application server.

These instructions use Tomcat as an example.

1. Define the server in Eclipse:
   a) Click **File ➤ New ➤ Other ➤ Server ➤ Server** and click **Next**.
   b) Select the server type as Tomcat v5.0 and click **Next**.
   c) Select the Tomcat v5.0 installation directory and click **Next**.
   d) Add your project to the list of configured Tomcat projects and click **Finish**.
2. Content services, identity services and SCI pages require additional configuration. You must add the custom class to the appropriate *Impl keys in the web.xml file in the WEB-INF directory. For details on Impl keys, see *Web Service Class Names (*Impl)* on page 421.
3. Run and debug the application:
   a) In Eclipse Project Explorer, right-click your project and click **Debug As ➤ Debug On Server** .
   b) Select the existing server and click **Finish**.
4. When Tomcat starts in a new **Servers** tab, hit `http://localhost:8080/<project name>/servlet/AxisServlet` to ensure that Axis has deployed correctly and the web service APIs are correctly configured.

Setting Up a Custom .NET IDK Project

These steps describe how to set up a custom .NET IDK project in Visual Studio.

These instructions assume you have installed the .NET version of the AquaLogic Interaction Development Kit (IDK).

1. Start Visual Studio and click **File ➤ New Project ➤ C# Projects ➤ ASP.NET Web Service** .
2. Type an intuitive name in the **Location** field.
3. Delete Service1.asmx and Web.config.
4. In the new project, click **File ➤ Add Existing Item** .
5. Browse to the **\devkit** folder in the IDK installation directory.
6. In the **File Types** mask, click **All Files**.
7. Select all the .asmx files and Web.config. Do not select the \bin directory.
8. Click **Open** . You will be prompted to create a class file for each .asmx file; click **No** for each file.
9. In the Solution Explorer (usually in the upper right), you should see the project you created in step 1. Add the IDK assemblies:
   a) Right-click **References** and click **Add Reference** .
   b) Browse to the **\devkit\bin** folder in the IDK installation directory.
   c) Select the assemblies to add to the bin directory: all the .dll files (Ctrl+A). These are the assemblies that resolve the references in the *.asmx files.
      - If you are using the standard (un-signed) version of the IDK, select all the .dll files (Ctrl+A).
      - If you are using the signed dll version of the IDK, select only Plumtree.openlog-framework_signed.dll. (You must deploy the other assemblies in the GAC as described in step f below.)

   d) Click **Open ➤ OK** .
   e) In the Solution Explorer References, confirm that you now see idk, openfoundation, etc.
   f) If you are using the signed dll version of the IDK, deploy the following assemblies in the GAC:
      - Plumtree.EDK_signed.dll

- OpenFoundation_signed.dll
- Plumtree.openkernel_signed.dll
- Plumtree.openlog-framework_signed.dll
- Plumtree.pmb_signed.dll
- Plumtree.RAT_signed.dll

10. Click **File ➤ Add New Item** to create new classes and complete your project.

Deploying a Custom .NET IDK Project in IIS

These steps describe how to deploy a custom .NET IDK project in IIS.

These instructions assume you have installed the .NET version of the AquaLogic Interaction Development Kit (IDK) and set up Visual Studio for IDK development.

1. Compile the class that implements the IDK interface(s).
2. Content services, identity services and SCI pages require additional configuration. You must add the class and the assembly that contains it to the appropriate *Assembly and *Impl keys in the web.config file in your project. For details, see *Web Service Class Names (*Impl)* on page 421.
3. If you do not already have a virtual directory in IIS for your services, add one using the steps below:
   a) Navigate to **Internet Services Manager (Internet Information Services)** in the Control Panel under Administrative Tools.
   b) Select **Default Web Site.**
   c) Click **Action ➤ New ➤ Virtual Directory** and type the name of your Visual Studio location.
   d) Click **Next** twice. Type the path to the home directory for the IDK: <installdir>\idk\6.0\devkit\dotnet.
   e) Check both the **Read** and **Scripts only** checkboxes if they are cleared (they should be checked by default). Click **Next** then click **Finish**.
4. Copy the compiled class files to the \bin folder in the <installdir>\idk\6.0\devkit\dotnet directory.

About ALI Logging Utilities

AquaLogic Interaction (ALI) provides a collection of debugging and logging solutions.

ALI Logging Utilities allow for a wide variety of logging solutions. The IDK provides a remote API that allows you to send logging messages from remote web applications.

For details, see the following topics:

- *About the IDK Logging API* on page 16
- *Using IDK Logging in Java* on page 17
- *Using IDK Logging in .NET* on page 21
- *Using IDK Logging from the Command Line* on page 27

In addition, the ALI System Health Monitor provides real-time access to performance information on remote servers, custom objects and ALI services. To access the System Health Monitor, go to portal administration and click **Select Utility ➤ System Health Monitor** . For details, see the portal online help.

About the IDK Logging API

The AquaLogic Interaction Development Kit (IDK) logging API allows you to send log messages from remote services and applications to a variety of logging receivers.

The com.plumtree.remote.logging package provides two interfaces:

- `LogFactory` provides static methods to configure logging, query configuration properties, and obtain `ILogger` instances.
- `ILogger` allows you to test if various log levels are enabled and provides logging methods. To create a logger object, call `LogFactory.getLogger()`.

For details, see the following topics:

- *Using IDK Logging in Java* on page 17
- *Using IDK Logging in .NET* on page 21
- *Using IDK Logging from the Command Line* on page 27
- *Configuring IDK Logging* on page 27: IDK logging is not enabled by default. You can enable logging options programmatically or using the web.xml or Web.config file distributed with the IDK. Verbose logging cannot be enabled programmatically; you must change a setting in the web.xml or Web.config file.

- *Configuring Java IDK Logging (web.xml)* on page 28
- *Configuring .NET IDK Logging (Web.config)* on page 29
- *IDK Logging Levels* on page 30
- *IDK Logging API Web Application Variables* on page 32

Using IDK Logging in Java

This example demonstrates how to enable and use IDK logging in a remote Java application.

1. The first step in this example is to enable logging programmatically, by defining the logging application name and setting the log to network option to true. For details on logging options, see *Configuring Java IDK Logging (web.xml)* on page 28.

```
import com.plumtree.remote.logging.ILogger;
import com.plumtree.remote.logging.LogFactory;

public class LoggingExample extends Thread
{
   private static final String INSTANCES_COMPONENT_NAME =
'Instances';
   private static final String MAIN_LOOP_COMPONENT_NAME = 'Main
 Loop';

   // set the application name
   // (legal characters: ASCII alphanumerics plus . - _ and
space)
   public static final String LOGGING_APPLICATION_NAME =
'Logging_API_Example-1';

   // set to true to multicast log messages to local network
   // set to false to send message only listeners on local
machine
   public static final boolean LOG_TO_NETWORK = true;

   private ILogger logger;   //instance logging class
   private static ILogger mainLogger;   // main component
logging class
```

2. Initialize `LogFactory`. The recommended way to initialize non-web applications is in a static block in the application's main class or a logging utility class. Always check to see if

`LogFactory` has already been initialized (for example, as part of an IDK-based web application).

```
if (!LogFactory.isInitialized())
{
   LogFactory.initialize(LOGGING_APPLICATION_NAME,
LOG_TO_NETWORK);
}
System.out.print('Set your logging receiver to the \'server\'
or \'application name\' ');
System.out.println(LogFactory.getApplicationName());
System.out.println('The logging component names are \'EDK\',
\'' + MAIN_LOOP_COMPONENT_NAME + '\' and \''
+ INSTANCES_COMPONENT_NAME + '\'.');

mainLogger = LogFactory.getLogger(MAIN_LOOP_COMPONENT_NAME,
LoggingExample.class);
```

This code creates the following messages in ALI Logging Spy. These messages are sent automatically by the IDK. For the sample code above, the <app name> entry would be Logging_API_Example-1.

1 <#> <app name> <date/time> Info EDK main LogFactory Initiating EDK logging on behalf of EDK: LogFactory.

2 <#> <app name> <date/time> Info EDK main LogFactory Verbose logging of internal EDK classes is off. It may be enabled by setting ptedk.VerboseLogging='true' .

3. Create an instance of `ILogger` by calling `LogFactory.getLogger`. In the code below, the `LoggingExample` method sends an Info level log message when an instance is created. The snippet below also uses `ILogger.functionBegin` and `ILogger.functionEnd` to log when a method is entered and exited, `ILogger.action` to log significant events, and `ILogger.performanceBegin` and `ILogger.performanceEnd` to log the time required to execute the methods.

```
public LoggingExample(String instanceName)
{
   setName(instanceName);
   this.logger = LogFactory.getLogger(INSTANCES_COMPONENT_NAME,
 LoggingExample.class);
   mainLogger.info('Created new instance named {0}',
instanceName);
}
public static void main(String[] args)
{
   final String methodName = 'main';
```

```
    mainLogger.functionBegin(methodName);

    // get a timestamp to measure performance of this function
    long performanceStartTicks = mainLogger.performanceBegin();


    mainLogger.action('Creating and starting instances');

    LoggingExample bill = new LoggingExample('Bill');
    bill.start();
    LoggingExample larry = new LoggingExample('Larry');
    larry.start();

    mainLogger.action('Done creating instances');

    // send log message with time since performanceBegin
   mainLogger.performanceEnd(methodName, performanceStartTicks);


    mainLogger.functionEnd(methodName);
}
```

This code creates the following messages in ALI Logging Spy.

3 <#> <app name> <date/time> Function Main Loop main LoggingExample Entering Function main

4 <#> <app name> <date/time> Action Main Loop main LoggingExample Creating and starting instances

5 <#> <app name> <date/time> Info Main Loop main LoggingExample Created new instance named Bill

6 <#> <app name> <date/time> Info Main Loop main LoggingExample Created new instance named Larry

7 <#> <app name> <date/time> Action Main Loop main LoggingExample Done creating instances

8 <#> <app name> <date/time> Performance Main Loop main LoggingExample main took 0 ms.

9 <#> <app name> <date/time> Function Main Loop main LoggingExample Leaving Function mainInfo

4. The code below demonstrates available logging levels and provides an example of how to use token substitution in formatting strings to construct messages. The thread runs through a small test of logging messages and transfers work to the next by calling `yield()`. Note: Wrap any complex message construction in a conditional block to avoid doing work if there are no listeners at that log level.

```
public void run()
{
   String levelDescriptionFormat = '{0} level messages are {1}
 by default in the log receiver.';
   logger.debug(levelDescriptionFormat, 'Debug', 'off');
   logger.info(levelDescriptionFormat, 'Info',  'off');
   logger.warn(levelDescriptionFormat, 'Warn',  'on');
   logger.error(levelDescriptionFormat, 'Error', 'on');
   logger.fatal(levelDescriptionFormat, 'Fatal', 'on');

   yield();

   // Exceptions may also be caught and logged, and may use
token substitution
   try
   {
      throw new InterruptedException(getName() + ' was
interrupted.');
   }
   catch (Exception eCaught)
   {
      logger.warn(eCaught, 'Caught an exception from {0}. ',
eCaught.getClass().getPackage().getName());
   }
}
```

This code creates the following messages in ALI Logging Spy:

10 <#> <app name> <date/time> Function Instances Larry LoggingExample Entering Function run

11 <#> <app name> <date/time> Action Instances Bill LoggingExample Action log messages are on by default in the log receiver.

12 <#> <app name> <date/time> Debug Instances Bill LoggingExample Debug level messages are off by default in the log receiver.

13 <#> <app name> <date/time> Info Instances Bill LoggingExample Info level messages are off by default in the log receiver.

14 <#> <app name> <date/time> Warning Instances Bill LoggingExample Warn level messages are on by default in the log receiver.

15 <#> <app name> <date/time> Error Instances Bill LoggingExample Error level messages are on by default in the log receiver.

16 <#> <app name> <date/time> Fatal Instances Bill LoggingExample Fatal level messages are on by default in the log receiver.

17 <#> <app name> <date/time> Action Instances Larry LoggingExample Action log messages are on by default in the log receiver.

18 <#> <app name> <date/time> Debug Instances Larry LoggingExample Debug level messages are off by default in the log receiver.

19 <#> <app name> <date/time> Info Instances Larry LoggingExample Info level messages are off by default in the log receiver.

20 <#> <app name> <date/time> Warning Instances Larry LoggingExample Warn level messages are on by default in the log receiver.

21 <#> <app name> <date/time> Error Instances Larry LoggingExample Error level messages are on by default in the log receiver.

22 <#> <app name> <date/time> Fatal Instances Larry LoggingExample Fatal level messages are on by default in the log receiver.

23 <#> <app name> <date/time> Warning Instances Bill LoggingExample Caught an exception from - java.lang. java.lang.InterruptedException: Bill was interrupted. - java.lang.InterruptedException: Bill was interrupted. at - com.plumtree.remote.logging.example.LoggingExample.run(LoggingExample.java:110)

24 <#> <app name> <date/time> Warning Instances Larry LoggingExample Caught an exception from - java.lang. java.lang.InterruptedException: Larry was interrupted. - java.lang.InterruptedException: Larry was interrupted. at - com.plumtree.remote.logging.example.LoggingExample.run(LoggingExample.java:110)

Using IDK Logging in .NET

This example demonstrates how to use IDK logging in a remote .NET application.

1. The first step in this example is to enable logging programmatically, by defining the logging application name and setting the log to network option to true. For details on logging options, see *Configuring .NET IDK Logging (Web.config)* on page 29.

```
using System;
using System.Threading;
using Plumtree.Remote.Logging;

public class LoggingCommandLineExample
{
   private static readonly String INSTANCES_COMPONENT_NAME =
'Instances';
   private static readonly String MAIN_LOOP_COMPONENT_NAME =
'Main Loop';

   // set the application name
   // (legal characters: ASCII alphanumerics plus . - _ and
space)
   public static readonly String LOGGING_APPLICATION_NAME =
'Logging_API_Example-1';

   // set to true to multicast log messages to local network
   // set to false to send message only listeners on local
machine
   public static readonly bool LOG_TO_NETWORK = true;

   private ILogger logger;   //instance logging class
   private static ILogger mainLogger;   // main component
logging class

   // thread for each instance of LoggingCommandLineExample
   private Thread _thread;
```

2. Initialize `LogFactory`. The recommended way to initialize non-web applications is in a static block in the application's main class or a logging utility class. Always check to see if `LogFactory` has already been initialized (for example, as part of an IDK-based web application).

```
if (!LogFactory.isInitialized())
{
   LogFactory.Initialize(LOGGING_APPLICATION_NAME,
LOG_TO_NETWORK);
}
Console.Out.WriteLine('Set your logging receiver to the
\'server\' or \'application name\' ');
```

```
Console.Out.WriteLine(LogFactory.GetApplicationName());
Console.Out.WriteLine('The logging component names are \'EDK\',
 \'' + MAIN_LOOP_COMPONENT_NAME + '\' and \'' +
INSTANCES_COMPONENT_NAME + '\'.');

mainLogger = LogFactory.GetLogger(MAIN_LOOP_COMPONENT_NAME,
typeof(LoggingCommandLineExample));
```

This code creates the following messages in ALI Logging Spy. These messages are sent automatically by the IDK. For the sample code above, the <app name> entry would be Logging_API_Example-1.

1 <#> <app name> <date/time> Info EDK main LogFactory Initiating EDK logging on behalf of EDK: LogFactory.

2 <#> <app name> <date/time> Info EDK main LogFactory Verbose logging of internal EDK classes is off. It may be enabled by setting ptedk.VerboseLogging='true'.

3. Create an instance of `ILogger` by calling `LogFactory.getLogger`. In the code below, the LoggingExample method sends an Info level log message when an instance is created. The snippet below also uses `ILogger.functionBegin` and `ILogger.functionEnd` to log when a method is entered and exited, `ILogger.action` to log significant events, and `ILogger.performanceBegin` and `ILogger.performanceEnd` to log the time required to execute the methods.

```
public LoggingCommandLineExample(String instanceName)
{
    _thread = new Thread(new ThreadStart(Run));
    _thread.Name = instanceName;
   this.logger = LogFactory.GetLogger(INSTANCES_COMPONENT_NAME,
 typeof(LoggingCommandLineExample));
    mainLogger.Info('Created new instance named {0}',
instanceName);
}
[STAThread]
public static void main(String[] args)
{
    String methodName = 'main';
    mainLogger.FunctionBegin(methodName);

    // get a timestamp to measure performance of this function
    long performanceStartTicks = mainLogger.PerformanceBegin();


    mainLogger.Action('Creating and starting instances');
```

```
    LoggingExample bill = new LoggingExample('Bill');
    bill.Thread.Start();
    LoggingExample larry = new LoggingExample('Larry');
    larry.Thread.Start();

    mainLogger.Action('Done creating instances');

    // send log message with time since performanceBegin
   mainLogger.PerformanceEnd(methodName, performanceStartTicks);


    mainLogger.FunctionEnd(methodName);
}
```

This code creates the following messages in ALI Logging Spy.

3 <#> <app name> <date/time> Function Main Loop main LoggingExample Entering Function main

4 <#> <app name> <date/time> Action Main Loop main LoggingExample Creating and starting instances

5 <#> <app name> <date/time> Info Main Loop main LoggingExample Created new instance named Bill

6 <#> <app name> <date/time> Info Main Loop main LoggingExample Created new instance named Larry

7 <#> <app name> <date/time> Action Main Loop main LoggingExample Done creating instances

8 <#> <app name> <date/time> Performance Main Loop main LoggingExample main took 0 ms.

9 <#> <app name> <date/time> Function Main Loop main LoggingExample Leaving Function mainInfo

4. The code below demonstrates available logging levels and provides an example of how to use token substitution in formatting strings to construct messages. The thread runs through a small test of logging messages and interleaves the messages using `Thread.Sleep`. Note: Wrap any complex message construction in a conditional block to avoid doing work if there are no listeners at that log level.

```
public void Run()
{
    String methodName = 'run';
```

```
   // send log message that function is starting
   logger.FunctionBegin(methodName);

   // get a timestamp to measure performance of this function
   long performanceStartTicks = mainLogger.PerformanceBegin();

   Thread.Sleep(1);    // interleaves work to the other thread


   String levelDescriptionFormat = '{0} level messages are {1}
 by default in the log receiver.';
   logger.Debug(levelDescriptionFormat, 'Debug', 'off');
   logger.Info(levelDescriptionFormat, 'Info',  'off');
   logger.Warn(levelDescriptionFormat, 'Warn',  'on');
   logger.Error(levelDescriptionFormat, 'Error', 'on');
   logger.Fatal(levelDescriptionFormat, 'Fatal', 'on');

   Thread.Sleep(1);    // interleaves work to the other thread


   // Exceptions may also be caught and logged, and may use
token substitution
   try
   {
      throw new ThreadInterruptedException(_thread.Name + ' was
 interrupted.');
   }
   catch (Exception eCaught)
   {
      logger.Warn(eCaught, 'Caught an exception from {0}. ',
eCaught.GetType().Name);
   }

   Thread.Sleep(1);    // interleaves work to the other thread


   // send log message with time since performanceBegin
  mainLogger.PerformanceEnd(methodName, performanceStartTicks);


   // send log message that function is ending
   logger.FunctionEnd(methodName);
}
public Thread Thread
```

```
{
   get
   {
      return _thread;
   }
}
```

This code creates the following messages in ALI Logging Spy:

10 <#> <app name> <date/time> Function Instances Larry LoggingExample Entering Function run

11 <#> <app name> <date/time> Action Instances Bill LoggingExample Action log messages are on by default in the log receiver.

12 <#> <app name> <date/time> Debug Instances Bill LoggingExample Debug level messages are off by default in the log receiver.

13 <#> <app name> <date/time> Info Instances Bill LoggingExample Info level messages are off by default in the log receiver.

14 <#> <app name> <date/time> Warning Instances Bill LoggingExample Warn level messages are on by default in the log receiver.

15 <#> <app name> <date/time> Error Instances Bill LoggingExample Error level messages are on by default in the log receiver.

16 <#> <app name> <date/time> Fatal Instances Bill LoggingExample Fatal level messages are on by default in the log receiver.

17 <#> <app name> <date/time> Action Instances Larry LoggingExample Action log messages are on by default in the log receiver.

18 <#> <app name> <date/time> Debug Instances Larry LoggingExample Debug level messages are off by default in the log receiver.

19 <#> <app name> <date/time> Info Instances Larry LoggingExample Info level messages are off by default in the log receiver.

20 <#> <app name> <date/time> Warning Instances Larry LoggingExample Warn level messages are on by default in the log receiver.

21 <#> <app name> <date/time> Error Instances Larry LoggingExample Error level messages are on by default in the log receiver.

22 <#> <app name> <date/time> Fatal Instances Larry LoggingExample Fatal level messages are on by default in the log receiver.

23 <#> <app name> <date/time> Warning Instances Bill LoggingExample Caught an exception from - java.lang. java.lang.InterruptedException: Bill was interrupted. - java.lang.InterruptedException: Bill was interrupted. at - com.plumtree.remote.logging.example.LoggingExample.run(LoggingExample.java:110)

24 <#> <app name> <date/time> Warning Instances Larry LoggingExample Caught an exception from - java.lang. java.lang.InterruptedException: Larry was interrupted. - java.lang.InterruptedException: Larry was interrupted. at - com.plumtree.remote.logging.example.LoggingExample.run(LoggingExample.java:110)

Using IDK Logging from the Command Line

These instructions explain how to run the IDK Logging API example code (Java or .NET) from the command line.

1. Scan the sample code and note the LOGGING_APPLICATION_NAME parameter declared near the top of the class. Change this value if you wish, and record it.
2. Java: Compile with all the idk jar files in the classpath. Make sure servlet.jar and all idk jar files are in the classpath. .NET: Compile the source with reference to idk.dll and its supporting DLLs.
3. Launch ALI Logging Spy. Go to the **Filters** dialog box and add a new server (right-click and select **Add Server**). Enter the value set for LOGGING_APPLICATION_NAME in the **Add Server** dialog box and click **OK** . Wait a few seconds until a new entry appears in the Filter Settings list .
4. Run the example from the command line. Note any messages displayed in ALI Logging Spy. Error and exception logs are included in the logging demonstration.
5. Go back to the **Filters** dialog in ALI Logging Spy. Click the gray selection box beside the 'server' entry to accept logging for all logging levels.
6. Run the example again. Note that the messages displayed now in ALI Logging Spy include examples of all logging levels, including error and exception logs.

Configuring IDK Logging

To enable and configure IDK logging, first determine how the IDK is deployed.

IDK logging is disabled by default. If logging is enabled, it is sent only to the local machine by default, requiring direct access to the machine to view the logs. These default settings were chosen to secure potentially sensitive information present in log messages.

- If the IDK is deployed as a **web application** to support Integration Service implementations, edit the distributed web application configuration file (web.xml or Web.config). For details, see *Configuring Java IDK Logging (web.xml)* on page 28 or *Configuring .NET IDK Logging (Web.config)* on page 29
- If the IDK is deployed as a **library** supporting a web application (for example, a portlet), copy and paste the configuration parameters from the IDK's distributed web.xml/Web.config into your web application configuration file. For details, see *Configuring Java IDK Logging (web.xml)* on page 28 or *Configuring .NET IDK Logging (Web.config)* on page 29.
- If the IDK is deployed as a **stand-alone application** outside a web application context, such as report-generating or data loading and dumping applications using the PRC, use programmatic configuration to initialize logging parameters. Programmatic logging configuration can be done at startup, or by using a static initialization call on a façade class that the Web application runtime code uses to obtain IDK logging components or logger instances. For details, see *About the IDK Logging API* on page 16.

To use the IDK Logging API, you must configure the logging receiver to read logs from the IDK. To configure the log receiver, you must know the logging application name. The IDK logging application name is configured in the web application configuration file or set via the `initialize()` method in the IDK Logging API.

Configuring Java IDK Logging (web.xml)

For web services using the Java IDK, the web.xml file is the standard way to configure log instrumentation.

The example below shows the logging settings only. The bulk of the web.xml file has been omitted; environment keys are inserted at the end according to the DTD.

```
 <?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE web-app
PUBLIC '-//Sun Microsystems, Inc.//DTD Web
Application 2.2//EN
'http://java.sun.com/j2ee/dtds/web-app_2.2.dtd'>
<web-app>
...
<env-entry>

<env-entry-name>ptedk.VerboseLogging</env-entry-name

    <env-entry-value>true</env-entry-value>
```

```
<env-entry-type>java.lang.Boolean</env-entry-type>
</env-entry>
<env-entry>


<env-entry-name>ptedk.LoggingApplicationName</env-entry-name>

    <env-entry-value>EDK</env-entry-value>

<env-entry-type>java.lang.String</env-entry-type>
</env-entry>
<env-entry>


<env-entry-name>ptedk.LogToNetwork</env-entry-name>

    <env-entry-value>true</env-entry-value>

<env-entry-type>java.lang.Boolean</env-entry-type>
  </env-entry>
<web-app>
```

Configuring .NET IDK Logging (Web.config)

If you are running the .NET IDK as a web application that hosts web services (that do not use the logging API), the Web.config file is the best way to configure log instrumentation.

The IDK Web.config follows the normal precedence rules of IIS Web.config: within a web application, machine.config is read first for configuration values, then overlaid with Web.config from each parent directory within the web application subtree down to the directory containing the running code. The example below shows the logging settings only.

All .NET web applications have Web.config files. If the configuration file does not have an <appSettings> section, it can be added along with the key-value pairs to configure logging.

```
<configuration>
  <appSettings>
      <add key="ptedk.LoggingApplicationName"
value="Bulk-Document-Loader" />
     <add key="ptedk.LogToNetwork" value="true" />

      <add key="ptedk.VerboseLogging" value="true"
```

```
 />
   </appSettings>
   <system.web>
  ...
   </system.web>
</configuration>
```

For stand-alone .NET applications outside a Web application context, use programmatic configuration. For details, see *Using IDK Logging in .NET* on page 21.

IDK Logging Levels

This page summarizes logging levels and their implementation in IDK logging.

The IDK ILogger interface provides access to all eight standard logging levels.

Severity-Based Logging Levels

Logging Level	Description	IDK Implementation
Debug	The most common and numerous log call, used for detailed call tracing and parameter logging. The message should contain a detailed descriptive message reporting a minor step taken in the code or providing variable values (or both).	Remote call tracing. Function parameters. ToString() of portlet settings or service request.
Info	Used for normal but significant events. Reports a common operation that is of possible interest, for example, serving a new user request or making a remote procedure call.	New portlet or service request. PRC session initialization (login). The IDK logging service sends an Info message to the "EDK main" logging component when it is initialized.
Warn	Used for minor problems. Indicates a possible problem which the person responsible for the application should be aware of.	Expected (application) exceptions. Portlet: non-gatewayed request, missing settings. The IDK logging service sends a Warn message to the 'EDK main' logging component when it is initialized if verbose logging is enabled, since the network or application administrator should be aware of possible security implications of

Logging Level	Description	IDK Implementation
		sending remote call parameters to a cleartext logging channel.
Error	Used for major problems affecting application function. Indicates a definite problem that should to be corrected. The message should state and explain the problem and suggest how to fix it.	Unexpected platform exceptions. Portlet: error parsing CSP headers.
Fatal	Used for problems so severe that the application cannot continue to function. The message should state the problem, explain why it is a problem, and suggest how to fix it. Examples include inability to obtain necessary system or network resources.	A Fatal message is logged when an instance of the class configured for the Web Service object cannot be instantiated. Otherwise reserved for application developer use.

Supplemental Logging Levels

Logging Level	Description	IDK Implementation
Action	Used for significant actions (between Info and Warn in severity). Examples include the beginning or ending of a startup routine or the start or completion of a new user request.	Initialize an application component or a new remote session.
Function	Used to bracket the beginning and ending of a function. Use at the very beginning and end of methods to illustrate code paths and provide context for messages occurring between the beginning and ending function messages.	Dispatching and receiving a remote call, and parsing request parameters.
Performance	Provides a millisecond timestamp (for example, operation X took # milliseconds). Use to measure operations that may be costly in time. Typically a pair of begin and end performance calls will bracket a blocking call to an operation of interest such as a disk	PRC remote calls. Web request lifecycle for services.

Logging Level	Description	IDK Implementation
	read or write, remote call, external process invocation, database query, or large sort operation.	

IDK Logging API Web Application Variables

To enable IDK logging, you must enter the application name and change the settings in the web.xml (Java) or Web.config (.NET) file distributed with the IDK. This page lists the applicable variables.

Setting	Default Value	Description
ptedk.LoggingApplicationName	**""** (No logging occurs if the application name is not set.)	OpenLog and ALI Logging Spy use a text string (OpenLog: 'Application' / PTSpy: 'server') to identify a specific log channel to which log appenders can send messages, and from which log receivers can receive messages. To receive messages sent to an OpenLog channel, a listening application must be configured with the same application name used by the log-generating application. To receive log messages from an existing IDK deployment in a web application, set values for the name and logging options according to the example in the IDK web.xml or Web.config file. To receive log messages from a non-Web application that uses the IDK (for example, batch or utility processes using IDK Remote APIs), set the logging application name programmatically. Use the value in the key ptedk.LoggingApplicationName to set a matching server name in the logging receiver. Note: If the application is already using OpenLog and also using the IDK, the IDK code

Setting	Default Value	Description
		must not attempt to initialize OpenLog with a different application name
ptedk.LogToNetwork	**false** (Logs to local machine only.)	Logging to the network is disabled by default. In this condition, log messages can only be received by OpenLog receiver processes on the local machine, including ALI Logging Spy, the File Logger, or receivers using the OpenLog-Log4J Bridge. Logging can be enabled by setting the value associated with ptedk.LogToNetwork to true in the web application configuration file. For non-web applications, you can enable network logging programmatically using the IDK.
ptedk.VerboseLogging	**false** (Does not log method parameters or return values unless requested.)	Verbose logging is disabled by default. Basic logging messages are still sent to the log receiver. The Portlet API sends an Info log message with each new portlet context created (each portlet request). Any exceptions, errors, or requests for missing settings are logged as Error or Warning as appropriate. If you enable verbose logging, additional messages and details are sent to the log receiver. The Portlet API sends a Warning message informing the log reader that sensitive information may be logged in cleartext. With each portlet request, the Portlet API sends a Debug message with a toString() of the PortletRequest object, containing request parameters and portlet settings; and a Debug message with a toString() of the PortletUser object, containing user settings.

About HTTP and CSP

HTTP is a protocol used mostly for transferring web page content and XML between a server and a client. CSP is a platform-independent protocol based on the open standard of HTTP 1.1 that defines the syntax of communication between the portal and remote servers.

HTTP communication is made up of Requests and Responses. Requests and Responses are essentially lists of name-value pairs of metadata in headers, along with an optional body. The body is the data that is being transferred (an HTML page or XML file). The metadata in the headers is information about the Request or Response itself (what language the content is in, or how long the browser should cache it). The Request and Response each contain specific information, outlined next. For more detailed information on HTTP, see RFC 2616 (*http://www.faqs.org/rfcs/rfc2616.html*).

The client sends the server an **HTTP Request**, asking for content. The Request body is used only for requests that transfer data to the server, such as POST and PUT.

HTTP Request Format:

```
[METHOD] [REQUEST-URI] HTTP/[VERSION]
[fieldname1]: [field-value1]
[fieldname2]: [field-value2]
[request body, if any]
```

HTTP Request Example:

```
GET /index.html HTTP/1.1
Host: www.plumtree.com
User-Agent: Mozilla/3.0 (compatible; Opera/3.0; Windows 95/NT4)
Accept: */*
Cookie: username=JoeSmith
```

The server sends back an **HTTP Response** that contains page content and important details, such as the content type, when the document was last modified, and the server type. The Response contains an error message if the requested content is not found.

HTTP Response Format:

```
HTTP/[VERSION] [CODE] [TEXT]
[fieldname1]: [field-value1]
[fieldname2]: [field-value2]
[response body, if any (document content here)]
```

HTTP Response Example:

```
HTTP/1.0 200 Found
Last-modified: Thursday, 20-Nov-97 10:44:53
Content-length: 6372
Content-type: text/html
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 3.2 Final// EN'><HTML>
...followed by document content...
```

Custom HTTP headers can be configured to include specialized information.

Note: Header size limits are controlled by the server that hosts the code. The standard limit for IIS/ASP is 60K. Java Application Servers range from 2K to 10K. These limits are generally configurable; see your server documentation for details.

Services can also access standard HTTP headers, such as the Set-Cookie header or HTTP 1.1 basic authentication header. If you want to investigate HTTP further, you can view all the headers being passed back and forth between your browser and web server using a tunnel tool. HTTP is used in conjunction with SSL to serve up secure content. Single Sign-On (SSO) also uses HTTP headers for basic authentication.

CSP

CSP extends HTTP and defines proprietary headers to pass settings between the portal and remote server. CSP outlines how ALI services use HTTP to communicate and modify settings.

The current version is CSP 1.4, which includes backward compatibility with previous versions. CSP 1.2 is used in portal version 5.x. CSP 1.1 is used in portal version 4.5. Versions 4.x and below use CSP 1.0. For links to the latest versions of the CSP specification, see *Additional Development References* on page 523.

The IDK provides simplified, stable interfaces that allow you to write code that communicates using CSP.

ALI Headers

AquaLogic Interaction (ALI) portal uses a group of custom headers to communicate system and user configuration variables. These headers include information that can be used by services.

All the useful information stored in these headers should be accessed using the AquaLogic Interaction Development Kit (IDK). Additional proprietary headers contain the protocol version, gateway type, and aggregation mode. All the key information in these headers is accessible through the `IPortletUser` and `IPortletRequest` interfaces in the IDK.

Header Name	IDK Method	Description
User ID	`IPortletUser.GetUserID`	The User ID of the currently logged in user. This value can be used to determine if the session has expired. If UserID=2, the default 'Guest' user is logged in; any other user's session has ended.
User Name	`IPortletUser.GetUserName`	The name of the logged in user. The user's name can be used to personalize display or pre-fill form fields.
Locale	`IPortletUser.GetUserCharacterSet`	The current user's language and character set. This value is essential when determining the correct content to return in an internationalized implementation of the portal.
Time Zone	`IPortletRequest.GetTimeZone`	The time zone of the current user in the format used by the portal. This value can be used to synchronize remote server time with the portal.
Image Service URL	`IPortletRequest.GetImageServerURI`	The URL to the root virtual directory of the Image Service in the user's implementation of the portal. This location should be used for all static images used in services.
Stylesheet URL	`IPortletRequest.GetStylesheetURI`	The URL to the current user's style sheet. In each implementation of the portal, the UI is customized. In some portals, users can choose between a selection of stylesheets. Using

Header Name	IDK Method	Description
		ALI-defined styles ensures that portlets appear in the style of the current user's portal.
Community ID	`IPortletRequest.GetCommunityID`	The Community ID for the Community in which the portlet appears. This value is used internally to identify the correct group of settings. (If CommunityID=0, the portlet appears on a My Page.)
Page ID	`IPortletRequest.GetPageID`	The Page ID for the current portal page. This value allows a single Portlet object to display different content on different portal pages.
Portlet ID	`IPortletRequest.GetPortletID`	The Portlet ID for the current portlet. This value is useful for appending to the names of HTML forms and client-side JavaScript functions to ensure unique form and function names on the My Page to avoid name conflicts.
Return URL	`IPortletRequest.GetReturnURI`	The URL to the page that the portlet should return to when finished, usually the page that hosts the portlet. Preference pages need this URL to return the user to the correct page after settings are configured.
Content Mode	`IPortletRequest.GetPortletMode`	The current content mode. This value is used to display portlet content in the appropriate manner.
Browser Type	`IPortletRequest.GetUserInterface`	The type of device being used to access the portal. The portal can

Header Name	IDK Method	Description
		support wireless handheld devices that communicate with HDML, WML, or HTML.

About SOAP

SOAP is a text-based protocol to wrap XML data for any type of transport, providing an efficient way to communicate structured data.

The SOAP 1.1 specification describes SOAP as follows: “SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.”

SOAP is based on Web standards. Like HTML, SOAP uses tags to indicate the role of each piece of information. In most implementations, SOAP uses HTTP for its transport protocol. A SOAP request is an XML document that describes a method to invoke on a remote machine and any parameters to be used. A program sends a SOAP request to a SOAP server. The SOAP server tries to execute the method with the parameters it was passed, and it sends back a SOAP response (the result or an error message). A SOAP endpoint is an HTTP-based URL identifying a target for method invocation.

A common analogy illustrates this concept well. If your XML code was a letter, SOAP would be the envelope; like an envelope, SOAP protects content from unauthorized access and provides information about the sender and the addressee. All the elements of the SOAP envelope are defined by a schema. The schema URI is also the identifier for the SOAP envelope namespace: *http://schema.xmlsoap.org/soap/envelope*.

As in standard XML, SOAP uses namespaces to segregate content. The formal designation of a namespace is a URI, usually a URL. Namespaces ensure unique element references, and they allow a processor to pick out which instructions it should obey and treat instructions for other processors as simple data. Processors are set up to handle elements from a particular namespace. Elements that have no namespace are treated as data.

SOAP Message in HTTP Request:

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset='utf-8'
Content-Length: nnnn
SOAPAction: 'Some-URI'

<SOAP-ENV:Envelope
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
SOAP-ENV:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>

<SOAP-ENV:Body>
<m:GetLastTradePrice xmlns:m='Some-URI'>
<symbol>DIS</symbol>
</m:GetLastTradePrice>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Message in HTTP Response:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset='utf-8'
Content-Length: nnnn

<SOAP-ENV:Envelope
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
SOAP-ENV:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>

<SOAP-ENV:Body>
<m:GetLastTradePriceResponse xmlns:m='Some-URI'>
<Price>34.5</Price>
</m:GetLastTradePriceResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

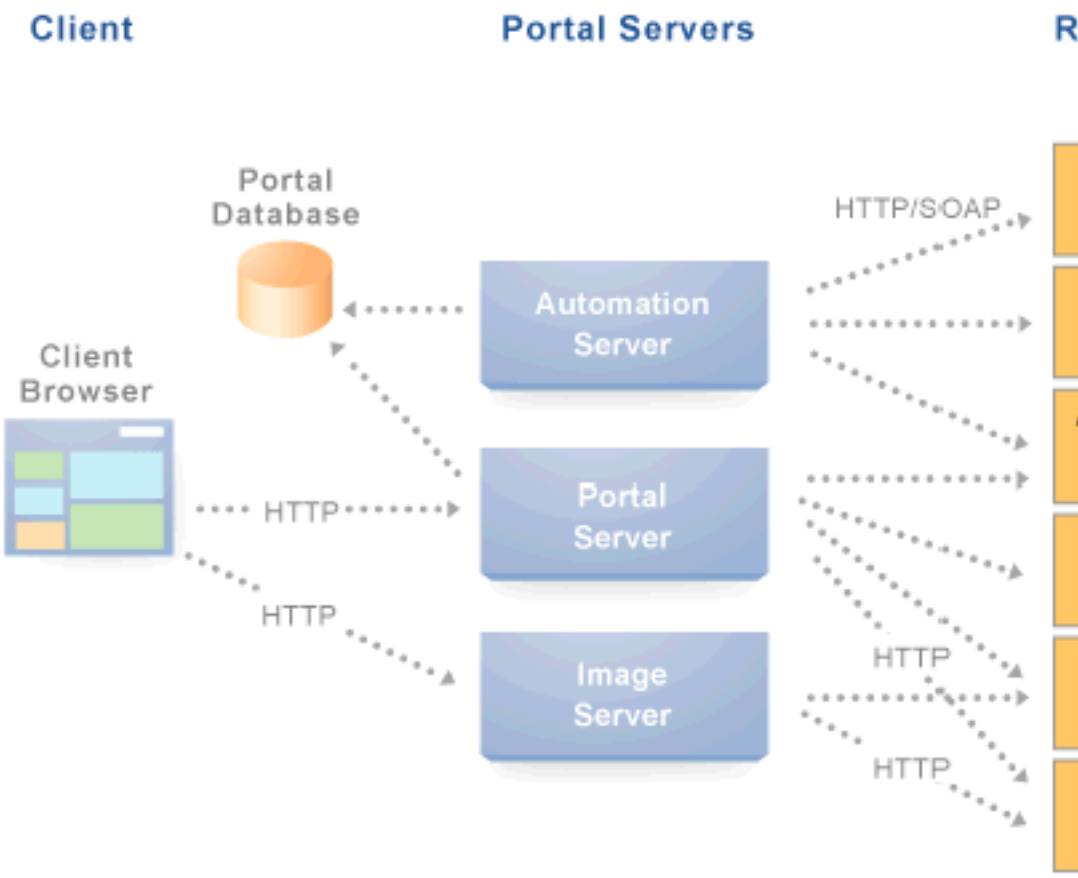
Download the complete SOAP 1.1 specification from the World Wide Web Consortium at *http://www.w3c.org/TR/SOAP/*.

ALI's SOAP API exposes commonly used elements of the traditional portal API, focused on the functions required to develop applications that access portal users, Communities, Portlets, and Knowledge Directory functions. The IDK's PRC API provides an efficient, object-oriented way to call into the portal's SOAP API. For details, see *About Programmable Remote Client (PRC) Remote APIs* on page 54.

About Server Communication and the Gateway

ALI and Ensemble both act as a gateway server, brokering transactions between client computers and remote servers.

Services on remote servers communicate with the portal via HTTP and SOAP as shown in the simplified diagram below. For example, when a browser requests a My Page from the portal, the portal makes simultaneous requests to each remote server to retrieve the portlet content for the page. The remote server reads the current user's preferences from the HTTP headers sent by the portal and sends back the appropriate HTML. The portal inserts the HTML into the table that makes up the My Page. Any images stored in the Image Service are retrieved and displayed by the browser.



HTTP and SOAP are both necessary because each standard fits the specific needs of different tasks. SOAP involves posting and returning XML documents and is appropriate for exchanging highly structured data. SOAP is used in the server-to-server communication required for content services, identity services, and importing documents. HTTP is a much more lightweight protocol, used in the portal for UI presentation, basic configuration and click-through, and caching. For an introduction to SOAP, see *About SOAP* on page 38.

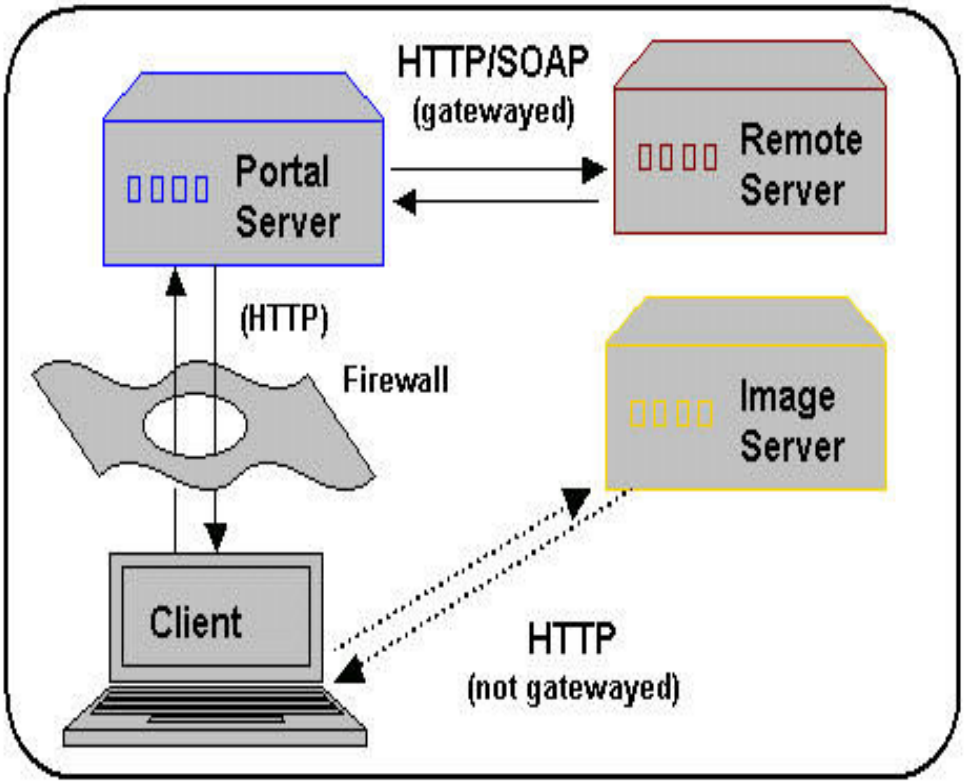
CSP is a platform-independent protocol based on the open standard of HTTP 1.1. The syntax of communication between the portal and remote servers is defined by CSP. CSP defines custom headers and outlines how ALI services use HTTP to communicate and modify settings. For details on CSP, see *About HTTP and CSP* on page 34.

The Gateway

A 'gateway server' acts as a middleman, brokering transactions between a client computer and another server. This configuration is typically used to serve content to clients that would otherwise be unable to access the remote server, but it can be used to impose additional security restrictions on the client. The gateway hides the remote server; to the end user, the content appears to come directly from the gateway server.

This architecture makes the portal the single point of access for content, and allows remote servers to reside on a private network or behind a firewall. As long as the portal can connect to the remote server, users can view the content, even if they cannot access it directly. To the browser, the portal appears to be the source of content on the remote server.

When a user interacts with a ALI service, any request made to a URL in the gateway is automatically rerouted through the portal. To the user, the content appears to come from the portal; the remote server is an unknown back-end system.



There are many benefits to this configuration. The most useful to ALI services are:

- **Dynamic functionality and personalization**: The portal intercepts requests from portlets, which allows it to include information stored in the ALI database in HTTP requests and responses. Most of this information is accessible through IDK methods. In many situations, an adaptive tag provides the functionality you need, including navigation and login elements. Custom tags can be created for additional functionality.
- **Security**: Services can allow users to access content that is not publicly available. Files stored on a secure server can be made available by including specific URLs in the configuration of the gateway. NOTE: The gateway is a powerful feature, and can compromise security if incorrectly configured. Allowing direct access to a remote server that hosts unprotected private content could create a dangerous security hole.
- **Performance**: The portal caches gatewayed content, decreasing response time for end users and improving performance on the remote server. While gatewaying works efficiently for

content like HTML, it is generally not appropriate for binary data like static images. Images do not need to be transformed, and gatewaying large images can adversely affect the performance of the portal. This is one reason the Image Service should be used to prevent routing static images through the gateway.

The collection of URLs that should be gatewayed for a service is configured in the Web Service editor on the HTTP Configuration page. In the Gateway URL Prefixes list, you must enter the base URLs for any directories that should be gatewayed.

Keep the following warnings and best practices in mind when implementing services that use the gateway:

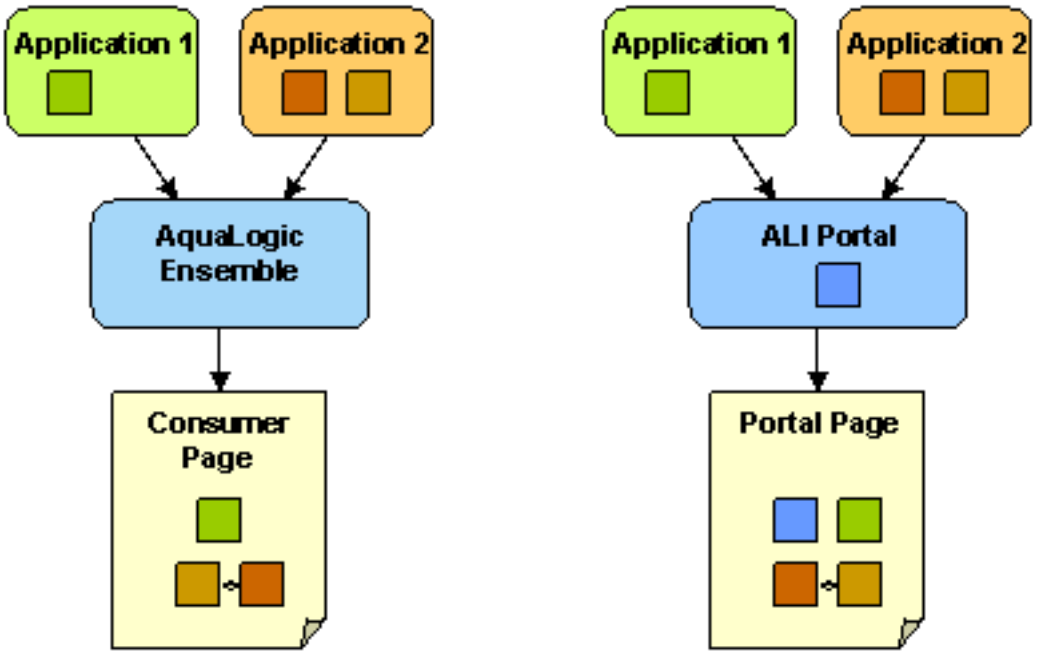
- **URL transformation**: The portal must transform code so that gatewayed URLs open correctly. Before the portal sends a response, it parses the HTML and looks for any URLs included in the Gateway URL Prefixes list for the associated Web Service object. The portal transforms any URLs that should be gatewayed before returning the response to the client. Relative URLs are transformed to point to the correct location.
- **Scripting limitations**: JavaScript constructs that dynamically create URLs can cause problems, because they are run after content is already transformed. VBScript is not transformed by the gateway; you can continue to use dynamic scripts and VBScript as long as your code is gateway-aware. To manually mark a URL for transformation, use the pt:url tag. To disable transformation, use pt:transformer with a pt:fixurl attribute of 'off.' For details, see *Transforming URLs Using Adaptive Tags* on page 211.
- **URL encoding**: It is a best practice to encode all headers that are URLs to prevent unexpected transformation. In JSP, encode all URLs that are written. If the code writes URLs in the body of a page (for example, a link to a preferences page) it should be encoded. The standard Java servlet command response.encodeURL() is the preferred method, but you can also use URLEncoder.encode(url). In the .NET Framework, the HttpUtility.URLEncode class provides the necessary functionality. Note: In .NET, there is no need to encode the redirect URL; this is handled automatically on the back end.

For details on pagelets and the gateway, see *About Pagelets and the Gateway* on page 281.

About Pagelets and the Gateway

All pagelets are designed to be displayed with other pagelets. ALI and Ensemble both act as a gateway, processing and combining pagelets from multiple applications to create a single, unified page with a range of functionality.

The code returned by a pagelet is parsed by the gateway server and inserted into the appropriate cell in the HTML table that makes up the mashup page. Pagelets from the same back-end application can interact with each other within the page.



The same pagelets can be used in both Ensemble and ALI portal, but a different process is used to implement the mashup page.

- Modeled after the “window” metaphor from desktop user interfaces, the **ALI portal** displays pagelets as a series of windows or boxes arranged in columns on a page with borders, title bars, buttons, headers and footers rendered by the portal framework. The page layout is defined through the portal's administrative UI. Each My Page or community page is made up of many pagelets, selected and arranged using portal editors. Portlets are pagelets specifically designed for use in the AquaLogic Interaction portal.
- In **Ensemble**, a consumer page defines the layout and includes specific pagelets in the page using adaptive tags. The portal header is not included by default, but portal header navigation can be added using tags.

For an introduction to the gateway, see *About Server Communication and the Gateway* on page 40.

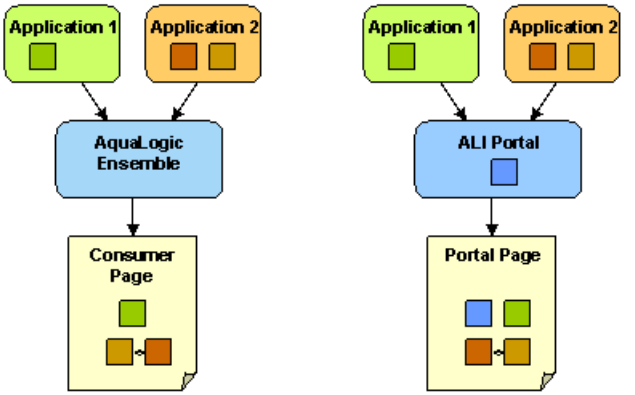
About Pagelets and Portlets

A pagelet is a web application that produces a self-contained, reusable user interface widget. A portlet is a pagelet designed for use with AquaLogic Interaction portal. Pagelets can be used for everything from displaying useful information to building integrated applications that combine functionality from multiple systems.

Every pagelet executes its functionality in a separate process. Most pagelets connect to a back-end application for data or functionality. The back-end for a pagelet can be any web application that returns HTML or XML over HTTP. The example below is a AquaLogic Interaction Collaboration pagelet that searches for projects based on a user's selections.

All pagelets are designed to be displayed with other pagelets. AquaLogic Interaction (ALI) portal and AquaLogic Ensemble both act as a gateway, processing and combining pagelets from multiple applications to create a single, unified page with a range of functionality. The code returned by a pagelet is parsed by the portal or proxy server and inserted into the appropriate location in the mashup page. Pagelets

from the same back-end application can interact with each other within the



page.

The AquaLogic Interaction Development Kit (IDK) allows pagelets to access useful information and integrate dynamic functionality from ALUI components.

For more information on the gateway, see *About Server Communication and the Gateway* on page 40. For details on pagelet development, see the following topics:

- *About Cross-Platform Pagelet Development* on page 48
- *About ALI Portlet Development* on page 283
- *About Ensemble Pagelet Development* on page 327

About Cross-Platform Pagelet Development

ALI and Ensemble both support AquaLogic development tools, including the AquaLogic Development Kit (IDK), Adaptive Tags, and the ALI Scripting Framework. These tools provide cross-product APIs that allow you to write a single pagelet for both ALI portal and Ensemble, as well as product-specific APIs to implement advanced functionality.

- *About the IDK Proxy API* on page 328: The bea.alui.proxy package supports cross-product pagelet development. The interfaces in this package provide access to information about the environment in which the pagelet is displayed and the user currently accessing the pagelet, including session preferences associated with that user. This package also includes Ensemble-specific methods to implement security and access XML payloads. For details on creating pagelets with the IDK Proxy API, see *Creating a Custom Pagelet with the Java IDK*

Proxy API on page 329 and *Creating a Custom Pagelet with the .NET IDK Proxy API* on page 330.

- *About Programmable Remote Client (PRC) Remote APIs* on page 54: The plumtree.remote.prc package includes a collection of APIs that provide access to functionality within the ALI portal, Collaboration, Publisher, and Search Service. These APIs are supported by Ensemble and ALI portal, and can be used by any pagelet deployed in an environment with access to these applications.
- *About Adaptive Tags* on page 202: Adaptive Tags are used to display contextual data and control Ensemble and ALI portal functionality from remote pagelets. Unlike the IDK, Adaptive Tags use XML in pagelet content instead of code, which avoids a network round trip. Tags can be included in the markup returned by any proxied page (HTML, JSP or ASP.Net). Using the attributes defined in the tag, the Ensemble or ALI portal gateway transforms the XML and replaces it with standard HTML and/or executes the relevant operations. The Adaptive Tag collection currently includes libraries for use in both ALI portal and Ensemble, as well as libraries that are specific to each environment.
- *About the ALI Scripting Framework* on page 260: The ALI Scripting Framework is a set of client-side JavaScript libraries that provide services to pagelets and proxied/gatewayed pages.
- *About Pagelet Caching* on page 318: Caching is the functionality that allows ALI and Ensemble to request pagelet content, save the content, and return the saved content to users when appropriate. The importance of caching cannot be overstated.
- *About Pagelet Internationalization* on page 317: These tips and best practices apply to all pagelets that will be translated into multiple languages.

About the IDK Proxy API

The IDK Proxy API supports cross-product pagelet development. The interfaces in this package provide access to information about the environment in which the pagelet is displayed and the user currently accessing the pagelet, including session preferences associated with that user. This package also includes Ensemble-specific methods to implement security and access XML payloads.

This page provides an introduction to the IDK Proxy API. For more details on objects and methods, see the IDK API documentation.

The bea.alui.proxy package/namespace includes the following interfaces:

- `IProxyContext`
- `IProxyRequest`
- `IProxyResponse`
- `IProxyUser`

In general, these interfaces are called in the following order:

1. A pagelet uses `ProxyContextFactory.getInstance().createProxyContext` to initiate a connection for communicating with Ensemble or the portal.
2. The `IProxyContext` object returned allows the pagelet to access information about the request and response, the current user, and the session. The pagelet uses this information as needed, in arbitrary order, to generate a proper response. Using `IProxyContext`, the pagelet can access `IProxyRequest`, `IProxyUser`, `IRemoteSession` and `IProxyResponse`.
3. The pagelet retrieves parameters from the request using `IProxyRequest`.
4. The pagelet retrieves user information and preferences from `IProxyUser`.
5. The pagelet can access functionality in AquaLogic applications using `IRemoteSession`. For details, see *About Programmable Remote Client (PRC) Remote APIs* on page 54.
6. The pagelet constructs a response using `IProxyResponse`. The response includes content to be displayed and any settings to be stored or removed.

For examples of using IProxy interfaces in a pagelet, see *Creating a Custom Pagelet with the Java IDK Proxy API* on page 329 and *Creating a Custom Pagelet with the .NET IDK Proxy API* on page 330.

Creating a Custom Pagelet with the Java IDK Proxy API

This example creates a simple pagelet that displays information from the proxy, including setting values.

1. Before writing any code, create a new IDK project as described in *Setting Up a Custom Java IDK Project in Eclipse Stand-Alone (without WTP)* on page 12 or *Setting Up a Custom Java IDK Project in Eclipse with WTP* on page 10.
2. In the new project, create a new JSP page for the pagelet (pagelet.jsp).
3. Implement your code. The pagelet code shown below instantiates the IDK and uses the `IProxyContext` interface to retrieve `IProxyRequest` and `IProxyUser` objects to access information about the user and the settings associated with the pagelet.

   Note: There is no need to include html, head and body tags; the display is handled by the Consumer resource.

```
<%@ page language='java' import='com.bea.alui.proxy.*' %>
<%
String Att1 = 'no setting';
String Att2 = 'no setting';
String sessionVariable = 'no setting';

//get the idk
```

```
IProxyContext proxyContext =
ProxyContextFactory.getInstance().createProxyContext(request,
response);
IProxyRequest proxyRequest = proxyContext.getProxyRequest()

IProxyUser proxyUser = proxyRequest.getUser();
String userName = proxyUser.getUserName();
int userID = proxyUser.getUserID();

Att1 = proxyRequest.getSetting('Att1')
Att2 = proxyRequest.getSetting('Att2');
sessionVariable = proxyRequest.getSetting('sessionVar');

byte[] payload = proxyRequest.getPayload().getText();
String payloadStr = new String(payload)
%>

<p>User name: <%=userName%><br/>
User ID: <%=userID%><br/>
Attribute 1: <%=Att1%><br/>
Attribute 2: <%=Att2%><br/>
Session variable: <%=sessionVariable%><br/>
Payload: <textarea name=xml cols=80 rows=6> <%=payloadStr%>
</textarea>
</p>
```

Creating a Custom Pagelet with the .NET IDK Proxy API

This example creates a simple pagelet that displays information from the proxy, including setting values. .NET pagelets use a code-behind page (.aspx.cs) to retrieve settings and a web form (.aspx) to display the pagelet content.

1. Before writing any code, create a new IDK project as described in *Setting Up a Custom .NET IDK Project* on page 14.
2. In the new project, implement your code. The example below uses a code-behind page and a web form.

The code-behind page (IDKPagelet.aspx.cs) instantiates the IDK and uses the `IProxyContext` interface to retrieve `IProxyRequest` and `IProxyUser` objects to access information about the user and the settings associated with the pagelet.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Plumtree.Remote.Portlet;
using System.Xml;
using System.Text;
using Bea.Alui.Proxy;

namespace IDKProxyWS
{
/// <summary>
/// Hello World Pagelet
/// </summary>
   public class IDKPagelet : System.Web.UI.Page
   {
      public String name;
      public bool isGuest;
      public int userID;
      public String envType;
      public String payload;
      public String Att1,Att2;
      public String SessionVar;
      private void Page_Load(object sender, System.EventArgs e
      {
         // Put user code to initialize the page here
         InitializeCSP();
      }
      private void InitializeCSP()
      {
         IProxyRequest proxyRequest;
         IProxyResponse proxyResponse;
         IProxyUser proxyUser;
         IProxyContext proxyContext;
         ProxyContextFactory factory;
         HttpRequest request = HttpContext.Current.Request;
         HttpResponse response = HttpContext.Current.Response;

         try
         {
            factory = ProxyContextFactory.getInstance();
            proxyContext = factory.CreateProxyContext(request,
response);
```

```
            proxyRequest = proxyContext.GetProxyRequest();
            proxyResponse = proxyContext.GetProxyResponse();
            envType =
proxyRequest.GetEnvironment().GetType().ToString();
            proxyUser = proxyRequest.GetUser();
            isGuest = proxyUser.IsAnonymous();
            name= proxyUser.GetUserName();
            userID = proxyUser.GetUserID();

            Att1 = (String)proxyRequest.GetSetting('attr1');
            Att2 = (String)proxyRequest.GetSetting('attr2');
           Att2 = (String)proxyRequest.GetSetting('SessionVar');


           byte[] bpayload = proxyRequest.GetPayload().GetText()

            System.Text.ASCIIEncoding enc = new
System.Text.ASCIIEncoding()
            payload = enc.GetString(bpayload)
         }
         catch(Bea.Alui.Proxy.NotGatewayedException e)
         {
         }
      }
   }
#region Web Form Designer generated code
...
#endregion
}
```

The web form that displays the pagelet (IDKPagelet.aspx) displays the information retrieved by the code-behind page above.

```
<%@ Page Language='c#' runat='server'
CodeBehind='IDKPagelet.aspx.cs' AutoEventWireup='false'
inherits='IDKProxyWS.IDKPagelet' %>
<%@ import Namespace='System.Collections' %>
<%@ import Namespace='System.Web' %>
<%@ import Namespace='System.Web.UI' %>

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN' >
<html>
<head>
<title>IDKPagelet</title>
<meta name='GENERATOR' Content='Microsoft Visual Studio .NET 7.1'>
<meta name='CODE_LANGUAGE' Content='C#'>
```

```
<meta name='vs_defaultClientScript' content='JavaScript'>
<meta name='vs_targetSchema'
content='http://schemas.microsoft.com/intellisense/ie5'>
</head>

<body MS_POSITIONING='GridLayout'>
<span xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>

Proxy Pagelet <BR>
<%
   Response.Write('IDK Proxy Pagelet<BR>');
   Response.Write('Environment Type ' + envType + '<BR>');
   Response.Write('Guest User? ' + isGuest + '<BR>');
   Response.Write('User Name: ' + name + '<BR>');
   Response.Write('User ID: ' + userID + '<BR>');
   Response.Write('<P>');

   Response.Write('Pagelet Attributes:<BR>');
   Response.Write('Attribute1: ' + Att1 + '<BR>');
   Response.Write('Attribute2: ' + Att2 + '<BR>')
   Response.Write('SessionVar: ' + SessionVar + '<BR>')
   Response.Write('<P>')

   Response.Write('Pagelet XML Payload:<BR>');
   Response.Write('<textarea name=xml cols=80 rows=6>' + payload
 + '</textarea>');
   Response.Write('<P>');
%>
</span>
</body>
</html>
```

About Programmable Remote Client (PRC) Remote APIs

The Programmable Remote Client (PRC) provides an object-oriented way to call into AquaLogic SOAP APIs. The PRC can be used to write applications that access the ALI portal and search, AL Collaboration, and AL Publisher.

The PRC APIs free you from serializing SOAP messages and minimize the amount of data that travels between the portal and the remote server, improving performance.

The PRC is included with both the Java and .NET versions of the AquaLogic User Interaction Development Kit (IDK). The Java version includes Apache AXIS 1.0; the .NET version uses the platform-native SOAP client stack. Java clients can call .NET portals and vice-versa; the PRC

takes care of the communication between AXIS and .NET. Pagelets that use the PRC can be deployed in either ALI or Ensemble.

The PRC provides access to functionality in a range of AquaLogic products:

- *About Remote ALI APIs* on page 58
- *About Remote Collaboration APIs* on page 103
- *About Remote Publisher APIs* on page 162

About the PRC Session Object

When using IDK remote APIs. the Session object is the master object; most other portal objects must be derived from it.

A Session object is created whenever any user logs in to the ALI system through the web or a client application. All subsequent access is made in the security context of the connected user. Users in the Administrators group have superuser access. Users in Content Manager and Content Maintainer groups also have privileged access. For details on portal groups and specific privileges, see the *Administrator Guide for AquaLogic Interaction*.

The Session object supports the `IPTSession` interface, represented in the PRC by the `IRemoteSession` interface. The Session object is comprised of:

- A set of Object Managers for the objects of the ALI system. An Object Manager is like a super collection. It provides advanced querying capabilities as well as create, delete, and clone methods. The Session object includes managers for almost every ALI portal, ALI Search, AL Collaboration and AL Publisher object. Managers are added as new object classes are introduced.
- A User object representing the current user of the system.
- A Catalog object representing the structure of the ALI catalog.
- Version information.
- Access to global ALI objects such as the MyPortal object, the scheduler, and global mapping objects.

Initiating a session is the first step in any implementation of the PRC. For detailed instructions, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.

Initiating a PRC Session to Use IDK Remote APIs

To use the IDK's Programmable Remote Client (PRC) Remote APIs, you must first establish a session with ALI or Ensemble.

The session is used to manipulate objects via the PRC. Once you have initiated a session, you can use PRC methods to manipulate AquaLogic objects.

Note: Before writing any code, you must prepare a custom project that references the standard IDK library (idk.jar/idk.dll).

To establish a session with the portal, acquire a reference to an `IRemoteSession` object. The simple code examples below do the following:

1. Create a new class (HelloWorldSession).
2. Create a new remote session using `RemoteSessionFactory`. The code below logs in with a user name of "administrator" and no password. You can also access an `IRemoteSession` through the IDK portlet and proxy APIs (`IPortletContext.GetRemoteSession` or `IProxyContext.GetRemoteSession`).

   Note: You must configure ALI or Ensemble to send a login token to any pagelet that uses the PRC. In ALI, the login token option is on the Advanced Settings page of the Web Service editor. In Ensemble, this option is on the CSP tab in Resource configuration.

3. Print out the portal API version from the remote session.

Java:

```
import java.net.URL;
import com.plumtree.remote.prc.*;
public class HelloWorldSession
{
public static void main(String[] args) throws Exception
{
try
{
IRemoteSession session =
RemoteSessionFactory.getExplicitLoginContext(
new URL("http://portalserver/ptapi/services/QueryInterfaceAPI"),
 "administrator","");

 System.out.println(session.getAPIVersion());
 }
 catch(Exception e)
 {
 System.err.println(e.getMessage());
 e.printStackTrace(System.err);
 }
 }
}
```

.NET (C#):

```
using System;
using Plumtree.Remote.PRC;
public class HelloWorldSession
{
 public static void Main(string[] args)
 {
 try
 {
 IRemoteSession session =
RemoteSessionFactory.GetExplicitLoginContext(
 new Uri("http://portalserver/ptapi/services/QueryInterfaceAPI"),

 "administrator","");

 Console.Out.WriteLine(session.GetAPIVersion());
 }
 catch(Exception e)
 {
 Console.Error.WriteLine(e.Message);
 Console.Error.WriteLine(e.StackTrace);
 }
 }
}
```

.NET (VB):

```
Imports System
Imports Plumtree.Remote.PRC
Module HelloWorldSession

 Sub Main()
 Try
 Dim session As IRemoteSession
 session = RemoteSessionFactory.GetExplicitLoginContext( _
 New Uri("http://portalserver/ptapi/services/QueryInterfaceAPI"),
 _
 "administrator", _"")
 Console.Out.WriteLine(session.GetAPIVersion())
 Catch e As Exception
 Console.Error.WriteLine(e.Message)
 Console.Error.WriteLine(e.StackTrace)
 End Try
 End Sub
```

```
End Module
```

IDK PRC Remote API Development Tips

These development tips apply to any application that uses the PRC.

- **You must configure ALI or Ensemble to send a login token to any pagelet that uses the PRC.** In ALI, the login token option is on the Advanced Settings page of the Web Service editor. In Ensemble, this option is on the CSP tab in Resource configuration.
- **Perform expensive processing outside the interface method, in a separate thread, or use back-end caching such that the interface method can respond in a timely fashion.** For example, an Active Directory Authentication Source Identity Service might employ user signatures to minimize reads/writes from the AD database during remote calls like `IProfileProvider.attachToUser`. The IDK PRC manager interfaces generally make remote calls. PRC object interfaces are normally local accessors/mutators and do not make remote calls (with the exception of store methods). Avoid unnecessary, repeated use of manager interface methods and maximize your application's use of PRC object methods. Avoid looping remote calls wherever possible. Maintaining local copies of PRC objects can improve your application's performance but be aware that your local state may not match the server state if another application modifies server state after you receive your local copy. For example, a portlet using PRC Collaboration to display the current user's personal Collaboration project area corresponding to "Username-Project" ensures that `IProjectManager.queryProjects` is used once. The resulting `IProject` object can be cached by the portlet per user session rather than performing a query on every portlet refresh. The user's project is never deleted, so the local caching is "safe."

About Remote ALI APIs

The portal provides the framework for applications and integrates AquaLogic Interaction components into a cohesive Web work environment. Administration is the core of the portal, where all portal objects and operations are configured.

The PRC's remote APIs provide access to key administrative components, as explained in the following topics:

- *About Remote ALI Object Management* on page 59: Everything in the portal, except users and documents, is represented by a portal object stored in the ALI database. This includes portlets, content crawlers, authentication sources, profile sources, remote servers, and content sources. The `IObjectManager` interface allows you to access portal objects from your remote services. You can look up information about a specific object, or query for objects using a

range of methods, including location, class, and custom filters. You can also query and manipulate the security for portal objects.

- *About Remote Portlet Operations* on page 74: There are many settings and options that apply only to portlets. In addition to manipulating portlet objects via `IObjectManager`, the PRC supports advanced portlet operations. Using the `IPortlet*` interfaces, you can create and edit portlets and portlet templates, and manage administrative and communityportlet preferences for a specific portlet instance.
- *About Remote Knowledge Directory Operations* on page 79: The Knowledge Directory stores links to documents in a hierarchical structure of folders and subfolders. These documents can be external or internal web pages, Office documents, or essentially any file of interest. The `IDocument*` interfaces allow you to query for documents and document properties, create new documents, and edit the properties for existing documents.
- *About Remote User Operations* on page 86: Portal users are organized into groups and sub-groups. This role-based hierarchy allows administrators to customize the portal display for specific audiences and assign object security for collections of users. The `IUserManager` and `IUserGroupManager` interfaces allow you to leverage the portal's user hierarchy. You can query for the current user's ID and group information, create new groups, and manage group membership.
- *About Remote Search Operations* on page 90: Using the PRC search API, you can query document, folder, user and Community objects using a standard request-response model. The API allows you to add multiple constraints and filter searches by location or object type.
- *Starting Portal Jobs Using IDK Remote APIs* on page 101: A job is a collection of related portal operations. Each operation is one task, such as a crawl for documents, an import of users, or one of the system maintenance tasks. To start an existing ALI job from a remote application, use the `IJobManager` interface.

About Remote ALI Object Management

Everything in the portal, except users and documents, is represented by a portal object stored in the ALI database. This includes portlets, content crawlers, authentication sources, profile sources, remote servers, and content sources. The PRC `IObjectManager` interface in the IDK allows applications to access portal objects from remote services.

Using the PRC, you can look up information about a specific object, or query for objects using a range of methods, including location, class, and custom filters. You can also query and manipulate the security for portal objects.

For details on using remote ALI object management, see the following topics:

- *Retrieving ALI Object Managers Using IDK Remote APIs* on page 60
- *Querying ALI Objects Using IDK Remote APIs* on page 60

- *Querying ALI Object Properties Using IDK Remote APIs* on page 67
- *Managing Object Security (ACLs) Using IDK Remote APIs* on page 71
- *ALI Object Type Class IDs and Modes* on page 65

Retrieving ALI Object Managers Using IDK Remote APIs

To access ALI objects from a remote application, first retrieve an `IObjectManager` object from the `IRemoteSession` object.

To retrieve an `IObjectManager` object from the `IRemoteSession` object, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an instance of `IObjectManager` from the `IRemoteSession` object, as shown in the sample code below. This example demonstrates how to retrieve an Object Manager to query communities, using an existing `IRemoteSession` instance. For a list of object types, see *ALI Object Type Class IDs and Modes* on page 65.

Java:

```
IObjectManager objectManager =
session.getObjectManager(ObjectClass.Community);
```

.NET (C#):

```
IObjectManager objectManager =
session.GetObjectManager(ObjectClass.Community);
```

.NET (VB):

```
Dim objectManager as IObjectManager
objectManager= session.GetObjectManager(ObjectClass.Community)
```

Querying ALI Objects Using IDK Remote APIs

To query for ALI objects from a remote application, use the `IObjectManager` interface in the PRC.

The `IObjectManager` interface in the PRC allows you to query for objects using a range of methods, including location, class, and custom filters. To query for portal objects, follow the steps below. You can also use the PRC Search API to query for portal objects; for details, see *Querying ALI Objects Using the IDK Remote Search API* on page 90.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.

2. Retrieve an Object Manager for the type of object you are querying. For details, see *Retrieving ALI Object Managers Using IDK Remote APIs* on page 60.
3. Execute the query, as shown in the sample code below. This example demonstrates how to query for portal users matching specific criteria within the portal, using the following process:
   a) Declare and prepare all parameters to be passed to the query method. This example uses the most flexible query call, with the following parameters. If any of these parameters is omitted, the default value will be used. (There are simpler calls with fewer parameters; for details, see the IDK API documentation.)

Parameter	Description	Default
folderID	The folder to search.	all folders (folderID = -1)
startRow	The row on which to start the search.	the initial row (startRow = 0)
maxRows	The maximum number of rows to search.	unlimited (maxRow = -1)
sortProperty	The object property on which to sort results.	object ID (sortProperty = ObjectProperty.ObjectID)
ascending	The sort order for results (ascending or descending).	ascending (ascending = true)
propsToReturn	The properties to return.	all properties
filters	The values on which to filter results.	no filters

   b) Execute the query to retrieve an `IObjectQuery` instance.
   c) Enumerate through the query, displaying interesting information.

Java:

```
public static voidqueryObjects(String loginToken)
{
 IObjectManager objectManager =
getSession(loginToken).getObjectManager(ObjectClass.User);

 int folderID = -1;  //search all folders
 int startRow = 0;  //start at the first found
 int maxRows = -1;  //return unlimited results
 ObjectProperty sortProperty = UserProperty.UniqueName;  //sort
on the unique name
 boolean ascending = true;  //sort ascending
```

```
 ObjectProperty[] propsToReturn = new ObjectProperty[4];  //return
 specific properties
 propsToReturn[0] = UserProperty.SimpleName;
 propsToReturn[1] = UserProperty.UniqueName;
 propsToReturn[2] = UserProperty.AuthName;
 propsToReturn[3] = ObjectProperty.Created;

 QueryFilter[] filters = new QueryFilter[2];  //filter the results

 //simple name contains "user"
 filters[0] = new StringQueryFilter(UserProperty.SimpleName,
Operator.Contains,"user");
 //created at most a day ago
 GregorianCalendar filterDate = new GregorianCalendar();
 filterDate.add(Calendar.DATE, -1);
 Date yesterday = filterDate.getTime();
 filters[1] = new DateQueryFilter(ObjectProperty.Created,
Operator.GreaterThan, yesterday);
 try
 {
 IObjectQuery queryResults = objectManager.queryObjects(
 folderID,
 startRow,
 maxRows,
 sortProperty,
 ascending,
 propsToReturn,
 filters);

 for(int i = 0; i < queryResults.getRowCount(); i++)
 {
 IObjectQueryRow queryObject = queryResults.getRow(i);
 System.out.println(
 "User: " + queryObject.getStringValue(UserProperty.SimpleName)
+
 ", Created:" + queryObject.getCreated());
 }
 }
 catch(Exception e)
 {
 System.err.println(e.getMessage());
 e.printStackTrace(System.err);
 }
}
```

.NET (C#):

```
public static voidQueryObjects(stringloginToken)
{
 IObjectManager objectManager =
GetSession(loginToken).GetObjectManager(ObjectClass.User);

 int folderID = -1;  //search all folders
 int startRow = 0;  //start at the first found
 int maxRows = -1;  //return unlimited results
 ObjectProperty sortProperty = UserProperty.UniqueName;  //sort
on the unique name
 bool ascending = true;  //sort ascending

 ObjectProperty[] propsToReturn = new ObjectProperty[4];  //return
 specific properties
 propsToReturn[0] = UserProperty.SimpleName;
 propsToReturn[1] = UserProperty.UniqueName;
 propsToReturn[2] = UserProperty.AuthName;
 propsToReturn[3] = ObjectProperty.Created;

 DateTime yesterday =  new DateTime();
 yesterday = DateTime.Now.AddDays(-1);

 QueryFilter[] filters = new QueryFilter[2];  //filter the results

 //simple name contains "user"
 filters[0] = new StringQueryFilter(UserProperty.SimpleName,
Operator.Contains, "user");
 //created at most a day ago
 filters[1] = new DateQueryFilter(ObjectProperty.Created,
Operator.GreaterThan, yesterday);

 try
 {
 IObjectQuery queryResults = objectManager.QueryObjects(
 folderID,
 startRow,
 maxRows,
 sortProperty,
 ascending,
 propsToReturn,
 filters);

 for(int i = 0; i < queryResults.GetRowCount(); i++)
```

```
 {
 IObjectQueryRow queryObject = queryResults.GetRow(i);
 Console.Out.WriteLine(
 "User: " + queryObject.GetStringValue(UserProperty.SimpleName)
+
 ", Created:" + queryObject.GetCreated());
 }
 }
 catch(Exception e)
 {
 Response.Write(e.Message + "<br>");
 Response.Write(e.StackTrace + "<br><br>");
 }
}
```

.NET (VB):

```
Public Shared SubQueryObjects(ByVal loginToken As String)

 Dim objectManager As IObjectManager
 Dim session As IRemoteSession =
portletContext.GetRemotePortalSession

 objectManager = session.GetObjectManager(ObjectClass.User)

 Dim folderID As Integer
 folderID = -1 'search all folders

 Dim startRow As Integer
 startRow = 0 'start at the first found

 Dim maxRows As Integer
 maxRows = -1 'return unlimited results

 Dim sortProperty As ObjectProperty
 sortProperty = UserProperty.UniqueName 'sort on the unique name

 Dim ascending As Boolean
 ascending = True 'sort ascending

 Dim propsToReturn(4) As ObjectProperty 'return specific properties

 propsToReturn(0) = UserProperty.SimpleName
 propsToReturn(1) = UserProperty.UniqueName
 propsToReturn(2) = UserProperty.AuthName
 propsToReturn(3) = ObjectProperty.Created
```

```
 Dim yesterday As DateTime
 yesterday = DateTime.Now.AddDays(-1)

 Dim filters(2) As QueryFilter 'filter the results
 'simple name contains "user"
 filters(0) = New StringQueryFilter(UserProperty.SimpleName,
_Operator.Contains, _"user")
 'created at most a day ago
 filters(1) = New DateQueryFilter(ObjectProperty.Created,
_Operator.GreaterThan, yesterday)

 Try

 Dim queryResults As IObjectQuery
 queryResults = objectManager.QueryObjects(folderID, startRow,
maxRows, sortProperty,
ascending, propsToReturn, filters)

 Dim i As Integer
 Dim queryObject As IObjectQueryRow
 For i = 0 To queryResults.GetRowCount()-1

 queryObject = queryResults.GetRow(i)
 Response.Write(_ "User: " &
queryObject.GetStringValue(UserProperty.SimpleName)& _
 ", Created:" & queryObject.GetCreated() + "<br>")
 Next

 Catch e As Exception
 Response.Write(e.Message + "<br>")
 Response.Write(e.StackTrace)

 End Try
EndSub
```

ALI Object Type Class IDs and Modes

This table lists class IDs for all ALI object types and describes how modes are implemented by each.

Object Type	Class ID	Mode 1:Open	Mode 2:View	Mode 3:View Metadata
Administrative Folder	20	Edit	View Contents	View properties

Authentication Source	3	Edit	-	View properties
Community	512	Edit	Preview community	View properties
Community Page	514	Edit	Preview community page	View properties
Community Template	54	Edit	-	View properties
Content Crawler	38	Edit	-	View properties
Content Source	35	Edit	-	View properties
Directory Link	18	Edit	-	View properties
Directory Folder	17	Edit	View contents	View properties
Content Type	37	Edit	-	View properties
Experience Definition	8	Edit	-	View properties
External Operation	58	Edit	-	View properties
Federated Search	46	Edit	-	View properties
Filter	32	Edit	-	View properties
Invitation	44	Edit	-	View properties
Job	256	Edit	-	View properties
Page Template	56	Edit	-	View properties
Portlet	43	Edit	Preview portlet	View properties
Portlet Bundle	55	Edit	-	View properties
Portlet Template	61	Edit	-	View properties
Profile Source	7	Edit	-	View properties
Property	36	Edit	-	-
Remote Server	48	Edit	-	View properties
Site Map Folder	515	Edit	-	View properties
Smart Sort	42	Edit	-	View properties
Snapshot Query	33	Edit	-	View properties

User	1	Edit	View user profile	View properties
User Group	2	Edit	-	View properties
Web Service	47	Edit	-	View properties

Querying ALI Object Properties Using IDK Remote APIs

To query the properties of a specific portal object from a remote application, use the instance of IObjectQueryRow that represents the portal object.

To query the properties of a specific portal object, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an Object Manager for the type of object you are querying. For details, see *Retrieving ALI Object Managers Using IDK Remote APIs* on page 60.
3. Use the Object Manager to query for the object. For details, see *Querying ALI Objects Using IDK Remote APIs* on page 60.

   Note: You must include the properties you want to query in the `propsToReturn` parameter when you query for the object.

4. Use the instance of `IObjectQueryRow` that represents the portal object to query for ALI object properties or for custom properties. The code below uses the `getValue` call to retrieve object property values. The `IObjectQueryRow` interface also provides methods that cast the requested property to a known data type. Attempting to retrieve a property as an incorrect data type will result in an exception.

   Note: If you attempt to retrieve a property that was not specified as a field in the `propsToReturn` parameter in the query for the object, a `PropertyNotRequestException` will be thrown.

The following sample code demonstrates how to query the ALI object properties of a specific community. This example prints out most of the standard properties available on community objects, including generic object properties and all community-specific properties (all fields in `CommunityProperty`).

Java

```
public static void printCommunityProperties(IObjectQueryRow
communityObject) throws
PropertyNotRequestedException
{
```

```
    System.out.println("Object ID is " + communityObject.getID());
     System.out.println("Created Date is " +
communityObject.getCreated());
     System.out.println("Description is " +
communityObject.getDescription());
     System.out.println("Name is " + communityObject.getName());
     System.out.println("Last Modified Date is " +
communityObject.getLastModified());
     System.out.println("Owner ID is " +
communityObject.getOwner());
     System.out.println("Parent folder ID is " +
communityObject.getParentFolderID());

    if(communityObject.getObjectClass() == ObjectClass.Community)
 //only one instance so reference comparison is ok
     {
         System.out.println("Template ID is " +
communityObject.getValue(CommunityProperty.CommunityTemplateID));
         System.out.println("Footer ID is " +
communityObject.getValue(CommunityProperty.FooterID));
         System.out.println("Header ID is " +
communityObject.getValue(CommunityProperty.HeaderID));
         System.out.println("MandatoryTabOrder is " +
communityObject.getValue(CommunityProperty.MandatoryTabOrder));
         System.out.println("OwnerInfo is " +
communityObject.getValue(CommunityProperty.OwnerInfo));
         System.out.println("SiteMapRoot ID is " +
communityObject.getValue(CommunityProperty.SiteMapRootID));
     }
     else System.out.println("Not a community object!");
}
```

.NET (C#)

```
public static void PrintCommunityProperties(IObjectQueryRow
communityObject)
{
     Console.Out.WriteLine("Object ID is " +
communityObject.GetID());
     Console.Out.WriteLine("Created Date is " +
communityObject.GetCreated());
     Console.Out.WriteLine("Description is " +
communityObject.GetDescription());
    Console.Out.WriteLine("Name is " + communityObject.GetName());
     Console.Out.WriteLine("Last Modified Date is " +
communityObject.GetLastModified());
```

```
    Console.Out.WriteLine("Owner ID is " +
communityObject.GetOwner());
    Console.Out.WriteLine("Parent folder ID is " +
communityObject.GetParentFolderID());

   if(communityObject.GetObjectClass() == ObjectClass.Community)
 //only one instance so reference comparison is ok
    {
        Console.Out.WriteLine("Template ID is " +
communityObject.GetValue(CommunityProperty.CommunityTemplateID));
        Console.Out.WriteLine("Footer ID is " +
communityObject.GetValue(CommunityProperty.FooterID));
        Console.Out.WriteLine("Header ID is " +
communityObject.GetValue(CommunityProperty.HeaderID));
        Console.Out.WriteLine("MandatoryTabOrder is "+
communityObject.GetValue(CommunityProperty.MandatoryTabOrder));
        Console.Out.WriteLine("OwnerInfo is " +
communityObject.GetValue(CommunityProperty.OwnerInfo));
        Console.Out.WriteLine("SiteMapRoot ID is " +
communityObject.GetValue(CommunityProperty.SiteMapRootID));
    }
    else Console.Out.WriteLine ("Not a community object!");
}
```

.NET (VB)

```
Public Shared Sub PrintCommunityProperties(ByVal communityObject
 As IObjectQueryRow)

    Console.Out.WriteLine("Object ID is " &
communityObject.GetID())
    Console.Out.WriteLine("Created Date is " &
communityObject.GetCreated())
    Console.Out.WriteLine("Description is " &
communityObject.GetDescription())
   Console.Out.WriteLine("Name is " & communityObject.GetName())
    Console.Out.WriteLine("Last Modified Date is " &
communityObject.GetLastModified())
    Console.Out.WriteLine("Owner ID is " &
communityObject.GetOwner())
    Console.Out.WriteLine("Parent folder ID is " &
communityObject.GetParentFolderID())

    If communityObject.GetObjectClass() = ObjectClass.Community
Then 'only one instance so reference comparison is ok
        Console.Out.WriteLine("Template ID is " &
```

```
communityObject.GetValue(CommunityProperty.CommunityTemplateID))
        Console.Out.WriteLine("Footer ID is " &
communityObject.GetValue(CommunityProperty.FooterID))
        Console.Out.WriteLine("Header ID is " &
communityObject.GetValue(CommunityProperty.HeaderID))
        Console.Out.WriteLine("MandatoryTabOrder is " &
communityObject.GetValue(CommunityProperty.MandatoryTabOrder))
        Console.Out.WriteLine("OwnerInfo is " &
communityObject.GetValue(CommunityProperty.OwnerInfo))
        Console.Out.WriteLine("SiteMapRoot ID is " &
communityObject.GetValue(CommunityProperty.SiteMapRootID))
    Else
        Console.Out.WriteLine("Not a community object!")
    End If
EndSub
```

To query for custom properties of an object, use the `IExtendedData` interface. This example prints out all the custom properties available on a portal object by enumerating through the available properties and printing out their name and value. To retrieve property IDs, use the standard object manager object querying method with `ObjectClass.Property`, and use the ID on the object returned to query for the properties you need.

Java

```
public static void printCustomProperties(IObjectQueryRow
portalObject)
    throws PortalException, MalformedURLException, RemoteException
{
    IExtendedData customProperties =
portalObject.getExtendedData();

    Enumeration propertyNames = customProperties.getNames();
    String propertyName;

    while(propertyNames.hasMoreElements())
    {
        propertyName = (String)propertyNames.nextElement();
        System.out.println("Property "+ propertyName + " is "+
customProperties.getValue(propertyName));
    }
}
```

.NET (C#)

```
public static void PrintCustomProperties(IObjectQueryRow
portalObject)
```

```
{
    IExtendedData customProperties =
portalObject.GetExtendedData();

    IEnumerator propertyNames = customProperties.GetNames();
       string propertyName;

    while(propertyNames.MoveNext())
    {
        propertyName = (string)propertyNames.Current;
        Console.WriteLine("Property " + propertyName + " is " +
customProperties.GetValue(propertyName));
    }
}
```

.NET (VB)

```
Public Shared Sub PrintCustomProperties(ByVal portalObject As
IObjectQueryRow)

    Dim customProperties As IExtendedData =
portalObject.GetExtendedData()
    Dim propertyNames As IEnumerator = customProperties.GetNames()
    Dim propertyName As String

    While propertyNames.MoveNext()

        propertyName = propertyNames.Current
        Console.WriteLine("Property " & propertyName & " is " &
customProperties.GetValue(propertyName))

    End While
EndSub
```

Managing Object Security (ACLs) Using IDK Remote APIs

To manipulate object security, use the `IACL` interface in the IDK.

The `IACL` interface provides full access to object security, allowing you to add and remove users from an object's Access Control List. To access an ACL using the PRC, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an object manager for the type of object you are querying. For details, see *Retrieving ALI Object Managers Using IDK Remote APIs* on page 60.

3. Use the Object Manager to query for the object and use the instance of `IObjectQueryRow` that represents the portal object to determine the object ID. For details, see*Querying ALI Objects Using IDK Remote APIs* on page 60 and *Querying ALI Object Properties Using IDK Remote APIs* on page 67.
4. Use `IACL` to query the ACL of the object and enumerate or modify entries. The following sample code demonstrates how to edit the ACL of a specific portal object. The code accesses the ACL, removes an existing entry, adds a new entry, and saves the updated ACL. It then enumerates the users with admin access to the object.

Java

```
publicstatic void updateACL(IObjectManager objectManager, int
objectID)
 throws PortalException, MalformedURLException, RemoteException
{
 IACL acl = objectManager.queryACL(objectID);

 // Remove user with ID 101 from the ACL - will be ignored if the
 user is not present
 acl.removeUserEntry(101);

 // Add user with ID 10 to the ACL with Admin access
 acl.addUserGroupEntry(10, AccessLevel.ADMIN);

 //store changes to the portal
 objectManager.updateACL(objectID, acl);

 IACLEntry[] entries = acl.entries();

 for(int i = 0; i < entries.length; i++)
 {
 if(entries[i].getAccessLevel().equals(AccessLevel.ADMIN))
 System.out.println(
 entries[i].getPrincipalObjectClass() + " with ID " +
 entries[i].getPrincipalID() + " has admin access");
 }
}
```

.NET (C#)

```
publicstatic void UpdateACL(IObjectManager objectManager, int
objectID)
{
 IACL acl = objectManager.QueryACL(objectID);
```

```
 // Remove user with ID 101 from the ACL - will be ignored if the
 user is not present
 acl.RemoveUserEntry(101);

 // Add user with ID 10 to the ACL with Admin access
 acl.AddUserGroupEntry(10, AccessLevel.ADMIN);

 //store changes to the portal
 objectManager.UpdateACL(objectID, acl);

 IACLEntry[] entries = acl.Entries();

 for(int i = 0; i < entries.Length; i++)
 {
 if(entries[i].GetAccessLevel().equals(AccessLevel.ADMIN))
 Console.WriteLine(
 entries[i].GetPrincipalObjectClass() + " with ID " +
 entries[i].GetPrincipalID() + " has admin access");
 }
}
```

.NET (VB)

```
PublicShared Sub UpdateACL(ByVal objectManager As IObjectManager,
 ByVal objectID
As Integer)

 Dim acl As IACL = objectManager.QueryACL(objectID)

 ' Remove user with ID 101 from the ACL - will be ignored if the
 user is not present
 acl.RemoveUserEntry(101)

 ' Add user with ID 10 to the ACL with Edit access
 acl.AddUserGroupEntry(10, AccessLevel.EDIT)

 ' store changes to the portal
 objectManager.UpdateACL(objectID, acl)

 Dim entries() As IACLEntry = acl.Entries()
 Dim i As Integer

 For i = 0 To entries.Length
 If entries(i).GetAccessLevel() Is AccessLevel.ADMIN Then
 Console.WriteLine( _
 entries(i).GetPrincipalObjectClass() & " with ID " & _
```

```
 entries(i).GetPrincipalID() & " has admin access")
 End If
 Next i

EndSub
```

Access Control List (ACL) Privileges

Security for portal objects is implemented using Access Control Lists (ACLs) that can be applied to folders or individual objects. The ACL defines the access privileges for portal users and groups.

Users in the Administrators group have full access to all portal objects. Other users can be assigned the following access privileges.

Privilege	Description
Read	Allows users or groups to see an object.
Select	Allows users or groups to add an object to other objects. For example, it allows users to add Portlets to their My Pages, add users to groups, or associate Remote Servers with Web Services. Object selection lists display only those objects to which you have Select access.
Edit	Allows users or groups to modify an object, including moving or copying an object.
Admin	Allows users or groups full administrative control of an object, including deleting the object or approving it for migration.

About Remote Portlet Operations

There are many settings and options that apply only to portlets. In addition to manipulating portlet objects via IObjectManager, the IDK supports advanced remote portlet operations. Using the PRC `IPortlet*` interfaces, you can create and edit portlets and portlet templates, and manage Administrative and CommunityPortlet preferences for a specific portlet instance.

Note: The PRC `IPortlet*` interfaces in the IDK (com.plumtree.remote.prc) are different from the IDK portlet API (com.plumtree.remote.portlet). The interfaces in the portlet API are used to manage communication between a portlet and the portal, while the PRC `IPortlet*` interfaces provide access to administrative functionality related to the portlet objects stored in the portal.

For details on using remote portlet operations, see the following topics:

- *Creating Portlets and Portlet Templates Using IDK Remote APIs* on page 75
- *Editing Portlets and Portlet Templates Using IDK Remote APIs* on page 77

Creating Portlets and Portlet Templates Using IDK Remote APIs

To manipulate portlet and portlet template objects in ALI administration from a remote application, use the `IPortlet*` interfaces in the IDK.

Creating portlets and portlet templates is very similar; if you create a portlet from a portlet template, it inherits the settings and properties from the template. There is no further relationship between the two objects; changes in a template are not reflected in individual portlet instances made from that template. To create a new portlet or portlet template, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an `IPortletManager IPortletTemplateManager` object by calling `IRemoteSession.getPortletManager` or `getPortletTemplateManager`.
3. Create the portlet or portlet template as shown in the sample code below.
   a) Create a new method to create a new portlet template or portlet.
   b) Create the portlet template object using the parent folder ID and the web service ID (to create a portlet, you would provide the portlet template ID).
      - There are three ways to retrieve an administrative folder ID: (1) Use PRC search to perform a search for administrative folder objects, (2) Query for an existing portlet or portlet template and use its parent folder ID (available from the `IPortlet` or `IPortletTemplate` object), or (3) Let the user select a folder by using a pt:treeLink tag with classID = 20. (There is no Object Manager for administrative folders.) For details on tags, see *About Adaptive Tags* on page 202.
      - To query for the web service ID, execute a standard object query using `ObjectClass.WebService`.
   c) Set the name and description for the portlet template or portlet object.
   d) Save the portlet template or portlet.
   e) Return the ID for the newly created portlet template or portlet.

This example demonstrates how to create a new portlet template based on a web service. To create a new portlet, replace all instances of "PortletTemplate" with "Portlet" and pass in a portlet template ID instead of the web service ID.

Java

```
public static int createPortletTemplate(IPortletTemplateManager
portletTemplateManager,
int parentfolderID, int webserviceID)
    throws PortalException, MalformedURLException, RemoteException
{
    IPortletTemplate portletTemplate =
portletTemplateManager.createPortletTemplate(parentFolderID,
webserviceID);
    portletTemplate.setName("IDK Test Template");
    portletTemplate.setDescription("Created in IDK example");
    int portletTemplateID = portletTemplate.save();
    return portletTemplateID;
}
```

.NET (C#)

```
public static int CreatePortletTemplate(IPortletTemplateManager
portletTemplateManager,
int parentfolderID, int webserviceID)
{
    IPortletTemplate portletTemplate =
portletTemplateManager.CreatePortletTemplate(parentFolderID,
webserviceID);
    portletTemplate.SetName("IDK Test Template");
    portletTemplate.SetDescription("Created in IDK example");
    int portletTemplateID = portletTemplate.Save();
    return portletTemplateID;
}
```

.NET (VB)

```
Public Shared Function CreatePortletTemplate( _
    ByVal portletTemplateManager As IPortletTemplateManager, ByVal
 parentfolderID
As Integer, ByVal webserviceID As Integer) As Integer

    Dim portletTemplate As IPortletTemplate =
portletTemplateManager.CreatePortletTemplate(parentfolderID,webserviceID)
    portletTemplate.SetName("IDK Test Template");
    portletTemplate.SetDescription("Created in IDK example");
    Dim portletTemplateID As Integer = portletTemplateID =
portletTemplate.Save()

    Return portletTemplateID
```

```
End Function
```

Editing Portlets and Portlet Templates Using IDK Remote APIs

To modify settings for a portlet or portlet template object from a remote application, use the `IPortlet` and `IPortletTemplate` interfaces in the IDK.

The `IPortlet*` interfaces allow you to set the name, description and alignment for a portlet or portlet template. You can also edit Administrative settings for a portlet or portlet template, or modify CommunityPortlet settings for a portlet. To edit an existing portlet or portlet template, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an `IPortletManager` or `IPortletTemplateManager` object by calling `IRemoteSession.getPortletManager` or `getPortletTemplateManager`.
3. Retrieve an existing portlet or portlet template using `IPortletManager.getPortlet` or `IPortletTemplateManager.getPortletTemplate`. (To query for the portlet or portlet template ID, execute a standard object query.)
4. Edit the portlet or portlet template as shown in the sample code below.

This example demonstrates how to change the alignment, name and description, and edit a administrative setting for a portlet template. To make the same changes to a portlet, replace all instances of "portletTemplate" with "portlet".

Java

```
public static void editPortletTemplate(IPortletTemplate
portletTemplate)
 throws PortalException, MalformedURLException, RemoteException
{
 portletTemplate.setAlignment(Alignment.Narrow);
 portletTemplate.setName("IDK Test Document EDITED");
 portletTemplate.setDescription("Edited by IDK example");
 portletTemplate.save();
}

public static void addAdminSetting(IPortletTemplate
portletTemplate,
String settingName, String settingValue)
 throws PortalException, MalformedURLException, RemoteException
{
 portletTemplate.addAdminSetting(settingName, settingValue);
```

```
 portletTemplate.save();
}
```

.NET (C#)

```
public static void EditPortletTemplate(IPortletTemplate
portletTemplate)
 throws PortalException, MalformedURLException, RemoteException
{
 portletTemplate.SetAlignment(Alignment.Narrow);
 portletTemplate.SetName("IDK Test Document EDITED");
 portletTemplate.SetDescription("Edited by IDK example");
 portletTemplate.Save();
}

public static void AddAdminSetting(IPortletTemplate
portletTemplate,
string settingName, string settingValue)
{
 portletTemplate.AddAdminSetting(settingName, settingValue);
 portletTemplate.Save();
}
```

.NET (VB)

```
Public Shared Sub EditWebLinkDocument(ByVal portletTemplateManager
 As IPortletTemplate)

 portletTemplate.SetAlignment(Alignment.Narrow)
 portletTemplate.SetName("IDK Test Document EDITED")
 portletTemplate.SetDescription("Edited by IDK example")
 portletTemplate.Save()

EndSub

Public Shared Sub AddAdminSetting(ByRef portletTemplateManager As
 IPortletTemplate,
ByVal settingName As String, ByVal settingValue As String)

 portletTemplate.AddAdminSetting(settingName, settingValue)
 portletTemplate.Save()

EndSub
```

About Remote Knowledge Directory Operations

The PRC `IDocument*` interfaces in the IDK allow you to query for documents and document properties, create new documents, and edit the properties for existing documents.

The ALI Knowledge Directory displays links to documents in a hierarchical structure of folders and subfolders. These documents can be external or internal web pages, Office documents, or essentially any file of interest. In ALI , documents are referenced by document ID. File metadata is interpreted based on the Content Type. For details on Content Types, see the *Administrator Guide for AquaLogic Interaction* or the ALI online help.

The documents displayed in the Knowledge Directory are not stored in the portal; the ALI database contains only the file properties, including a link to the source file.

Note: The folders in the Knowledge Directory are different from the folders in portal Administration. For information on manipulating the portal objects found in administrative folders, see *About Remote ALI Object Management* on page 59.

For details on using remote Knowledge Directory operations, see the following topics:

- *Querying Documents in the Knowledge Directory Using IDK Remote APIs* on page 79
- *Creating Documents in the Knowledge Directory Using IDK Remote APIs* on page 82
- *Editing Document Properties in the Knowledge Directory Using IDK Remote APIs* on page 84

Querying Documents in the Knowledge Directory Using IDK Remote APIs

To query for documents in the ALI Knowledge Directory from a remote application, use the `IDocumentManager` interface in the IDK.

To query for documents, follow the steps below. You can also query documents using the PRC Search API; for details, see *Querying ALI Objects Using the IDK Remote Search API* on page 90.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an `IDocumentManager` object by calling `IRemoteSession.getDocumentManager`.
3. Execute the query, as shown in the sample code below.
   a) Create a new method to print out information about new documents in a folder.
   b) Create a query within the specified folder. There are two ways to retrieve a KD folder ID: (1) Use the PRC search API to perform a search for document folder objects, or (2) Let

the user select a folder by using a pt:treeLink tag with classID = 17. (There is no Object Manager for document folders.) For details on tags, see *About Adaptive Tags* on page 202.

c) Set up a query filter with the appropriate parameters.

d) Execute the query.

e) Loop through the results and print out document details.

This example demonstrates how to query for documents matching specific criteria (new documents) within a specific folder.

Java

```
public static void printNewDocumentDetails(IDocumentManager
documentManager, int
folderID, int daysOld) throws PortalException, RemoteException
{
 IDocumentQuery documentQuery =
documentManager.createQuery(folderID);

 // Set up a filter to query only documents up to the specified
age
 GregorianCalendar createdAge = new GregorianCalendar();
 createdAge.add(Calendar.DATE, -daysOld);
 QueryFilter ageFilter = new
DateQueryFilter(ObjectProperty.Created, Operator.GreaterThan,
createdAge.getTime());
 documentQuery.setFilters(new QueryFilter[]{ageFilter});

 IObjectQuery queryResults = documentQuery.execute();

 for (int i = 0; i < queryResults.getRowCount(); i++)
 {
  IObjectQueryRow document = queryResults.getRow(i);
  //Print out standard properties
  System.out.println("Document: " + document.getName());
  System.out.println("Created: " + document.getCreated());
  System.out.println("Description" + document.getDescription());
  //Print out a Document-specific property
  System.out.println("Located at URL: " +
document.getStringValue(DocumentProperty.URL));
 }
}
```

.NET (C#)

```
public static void PrintNewDocumentDetails(IDocumentManager
documentManager, int
folderID, int daysOld)
{
 IDocumentQuery documentQuery =
documentManager.CreateQuery(folderID);

 // Set up a filter to query only documents up to the specified
age
 DateTime createdAge = DateTime().AddDays(-daysOld);
 QueryFilter ageFilter = new
DateQueryFilter(ObjectProperty.Created, Operator.GreaterThan,
createdAge);
 documentQuery.SetFilters(new QueryFilter[]{ageFilter});

 IObjectQuery queryResults = documentQuery.Execute();

 for (int i = 0; i < queryResults.GetRowCount(); i++)
 {
  IObjectQueryRow document = queryResults.GetRow(i);
  //Print out standard properties
  Console.WriteLine("Document: " + document.GetName());
  Console.WriteLine("Created: " + document.GetCreated());
  Console.WriteLine("Description" + document.GetDescription());
  //Print out a Document-specific property
  Console.WriteLine("Located at URL: " +
document.GetStringValue(DocumentProperty.URL));
 }
}
```

.NET (VB)

```
Public Shared SubPrintNewDocumentDetails (ByVal documentManager
As IDocumentManager,
ByVal folderID As Integer, ByVal daysOld As Integer)

 Dim documentQuery As IDocumentQuery =
documentManager.CreateQuery(folderID)

 ' Set up a filter to query only documents up to the specified
age
 Dim createdAge As DateTime = New DateTime().AddDays(-daysOld)
 Dim ageFilter As QueryFilter = New
DateQueryFilter(ObjectProperty.Created,
```

```
Operator.GreaterThan,createdAge)
 Dim ageFilters() As QueryFilter = {ageFilter} 'Put the filter
into an array

 documentQuery.SetFilters(ageFilters)

 Dim queryResults As IObjectQuery = documentQuery.Execute()
 Dim i As Integer

 For i = 0 To queryResults.GetRowCount -1
Dim document As IObjectQueryRow = queryResults.GetRow(i)
'Print out standard properties
Console.WriteLine("Document: " + document.GetName())
Console.WriteLine("Created: " + document.GetCreated())
Console.WriteLine("Description: " + document.GetDescription())
'Print out a Document-specific property
Console.WriteLine("Located at URL:"+
document.GetStringValue(DocumentProperty.URL))
 Next

EndSub
```

Creating Documents in the Knowledge Directory Using IDK Remote APIs

To create new remote documents in the ALI Knowledge Directory from a remote application, use the `IRemoteDocument` and `IWebLinkDocument` interfaces in the IDK.

The `IWebLinkDocument` interface can only be used for HTML pages. To create a remote document of another type, use `IRemoteDocument` and set the content type. To create a new document, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an `IDocumentManager` object by calling `IRemoteSession.getDocumentManager`.
3. Create the document, as shown in the sample code below.
   a) Create a new method to create a new Web Link document.
   b) Create the document with the specified parameters: the folder ID, Content Source ID, and the URL to the html page.
      - There are two ways to retrieve a folder ID: (1) Use the PRC search API to perform a search for document folder objects, or (2) Let the user select a folder by using a pt:treeLink tag with classID = 17. (There is no Object Manager for document folders.). For details on tags, see *About Adaptive Tags* on page 202.

- This example uses the standard World Wide Web Content Source (ID 104). To query for available Content Sources, execute a standard object query using `ObjectClass.DataSource`.

c) Override the document name.
d) Save the document.
e) Return the newly created document ID.

This example demonstrates how to create a new Web Link document (HTML page). The implementation of `IRemoteDocument` is identical to the sample code shown below with one exception: you must set the Content Type. To query for available Content Types, execute a standard object query using `ObjectClass.DocumentType`.

Java

```
public static void createWebLinkDocument(IDocumentManager
documentManager, int folderID, String URL)
 throws PortalException, MalformedURLException, RemoteException
{
 IWebLinkDocument webLinkDocument =
  documentManager.createWebLinkDocument(folderID,104, URL);// 104
 is WWW Content Source
 webLinkDocument.setOverrideName("EDK Test Document"); // overrride
 intrinsic name
 int documentID = webLinkDocument.save();
 return documentID;
}
```

.NET (C#)

```
public static void CreateWebLinkDocument(IDocumentManager
documentManager, int folderID, string URL)
 throws PortalException, MalformedURLException, RemoteException
{
 IWebLinkDocument webLinkDocument =
  documentManager.CreateWebLinkDocument(folderID,104, URL);// 104
 is WWW Content Source
 webLinkDocument.SetOverrideName("EDK Test Document"); // overrride
 intrinsic name
 int documentID = webLinkDocument.Save();
 return documentID;
}
```

.NET (VB)

```
Public Shared Function CreateWebLinkDocument( _
 ByVal documentManager As IDocumentManager, ByVal folderID As
Integer, ByVal URL As String)As Integer

 Dim webLinkDocument As IWebLinkDocument =
  documentManager.CreateWebLinkDocument(folderID,104, URL)' 104
is WWW ContentSource
 webLinkDocument.SetOverrideName("EDK Test Document")' overrride
 intrinsic name
 Dim documentID As Integer = webLinkDocument.Save()
 Return documentID

EndFunction
```

Editing Document Properties in the Knowledge Directory Using IDK Remote APIs

To edit properties for existing documents in the ALI Knowledge Directory from a remote application, use the `IDocumentManager` interface in the IDK.

To edit an existing document, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an `IDocumentManager` object by calling `IRemoteSession.getDocumentManager`.
3. Edit the document, as shown in the sample code below.

This example demonstrates how to edit the document name, title, description and last modified date for an existing Web Link document (HTML page).

Note: This example uses integers to set properties on the document, in contrast with the `ObjectProperty` parameters used to retrieve information about general objects. This is because document properties can contain custom properties defined in the portal, for which there are no standard `ObjectProperty` parameters. To retrieve property IDs to use for this API, use the standard object querying method with `ObjectClass.Property`, and use the ID on the object returned to query for the properties you need.

Java

```
public static void editWebLinkDocument(IDocumentManager
documentManager, int documentID)
throws PortalException, RemoteException
{
 IDocumentProperties documentProperties =
```

```
documentManager.queryDocumentProperties(documentID);

 documentProperties.setStringValue(1, "IDK Document EDITED"); //
 1 = name
 documentProperties.setStringValue(105, "IDK Document Title
EDITED"); // 105 = title
 documentProperties.setStringValue(2, "Edited in IDK example ");
 // 2 = description
 documentProperties.setDateValue(112, newDate()); // 112 = last
modified date

 documentManager.updateDocumentProperties(documentID,
documentProperties);
}
```

.NET (C#)

```
public static void EditWebLinkDocument(IDocumentManager
documentManager, int documentID)
{
 IDocumentProperties documentProperties =
documentManager.QueryDocumentProperties(documentID);

 documentProperties.SetStringValue(1, "IDK Document EDITED"); //
 1 = name
 documentProperties.SetStringValue(105, "IDK Document Title
EDITED"); // 105 = title
 documentProperties.SetStringValue(2, "Edited in IDK example ");
 // 2 = description
 documentProperties.SetDateValue(112, newDateTime()); // 112 =
last modified date

 documentManager.UpdateDocumentProperties(documentID,
documentProperties);
}
```

.NET (VB)

```
Public Shared Sub EditWebLinkDocument(ByVal documentManager As
IDocumentManager,
ByVal folderID As Integer)

Dim documentProperties As IDocumentProperties =
documentManager.QueryDocumentProperties(documentID)

 documentProperties.SetStringValue(1, "IDK Document EDITED")' 1
```

```
= name
 documentProperties.SetStringValue(105, "IDK Document Title
EDITED") ' 105 = title
 documentProperties.SetStringValue(2, "Edited in IDK example ")
' 2 = description
 documentProperties.SetDateValue(112, NewDateTime()) ' 112 = last
 modified date

 documentManager.UpdateDocumentProperties(documentID,
documentProperties)

EndSub
```

About Remote User Operations

The PRC `IUserManager` and `IUserGroupManager` interfaces in the IDK allow you to leverage the portal's user hierarchy. You can query for the current user's ID and group information, create new groups, and manage group membership.

Portal users are organized into groups and sub-groups. This hierarchy allows administrators to customize the portal display for specific audiences and assign object security for collections of users.

Note: The PRC `IUser*` interfaces provide access to administrative functionality related to users in the portal. To access user settings and user profile information, use the methods in the com.plumtree.remote.portlet and com.plumtree.remote.util packages. To manipulate user objects, create an Object Manager of type ObjectClass.User.

For details on using remote user operations, see the following topics:

- *Querying Users Using IDK Remote APIs* on page 86
- *Creating Groups and Adding Users Using IDK Remote APIs* on page 88

Querying Users Using IDK Remote APIs

To query for the current user's ID and group information from a remote application, use the `IUserManager` interface in the IDK.

The `IUserManager` interface only provides access to user-specific administrative functionality. To access user settings and user profile information, use the methods in the com.plumtree.remote.util package. To manipulate user objects, create an Object Manager of type `ObjectClass.User`. For details, see *Retrieving ALI Object Managers Using IDK Remote APIs* on page 60.

To query for the properties for an existing user, follow the steps below. (To retrieve a user ID, you can also execute a standard object query with type ObjectClass.User.)

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an IUserManager by calling IRemoteSession.getUserManager.
3. Query the current user's groups, as shown in the sample code below.

   Note: The current user is the user initially used to create the PRC session, the user associated with the login token.

This example retrieves the current user's group associations and prints out the group IDs.

Note: To print out the names of groups, you must look up each group using an IUserGroupManager (IObjectManager with ObjectClass.UserGroup); group names are available on each IObjectQueryRow.

Java

```
public static void printGroupIDs(IUserManager userManager)
 throws PortalException, MalformedURLException, RemoteException
{
 int[] ids = userManager.getCurrentUserGroups();
 for(int i = 0 ; i < ids.length ; i++)
 {
  System.out.println("Current user belongs to group with ID: " +
 ids[i]);
 }
}
```

.NET (C#)

```
public static void PrintGroupIDs(IUserManager userManager)
 throws PortalException, MalformedURLException, RemoteException
{
 int[] ids = userManager.GetCurrentUserGroups();
 for(int i = 0 ; i < ids.length ; i++)
 {
  Console.WriteLine("Current user belongs to group with ID: " +
ids[i]);
 }
}
```

.NET (VB)

```
Public Shared Sub PrintGroupIDs(ByVal userManager As IUserManager)


 Dim ids() As Integer = userManager.GetCurrentUserGroups()
```

```
 Dim i As Integer
 For i = 0 To ids.Length
  Console.WriteLine("Current user belongs to group with ID: " &
ids[i])

EndSub
```

Creating Groups and Adding Users Using IDK Remote APIs

To create new groups and manage group membership from a remote application, use the `IUserGroupManager` interface in the IDK.

To create a new group and add a user, follow the steps below.

Note: The PRC `IUserGroupManager` interface only provides access to group-specific administrative functionality. To manipulate group objects, create an Object Manager of type `ObjectClass.UserGroup`.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an `IUserGroupManager` by calling `IRemoteSession.getUserGroupManager`.
3. Create a new method to create a group.
4. Create a new group using the folder ID as shown in the sample code below. There are two ways to retrieve an administrative folder ID: (1) Use PRC search to perform a search for administrative folder objects, or (2) Let the user select a folder by using a pt:treeLink tag with classID = 20. (There is no IObjectManager for administrative folders.) For details on tags, see *About Adaptive Tags* on page 202.
5. Return the group ID for the newly created group.
6. Create a new method to add a user.
7. Add the user to the new group, using the group ID returned in the previous method. (To query for an existing group ID, execute a standard object query using `ObjectClass.Group`.)

Java

```
public static int createEmptyGroup(IUserGroupManager
userGroupManager, int adminFolderID)
 throws PortalException, RemoteException
{
 int newGroupID = userGroupManager.createGroup(
  "IDK Group",
  "Created in IDK example",
```

```
  adminFolderID,
  new int[0], //no member users
  new int[0]); //no member groups
 return newGroupID;
}

public static void addUserToGroup(IUserGroupManager
userGroupManager, int userIDToAdd,
int newGroupID)
 throws PortalException, RemoteException
{
 userGroupManager.addMemberUsers(newGroupID, new
int[]{userIDToAdd});
}
```

.NET (C#)

```
public static int CreateEmptyGroup(IUserGroupManager
userGroupManager, int adminFolderID)
{
 int newGroupID = userGroupManager.CreateGroup(
  "IDK Group",
  "Created in IDK example",
  adminFolderID,
  new int[0], //no member users
  new int[0]); //no member groups
 return newGroupID;
}

public static void AddUserToGroup(IUserGroupManager
userGroupManager, int userIDToAdd,
int newGroupID)
{
 userGroupManager.AddMemberUsers(newGroupID, new
int[]{userIDToAdd});
}
```

.NET (VB)

```
Public Shared Function CreateEmptyGroup(ByVal userGroupManager As
 IUserGroupManager,
ByVal adminFolderID As Integer)

 Dim emptyIntegerArray(0) As Integer
 Dim newGroupID As Integer = userGroupManager.CreateGroup( _
  "IDK Group", _
```

```
  "Created in IDK example", _
  adminFolderID, _
  emptyIntegerArray, _
  emptyIntegerArray) 'no member users or groups
 Return newGroupID

EndFunction

Public Shared Sub AddUserToGroup( _
 (ByVal userGroupManager As IUserGroupManager, ByVal userIDToAdd
 As Integer, ByVal newGroupID As Integer)

 Dim singleUserArray() As Integer= {userIDToAdd}

 userGroupManager.AddMemberUsers(newGroupID, singleUserArray)

EndSub
```

About Remote Search Operations

The remote search API (com.plumtree.remote.prc.search) provides a generic interface to ALI search operations.

Using the PRC search API, you can query document, folder, user and Community objects using a standard request-response model. The API allows you to add multiple constraints and filter searches by location or object type.

Note: AquaLogic Interaction Search is a full-text search engine optimized for dealing with text; it should not be used for precise storage of numeric values, such as currency values.

Portal properties are represented as standard fields that can be accessed in search results by name or by iteration. By default, searches return a set of standard properties; you can choose to retrieve additional properties.

For details on using remote search operations, see the following topics:

- *Querying ALI Objects Using the IDK Remote Search API* on page 90
- *Using Query Constraints with the IDK Remote Search API* on page 94
- *Managing Search Results Using the IDK Remote Search API* on page 96

For information on remote search services, see *About ALI Federated Search Services* on page 385.

Querying ALI Objects Using the IDK Remote Search API

To search for ALI objects and documents from a remote application, use the `IPortalSearchRequest` interface in the IDK.

To construct a query, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an `ISearchFactory` from the session by calling `IRemoteSession.getSearchFactory`.
3. Create a new `IPortalSearchRequest` to represent your query.
4. Create a new method and construct the query, as shown in the sample code that follows.

The sample code below demonstrates how to query for a folder ID by folder name. The folder ID is required to execute other PRC functionality, including creating portlets, documents and groups. This example uses query constraints; for more information, see *Using Query Constraints with the IDK Remote Search API* on page 94.

Java

```
//set the endpoint to the value of web services server
String endpoint =
"http://IP-GW-AS08:9080/ptapi/services/QueryInterfaceAPI";
URL url = new URL(endpoint);

//set username and password to log in
//hard-coding the values is only for demo purposes
String username = "Administrator";
String password = "";
IRemoteSession prcSession =
RemoteSessionFactory.getExplicitLoginContext(url, username,
password);
ISearchFactory searchFactory = prcSession.getSearchFactory();
IPortalSearchRequest searchRequest =
searchFactory.createPortalSearchRequest();
public static int getFolderId(IPortalSearchRequest searchRequest,
 String folderName)
 throws Exception
{
 int folderId = -1;

 //search for the given folder name
 searchRequest.setQuery(folderName);

 // only search for folders
 ObjectClass[] objectTypes = {ObjectClass.DocumentFolder};
 searchRequest.setObjectTypesToSearch(objectTypes);
```

```
 ISearchResponse searchResponse = searchRequest.execute();
 ISearchResultSet resultSet = searchResponse.getResultSet();
 int numResults = searchResponse.getReturnedCount();

 if(numResults > 0)
  {
  Enumeration results = resultSet.getResults(); //just get the
first element
  IPortalSearchResult result = (IPortalSearchResult)
results.nextElement();
  folderId = result.getObjectID();
  }

 returnfolderId;
}
```

.NET (C#)

```
//set the endpoint to the value of web services server
String endpoint =
"http://IP-GW-AS08:9080/ptapi/services/QueryInterfaceAPI";

//set username and password to log in
//hard-coding the values is only for demo purposes
String username = "Administrator";
String password = "";
IRemoteSession session =
RemoteSessionFactory.GetExplicitLoginContext(new
System.Uri(endpoint), username, password);
ISearchFactory searchFactory = session.GetSearchFactory();
IPortalSearchRequest searchRequest =
searchFactory.CreatePortalSearchRequest();

public static int GetFolderId(IPortalSearchRequest searchRequest,
 String folderName)
{
 int folderId = -1;

 //search for the given folder name
 searchRequest.SetQuery(folderName);

 // only search for folders
 ObjectClass[] objectTypes = {ObjectClass.DocumentFolder};
 searchRequest.SetObjectTypesToSearch(objectTypes);

 ISearchResponse searchResponse = searchRequest.Execute();
```

```
 ISearchResultSet resultSet = searchResponse.GetResultSet();
 int numResults = searchResponse.GetReturnedCount();

 if(numResults > 0)
  {
  Enumeration results = resultSet.GetResults(); //just get the
first element
  IPortalSearchResult result = (IPortalSearchResult)
results.Current;
  folderId = result.GetObjectID();
  }

 returnfolderId;
}
```

.NET (VB)

```
//set the endpoint to the value of web services server
String endpoint =
"http://IP-GW-AS08:9080/ptapi/services/QueryInterfaceAPI"

//set username and password to log in
//hard-coding the values is only for demo purposes
String username = "Administrator"
String password = ""

Dim session As IRemoteSession =
RemoteSessionFactory.GetExplicitLoginContext(New
System.Uri(endpoint), username, password)
Dim searchFactory As ISearchFactory = session.GetSearchFactory()
Dim searchRequest As IPortalSearchRequest =
searchFactory.CreatePortalSearchRequest()

Public Shared Function GetFolderID(ByVal searchRequest As
IPortalSearchRequest, ByVal
folderName As String)

 Dim folderID As Int32 = -1
 searchRequest.SetQuery(folderName)

 Dim objectTypes() As ObjectClass = {ObjectClass.DocumentFolder}
 searchRequest.SetObjectTypesToSearch(objectTypes)

 Dim searchResponse As ISearchResponse = searchRequest.Execute()
 Dim resultSet As ISearchResultSet = searchResponse.GetResultSet()
```

```
 Dim numResults As Int32 = searchResponse.GetReturnedCount()

 If (numResults > 0) Then

  Dim results As IEnumerator = resultSet.GetResults()
  results.MoveNext()
 Dim result As IPortalSearchResult = DirectCast(results.Current,
IPortalSearchResult)
  folderId = result.GetObjectID()

 End If

 Return folderId

End Function
```

Using Query Constraints with the IDK Remote Search API

To limit search results to an object type or filter on a specific object property, use constraints.

Portal properties are represented as standard fields (`PortalField`, `PlumtreeField`) that can be accessed in search results by name or by iteration. By default, searches return a set of standard properties; you can choose to retrieve additional properties. To find the property ID, edit the property and note the ObjectID in the query string (for example, &in_hi_ObjectID=206).

This example sets constraints to limit the results to documents that are less than a year old.

Note: The code below should be executed within a method that can throw a `SearchException` and a `RemoteException`.

Java

```
...
//add a property field to the search
//SetFieldsToReturn adds fields to the existing default fields
//make the PortalField based on the property ID
int propertyID = 206;
Field[] fieldsToReturn = new Field[]
{PortalField.forID(propertyID)};
searchRequest.setFieldsToReturn(fieldsToReturn);

//constrain the results to documents
ObjectClass[] objectClasses = new ObjectClass[]
{ObjectClass.Document};
searchRequest.setObjectTypesToSearch(objectClasses);

//return only documents that are less than a year old
```

```
Calendar cal = Calendar.getInstance();
cal.add(Calendar.YEAR, -1);
Date createdDate = cal.getTime();
IFilterClause filterClause = searchFactory.createAndFilterClause();
filterClause.addStatement(PlumtreeField.CREATED,
Operator.GreaterThan, createdDate);
searchRequest.setQuery(searchString, filterClause);

//execute the query and display the results
DisplayResults(searchRequest, propertyID);
...
```

.NET (C#)

```
...
//add a property field to the search
//SetFieldsToReturn adds fields to the existing default fields
//make the PortalField based on the property ID
int propertyID = 206;
Field[] fieldsToReturn = new Field[]
{PortalField.ForID(propertyID)};
searchRequest.SetFieldsToReturn(fieldsToReturn);

//constrain the results to documents
ObjectClass[] objectClasses = new ObjectClass[]
{ObjectClass.Document};
searchRequest.SetObjectTypesToSearch(objectClasses);

//return only documents that are less than a year old
DateTime createdDate = DateTime.Today.AddYears(-1);
IFilterClause filterClause = searchFactory.CreateAndFilterClause();
filterClause.AddStatement(PlumtreeField.CREATED,
Operator.GreaterThan, createdDate);
searchRequest.SetQuery(searchString, filterClause);

//execute the query and display the results
DisplayResults(searchRequest, propertyID);
...
```

.NET (VB)

```
...
'add a property field to the search
'note that SetFieldsToReturn only adds fields to the existing
default fields
'make the PortalField based on the property ID
```

```
Dim propertyID As Int32 = 206
Dim fieldsToReturn(0) As Field
fieldsToReturn(0) = PortalField.ForID(propertyID)
searchRequest.SetFieldsToReturn(fieldsToReturn)

'constrain the results to documents
Dim objectClasses(0) As ObjectClass
objectClasses(0) = ObjectClass.Document
searchRequest.SetObjectTypesToSearch(objectClasses)

'return only documents that are less than a year old
Dim createdDate As DateTime = DateTime.Today.AddYears(-1)
Dim filterClause As IFilterClause =
searchFactory.CreateAndFilterClause()
filterClause.AddStatement(PlumtreeField.CREATED,
Operator.GreaterThan, createdDate)
searchRequest.SetQuery(searchString, filterClause)

'execute the query and display the results
DisplayResults(searchRequest, propertyID)
...
```

Managing Search Results Using the IDK Remote Search API

To manage the results returned from a search in a remote application, use the `IPortalSearchResult` interface.

AquaLogic Interaction Search stores numeric and date fields (properties) as 32-bit floats. This means that any prc.search method that returns the value of a numeric field is potentially subject to roundoff. `getFieldAsInt` is converted to a float and then converted back to an int. `getFieldAsFloat` rounds the original value to return it as a float.

This example continues the sample code from *Querying ALI Objects Using the IDK Remote Search API* on page 90. The code samples below print out a result summary and display the returned properties.

Note: To access the additional property referenced in the `setFieldsToReturn` method of `IPortalSearchRequest`, this example uses `getFieldAsString`. This is because the field type is unknown; if you know the type of property being returned, use the appropriate type-specific field (e.g., `getFieldAsDate`, `getFieldAsFloat`).

Java

```
public static void displayResults(IPortalSearchRequest
searchRequest, int propertyID)
    throws SearchException, RemoteException
```

```
{

 //execute the search
 ISearchResponse searchResponse = searchRequest.execute();

 //get information about the number of results returned
 System.out.println("Total matches is " +
searchResponse.getTotalCount());
 System.out.println("First result is " +
searchResponse.getFirstResultIndex());
 System.out.println("Number returned is " +
searchResponse.getReturnedCount());

 //write out any warnings
 SearchWarning warning = searchResponse.getWarning();
 if (null != warning)
 {
  if (warning.getCode() ==
(SearchWarning.PROCESSING_TIMED_OUT.getCode()))
  {
   System.out.println("Search Warning: Timed out when processing
 search request; a partial search result was returned");
  }
  if (warning.getCode ==
(SearchWarning.TOO_MANY_WILDCARD_EXPANSIONS.getCode())
  {
   System.out.println("Search Warning: A wildcard query, such as
 \"a*\", matched a large number of patterns, only some of which
were used for your search.");
  }
 }

 //make the PortalField based on the property ID (from
IPortalSearchRequest.SetFieldsToReturn)
 Field propField = PortalField.forID(propertyID);

 //iterate through the results
 ISearchResultSet resultSet = searchResponse.getResultSet();
 IEnumerator results = resultSet.getResults();
 while (results.hasMoreElements())
 {

System.out.println("-------------------------------------------------");

  IPortalSearchResult result = (IPortalSearchResult)
```

```
results.nextElement();
  System.out.println("name is " + result.getName());
  System.out.println("class id is " + result.getClassID());
  System.out.println("created is " + result.getCreated());
  System.out.println("excerpt is " + result.getExcerpt());
  System.out.println("last modified is " +
result.getLastModified());
  System.out.println("object id is " + result.getObjectID());
  System.out.println("url is " + result.getURL());
  System.out.println("icon url is " + result.getIconURL());
  System.out.println("rank is " + result.getRank());
  //write out the property if the field exists
  Object value = result.getFieldAsObject(propField);
  if (null != value)
  {
   //use GetFieldAsString because type of field is unknown
   String propResult = result.GetFieldAsString(propField);
   System.out.println("property field is " + propResult);
  }
 }
}
```

.NET (C#)

```
public static void DisplayResults(IPortalSearchRequest
searchRequest, int propertyID)
{

 //execute the search
 ISearchResponse searchResponse = searchRequest.Execute();

 //get information about the number of results returned
 Console.WriteLine("Total matches is " +
searchResponse.GetTotalCount());
 Console.WriteLine("First result is " +
searchResponse.GetFirstResultIndex());
 Console.WriteLine("Number returned is " +
searchResponse.GetReturnedCount());

 //write out any warnings
 SearchWarning warning = searchResponse.GetWarning();
 if (null != warning)
 {
  if
(warning.GetCode().Equals(SearchWarning.PROCESSING_TIMED_OUT.GetCode()))
```

```
  {
   Console.WriteLine("Search Warning: Timed out when processing
search request; a partial search result was returned");
  }
  if
(warning.GetCode().Equals(SearchWarning.TOO_MANY_WILDCARD_EXPANSIONS.GetCode())

  {
   Console.WriteLine("Search Warning: A wildcard query, such as
\"a*\", matched a large number of patterns, only some of which
were used for your search.");
  }
 }

 //make the PortalField based on the property ID (from
IPortalSearchRequest.SetFieldsToReturn)
 Field propField = PortalField.ForID(propertyID);

 //iterate through the results
 ISearchResultSet resultSet = searchResponse.GetResultSet();
 IEnumerator results = resultSet.GetResults();
 while (results.MoveNext())
 {

Console.WriteLine("-------------------------------------------------");

  IPortalSearchResult result = (IPortalSearchResult)
results.Current;
  Console.WriteLine("name is " + result.GetName());
  Console.WriteLine("class id is " + result.GetClassID());
  Console.WriteLine("created is " + result.GetCreated());
  Console.WriteLine("excerpt is " + result.GetExcerpt());
  Console.WriteLine("last modified is " +
result.GetLastModified());
  Console.WriteLine("object id is " + result.GetObjectID());
  Console.WriteLine("url is " + result.GetURL());
  Console.WriteLine("icon url is " + result.GetIconURL());
  Console.WriteLine("rank is " + result.GetRank());
  //write out the property if the field exists
  Object value = result.GetFieldAsObject(propField);
  if (null != value)
  {
   //use GetFieldAsString because type of field is unknown
   String propResult = result.GetFieldAsString(propField);
   Console.WriteLine("property field is " + propResult);
```

```
  }
 }
}
```

.NET (VB)

```
Public Shared Sub DisplayResults(ByVal searchRequest As
IPortalSearchRequest, ByVal
propertyID As Int32)

 //execute the search
 Dim searchResponse As ISearchResponse = searchRequest.Execute()

 'get information about the number of results returned
 Console.WriteLine("Total matches is "&
searchResponse.GetTotalCount())
 Console.WriteLine("First result is "&
searchResponse.GetFirstResultIndex())
 Console.WriteLine("Number returned is "&
searchResponse.GetReturnedCount())

 'write out any warnings
 Dim warning AsSearchWarning = searchResponse.GetWarning()
 If Notwarning IsNothing Then

  If
(warning.GetCode().Equals(SearchWarning.PROCESSING_TIMED_OUT.GetCode()))
 Then
   Console.WriteLine("Search Warning: Timed out when processing
search request; a partial search result was returned")
 End If

  If
(warning.GetCode().Equals(SearchWarning.TOO_MANY_WILDCARD_EXPANSIONS.GetCode()))

  Then
   Console.WriteLine("Search Warning: A wildcard query, such as
a*, matched a large number of patterns, only some of which were
used for your search.")
  End If

 End If

 'make the PortalField based on the property ID (from
IPortalSearchRequest.SetFieldsToReturn)
 Dim propField As Field = PortalField.ForID(propertyID)
```

```
 'iterate through the results
 Dim resultSet AsISearchResultSet = searchResponse.GetResultSet()

 Dim results AsIEnumerator = resultSet.GetResults()
 While(results.MoveNext())


Console.WriteLine("-----------------------------------------------")

  Dim result As IPortalSearchResult = DirectCast(results.Current,
 IPortalSearchResult)
  Console.WriteLine("name is " + result.GetName())
  Console.WriteLine("class id is " + result.GetClassID())
  Console.WriteLine("created is " + result.GetCreated())
  Console.WriteLine("excerpt is " + result.GetExcerpt())
  Console.WriteLine("last modified is " + result.GetLastModified())

  Console.WriteLine("object id is " + result.GetObjectID())
  Console.WriteLine("url is " + result.GetURL())
  Console.WriteLine("icon url is " + result.GetIconURL())
  Console.WriteLine("rank is " + result.GetRank())
  'write out the property if the field exists
  Dim value As Object = result.GetFieldAsObject(propField)
  If Not value Is Nothing Then
   'use GetFieldAsString because type of field is unknown
   Dim propResult As String = result.GetFieldAsString(propField)
   Console.WriteLine("property field is " + propResult)
  End If

 End While

End Sub
```

Starting Portal Jobs Using IDK Remote APIs

To start an existing ALI job from a remote application, use the `IJobManager` interface in the IDK.

A job is a collection of related portal operations. Each operation is one task, such as a crawl for documents, an import of users, or one of the system maintenance tasks. The return code from starting a job indicates whether or not the call was successful (whether or not the object is locked). See the IDK API documentation for `LockStatus` for more information on what each return value indicates.

To start a portal job, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve an `IJobManager` by calling `IRemoteSession.getJobManager`.
3. Query for the objectID of the job. For details, see *Querying ALI Object Properties Using IDK Remote APIs* on page 67.
4. Create a new method to start a job and start the job as shown in the code below. It is a best practice to check the return code in the response and print out a description.

   Note: The `startJob` method does not technically start the portal job; it sets the start time for the job to the current time. If there are problems that prevent the job from running once it is rescheduled, they are not accessible from this call.

This example checks to see if the job object is locked. If it is unlocked, the code assumes that the job will start.

Java

```
public static voidstartJob(IJobManager jobManager, int jobID)
{
 int status = jobManager.startJob(jobID);

 if(status == LockStatus.UNLOCKED) //LockStatus.UNLOCKED = 0
 System.out.println("Job started successfully");
 else
 System.out.println("Job failed to start");
}
```

.NET (C#)

```
public static voidStartJob(IJobManager jobManager, int jobID)
{
 int status = jobManager.startJob(jobID);

 if(status == LockStatus.Unlocked) //LockStatus.Unlocked = 0
 Console.WriteLine("Job started successfully");
 else
 Console.WriteLine("Job failed to start");
}
```

.NET (VB)

```
Public Shared SubStartJob ByVal jobManager As IJobManager, ByVal
 jobID As Integer)
```

```
 Dim objectManager As Integer = jobManager.StartJob(jobID)

 If status = LockStatus.UnlockedThen 'LockStatus.Unlocked = 0
 Console.WriteLine "Job started successfully");
 Else
 Console.WriteLine("Job failed to start");
 End If
End Sub
```

About Remote Collaboration APIs

The IDK's remote Collaboration API (com.plumtree.remote.prc.collaboration) provides programmatic access to many of the objects stored within AquaLogic Interaction Collaboration. Use this remote programming interface to embed collaborative components and functions into any web application delivered through the ALI framework.

The PRC Collaboration API can be used to access existing ALI Collaboration objects. Each object interface provides a `GetDetailsURL` method that returns the URL to the associated detail page in ALI Collaboration. To create a URL to a Collaboration component, first obtain an instance of the associated object, then call the `getDetailsURL` method.

For details on remote Collaboration APIs, see the following topics:

- *About Remote Collaboration Project Operations* on page 103
- *About Remote Collaboration Discussion Operations* on page 123
- *About Remote Collaboration Document and Folder Operations* on page 137
- *About Remote Collaboration Task Operations* on page 150

Each object interface allows you to determine a user's permissions for a specific Collaboration object in two ways:

- Get the access level for each role (`getAccessLevel`).
- Determine whether a specific action is permitted (`isActionAllowed`).

For details, see *Managing Collaboration Project Roles Using the IDK Remote API* on page 119.

For more details on ALI Collaboration functionality, see the *Administrator Guide for AquaLogic Integration Collaboration* and the Collaboration online help.

About Remote Collaboration Project Operations

Every Collaboration task, document and discussion is associated with a project; the project must exist before you can create any component objects. Using the PRC Collaboration API in the IDK

(com.plumtree.remote.prc.collaboration.project), you can query, create and modify projects, and manage project security and subscriptions from a remote application.

Each Collaboration project has its own set of objects and properties that are not shared with other projects. The PRC Collaboration API provides access to the following project functionality:

- **Collaboration Workspace**: Create, copy, modify, and delete Collaboration projects.
- **Subscriptions**: Provide users with e-mail notifications when an activity occurs in a project, such as adding a folder or modifying a document.
- **Search**: Search collaboration projects, and create filters for focused results.

For details on using remote Collaboration project operations, see the following topics:

- *Querying Existing Collaboration Projects Using IDK Remote APIs* on page 104
- *Creating Collaboration Projects Using IDK Remote APIs* on page 109
- *Editing Collaboration Project Properties Using IDK Remote APIs* on page 116
- *Managing Collaboration Project Roles Using the IDK Remote API* on page 119
- *Managing Collaboration Subscriptions Using the IDK Remote API* on page 121

Querying Existing Collaboration Projects Using IDK Remote APIs

To search for collaboration projects by name from a remote application, use the `IProjectManager` interface in the IDK.

Results can be filtered in a variety of ways. The `IProjectManager` query method takes in an `IProjectFilter` object that allows you to set the following search options:

Search Text (Name)	Sets the string that will be used to search project names. If you do not set a search string, all projects will be returned. The name search string is case-insensitive by default; all projects will be returned that contain the text using the form *text*.
Maximum Results	Sets the maximum number of results returned. The default is to return all results.
Order-By Fields and Sort Order	Sets the fields to be displayed with an order-by functionality (name or last modified date) and sets the sort order (ascending or descending).
Security	Enables or disables the security filter that applies security to the result set with respect to the user that submitted the query. If the filter is enabled, the query result will only include objects for which the querying user has appropriate permission. The default is false (disabled); all objects matching the query criteria will be returned.

Result Filter: Project Type	Limits the query to those projects for which the current user is a project leader, or extend the search to all projects. For details on project roles, see *Managing Collaboration Project Roles Using the IDK Remote API* on page 119.

The code samples below are simplified for illustration.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Query for a project as shown in the code below.
   a) Retrieve an `IProjectManager`.
   b) Implement `IProjectFilter` to filter results and define the query.
   c) Execute the search and display results in table format.

Java

```
...

//perform the search
IProjectManager projectManager = getProjectManager(request,
response, out);
IProjectFilter projectFilter =
projectManager.createProjectFilter();

//hard-code the max results to 10
projectFilter.setMaximumResults(10);

//set the query
projectFilter.setNameSearchText(searchText);

//execute the search and print out the results
IProject[] projects = projectManager.queryProjects(projectFilter);
if (projects.length > 0)
{
 %>
 <table>
 <tr>
  <td>
  Search Results
  </td>
 </tr>
 <tr>
  <td>
  Project Name
```

```
  </td>
  <td>
  Project ID
  </td>
 </tr>
 <%
 for (int i = 0; i < projects.length; i++)
 {
  IProject project = projects[i];
  %>
  <tr>
   <td>
   <%out.println(project.getName());%>
   </td>
   <td>
   <%out.println(project.getID());%>
   </td>
  </tr>
  <%
 }
}
else
{
 %>
 <tr>
  <td colspan="2">
  <%out.println("No projects found using search query of " +
searchText);%>
  </td>
 </tr>
 </table>
...
```

.NET (C#)

```
...

//perform the search
Plumtree.Remote.PRC.Collaboration.Project.IProjectManager
projectManager = GetProjectManager(Request, Response);
Plumtree.Remote.PRC.Collaboration.Project.IProjectFilter
projectFilter = projectManager.CreateProjectFilter();

//hard-code the max results to 10
projectFilter.MaximumResults = 10;
```

```
//set the query
projectFilter.NameSearchText = searchText;

//execute the search and print out the results
Plumtree.Remote.PRC.Collaboration.Project.IProject[] projects =
projectManager.QueryProjects(projectFilter);
if (projects.Length > 0)
{
 %>
 <table>
 <tr>
  <td>
  Search Results
  </td>
 </tr>
 <tr>
  <td>
  Project Name
  </td>
  <td>
  Project ID
  </td>
 </tr>
 <%
 for (int i = 0; i < projects.Length; i++)
 {
  Plumtree.Remote.PRC.Collaboration.Project.IProject project =
projects[i];
  %>
  <tr>
   <td>
   <%Response.Write(project.Name);%>
   </td>
   <td>
   <%Response.Write(project.ID);%>
   </td>
  </tr>
  <%
 }
}
else
{
 Response.Write("No projects found using search query of " +
searchText);
```

```
}
...
```

.NET (VB)

```
...

'perform the search
dim projectManager as
Plumtree.Remote.PRC.Collaboration.Project.IProjectManager =
GetProjectManager(Request, Response)
dim projectFilter as
Plumtree.Remote.PRC.Collaboration.Project.IProjectFilter =
projectManager.CreateProjectFilter()

'hard-code the max results to 10
projectFilter.MaximumResults = 10

'set the query
projectFilter.NameSearchText = searchText

'execute the search and print out the results
dim projects() as
Plumtree.Remote.PRC.Collaboration.Project.IProject =
projectManager.QueryProjects(projectFilter)
if projects.Length > 0 then
 %>
 <tr>
  <td>
  Search Results
  </td>
 </tr>
 <tr>
  <td>
  Project Name
  </td>
  <td>
  Project ID
  </td>
 </tr>
 <%
 dim i as Integer
 for i = 0 to projects.Length -1
  dim project as Plumtree.Remote.PRC.Collaboration.Project.IProject
 = projects(i)
  %>
```

```
  <tr>
   <td>
   <%Response.Write(project.Name) %>
   </td>
   <td>
   <%Response.Write(Cstr(project.ID)) %>
   </td>
  </tr>
  <%
 next
else
 Response.Write("No projects found using search query of " +
searchText)
...
```

Creating Collaboration Projects Using IDK Remote APIs

To create a new Collaboration project from a remote application, use the `IProjectManager` interface in the IDK.

The `IProjectManager` interface provides a factory method for creating new projects that takes in a name and description, and returns an `IProject` object with a corresponding object ID and associated properties. The sample code below provides a simple example of creating a new project.

The `IProjectManager` interface also provides methods for copying content and metadata from existing collaboration projects. Prior to the call, both the source and target project must be a persisted `IProject`, and both projects must have a set start date. The user must be a Project Leader and have READ access in both the source and target projects.

Method	Description	Copies
`copyProjectContent`	Copies all project content from the source project to the target project. The copied project will be stored permanently. Security is mapped isomorphically from the source project to the target project; if the ProjectMember role in the source project has access level 1 on object X, and object X is copied to object Y, then the ProjectMember role in	• All document folders • All documents • All discussions (not messages) • All task lists • All tasks

Method	Description	Copies
	the target project will have access level 1 on object Y.	
`copyProjectMetadata`	Copies the basic metadata and all IRole objects from the source project to the target project. The copied project will be stored permanently. No store method is required. The old roles in the target project will be overwritten with the copied roles from the source project.	• Project description • All IRole objects • Start date information (if the target project's start date is not set and the source project's start date is available)

For details on these methods, see the IDK API documentation.

Note: Before writing any code, you must prepare a custom development project that references the standard IDK library (idk.jar/idk.dll).

The code samples below implement the following steps:

1. Initiate a PRC session. This example retrieves a login token using `IPortletContext`; you can also use `IRemoteSession`. (For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.)
2. Get the Project Manager.
3. Create a project with name and description.
4. Get the ID for the new project.

Java

```
<%@page import="com.plumtree.remote.prc.IRemoteSession,
com.plumtree.remote.prc.RemoteSessionFactory,
com.plumtree.remote.prc.collaboration.*,com.plumtree.remote.prc.collaboration.project.*,
com.plumtree.remote.portlet.*,java.util.*,java.text.*" %>

<%
//get the project manager
private IProjectManager getProjectManager(HttpServletRequest req,
 HttpServletResponse
res, JspWriter pout) throws Exception
{
```

```
 IProjectManager projectManager = null;
 IPortletContext portletContext =
PortletContextFactory.createPortletContext(req, res);
 IPortletRequest portletRequest = portletContext.getRequest();
 String loginToken = portletRequest.getLoginToken();
 if (null == loginToken)
 {
  pout.println("Unable to retrieve the login token. Confirm that
 the login token has been checked in the Advanced Settings page
of the Web Service.");
 }

 //get the remote session
 com.plumtree.remote.prc.IRemoteSession portalSession =
portletContext.getRemotePortalSession();

 //get a collab factory and a project manager
 ICollaborationFactory collabFactory =
portalSession.getCollaborationFactory();
 projectManager = collabFactory.getProjectManager();

 return projectManager;

}

//create a project and print out the project id
name = (null == name) ? "ExampleProject" : name;
description = (null == description) ? "ExampleProjectDescription"
 : description;

//create the project
IProjectManager projectManager = getProjectManager(request,
response, out);
IProject project = projectManager.createProject(name, description);

//to set additional properties, you must call store() after making
 changes
//for example:
/*
project.setStatus(ProjectStatus.NOT_STARTED);
project.setStartDate(new Date());
*/

//call store before asking for the ID.
project.store();
```

```
//get the new project ID
project.getID()

%>
```

.NET (C# - Project Page)

```
<%@ Page language="c#" Codebehind="ProjectSample.aspx.cs"
AutoEventWireup="false"
Inherits="WebProjectSample.ProjectSample" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<html>
<body>

//create a project and print out the project id
<%
{
 name = (null == name) ? "ExampleProject" : name;
 description = (null == description) ? "ExampleProjectDescription"
 : description;

 //create the project
 Plumtree.Remote.PRC.Collaboration.Project.IProjectManager
projectManager = GetProjectManager(Request, Response);
 Plumtree.Remote.PRC.Collaboration.Project.IProject project =
 projectManager.CreateProject(name, description);

 //to set additional properties, you must call store() after making
 changes
 //for example:
 /*
 project.Status = ProjectStatus.NotStarted;
 project.StartDate = new Date();
 */

 //call store before asking for the id.
 project.Store();
}
%>

<table>
 <tr>
  <td>
  <%
```

```
  Response.Write("ID of newly created project is " + project.ID);

  %>
  </td>
 </tr>
</table>
</body>
</html>
```

.NET (C# - Code-Behind Page)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Plumtree.Remote.PRC;
using Plumtree.Remote.PRC.Collaboration;
using Plumtree.Remote.PRC.Collaboration.Project;
using Plumtree.Remote.Portlet;

namespace WebProjectSample
{

 public class ProjectSample: System.Web.UI.Page
 {
  //name and description for new project
  public String name = "ExampleProject";
  public String description = "ExampleProjectDescription";

  //get the Project Manager
  public IProjectManager GetProjectManager(HttpRequest req,
HttpResponse res)
  {

   IProjectManager projectManager = null;
   IPortletContext portletContext =
PortletContextFactory.CreatePortletContext(req, res);
   IPortletRequest portletRequest = portletContext.GetRequest();
   String loginToken = portletRequest.GetLoginToken();
   if (null == loginToken)
```

```
   {
    res.Write("Unable to retrieve the login token. Confirm that
the the login token has been checked in the Advanced Settings page of
 the Web Service.");
   }

   //get the remote session
   Plumtree.Remote.PRC.IRemoteSession portalSession =
portletContext.GetRemotePortalSession();

   //get a collab factory and a project manager
   ICollaborationFactory collabFactory =
portalSession.GetCollaborationFactory();
   projectManager = collabFactory.GetProjectManager();
   return projectManager;
  }
 }
}
```

.NET (VB - Project Page)

```
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="ProjectSample.aspx.vb"
Inherits="TestProjectVB.ProjectSample"%>
<!DOCTYPE HTML PUBLIC "-'W3C'DTD HTML 4.0 Transitional'EN" >
<html>
<body>
<%

 'create a project and print out the project id
 name = "ExampleProject"
 description = "ExampleProjectDescription"

 'create the project
 dim projectManager as
Plumtree.Remote.PRC.Collaboration.Project.IProjectManager =
 GetProjectManager(Request, Response)
 dim project as Plumtree.Remote.PRC.Collaboration.Project.IProject
 = projectManager.CreateProject(name, description)

 'to set additional properties, you must call store() after making
 changes
 'for example:
 'project.Status = ProjectStatus.NotStarted
 'project.StartDate = new Date()
```

```
 'call store before asking for the id.
 project.Store()

%>

<table>
 <tr>
  <td>
  <%
   Response.Write("ID of newly created project is " +
Cstr(project.ID))
  %>
  </td>
 </tr>
</table>
</body>
</html>
```

.NET (VB - Code-Behind Page)

```
Imports System
Imports System.Collections
Imports System.ComponentModel
Imports System.Data
Imports System.Drawing
Imports System.Web
Imports System.Web.SessionState
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Web.UI.HtmlControls
Imports Plumtree.Remote.PRC
Imports Plumtree.Remote.PRC.Collaboration
Imports Plumtree.Remote.PRC.Collaboration.Project
Imports Plumtree.Remote.Portlet

Public Class ProjectSample
Inherits System.Web.UI.Page

 'get the project manager
 Public Function GetProjectManager(ByVal req As HttpRequest, ByVal
 res As HttpResponse)
 As IProjectManager
 Dim projectManager As IProjectManager = Nothing
 Dim portletContext As IPortletContext =
PortletContextFactory.CreatePortletContext(req, res)
 Dim portletRequest As IPortletRequest =
```

```
portletContext.GetRequest()
 Dim loginToken As String = portletRequest.GetLoginToken()
 If loginToken Is Nothing Then
  res.Write("Unable to retrieve the login token. Confirm that the
 login token has been checked in the Advanced Settings page of
the Web Service.")
 End If

 'get the remote session
 Dim portalSession As Plumtree.Remote.PRC.IRemoteSession =
portletContext.GetRemotePortalSession()

 'get a collab factory and a project manager
 Dim collabFactory As ICollaborationFactory =
portalSession.GetCollaborationFactory()
 projectManager = collabFactory.GetProjectManager()

 Return projectManager

End Function

End Class
```

The `IProjectManager` also allows you to remove projects. The `removeProject` method takes in an `IProject` object and removes the associated project from the system.

Note: This action cannot be undone. No call to store is required.

Editing Collaboration Project Properties Using IDK Remote APIs

To query or modify Collaboration project metadata from a remote application, use the `IProject` interface in the IDK.

Each Collaboration project has its own set of objects and properties that are not shared with other projects. The `IProject` interface provides access to the following project properties:

Property Name	Description	API Access
ID	The object ID for the current project.	Read Only
Name	The name of the current project.	Read/Write
Description	The description for the current project.	Read/Write
Details	The URL to the details page for the current project.	Read Only

Property Name	Description	API Access
Created Date	The date the current project was created (this information might not be available).	Read Only
Last-Modified Date	The date the current project was last updated (this information might not be available).	Read Only
Owner ID	The user ID of the project owner.	Read Only
Access Level	The permissions for the current user (edit, delete, edit security).	Read Only
Start Date	The start date for the current project.	Read/Write
Status	The status of the current project (not started, 25% complete, 50% complete, 75% complete, or completed).	Read/Write

The `IProject` interface also allows you to modify user access levels. For details, see *Managing Collaboration Project Roles Using the IDK Remote API* on page 119.

To edit settings for an existing project, follow the steps below.

1. Create a session with the portal. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve the project ID. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55 *Querying Existing Collaboration Projects Using IDK Remote APIs* on page 104.
3. Edit the properties as shown in the code below.
   a) Retrieve an `IProjectManager`.
   b) Change the settings. The simplified sample code below changes the name, description, start date, and current status.
   c) Store the settings.

Java

```
...

//get the project
IProjectManager projectManager = getProjectManager(request,
response, out);
IProject project = projectManager.getProject(projectID);
```

```
//set the name, description, start date and status
project.setName() = "Updated Name";
project.setDescription() = "Updated description";
project.setStatus(ProjectStatus.TWENTY_FIVE_PERCENT_COMPLETED);

//you must call store to persist changes.
project.store();

...
```

.NET (C#)

```
...

//get the project
Plumtree.Remote.PRC.Collaboration.Project.IProjectManager
projectManager = GetProjectManager(Request,Response);
IProject project = GetProject(projectID);

//set project metadata
project.Name = "Updated Name";
project.Description = "Updated Description";
project.Status = ProjectStatus.TwentyFivePercentCompleted;

//you must call store to persist changes
project.Store();

...
```

.NET (VB)

```
...

'get the project
dim projectManager as
Plumtree.Remote.PRC.Collaboration.Project.IProjectManager =
GetProjectManager(Request, Response)
dim project as Plumtree.Remote.PRC.Collaboration.Project.IProject
 = projectManager.GetProject(-1)

'set project properties
project.Name = "Updated Name"
project.Description = "Updated Description
project.Status = ProjectStatus.TwentyFivePercentCompleted

'you must call store to persist changes
```

```
project.Store()

...
```

Managing Collaboration Project Roles Using the IDK Remote API

To assign users to Collaboration project roles from a remote application, use the `IRole` interface in the IDK.

The project role defines the actions that users are able to perform in a given project. The default Collaboration roles are defined as follows:

- **Project Leaders** have **Admin** control over project objects, which includes Read, Write, and Edit permission for all objects, as well as the ability to set role permissions (access levels) for each object.
- **Project Members** have **Write** access to project objects and can participate in the project. This role can create tasks, add documents, attach links, and check files in and out. The access privileges for this role are configured by the Project Leader.
- **Project Guests** have **Read** access to project objects. This role cannot create objects; it is intended for users who simply want to monitor projects but not participate actively. The access privileges for this role are configured by the Project Leader.

The `IRole` interface allows you to assign users to each project role. Each instance of `IRole` applies to a specific role within a specific project. You can assign roles to individual users, or to all the users that fulfill a specific role in a community. Once you have defined roles, you can modify the default access levels for a project, or for an individual object in the project (task list, task, folder, document, or discussion). For a list of access levels for Collaboration components, see *Collaboration Access Levels* on page 162.

The `IProject.getRole` method takes in the role type (guest, member or leader) and returns the associated IRole object for the project. To define roles for a project, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get the project ID and retrieve the associated object.
3. Get the role to assign.
4. Add users to the role.
5. Set any specific access level restrictions.

   Note: You must call `store` after making any changes or they will not be persisted.

Java

```
...

//get the project
IProjectManager projectManager = getProjectManager(request,
response, out);
IProject project = projectManager.getProject(projectID);

//get the guest role for the project
IRole guestrole = project.getRole(RoleType.GUEST);

//add the guests from a community and an individual user to the
role
guestrole.addCommunityMember(CommunityID, COMMUNITY_GUEST);
guestrole.addMember(UserID, USER);

//set the access level for discussions to write
guestrole.setAccessLevel(FunctionalArea.DISCUSSION,
AccessLevel.WRITE);

//call store to persist the changes
guestrole.store()

...
```

.NET (C#)

```
...

//get the project
Plumtree.Remote.PRC.Collaboration.Project.IProjectManager
projectManager = GetProjectManager(Request, Response);
IProject project = GetProject(projectID);

//get the guest role for the project
IRole guestrole = project.GetRole(RoleTypes.Guest);

//add the guests from a community and an individual user to the
role
guestrole.AddCommunityMember(CommunityID, CommunityGuest);
guestrole.AddMember(UserID, User);

//set the access level for discussions to write
guestrole.SetAccessLevel(FunctionalAreas.Discussion,
AccessLevels.Write);
```

```
//call store to persist the changes
guestrole.Store();

...
```

.NET (VB)

```
...

//get the project
dim projectManager As
Plumtree.Remote.PRC.Collaboration.Project.IProjectManager =
GetProjectManager(Request, Response)
dim project As IProject = GetProject(projectID)

//get the guest role for the project
dim guestrole As IRole = project.GetRole(RoleTypes.Guest)

//add the guests from a community and an individual user to the
role
guestrole.AddCommunityMember(CommunityID, CommunityGuest)
guestrole.AddMember(UserID, User)

//set the access level for discussions to write
guestrole.SetAccessLevel(FunctionalAreas.Discussion,
AccessLevels.Write)

//call store to persist the changes
guestrole.Store()

...
```

Managing Collaboration Subscriptions Using the IDK Remote API

To provide users with e-mail notifications when an activity occurs in Collaboration, use a subscription. The `IProjectManager` interface allows you to query current subscriptions, and subscribe and unsubscribe users to Collaboration projects. You can also subscribe users to individual project components.

The `IProjectManager` allows you to manage subscriptions for a project. To manage subscriptions for individual project components (folders, documents, task lists and discussions), use the associated component manager:

- The `ITaskListManager` interface allows you to query current subscriptions, and subscribe and unsubscribe users to collaboration task lists.

- The `IDocumentManager` interface allows you to query current subscriptions, and subscribe and unsubscribe users to collaboration folders and documents.
- The `IDiscussionManager` interface allows you to query current subscriptions, and subscribe and unsubscribe users to collaboration discussions.

The subscription methods within each interface are identical aside from the type of object passed into the method as a parameter. The sample code below uses the `ITaskListManager` subscription methods as an example.

To subscribe a user to a project or project component, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get the project or project component ID and retrieve the associated object manager.
3. Get the IDs of any users to be subscribed. (To retrieve the users that are currently subscribed, use the `getSubscribedUserIDs` method.)
4. Add users to a role if they are not already assigned one.

   Note: The users to be subscribed must be in at least GUEST role, and the calling user has to have ADMIN access to the project, or else an exception will be thrown.

5. Subscribe the users to the project or project component..

Java

```
...

//userID1 and userID2 are both valid int user IDs that have not
been added to any project roles
int[] validUserIDs = new int(userID1, userID2);
IRole guestRole = project.getRole(RoleType.GUEST);

//Add the two users to the GUEST role.
guestRole.addMember(userID1, MemberType.USER);
guestRole.addMember(userID2, MemberType.USER);
guestRole.store();

//Subscribe the two users to the task list.
//No store() needs to be called after the call to subscribeUsers.
tasklistManager.subscribeUsers(tasklist, validUserIDs);

...
```

.NET (C#)

```
...

//userID1 and userID2 are both valid int user IDs that have not
been added to any project roles
int[] validUserIDs = new int(userID1, userID2);
IRole guestRole = project.GetRole(RoleTypes.Guest);

//Add the two users to the GUEST role
guestRole.AddMember(userID1, MemberTypes.User);
guestRole.AddMember(userID2, MemberTypes.User);
guestRole.Store();

//Subscribe the two users to the project, set notifyForAllCreation
 setting to true,
//so the two subscribed users will get notified upon all new object
 creations in this project.
//No Store needs to be called on the project after the call to
SubscribeUsers.
projectManager.SubscribeUsers(project, validUserIDs, true);

...
```

About Remote Collaboration Discussion Operations

Collaboration discussions provide a virtual forum where project users hold online conversations on subjects of interest. The PRC Collaboration API in the IDK (com.plumtree.remote.prc.collaboration.discussion) allows you to manage discussions remotely and embed discussion functionality in your remote applications.

The PRC Collaboration API provides access to the following discussion functionality:

- **Collaboration Workspace**: Query, create, approve, and delete discussions and messages.
- **User Assignment**: Add users to discussions, and assign moderators who approve messages before they are published.
- **Subscriptions**: Provide users with e-mail notifications when a new subject of conversation is started or when a new message is added to an existing subject.

For details on using remote Collaboration discussion operations, see the following topics:

- *Querying Existing Collaboration Discussions Using IDK Remote APIs* on page 124
- *Creating Collaboration Discussions Using IDK Remote APIs* on page 130
- *Creating Collaboration Discussion Messages Using IDK Remote APIs* on page 132

- *Editing Collaboration Discussion Properties Using IDK Remote APIs* on page 134

Querying Existing Collaboration Discussions Using IDK Remote APIs

To query Collaboration discussions and messages from a remote application, use the `IDiscussionManager` interface in the IDK.

The PRC Collaboration API allows you to query existing collaboration discussions and messages. Results can be filtered in a variety of ways.

- To query for existing discussions in a project, use `IDiscussionManager.queryDiscussions` using the project instance.
- To query for existing messages in a discussion, use `IDiscussionManager.queryDiscussionMessages` using the discussion instance.
- To query for existing messages in a project, use `IDiscussionManager.queryDiscussionMessages` using the project instance.

For any of these queries, the `IDiscussionFilter`/`IDiscussionMessageFilter` interfaces allow you to set the following search options:

Maximum Results	Sets the maximum number of results returned. The default is to return all results.
Order-By Fields and Sort Order	Messages only. Sets the fields to be displayed with an order-by functionality, and sets the sort order (ascending or descending). The following fields support the order-by option: created, most recent, last modified, project, replies, and owner.
Security	Enables or disables the security filter that applies security to the result set with respect to the user that submitted the query. If the filter is enabled, the query result will only include objects for which the querying user has appropriate permission. The default is false (disabled); all objects matching the query criteria will be returned.
Result Filter: Status	Messages only. Limits queries by status (approved or unapproved).
Result Filter: Moderator Type	Messages only. Limits queries to those discussions for which the current user is a moderator, or extends the search to all discussions.

To query for discussions and messages, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.

2. Get the project or discussion ID and retrieve the associated object.
3. Create a new method to query for discussions or messages.
4. Get the Discussion Manager.
5. Create a query filter and execute the query as shown in the code samples below.

Java

```
...

//perform the search
IDiscussionManager discussionManager =
getDiscussionManager(request, response);
IDiscussionMessageFilter discussionMessageFilter =
discussionManager.createDiscussionMessageFilter();

//disable security checking on the returned objects against the
user who performs this query,
//so that all objects will be returned
messageFilter.setRestoreSecurity(false);

//hard-code the max results to 10; setting to 0 will return all
results
discussionMessageFilter.setMaximumResults(10);

//search for ALL messages; other options include searching for
APPROVED or UNAPPROVED messages
messageFilter.setMessageStatusType(DiscussionMessageStatusFilterType.ALL);

//optionally, set the query orders
//example below sorts returned messages by CREATED date in
descending order
DiscussionMessageQueryOrder messageQueryOrder = new
DiscussionMessageQueryOrder(DiscussionMessageAttribute.CREATED,
false);
messageFilter.setQueryOrders(new
DiscussionMessageQueryOrder(messageQueryOrder));

//execute the search and print out the results
IDiscussionMessage[] discussionMessages =
discussionManager.queryDiscussionMessages(project,
discussionMessageFilter);
if (discussionMessages.length > 0)
{
 %>
```

```
<tr>
 <td colspan="2">
 Search Results
 </td>
</tr>
<tr>
 <td>
 Discussion Message Name- Link to Discussion Message
 </td>
 <td>
 Discussion ID
 </td>
</tr>
<%
for (int i = 0; i < discussionMessages.length; i++)
{
 IDiscussionMessage discussionMessage = discussionMessages[i];
 int id = discussionMessage.getID();
 name = discussionMessage.getSubject();
 String url = discussionMessage.getDetailsURL();
 %>
<tr>
 <td>
 <%out.print("<a href=\"" + url + "\">" + name + "</a>");%>
 </td>
 <td>
 <%out.print(id);%>
 </td>
</tr>

...
```

.NET (C#)

```
...

//get the project ID out of session- this should never be null as
 it is added in the page load event
Plumtree.Remote.PRC.Collaboration.Project.IProject project =
(Plumtree.Remote.PRC.Collaboration.Project.IProject)
Session[SESSION_PROJECT_KEY];

//perform the search
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionManager
discussionManager = GetDiscussionManager(Request, Response);
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessageFilter
```

```
 discussionMessageFilter =
discussionManager.CreateDiscussionMessageFilter();

//disable security checking on the returned objects against the
user who performs this query,
//so that all objects will be returned
messageFilter.RestoreSecurity = false;

//hard-code the max results to 10
discussionMessageFilter.MaximumResults = 10;

//search for ALL messages; other options include searching for
Approved or Unapproved messages
messageFilter.MessageStatusType =
DiscussionMessageStatusFilterTypes.All;

//optionally, set the query orders
//example below sorts returned messages by CREATED date in
descending order
DiscussionMessageQueryOrder messageQueryOrder = new
DiscussionMessageQueryOrder(DiscussionMessageAttributes.Created,
 false);
messageFilter.setQueryOrders(new
DiscussionMessageQueryOrder(messageQueryOrder));

//execute the search and print out the results
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessage[]
 discussionMessages
= discussionManager.QueryDiscussionMessages(project,
discussionMessageFilter);
if (discussionMessages.Length > 0)
{
 %>
 <tr>
  <td colspan="2">
  Search Results
  </td>
 </tr>
 <tr>
  <td>
  Discussion Message Name- Link to Discussion Message
  </td>
  <td>
  Discussion ID
  </td>
```

```
 </tr>
 <%
 for (int i = 0; i < discussionMessages.Length; i++)
 {
  Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessage
 discussionMessage = discussionMessages[i];
  int id = discussionMessage.ID;
  String name = discussionMessage.Subject;
  String url = discussionMessage.DetailsURL;
  %>
  <tr>
   <td>
   <%Response.Write("<a href=\"" + url + "\">" + name + "</a>");%>

   </td>
   <td>
   <%Response.Write(id);%>
   </td>
  </tr>
  <%
 }
}
else
{
 Response.Write("No discussion messages found.");
}
...
```

.NET (VB)

```
...

'get the project ID out of session- this should never be Nothing
 as it is added in the page load event
dim project as Plumtree.Remote.PRC.Collaboration.Project.IProject
 =
CType(Session.Item(SESSION_PROJECT_KEY),Plumtree.Remote.PRC.Collaboration.Project.IProject)

'perform the search
dim discussionManager as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionManager
= GetDiscussionManager(Request, Response)
dim discussionMessageFilter as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessageFilter
 = discussionManager.CreateDiscussionMessageFilter()
```

```
//disable security checking on the returned objects against the
user who performs this query,
//so that all objects will be returned
messageFilter.RestoreSecurity = false

'hard-code the max results to 10; setting to 0 will return all
messages
discussionMessageFilter.MaximumResults = 10

//search for ALL messages; other options include searching for
Approved, or Unapproved messages
messageFilter.MessageStatusType =
DiscussionMessageStatusFilterTypes.All

'optionally, set the query orders
'example below sorts returned messages by CREATED date in
descending order
DiscussionMessageQueryOrder messageQueryOrder = new
DiscussionMessageQueryOrder(DiscussionMessageAttributes.Created,
 false)
messageFilter.setQueryOrders(new
DiscussionMessageQueryOrder(messageQueryOrder))

'execute the search and print out the results
dim discussionMessages() as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessage
= discussionManager.QueryDiscussionMessages(project,
discussionMessageFilter)
if discussionMessages.Length > 0 then
 %>
 <tr>
  <td colspan="2">
  Search Results
  </td>
 </tr>
 <tr>
  <td>
  Discussion Message Name- Link to Discussion Message
  </td>
  <td>
  Discussion Message ID
  </td>
 </tr>
 <%
 dim i as Integer
```

```
 for i = 0 to discussionMessages.Length -1
  dim discussionMessage as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessage
= discussionMessages(i)
  dim id as Integer = discussionMessage.ID
  dim name as String = discussionMessage.Subject
  dim url as String = discussionMessage.DetailsURL
  %>
  <tr>
   <td>
   <%Response.Write("<a href=""" & url & """>" & name & "</a>")
%>
   </td>
   <td>
   <%Response.Write(CStr(id)) %>
   </td>
  </tr>
  <%
 next
else
 Response.Write("No discussion messages found.")
end if
...
```

Creating Collaboration Discussions Using IDK Remote APIs

To create Collaboration discussions from a remote application, use the `IDiscussionManager` interface in the IDK.

The `IDiscussionManager` interface allows you to create new discussions in an existing project. To create a new discussion, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve the project ID (a source project must exist before you can create any Collaboration component objects). For details, see *Querying Existing Collaboration Projects Using IDK Remote APIs* on page 104.
3. Get the Discussion Manager and create a new discussion as shown in the code samples below.

   Note: You must call `store` after creating a discussion, or it will not be persisted.

Java

```
...
```

```
//create the discussion
IDiscussionManager discussionManager =
getDiscussionManager(request, response);
IDiscussion discussion =
discussionManager.createDiscussion(project, name, description);

//call store before asking for the ID
discussion.store();
String url = discussion.getDetailsURL();
int id = discussion.getID();
String detailsUrl = "DiscussionMessage.jsp?" +
SESSION_DISCUSSION_KEY + "=" + id;

...
```

.NET (C#)

```
...

//get the project ID out of session- this should never be null as
 it is added in the page load event
Plumtree.Remote.PRC.Collaboration.Project.IProject project =
(Plumtree.Remote.PRC.Collaboration.Project.IProject)
Session[SESSION_PROJECT_KEY];

//create the discussion
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionManager
discussionManager = GetDiscussionManager(Request, Response);
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussion discussion
 =
discussionManager.CreateDiscussion(project, name, description);

//call store before asking for the id.
discussion.Store();
String url = discussion.DetailsURL;
int id = discussion.ID;
String detailsUrl = "DiscussionMessage.aspx?" +
SESSION_DISCUSSION_KEY + "=" + id;

...
```

.NET (VB)

```
...

'get the project ID out of session- this should never be Nothing
```

```
 as it is added in the page load event
dim project as Plumtree.Remote.PRC.Collaboration.Project.IProject
 =
CType(Session.Item(SESSION_PROJECT_KEY),Plumtree.Remote.PRC.Collaboration.Project.IProject)

'create the discussion
dim discussionManager as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionManager
= GetDiscussionManager(Request, Response)
dim discussion as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussion =
discussionManager.CreateDiscussion(proje ct, name, description)

'call store before asking for the id.
discussion.Store()
dim url as String = discussion.DetailsURL
dim id as Integer = discussion.ID
dim detailsUrl as String = "DiscussionMessage.aspx?" &
SESSION_DISCUSSION_KEY & "=" & CStr(id )

...
```

Creating Collaboration Discussion Messages Using IDK Remote APIs

To create Collaboration discussion messages and reply messages from a remote application, use the `IDiscussion` interface in the IDK.

Messages and replies are structured in a tree hierarchy. Each message and reply in the hierarchy is represented by an instance of `IDiscussionMessage`.

- To create a new message thread in a discussion, use the `IDiscussion.createMessage` method to create a new `IDiscussionMessage` object.
- To create a reply to a message, use the associated `IDiscussionMessage.createDiscussionReplyMessage` method. This creates a child message (reply) that is also represented by an instance of `IDiscussionMessage`. You can create a reply at any level in the hierarchy (the root message, the latest message, or a specific message in the thread).

These methods use the same syntax, and take in the subject and body for the message. You can also set the description and approval status when you create a message. To create a new discussion message, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.

2. Retrieve the discussion ID and retrieve the discussion instance.
3. Create a new discussion message as shown in the code samples below.

   Note: You must call `store` after creating a discussion message, or it will not be persisted.

Java

```
...

//create the discussion message
IDiscussionMessage discussionMessage =
discussion.createDiscussionMessage(subject, body);

//call store before asking for the id.
discussionMessage.store();
int id = discussionMessage.getID();
String url = discussionMessage.getDetailsURL();

%>
<tr>
 <td>
 <%
 out.println("<a href=\"" + url + "\">Link to collab message " +
 id + "</a>");
 %>
 </td>
</tr>

...
```

.NET (C#)

```
...

//create the discussion message
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessage
discussionMessage = discussion.CreateDiscussionMessage(subject,
body);

//call store before asking for the id.
discussionMessage.Store();
int id = discussionMessage.ID;
String url = discussionMessage.DetailsURL;

%>
```

```
<tr>
 <td colspan="6">
 <%
 Response.Write("<a href=\"" + url + "\">Link to collab message
" + id + "</a>");
 %>
 </td>
</tr>
...
```

.NET (VB)

```
...

'create the discussion message
dim discussionMessage as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessage
= discussion.CreateDiscussionMessage(subject, body)

'call store before asking for the id.
discussionMessage.Store()
dim id as Integer = discussionMessage.ID
dim url as String = discussionMessage.DetailsURL

%>
<tr>
 <td colspan="6">
 <%
 Response.Write("<a href=""" & url & """>Link to collab message
" & Cstr(id) & "</a>")
 %>
 </td>
</tr>
...
```

Editing Collaboration Discussion Properties Using IDK Remote APIs

To query and modify Collaboration discussion and message properties from a remote application, use the `IDiscussion` and `IMessage` interfaces in the IDK.

The `IDiscussion` and `IMessage` interfaces allow you to change the subject, description, and body of the message before approving it. These interfaces provide access to the following metadata:

Property Name	Description	API Access
ID	The object ID for the current discussion or message.	Read Only

Property Name	Description	API Access
Name	The name of the current discussion or message.	Read/Write
Subject	Messages only. The subject of the current message	Read/Write
Description	The description for the current discussion or message.	Read/Write
Details	The URL to the details page for the current discussion or message.	Read Only
Created Date	The date the current discussion or message.was created (this information might not be available).	Read Only
Last-Modified Date	The date the current discussion or message.was last updated (this information might not be available).	Read Only
Approval Status	Messages only. The approval status of the message.	Read/Write
Owner ID	The user ID of the message owner.	Read Only
Moderators	Discussions only. The user IDs of the dicussion moderators (if any).	Read/Write
Access Level	The permissions for the defined roles on the current discussion or message (edit, delete, edit security). You can only change permissions for the folder if the default project security is set to false.	Read/Write
Permissions	The permissions for the current user on the current discussion or message (post, attach links, create, edit, edit security, delete).	Read Only
Discussion	Messages only. The discussion that contains the current message.	Read Only
Project	The parent project that contains the current discussion or message.	Read Only
Default Project Security	Whether or not default project security should be applied to the discussion or message. If default project security is enabled, you cannot change the security for the folder.	Read/Write

To modify discussion or message properties, follow the steps below.

1. Initiate a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get the Discussion Manager.
3. Get the discussion or message and modify properties as shown in the code samples below.

   Note: You must call `store` after making any changes or they will not be persisted.

Java

```
...

//get the discussion message
IDiscussionManager discussionManager =
getDiscussionManager(request, response);
IDiscussion discussionMessage =
discussionManager.getDiscussionMessage(messageID);

//update properties
discussionMessage.setName() = "Updated Name";
discussionMessage.setDescription() = "Updated Description";

//approve the message
discussionMessage.setApproved();

//call store to persist your changes
discussionMessage.store();

...
```

.NET (C#)

```
...

//get the discussion message
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionManager
discussionManager = GetDiscussionManager(Request, Response);
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessage
discussionMessage =
discussionManager.GetDiscussionMessage(messageID);

//update properties
discussionMessage.Name = "Updated Name";
discussionMessage.Description = "Updated Description";

//approve the message
```

```
discussionMessage.Approved = true;

//call store to persist your changes
discussionMessage.Store();

...
```

.NET (VB)

```
...

'get the discussion message
dim discussionManager as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionManager
= GetDiscussionManager(Request, Response)
dim discussion as
Plumtree.Remote.PRC.Collaboration.Discussion.IDiscussionMessage
= discussionManager.GetDiscussionMessage(messageID)

'update properties
discussionMessage.Name = "Updated Name"
discussionMessage.Description = "Updated Description"

'approve the message
discussionMessage.Approved = true

'call store to persist your changes
discussionMessage.Store()

...
```

About Remote Collaboration Document and Folder Operations

Documents are any kind of file uploaded to a Collaboration project, including spreadsheets, presentations, images, and PDF files. Documents are organized in a standard folder taxonomy. The PRC Collaboration API in the IDK (com.plumtree.remote.prc.collaboration.document) provides full access to documents and folders, allowing you to query, create, or modify these objects.

The PRC Collaboration API provides access to the following document functionality:

- **Collaboration Workspace**: Query, create, copy, modify, and delete documents and folders in Collaboration projects.
- **Folder Organization**: Create new folders and subfolders, copy existing folders and documents, and insert new documents.

- **Version Control**: Check in and check out documents, and query version information. The system retains a history of all versions.
- **Subscriptions**: Provide users with e-mail notifications when an activity occurs, such as deleting or modifying a document.

For details on using remote Collaboration document and folder operations, see the following topics:

- *Querying Collaboration Folders and Documents Using IDK Remote APIs* on page 138
- *Managing Collaboration Documents Using IDK Remote APIs* on page 141
- *Creating Collaboration Folders and Documents Using IDK Remote APIs* on page 143
- *Editing Collaboration Folder and Document Properties Using IDK Remote APIs* on page 147

Querying Collaboration Folders and Documents Using IDK Remote APIs

To query existing Collaboration documents and folders from a remote application, use the `IDocumentManager` interface in the IDK.

The PRC Collaboration API allows you to query existing folders or documents in a given project or folder. Results can be filtered in a variety of ways.

- To query for existing folders in a parent folder, use `IDocumentManager.queryFolders` using the parent folder instance. To return all folders in a project, get the top level document folder for the project using `getTopLevelFolder`, then query the top level document folder for all the document folders it contains.
- To query for existing documents in a folder, use `IDocumentManager.queryDocuments` using the folder instance.
- To query for existing documents in a project, use `IDocumentManager.queryDocuments` using the project instance.

For any of these queries, the `IDocumentFolderFilter`/`IDocumentFilter` interfaces allow you to set the following search options:

Maximum Results	Sets the maximum number of results returned. The default is to return all results.
Order-By Fields and Sort Order	Sets the fields to be displayed with an order-by functionality (name or last modified date), and sets the sort order (ascending or descending). The following fields support the order-by option for documents: name, author, project, parent folder, content type, size (bytes), timestamp, last modified, last check in user, check out user.

Security	Enables or disables the security filter that applies security to the result set with respect to the user that submitted the query. If the filter is enabled, the query result will only include objects for which the querying user has appropriate permission. The default is false (disabled); all objects matching the query criteria will be returned.
Result Filter: Checkin Status	Documents only. Limits queries by document status (checked in or checked out).

To query for folders or documents in a folder or project, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get the Document Manager.
3. Get the ID of the parent folder or project. To query all the folders in a project, get the top level document folder for the project using `getTopLevelFolder`.
4. Create a query filter and execute the query as shown in the code samples below.

The examples below query a project for all checked-out documents and order results by size and then last-modified date.

Java

```
...

//create a query filter
IDocumentFilter filter = manager.createDocumentFilter();

//set to only search for checked-out documents
filter.filterType = DocumentFilterTypes.checkedOutByCurrentUser;

//order results by size, then last modified date
DocumentQueryOrder sizeOrder = new
DocumentQueryOrder(DocumentFolderAttribute.NUMBYTES, true);
DocumentQueryOrder lastModifiedOrder = new
DocumentQueryOrder(DocumentFolderAttribute.LASTMODIFIED, true);
DocumentQueryOrder[] orders = new DocumentQueryOrder(sizeOrder,
lastModifiedOrder);
filter.setQueryOrders(orders);

//perform the query
IDocument[] foundDocuments = manager.queryDocuments(project,
filter);
```

```
...
```

.NET (C#)

```
...

//create a query filter
IDocumentFilter filter = documentManager.CreateDocumentFilter();

//set to only search for checked-out documents
filter.FilterType = DocumentFilterTypes.CheckedOutByCurrentUser;

//order results by size, then last modified date
DocumentQueryOrder sizeOrder = new
DocumentQueryOrder(DocumentAttributes.NumBytes, true);
DocumentQueryOrder lastModifiedOrder = new
DocumentQueryOrder(DocumentAttributes.LastModified, true);
DocumentQueryOrder[] orders = new DocumentQueryOrder(sizeOrder,
lastModifiedOrder);
filter.QueryOrders = orders;

//perform query
IDocument[] foundDocuments =
documentManager.QueryDocuments(project, filter);

...
```

.NET (VB)

```
...

'create a query filter
dim filter As IDocumentFilter =
documentManager.CreateDocumentFilter()

'set to only search for checked-out documents
filter.FilterType = DocumentFilterTypes.CheckedOutByCurrentUser;

'order results by size, then last modified date
dim sizeOrder As DocumentQueryOrder = new
DocumentQueryOrder(DocumentAttributes.NumBytes, true)
dim lastModifiedOrder As DocumentQueryOrder = new
DocumentQueryOrder(DocumentAttributes.LastModified, true)
dim orders As DocumentQueryOrder[] = new
DocumentQueryOrder(sizeOrder, lastModifiedOrder)
```

```
filter.QueryOrders = orders

'perform query
dim foundDocuments As IDocument[] =
documentManager.QueryDocuments(project, filter)

...
```

Managing Collaboration Documents Using IDK Remote APIs

To check in and check out Collaboration documents from a remote application, use the `IDocumentManager` interface in the IDK.

Only one person can check out a document at a time. To check out a document, the current user must have at least WRITE access to the document.

Checking in a document saves a new version of the document, increments the current version number, and makes a new entry in the document's history. When you check in a document, you can set the following properties:

Check in comment	Required. A string that will be added as the first check in comment for the new document.
Input stream	Required. An InputStream from which the contents of the new document can be read.
Language	The ISO 639-1 language code for the content in the document (for example, en for english). If null, the language is set to that of the current user.
Keep checked out	If set to true, the document will be checked in and automatically checked out again. The default is false.

To check out or check in documents, follow the steps below.

1. Initiate a PRC session. This example retrieves a login token using the `IPortletContext`; you can also use `IRemoteSession`. (For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.)
2. Get the Document Manager.
3. Get the document ID and retrieve the document.
   - To check out the document, pass in the document instance.
   - To check in the document, pass in the document instance, the check in comment, the input stream, language (optional) and whether or not the file should be checked out again as shown in the code samples below.

Java

```
...

IRemoteSession remoteSession =
portletContext.getRemotePortalSession();
IDocumentManager documentManager =
remoteSession.getCollaborationFactory().getDocumentManager();

//get the document
IDocument checkedOutDocument =
documentManager.getDocument(documentID);

//Open an inputstream for the document contents - this can be any
 InputStream
InputStream fileInputStream = new
FileInputStream("c:\\myNewDocument.doc");

//Check in the new version
documentManager.checkInDocument(checkedOutDocument, "updated
version of the document", fileInputStream, "en", false);

...
```

.NET (C#)

```
...

remoteSession = portletContext.GetRemotePortalSession();
documentManager =
remoteSession.GetCollaborationFactory().GetDocumentManager();

//get the document
IDocument checkedOutDocument =
documentManager.GetDocument(documentID);

//open an inputstream for the document contents - this can be any
 readable Stream
Stream fileInputStream = File.OpenRead("c:\\MyNewDocument.doc");

//check in the new version
documentManager.CheckInDocument(checkedOutDocument, "updated
version of the document", fileInputStream, "en", false);

...
```

.NET (VB)

```
...

dim documentManager As IDocumentManager
dim remoteSession As Plumtree.Remote.PRC.IRemoteSession
remoteSession = portletContext.GetRemotePortalSession()
documentManager =
remoteSession.GetCollaborationFactory().GetDocumentManager()

'get the document
IDocument checkedOutDocument =
documentManager.GetDocument(documentID)

'Open an inputstream for the document contents - this can be any
 readable Stream
dim fileInputStream As Stream =
File.OpenRead("c:\\MyNewDocument.doc")

'Check in the new version
documentManager.CheckInDocument(checkedOutDocument, "updated
version of the document", fileInputStream, "en", false)

...
```

Creating Collaboration Folders and Documents Using IDK Remote APIs

To create new Collaboration folders and documents from a remote application, use the `IDocumentManager` interface in the IDK.

The `IDocumentManager` interface allows you to create new folders, subfolders and documents. The `IDocumentManager` interface also allows you to copy existing documents and folders to target folders in any project.

- To create a new folder, create it and insert it into an existing folder. The `insertNewFolder` method takes in the target parent folder, the new folder, and an optional third parameter to set the new folder to inherit security from the parent folder.
- To create a new document, create it and insert it into an existing folder. The parameters in the `insertNewDocument` method allow you to set the following properties:

Check in comment	Required. A string that will be added as the first check in comment for the new document.
Input stream	Required. An InputStream from which the contents of the new document can be read.

Language	The ISO 639-1 language code for the content in the document (for example, "en" for english). If null, the language is set to that of the current user.
Inherit security	If set to true, the new document will inherit security from the parent folder.

Note: If there is already a document or subfolder in the parent folder with the same name as a supplied document, the name of the newly inserted document will be changed. For example, if a document is submitted with a name of report.doc and a file with this name already exists, the name of the new file will be changed to report_1.doc (or report_2.doc if report_1.doc also already exists). You can check the name of the returned `IDocument` to see if it differs from the name in the document parameter.

- To copy a folder or document, use the `IDocumentManager.copyToFolder` method and pass in the source folder, target folder, and document(s) or folder(s) to copy.
- To create a new folder or document, follow the steps below.

1. Create a PRC session. This example retrieves a login token using the `IPortletContext`; you can also use `IRemoteSession`. (For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.)
2. Get the `IDocumentManager`.
3. Get the ID of the parent folder. To insert a folder or document at the root level, get the top level document folder for the project using `getTopLevelFolder`.
4. Create the folder or document and insert it in the parent folder as shown in the code samples below.

These examples create a new folder in the top level folder and a new document in the new folder.

Java

```
...

IRemoteSession remoteSession =
portletContext.getRemotePortalSession();
IDocumentManager documentManager =
remoteSession.getCollaborationFactory().getDocumentManager();

//get top level folder in project to create new root folder
topLevelFolder =
documentManager.getTopLevelFolder(containingProject);

//create a new folder
newFolder = documentManager.createNewFolder("Example Name",
```

```
"Example Description");

//insert the folder, set to inherit security from the top level
folder
IDocumentFolder storedFolder =
documentManager.insertNewFolder(topLevelFolder, newFolder, true);

//create the document
IDocument newDocument = documentManager.createNewDocument("Example
 Document Name", "Example Document Description");

//set additional properties before inserting the document or they
 will not be persisted
newDocument.setAuthor("joe bloggs");
newDocument.setContentType("text/vnd.ms-word");

//open an inputstream for the document contents
InputStream fileInputStream = new
FileInputStream("c:\\report.doc");

//insert the document, inheriting the containing folder's security
documentManager.insertNewDocument(storedFolder, newDocument,
"initial check-in", fileInputStream, "en", true);

...
```

.NET (C#)

```
...

remoteSession = portletContext.GetRemotePortalSession();
documentManager =
remoteSession.GetCollaborationFactory().GetDocumentManager();

//get the top level folder for the project to create a root folder
IDocumentFolder rootFolder =
documentManager.GetTopLevelFolder(project);

//create a new folder
IDocumentFolder newFolder =
documentManager.CreateDocumentFolder("Example Name", "Example
Description");

//insert the new folder into the top level folder, set to inherit
 security
IDocumentFolder storedFolder =
```

```
documentManager.InsertNewFolder(topLevelFolder, newFolder,true);

//create the document
IDocument newDocument = documentManager.CreateNewDocument("Example
 Document Name", "Example Document Description");

//set additional properties before inserting the document or they
 will not be persisted
newDocument.Author = "joe bloggs";
newDocument.ContentType = "text/vnd.ms-word";

//open a Stream for the document contents
Stream fileInputStream = new FileStream("c:\\report.doc");

//insert the document, set to inherit security from the parent
folder
documentManager.InsertNewDocument(storedFolder, newDocument,
"initial check-in", fileInputStream, "en", true);

...
```

.NET (VB)

```
...

dim documentManager As IDocumentManager
dim remoteSession As Plumtree.Remote.PRC.IRemoteSession
remoteSession = portletContext.GetRemotePortalSession()
documentManager =
remoteSession.GetCollaborationFactory().GetDocumentManager()

'get the top level folder for the project to create a root folder
dim rootFolder As IDocumentFolder =
documentManager.GetTopLevelFolder(project)

'create the new folder
dim newFolder As IDocumentFolder =
documentManager.CreateDocumentFolder("Example Name", "Example
Description")

'Insert the new folder into the top level folder, set to inherit
 security
dim storedFolder As IDocumentFolder =
documentManager.InsertNewFolder(topLevelFolder, newFolder, true)

'create the document
```

```
dim newDocument As IDocument =
documentManager.CreateNewDocument("Example Document Name", "Example
 Document Description")

'set additional properties before inserting the document or they
 will not be persisted
newDocument.Author = "joe bloggs"
newDocument.ContentType = "text/vnd.ms-word"

'open a Stream for the document contents
dim fileInputStream as Stream = new FileStream("c:\\report.doc")

'insert the document, set to inherit security from the parent
folder
documentManager.InsertNewDocument(storedFolder, newDocument,
"initial check-in", fileInputStream, "en", true)

...
```

Editing Collaboration Folder and Document Properties Using IDK Remote APIs

To query and modify Collaboration folder and document properties from a remote application, use the `IDocumentFolder` and `IDocument` interfaces in the IDK.

The `IDocumentFolder` and `IDocument` interfaces allow you to update metadata and manipulate security settings. These interfaces provide access to the following metadata:

Property Name	Description	API Access
ID	The object ID for the current folder or document.	Read Only
Name	The name of the current folder or document.	Read/Write
Description	The description for the current folder or document.	Read/Write
Details	The URL to the details page for the current folder or document.	Read Only
Content Type	Documents only. The Content Type of the current document.	Read/Write
Content URL	The URL at which the document content can be downloaded.	Read Only
Path	The path to the document (as a string).	Read Only

Property Name	Description	API Access
Author	Documents only. The user ID of the author of the current document.	Read Only
Created Date	The date the current folder or document was created (this information might not be available).	Read Only
Last-Modified Date	The date the current folder or document was last updated (this information might not be available).	Read Only
Checked-Out Date	The date the document was last checked out. (Returns null if the document is not checked out.)	Read Only
Owner	The user ID of the folder or document owner.	Read Only
Access Level	The permissions for the defined roles on the current folder or document (edit, delete, edit security). You can only change permissions for the folder if the default project security is set to false.	Read/Write
Permissions	Documents only. The permissions for the current user on the current document (check out, attach links, copy, edit, edit security, delete).	Read Only
Parent Folder	The parent folder that contains the current folder or document.	Read Only
Project	The parent project that contains the current folder or document.	Read Only
Default Project Security	Whether or not default project security should be applied to the folder or document. If default project security is enabled, you cannot change the security for the folder.	Read/Write

To modify folder or document properties, follow the steps below.

1. Initiate a PRC session. This example retrieves a login token using the `IPortletContext`; you can also use `IRemoteSession`. (For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.)
2. Get the Document Manager.
3. Get the folder or document and modify properties as shown in the code samples below.

   Note: You must call store after making any changes or they will not be persisted.

Java

```
...

IRemoteSession remoteSession =
portletContext.getRemotePortalSession();
IDocumentManager documentManager =
remoteSession.getCollaborationFactory().getDocumentManager();

//get the document
IDocument document = documentManager.getDocument(documentID);

//set properties
document.setName() = "Updated Name";
document.setDescription() = "Updated Description ";

//update security
document.setAccessLevel(RoleType.MEMBER, AccessLevel.WRITE);

//call store to persist your changes
document.store();

...
```

.NET (C#)

```
...

remoteSession = portletContext.GetRemotePortalSession();
documentManager =
remoteSession.GetCollaborationFactory().GetDocumentManager();

//get the document
IDocument document = documentManager.GetDocument(documentID);

//set properties
document.Name = "Updated Name";
document.Description = "Updated Description";

//update security
document.SetAccessLevel(RoleTypes.Member, AccessLevels.Write);

//call store to persist your changes
document.Store();

...
```

.NET (VB)

```
...

dim documentManager As IDocumentManager
dim remoteSession As Plumtree.Remote.PRC.IRemoteSession
remoteSession = portletContext.GetRemotePortalSession()
documentManager =
remoteSession.GetCollaborationFactory().GetDocumentManager()

'get the document
dim document As IDocument = documentManager.GetDocument(documentID)

'set properties
document.Name = "Updated Name"
document.Description = "Updated Description"

'update security
document.SetAccessLevel(RoleTypes.Member, AccessLevels.Write)

//call store to persist your changes
document.Store()

...
```

About Remote Collaboration Task Operations

Collaboration tasks can be used to track workflow and process in a wide range of applications. Using the PRC Collaboration API in the IDK (com.plumtree.remote.prc.collaboration.tasklist), you can query, create, and modify tasks and task lists, as well as manage workflow and task dependencies.

Collaboration tasks define work that needs to be done, who will do it, when it should be completed, and how it relates to other tasks. Individual tasks can contain up to three levels of subtasks, and you can set dependencies between tasks in the same project. Task lists serve as structured to-do lists of tasks to be completed for a project or a phase of a project. Security for tasks is implemented through the associated task list.

The PRC Collaboration API provides access to the following task functionality:

- **Collaboration Workspace**: Query, create, copy, modify, and delete task lists, tasks and subtasks.
- **Workflow**: Add subtasks and task dependencies, assign users, and track task status and risk.
- **Subscriptions**: Provide users with e-mail notifications when an event occurs, such as when new task list is created or a task is assigned.

For details on using remote Collaboration task operations, see the following topics:

- *Querying Collaboration Tasks and Task Lists Using IDK Remote APIs* on page 151
- *Creating Collaboration Tasks and Task Lists Using IDK Remote APIs* on page 154
- *Editing Collaboration Task and Task List Properties Using IDK Remote APIs* on page 157
- *Managing Collaboration Task Workflow Using IDK Remote APIs* on page 160

Querying Collaboration Tasks and Task Lists Using IDK Remote APIs

To query Collaboration task lists, tasks and subtasks from a remote application, use the `ITaskListManager` interface in the IDK.

The PRC Collaboration API allows you to query existing collaboration tasks and task lists. Results can be filtered in a variety of ways.

- To query for task lists in a given project, use `ITaskListManager.queryTaskLists` using the project instance.
- To query for tasks in a given project, use `ITaskListManager.queryTasks` using the project instance.
- To query for tasks in a given task list, use `ITaskListManager.queryTasks` using the task list instance.

For any of these queries, the `ITaskListFilter`/`ITaskFilter` interfaces allow you to set the following search options:

Maximum Results	Sets the maximum number of results returned. The default is to return all results.
Order-By Fields and Sort Order	Sets the fields to be displayed with an order-by functionality, and sets the sort order (ascending or descending). The following fields support the order-by option: name, start date, end date, status, assigned to (tasks and subtasks only) and order (location of the task or subtask in the hierarchy - tasks and subtasks only).
Security	Enables or disables the security filter that applies security to the result set with respect to the user that submitted the query. If the filter is enabled, the query result will only include objects for which the querying user has appropriate permission. The default is false (disabled); all objects matching the query criteria will be returned.
Result Filter: Status	Limits queries by status (completed, pending or overdue).

Result Filter: User	Tasks and subtasks only. Limits queries to those tasks assigned to a specific user.
Result Filter: Assignment	Tasks and subtasks only. Limits queries to unassigned tasks.

To query for task lists, tasks and subtasks, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get the project or task list ID and retrieve the associated object.
3. Create a new method to query for task lists or tasks.
4. Get the Task List Manager.
5. Create a query filter as shown in the code samples below.
6. Execute the query.

The following examples query for task lists in a project. To create a query filter for tasks, replace `ITaskListFilter` with `ITaskFilter`.

Java

```
...

ITaskListFilter taskListFilter =
tasklistManager.createTaskListFilter();

//set the query to search for all task lists
//options can be set to search for task lists that contain only
pending tasks,
//completed tasks, or overdue tasks
taskListFilter.setCompletionType(TaskListCompletionFilterType.ALL);

//limit the return results to be 10
taskListFilter.setMaximumResults(10);

//disable security checking against the user who performs the
query,
//so that all objects will be returned
taskListFilter.setRestoreSecurity(false);

//use TaskListQueryOrder to sort the query result by NAME in
ascending order
TaskListQueryOrder taskListQueryOrder = new
```

```
TaskListQueryOrder(TaskListAttribute.NAME, true);
taskListFilter.setQueryOrders(new
TaskListQueryOrder(taskListQueryOrder));

//an array of ITaskList objects are returned from queryTaskLists();

// if no result is retrieved, a zero-length array will be returned
ITaskList[] retrievedTaskLists =
tasklistManager.queryTaskLists(project, taskListFilter);

...
```

.NET (C#)

```
...

ITaskListFilter taskListFilter =
tasklistManager.CreateTaskListFilter();

//set the query to search for all task list
//options can be set to search for task lists that contain only
pending tasks,
//completed tasks, or overdue tasks.
taskListFilter.CompletionType = TaskListCompletionFilterTypes.All;

//limit the return results to be 10
taskListFilter.MaximumResults = 10;

//disable security checking against the user who performs the
query,
//so that all objects will be returned
taskListFilter.RestoreSecurity = false;

//use TaskListQueryOrder to sort the query result by NAME in
ascending order
TaskListQueryOrder taskListQueryOrder = new
TaskListQueryOrder(TaskListAttributes.Name, true);
taskListFilter.QueryOrders = new
TaskListQueryOrder(taskListQueryOrder);

//an array of ITaskList objects are returned from queryTaskLists();

//if no result is retrieved, a zero-length array will be returned
ITaskList[] retrievedTaskLists =
tasklistManager.QueryTaskLists(project, taskListFilter);
```

```
...
```

.NET (VB)

```
...

dim taskListFilter As ITaskListFilter =
tasklistManager.CreateTaskListFilter();

'set the query to search for all task list
'options can be set to search for task lists that contain only
pending tasks,
'completed tasks, or overdue tasks.
taskListFilter.CompletionType = TaskListCompletionFilterTypes.All

'limit the return results to be 10
taskListFilter.MaximumResults = 10

'disable security checking against the user who performs the query,
'so that all objects will be returned
taskListFilter.RestoreSecurity = false

'use TaskListQueryOrder to sort the query result by NAME in
ascending order
dim taskListQueryOrder As TaskListQueryOrder = new
TaskListQueryOrder(TaskListAttributes.Name, true)
taskListFilter.QueryOrders(0) = taskListQueryOrder

'an array of ITaskList objects are returned from queryTaskLists()
'if no result is retrieved, a zero-length array will be returned
dim retrievedTaskLists() As ITaskList =
tasklistManager.QueryTaskLists(project, taskListFilter)

...
```

Creating Collaboration Tasks and Task Lists Using IDK Remote APIs

To create new Collaboration task lists, tasks and subtasks from a remote application, use the `ITask*` interfaces in the IDK.

The `ITaskListManager.createTaskList` method takes in a project ID, name and description, and returns an `ITaskList` object with a corresponding object ID and associated properties. In some cases, an existing task list can be used as a template. The `ITaskListManager.copyTaskLists` method allows you to copy existing task lists from one project to another.

Once a task list is created, you can create tasks and subtasks. The `ITaskList.createTask` method takes in a name, description, start date and end date, and returns an `ITask` object with a corresponding object ID and associated properties. The `ITask.createSubTask` method allows you to create subtasks using the same parameters. Subtasks are represented by an instance of `ITask`. For more information on subtasks, see *Managing Collaboration Task Workflow Using IDK Remote APIs* on page 160.

To create a new task list, follow the steps below.

1. Create a session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Retrieve the project ID (a source project must exist before you can create any Collaboration component objects). For details, see *Querying Existing Collaboration Projects Using IDK Remote APIs* on page 104.
3. Create a new task list as shown in the code samples below.
4. Use the new task list to add tasks and subtasks as shown in the code samples below.

Java

```
...

ICollaborationFactory collabFactory =
portalSession.getCollaborationFactory();
ITaskListManager tasklistManager =
collabFactory.getTaskListManager();

//create the task list
ITaskList tasklist = tasklistManager.createTaskList(project, name,
 description);

//call store() to persist the task list
tasklist.store();

//get the details URL and ID for the new task list
string url = tasklist.getDetailsURL();
int id = tasklist.getID();

//create the task
ITask task = tasklist.createTask(taskname, taskdescription,
startTime, endTime);

//call store to persist the task
task.store();
```

```
......
```

.NET (C#)

```
...

//get the project ID out of session- this should never be null as
 it is added in the page load event
Plumtree.Remote.PRC.Collaboration.Project.IProject project =
(Plumtree.Remote.PRC.Collaboration.Project.IProject)
Session[SESSION_PROJECT_KEY];

//create the task list
ITaskList tasklist = tasklistManager.CreateTaskList(project, name,
 description);

//call Store() to persist the task list
tasklist.Store();

//create the task
ITask task = tasklist.CreateTask(taskname, taskdescription,
startTime, endTime);

//call Store() to persist the task
task.Store();
...
```

.NET (VB)

```
...

name = "ExampleTaskList"
description = "ExampleTaskListDescription"

'get the project ID out of session- this should never be Nothing
 as it is added in the page load event
dim project as Plumtree.Remote.PRC.Collaboration.Project.IProject
 =
CType(Session.Item(SESSION_PROJECT_KEY),Plumtree.Remote.PRC.Collaboration.Project.IProject)

'create the task list
Dim tasklist As ITaskList = tasklistManager.CreateTaskList(project,
 name, description)

'call Store() to persist the task list
```

```
tasklist.Store()

'create the task
Dim tasklist As ITaskList = tasklistManager.CreateTaskList(project,
 name, description)

'call Store() to persist the task
task.Store()

...
```

Editing Collaboration Task and Task List Properties Using IDK Remote APIs

To query and modify Collaboration task list and task properties from a remote application, use the `ITaskList` and `ITask` interfaces in the IDK.

The `ITaskList` interface allows you to query and update metadata and manipulate security settings for task lists. The `ITask` interface allows you to assign users and manipulate key settings for individual tasks, including start date, due date, status and risk.

These interfaces provide access to the following metadata:

Property Name	Description	API Access
ID	The object ID for the current task list or task.	Read Only
Name	The name of the current task list or task.	Read/Write
Description	The description for the current task list.	Read/Write
Details	The URL to the details page for the current task list or task.	Read Only
Start Date	Tasks only. The assigned start date for the current task.	Read/Write
End Date	Tasks only. The assigned due date for the current task.	Read/Write
Created Date	The date the current task list or task was created (this information might not be available).	Read Only
Last-Modified Date	The date the current task list or task was last updated (this information might not be available).	Read Only
Owner	The user ID of the task list or task owner.	Read Only
Assigned Users	Tasks only. The IDs of the users assigned to the task.	Read/Write
Status	Tasks only. The status of the current task (pending, 25% complete, 50% complete, 75% complete, or completed).	Read/Write

Property Name	Description	API Access
Risk	Tasks only. The risk for the current task (high, low or medium).	Read/Write
Access Level	The permissions for defined roles on the current task list or task (edit, delete, edit security). You can only change permissions for the task list if the default project security is set to false.	Read/Write
Project	The ID of the project that contains the current task list.	Read Only
Default Project Security	Task lists only. Whether or not default project security should be applied to the task list. If default project security is enabled, you cannot change the security for the task list.	Read/Write
Task List	Tasks only. The ID of the task list that contains the current task.	Read Only
Level	Tasks only. The level of the task in the task hierarchy (0-3)	Read Only
Parent Task	Tasks only. The parent task for the current task (returns null if this is the root task).	Read Only
Subtasks	Tasks only. The subtasks of the current task.	Read/Write
Dependent Tasks	Tasks only. The tasks that are defined as dependent on the current task.	Read Only
Task Dependencies	Tasks only. The tasks for which the current task is defined as dependent.	Read/Write

To edit task list or task properties, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get the task or task list ID and retrieve the associated object.
3. Edit the task or task list properties as shown in the code samples below.

   Note: You must call `store` after making any changes or they will not be persisted.

The following examples modify the status and risk for a task and assign a user.

Java

```
...

//get the task
ITask task = tasklistManager.getTask(taskID);

//set properties
task.setStatus(TaskStatus.TWENTY_FIVE_PERCENT_COMPLETED);
task.setRisk(TaskRisk.MEDIUM);

//assign the task
task.addAssignedUser(userID);

//call store() to persist the task
task.store();

...
```

.NET (C#)

```
...

//get the task
ITask task = tasklistManager.GetTask(taskID);

//set properties
task.Status = TaskStatus.TwentyFivePercentCompleted;
task.Risk = TaskRisk.Medium;

//assign the task
task.AddAssignedUser(UserID);

//call Store() to persist the task
task.Store();

...
```

.NET (VB)

```
...

'get the task
dim task As ITask = tasklistManager.GetTask(taskID)

'set properties
task.Status = TaskStatus.TwentyFivePercentCompleted
```

```
task.Risk = TaskRisk.Medium

'assign the task
task.AddAssignedUser(UserID)

'call Store() to persist the task
task.Store()

...
```

Managing Collaboration Task Workflow Using IDK Remote APIs

To set dependencies between Collaboration tasks and create subtasks from a remote application, use the `ITask` interface in the IDK.

Almost every task can be broken up into detailed subtasks, and most tasks are related to other tasks in the same project. The PRC Collaboration API allows you to create up to three levels of subtasks, and set dependencies between tasks. You can also manipulate assignments, task status and risk settings. For details, see *Editing Collaboration Task and Task List Properties Using IDK Remote APIs* on page 157.

The `ITask` interface allows you to create subtasks for a given task and define the name, description, start date and due date. Subtasks are also represented by an instance of `ITask`. You can also manipulate dependencies between tasks in the same project using `ITask.addDependentTask`.

Note: Tasks with associated subtasks cannot be added as dependents.

To add a subtask to an existing task, follow the steps below.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get the task ID and retrieve the associated object.
3. Create a subtask as shown in the code samples below.

   Note: The `createSubTask` method creates a persisted task, so no call to `store` is required unless you modify properties after creating the subtask.

Java

```
...

//get the parent task
ITask task = tasklistManager.getTask(taskID);
```

```
//create the subtask
ITask subtask = parentTask.createSubTask(name, description,
startTime, endTime);

//to set additional properties, you must call store() to persist
 the subtask
subtask.AddAssignedUser(UserID);
subtask.store();

...
```

.NET (C#)

```
...

//get the parent task
ITask task = tasklistManager.GetTask(taskID);

//create a subtask
ITask subtask = parentTask.CreateSubTask(name, description,
startTime, endTime);

//To set additional properties, make sure that Store() is called
subtask.Risk = TaskRisks.Low;
subtask.Status = TaskStatuses.FiftyPercentCompleted;
subtask.Store();

...
```

.NET (VB)

```
...

'get the parent task
dim task As ITask = tasklistManager.GetTask(taskID)

'create a subtask
Dim subtask As ITask = parentTask.CreateSubTask(name, description,
 startTime, endTime)

'To set additional properties, make sure that Store() is called
subtask.Risk = TaskRisks.Low
subtask.Status = TaskStatuses.FiftyPercentCompleted
subtask.Store()

...
```

Collaboration Access Levels

The following matrix describes the default permissions for each access level and Collaboration component.

By default, all objects within a project inherit the default project security settings. You can override default project security and define object-level permissions for each role. If default project security is enabled for the object, you cannot change the access level settings. (As with any security system, permissions are defined in a hierarchy; Admin permissions include Edit permissions, Edit permissions include Write permissions, etc.)

Collaboration Component	Read	Write	Edit	Admin
Projects	View project	View project	Save project, modify project properties	Save project, modify project properties
Tasks	View Tasks	Create tasks, update task status	Modify task list and task properties, create task lists, assign owners	Delete task lists and tasks, configure task list security
Folders	View folders	Add documents	Modify folder properties, rename folders, copy folders, create folders	Modify folder properties, rename folders, copy folders, create folders
Documents	View files	Check files in and out, undo check-out	Modify file properties, copy files	Delete files, move files, configure file security
Discussions	View Discussions	Post messages, reply to messages	Modify discussion properties, create new discussions	Delete discussions and messages, configure discussion security, edit messages, approve or reject messages

About Remote Publisher APIs

The IDK's remote Publisher API (com.plumtree.remote.prc.content) provides programmatic access to many of the objects stored within AquaLogic Interaction Publisher. Use this remote programming

interface to embed Publisher components and functions into any web application delivered through the ALI framework.

You can publish simple HTML pages by creating and publishing either content items or Presentation Templates. The published HTML content is managed by Publisher—which stores the published pages in your portal's subnet, controls who can access them, and makes them searchable.

Note: Published content is not searchable by default. For more information on indexing and making content available for search, see *Publishing Publisher Content Items Using IDK Remote APIs* on page 176.

You can create published content portlets, which provide access to content items from within a portal page. The following portlet types can be created using the Publisher portlet templates and the Published Content Web Service:

- Announcement portlets display a formatted text message that Community Managers (or any user with an Editor or higher role for the folder) can edit and publish. The portlet can display one or more content items that include images, links to Web sites, or other content.
- News portlets display a list of links to published content items (articles). The articles themselves, which can be accessed by clicking a link in the News portlet, are launched on individual article pages.
- Community Directory portlets display articles in a folder structure.

For details on using remote Publisher APIs, see the following topics:

- *About Remote Publisher Folder Operations* on page 163
- *About Remote Publisher Content Item Operations* on page 166
- *About Remote Publisher Data Entry Template Operations* on page 177
- *About Remote Publisher Presentation Template Operations* on page 184
- *About Publisher Properties and Remote Property Operations* on page 190

For details on Publisher functionality, see the *Administrator Guide for AquaLogic Interaction Publisher* and the Publisher online help.

About Remote Publisher Folder Operations

Folders are containers that enable you to access and manage Publisher objects. The IDK remote Publisher API (com.plumtree.remote.prc.content.folder) allows you to query, create, or modify folders.

Folders organize content items, Presentation Templates, Data Entry Templates, image files, and selection lists into a structured hierarchy. Each folder inherits the security of its parent folder by

default. Folder security is managed in Publisher Explorer, and cannot be configured using the remote API.

Note: The Publisher folder organization is also used to create the folder structure on the web server where published content items are stored (the publishing target). Administrative users can override this default mirroring of the Publisher and web server folder structures through Publisher Explorer. The remote Publisher API does not provide access to this functionality.

For details on using remote Publisher folder operations, see the following topics:

- *Managing Publisher Folders Using IDK Remote APIs* on page 166
- *Creating Publisher Folders Using IDK Remote APIs* on page 164

The remote Publisher API provides methods to manage associations between folders and portlets, but the recommended way to associate a portlet with a folder is through ALI administration. For details on managing folder security, publishing targets and portlet association, see the *Administrator Guide for AquaLogic Interaction Publisher* or the Publisher online help.

Creating Publisher Folders Using IDK Remote APIs

To create folders using the IDK remote Publisher API, use the `IFolderManager` interface.

The `IFolderManager` interface includes the `createFolder` method with two parameters: the target folder and the new folder name. (To rename a folder, use `IFolder.setName`.) A folder name cannot have more than 255 characters. Publisher ignores capitalization and spaces in the characters. You must invoke `IFolder.store` to store the newly created folder.

Note: When you create a new folder with the IDK remote Publisher API, it inherits the security of its parent folder. To configure folder security, you must use Publisher Explorer.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get a Content Factory object using `IRemoteSession.getContentFactory`.
3. Get a Folder Manager, retrieve the target folder, and create the new folder as shown in the code samples below.
   - To retrieve the root folder, use `IFolderManager.getRootFolder`.
   - To retrieve the subfolder of the current folder, use `IFolderManager.getSubFolder`. (To retrieve all the folders in Publisher, retrieve the root folder with the `getRootFolder` method, then retrieve all subfolders with the `getSubFolder` method.)
   - To retrieve a folder by UUID, use `IFolderManager.getFolder`. (To obtain the UUID of a folder, use `IFolder.getUUID`.)
   - To retrieve a folder by path, use `IFolderManager.getFolderByPath`. (To obtain the path of a folder, use `IFolder.getPath`.)

The examples below create a folder in the root Publisher directory.

4. Call `store` to persist the folder; it will not exist in Publisher until this call is executed.

Java

```
...

// Get a folder manager for working with Publisher folders.
IFolderManager folderManager = factory.getFolderManager();

// Retrieve the Publisher root folder, which always exists.
IFolder rootFolder = folderManager.getRootFolder();

// Create a new folder.
IFolder subFolder = folderManager.createFolder(rootFolder, "Java
 Example - Support Incidents");
subFolder.store();

...
```

.NET (C#)

```
...

// Get a folder manager for working with Publisher folders.
IFolderManager folderManager = factory.GetFolderManager();

// Retrieve the Publisher root folder, which always exists.
IFolder rootFolder = folderManager.GetRootFolder();

// Create a new folder.
IFolder subFolder = folderManager.CreateFolder(rootFolder, ".NET
 Example - Support Incidents");
subFolder.Store();

...
```

.NET (VB)

```
...

' Get a folder manager for working with Publisher folders.
Dim folderManager As IFolderManager = factory.GetFolderManager()

' Get a reference to the root folder. It always exists.
Dim rootFolder As IFolder = folderManager.GetRootFolder()
```

```
' Create a new folder.
subFolder = folderManager.CreateFolder(rootFolder, "VB.NET Web
Example - DET")
subFolder.Store()

...
```

Managing Publisher Folders Using IDK Remote APIs

To copy, move and delete Publisher folders using the IDK remote Publisher API, use the `IFolderManager` interface.

The remote Publisher API provides access to basic folder management operations. Only users with a Producer or a higher role can execute these actions.

Note: If an error occurs while processing one of the objects in a folder, the operation will stop and all changes will be rolled back. Before you make the call, all folders must be persisted.

- To copy folder contents, use `IFolderManager.copyFolder`. This method copies all the objects from the source folder to the destination folder, including content items, Data Entry Templates, Presentation Templates, selection lists, and subfolders. You cannot copy a folder to any of its subfolders or to itself. When you copy published content items, the original content items remain published, but the copies are not automatically published. Also, copied content items are not searchable because the associated portlet IDs are not copied.
- To move folder contents, use `IFolderManager.moveFolder`. This method moves all objects from the source folder to the destination folder. You cannot move a folder to any of its subfolders or to itself. Other folders that refer to the folder being moved must be refreshed from the server.
- To delete a folder, use `IFolderManager.removeFolder`. This method deletes the folder, any subfolders, and all contents. If any content items in the folder have been published, set the content item to expire first, so that the published content is removed from the portal. If you do not expire published content items, users will still be able to access them.

  Note: When you remove a folder, it is deleted permanently. There is no undo.

About Remote Publisher Content Item Operations

Publisher content items are the primary publishable objects of Publisher. Content items can be objects created from a Data Entry Template, or they can be imported files and images. Content items can be created and managed remotely using the IDK remote Publisher API.

In content items created from a Data Entry Template, the Content Item Editor displays properties as editable fields, such as text boxes, defined selection lists, fields for downloading files, Boolean

True/False radio buttons, and so forth. Once the user has entered or selected values for the properties that make up a content item and saved the changes, the content item becomes a page on the preview site, with its data structured according to the Data Entry Template and its appearance and format defined by the associated Presentation Template. When you publish the content item, it becomes a web page or an object in the portal Knowledge Directory.

Note: A Data Entry Template can only be associated with a single Presentation Template, so a single content item cannot be formatted in multiple ways.

File content items are content items created by uploading images or documents to Publisher. The remote Publisher API does not allow you to create file content items, but you can include existing file content items as links or properties in other content items.

The IDK remote Publisher API provides access to the following document functionality:

- **Content Item Management**: Create, copy, modify, and delete content items.
- **Content Item Queries**: Query and retrieve content items, as well as content item metadata and properties. The `IContentItem` interface provides access to a large collection of information about individual content items. For details, see the IDK API documentation.
- **Publication**: Preview, publish, and expire content items.
- **Portlet Association**: Create associations between content items and portlets. The remote Publisher API provides methods to manage portlet association, but the recommended way to associate a portlet with a content item is through ALI administration.

For details on using remote Publisher content item operations, see the following topics:

- *Creating Publisher Content Items Using IDK Remote APIs* on page 167
- *Editing Publisher Content Items Using IDK Remote APIs* on page 172
- *Publishing Publisher Content Items Using IDK Remote APIs* on page 176

Creating Publisher Content Items Using IDK Remote APIs

To create new content items based on existing Data Entry Templates and Presentation Templates, use the `IContentItemManager` and `IContentItem` interfaces in the IDK.

The `IContentItemManager` interface allows you to create new content items from a Data Entry Template with defined properties and its associated Presentation Template. You can create multiple content items from a single Data Entry Template; all the content items will be formatted according to the attached Presentation Template.

Several processes are required before you can create a content item. The steps below provide a summary.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Create a Data Entry Template. For details, see *Creating Publisher Data Entry Templates Using IDK Remote APIs* on page 177.
3. Create a Presentation Template. For details, see *Creating Publisher Presentation Templates Using IDK Remote APIs* on page 185. If associated content items should not be included in the search index, set the searchable flag on the Presentation Template object to false.
4. Attach the Presentation Template to the Data Entry Template using the `IDataEntryTemplate.attachPresentationTemplate` method. You must call store to persist the attachment before creating a content item.
5. Once these processes are complete, you can create a content item and set values for the associated properties as shown in the sample code below. These examples retrieve the properties from the Data Entry Template as an array and set the associated values.

   Note: To persist a new content item, you must check it in to Publisher using `IContentItemManager.checkInItem`.

Java

```
...
// Get a content item manager object.
IContentItemManager ciManager = factory.getContentItemManager();

// Create a new content item.
IContentItem contentitem =
ciManager.createContentItem(det.getContainingFolder(), "Incident
 123", det);

// Retrieve the properties from the Data Entry Template.
IBaseProperty[] props = det.getAllProperties();

// Set the incident ID as an integer property.
contentitem.setIntegerPropertyValue(props[0], 123);

// Set the resolution status as a Boolean property.
contentitem.setBooleanPropertyValue(props[1], false);

// Set the severity level as a real property.
contentitem.setDoublePropertyValue(props[2], 0.70 );

// Set the date the incident was opened as a date property.
contentitem.setDatePropertyValue(props[3], new Date());
```

```
// Set the name of the customer as a text property.
contentitem.setTextLinePropertyValue(props[4], "Joe Customer");

// Set the incident description as a long text property
contentitem.setTextBlockPropertyValue(props[5], "Can't open the
application.");

// Link to another content item that contains customer information.
IContentItem customerData=
ciManager.getContentItem(det.getContainingFolder(), "Customer
Information");
contentitem.setItemReferencePropertyValue(props[6], customerData);

// Select the product from a selection list.
// When setting a selection list value, the passed-in string should

// exactly match one of the values in the selection list.
contentitem.setSelectionListPropertyValue(props[7], "ALI");

// Set the value of the log file property.
// The name of the file can be set to a different file name
FileInputStream log = new
FileInputStream("C:\uploads\logFile.txt");
contentItem.setFilePropertyValue(props[8], "logFile.txt", log);

// Set the image property value.
// The name of the image can be set to a different name
FileInputStream screenShot = new
FileInputStream("C:\uploads\screenshot.jpg");
contentItem.setImagePropertyValue(props[9], "screenshot.jpg",
screenShot);

// Store the content item by checking it in.
ciManager.checkInItem(contentitem, "New incident.");

...
```

.NET (C#)

```
...

// Get a content item manager object.
IContentItemManager ciManager = factory.GetContentItemManager();

// Create a new content item.
```

```
IContentItem contentitem =
ciManager.CreateContentItem(det.GetContainingFolder(), "Incident
 123", det);

// Retrieve the properties from the Data Entry Template.
IBaseProperty[] props = det.GetAllProperties();

// Set the incident ID as an integer property.
contentitem.SetIntegerPropertyValue(props[0], 123);

// Set the resolution status as a Boolean property.
contentitem.SetBooleanPropertyValue(props[1], false);

// Set the severity level as a real property.
contentitem.SetDoublePropertyValue(props[2], 0.70 );

// Set the date the incident was opened as a date property.
contentitem.SetDatePropertyValue(props[3], DateTime.Now);

// Set the name of the customer as a text property.
contentitem.SetTextLinePropertyValue(props[4], "Joe Customer");

// Set the incident summary as a long text property.
contentitem.SetTextBlockPropertyValue(props[5], "Can't open the
application.");

// Link to another content item that contains customer information.

IContentItem customerData =
ciManager.GetContentItem(det.GetContainingFolder(), "Customer
Information");
contentitem.SetItemReferencePropertyValue(props[6], customerData);

// Select the product from a selection list.
// When setting a selection list value, the passed-in string should
 exactly
// match one of the values in the selection list.
contentitem.SetSelectionListPropertyValue(props[7], "ALI");

// Set the value of the log file property.
// The name of the file can be set to a different file name
FileStream log = new FileStream("C:\uploads\logFile.txt",
FileMode.Open);
contentItem.setFilePropertyValue(props[8], "logFile.txt", log);
```

```
// Set the image property value.
// The name of the image can be set to a different name
FileStream screenShot = new FileStream("C:\uploads\screenshot.jpg",
 FileMode.Open);
contentItem.setImagePropertyValue(props[9], "screenshot.jpg",
screenShot);

// Store the content item by checking it in.
ciManager.CheckInItem(contentitem, "Automatically created an
article.");
...
```

.NET (VB)

```
...

'Get the Content Item Manager.
Dim contentFactory As IContentFactory =
remoteSession.GetContentFactory()
Dim contentItemManager As IContentItemManager =
contentFactory.GetContentItemManager()

' Create a new Content Item.
Dim contentItemName As String = CType(pnlMain.FindControl("Incident
 123"), TextBox).Text
Dim contentItem As IContentItem =
contentItemManager.CreateContentItem(dataEntryTemplate.GetContainingFolder,
 contentItemName, dataEntryTemplate)

' Retrieve the properties from the Data Entry Template.
Dim props() As IBaseProperty = det.GetAllProperties()

' Set the incident ID as an integer property.
contentitem.SetIntegerPropertyValue(props(0), 123)

' Set the resolution status as a Boolean property.
contentitem.SetBooleanPropertyValue(props(1), False)

' Set the severity level as a real property.
contentitem.SetDoublePropertyValue(props(2), 0.7)

' Set the date  the incident was opened as a date property.
contentitem.SetDatePropertyValue(props(3), DateTime.Now)

' Set the name of the customer as a text property.
contentitem.SetTextLinePropertyValue(props(4), "Joe Customer")
```

```
' Set the incident summary as a long text property.
contentitem.SetTextBlockPropertyValue(props(5), "Can't open the
application.")

' Link to another content item that contains customer information.
Dim customerData As IContentItem =
ciManager.GetContentItem(det.GetContainingFolder(), "Customer
Information")
contentitem.SetItemReferencePropertyValue(props(6), customerData)

' Select the product from a selection list.
' When setting a selection list value, the passed-in string should
 exactly
' match one of the values in the selection list.
contentitem.SetSelectionListPropertyValue(props(7), "ALI")

' Set the value of the file property.
' The name of the file can be set to a different file name
Dim log As FileStream = New FileStream("C:\uploads\logFile.txt",
 FileMode.Open)
contentItem.SetFilePropertyValue(props(8), "logFile.txt", log)

' Set the value of the image property
' The name of the image can be set to a different name
Dim screenShot As FileStream = New
FileStream("C:\uploads\screenshot.jpg", FileMode.Open)
contentItem.SetImagePropertyValue(props(9), "screenshot.jpg",
screenShot)

' Store the content item by checking it in.
ciManager.CheckInItem(contentitem, "Automatically created an
article.")
...
```

Editing Publisher Content Items Using IDK Remote APIs

To edit an existing Publisher content item, use the `IContentItemManager` and `IContentItem` interfaces in the IDK.

To edit an existing content item programmatically, follow the steps below. (To open a content item for editing in Publisher Explorer, use the URL returned by `IContentItem.getEditorURL`.)

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.

2. Retrieve the content item.
   - To retrieve a content item in a specific folder by name, use `IContentItemManager.getContentItem` and pass in the folder object and content item name.
   - To retrieve a content item by its UUID, use `IContentItemManager.getContentItem` and pass in the content item UUID.
   - To retrieve all the content items in a folder, use `IContentItemManager.getContentItems` and pass in the folder object.
   - To retrieve all the content items created from a specific Data Entry Template, use `IContentItemManager.getContentItems` and pass in the folder object and the Data Entry Template object.
3. Check out the content item using `IContentItemManager.checkOutItem`.

   Note: You must check out a content item before it can be edited.
4. Retrieve the properties of the content item using `IContentItem.getAllProperties` or the appropriate `getPropertyValue` method.
5. Set the new value for the property using the appropriate `setPropertyValue` method for the property.
6. Store your changes by checking in the content item using `IContentItem.checkInContentItem`.

These examples upgrade the severity of a support incident by updating two values in the associated content item. To find the appropriate property to update, this example checks for the property type.

Java

```
...
// check out the content item
itemManager.checkOutItem(contentItem);

//Retrieve all the properties of the content item.
IBaseProperty[] allProps = contentItem.getAllProperties();
for (int i = 0; i < allProps.length; i++)
{
 //Check for a text property.
 if (allProps[i] instanceof ITextBlockProperty)
 {
  //Retrieve the value of the text property.
  String oldValue =
```

```
contentItem.getTextBlockPropertyValue((ITextBlockProperty)allProps[i]);


  //Add a message to the existing string.

contentItem.setTextBlockPropertyValue((ITextBlockProperty)allProps[i],
 "Severity upgraded. " + oldValue);
 }

 //Check for a double property.
 else if (allProps[i] instanceof IDoubleProperty)
 {
  //Set the new value of the existing property

contentItem.setIntegerPropertyValue((IIntegerProperty)allProps[i],
 .9);
 }
}
//Check in the content item to maintain the persistence of the
updated values.
itemManager.checkInItem(contentItem, "Severity Upgraded");
...
```

.NET (C#)

```
...
//Check out the content item.
itemManager.CheckOutItem(contentItem);

//Retrieve all the properties on the content item.
IBaseProperty[] allProps = contentItem.GetAllProperties();
for (int i = 0; i < allProps.Length; i++)
{
   //Check for a text block property.
   if (allProps[i] is ITextBlockProperty)
   {
  String oldValue = "";
  if (contentItem.HasPropertyValue(allProps[i]))
  {
   //Retrieve the old value.
   oldValue = contentItem.GetTextBlockPropertyValue(allProps[i]);

  }
  //Set the new value.
  contentItem.SetTextBlockPropertyValue(allProps[i], "Severity
upgraded. " + oldValue);
```

```
 }
 //Check for an integer property.
 else if (allProps[i] is IDoubleProperty)
 {
  //Set the new value for the existing property
  contentItem.SetIntegerPropertyValue(allProps[i], .9);
  }
}
//Check in the content item to store the updated values.
itemManager.CheckInItem(contentItem, "Severity Upgraded");
...
```

.NET (VB)

```
...
' Check out the content item.
contentItemManager.CheckOutItem(contentItem)

' Retrieve all the properties on the content item.
Dim allProps As IBaseProperty() = contentItem.GetAllProperties
Dim i As Integer
For i = 0 To allProps.Length - 1
 ' Check for a text block property.
 If TypeOf allProps(i) Is ITextBlockProperty Then
  Dim oldValue As String = ""
  If contentItem.HasPropertyValue(allProps(i)) Then
   'Retrieve the old text block value if it has been set
previously.
   oldValue = contentItem.GetTextBlockPropertyValue(allProps(i))
  End If
  ' Set the new value.
  contentItem.SetTextBlockPropertyValue(allProps(i), ("Severity
upgraded. " & oldValue))
 Else
  ' Test if the property is an integer property.
  If TypeOf allProps(i) Is IIntegerProperty Then
   ' Set the new value for the existing property
   contentItem.SetIntegerPropertyValue(allProps(i), (.9))
  End If
End If
Next i

'Check in the content item to store the updated values.
contentItemManager.CheckInItem(contentItem, "Severity Upgraded")
...
```

Publishing Publisher Content Items Using IDK Remote APIs

To preview or publish an existing content item, use the `IContentItemManager` interface in the IDK.

When you publish a content item, Publisher formats the content according to the associated Presentation Template and saves the formatted files to the server directory specified in the publishing target. You can publish one content item at a time or multiple content items simultaneously. Publishable Presentation Templates can be published without a content item.

To preview or publish a content item, the following conditions must be met:

- The Data Entry Template from which the content item was created must have an attached Presentation Template.
- The folder in which the content item resides must have a configured previewing and/or publishing target. The target is the location on the web server where the processed content item will reside. Targets must be configured in Publisher Explorer and cannot be modified using the remote Publisher API.
- The user must have an Editor role or higher for the folder in which the content item resides (required for publication only).

Once a content item or Presentation Template has been published, it can be removed from the publishing target through expiration. When you set a content item to expire, Publisher removes the published content files and all included images and files from the publishing target and from the Knowledge Directory and search index. Any published content items that reference the expired content item are refreshed and republished. The content item itself is not deleted from Publisher.

For more information about publication and expiration, see the *Administrator Guide for AquaLogic Interaction Publisher* or the Publisher online help.

The options below also apply to publishable Presentation Templates; the associated methods can be found in the `IPresentationTemplateManager` interface.

- To preview a content item, use `IContentItemManager.previewContentItem` and pass in the content item object. To retrieve the preview URL, use `IContentItem.getPreviewURL`.
- To publish a single content item use `IContentItemManager.publishContentItem` and pass in the content item object. (To check if a content item has an associated Presentation Template and is ready for publication, use `IContentItem.isContentItemPublishable`.) To retrieve the publication URL, use `IContentItem.getPublishURL`.
- To publish all the content items in folder, use `IContentItemManager.publishContentItems` and pass in the folder object. If

subfolders should also be published, set a second parameter to true. If a content item in a folder fails to publish, it does not cause complete failure. Publisher will publish all the content items in the folder except for the ones with errors. If any content items contained in the specified folder are not published, an exception will be thrown.

- To expire a content item, use `IContentItemManager.expireContentItem` and pass in the content item object.
- To expire a folder, use `IContentItemManager.expireContentItems` and pass in the folder object.
- To republish expired content item, use `IContentItemManager.unexpireContentItem` and pass in the content item object.

About Remote Publisher Data Entry Template Operations

Data Entry Templates consist of a set of properties that define the fields available for creating a content item. The IDK remote Publisher API (com.plumtree.remote.prc.content.dataentrytemplate) allows you to create and manage Data Entry Templates, add and remove properties, and attach and detach Presentation Templates.

The fields available for creating a content item appear in the Content Item Editor as editable fields, such as text boxes, selection lists, fields for downloading files, Boolean radio buttons, etc. When you publish content items created from Data Entry Templates, Publisher uses the associated Presentation Template to define how the property values are displayed to the end user.

For details on using remote Publisher Data Entry Template operations, see the following topics:

- *Creating Publisher Data Entry Templates Using IDK Remote APIs* on page 177
- *Creating Publisher Properties Using IDK Remote APIs* on page 190
- *Creating Publisher Selection Lists Using IDK Remote APIs* on page 191

To publish content items created from a Data Entry Template, you must attach a Presentation Template. The Presentation Template determines the layout of the published content item. Only one Presentation Template can be attached to a Data Entry Template at a time. If a Presentation Template is already attached to the Data Entry Template, you must first detach the existing Presentation Template and store the detachment before you can attach a different Presentation Template. To add or remove properties from an existing Data Entry Template, use the `IDataEntryTemplate.addProperty` and `removeProperty` methods. Changes will not affect existing content items created from the Data Entry Template.

Creating Publisher Data Entry Templates Using IDK Remote APIs

To create or retrieve Publisher Data Entry Templates from a remote application, use the IDK `IDataEntryTemplateManager` interface.

Creating a Data Entry Template involves assembling properties for each variable component of a content item. Additional properties can be added to existing Data Entry Templates at any time, but these changes will not affect existing content items. To create a Data Entry Template, follow the steps below.

Note: Only users with Producer or higher roles can create and edit Data Entry Templates.

1. Create a PRC session. For details, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
2. Get a Content Factory object using `IRemoteSession.getContentFactory`.
3. Get a Data Entry Template Manager and create the new Data Entry Template.
4. Add properties for each variable component. For a list of properties, see *Publisher Property Types* on page 193.
5. Call `store` to persist the Data Entry Template; it will not exist in Publisher until this call is executed.

Java

```
...

// Get manager objects needed for creating a Data Entry Template.
IDataEntryTemplateManager detManager =
factory.getDataEntryTemplateManager();
IPropertyManager propertyManager = factory.getPropertyManager();
ISelectionListManager selectionListManager =
factory.getSelectionListManager();

// Create a Data Entry Template object.
IDataEntryTemplate det =
detManager.createDataEntryTemplate(supportfolder, "Technical
Support Incident Data Entry Template");

// Add an integer property.
IIntegerProperty incident =
propertyManager.createIntegerProperty("Incident ID", "The ticket
 number for this incident.");
det.addProperty(incident);

// Add a Boolean property.
IBooleanProperty resolved =
propertyManager.createBooleanProperty("Resolved", "True if the
incident has been closed; false if it is still open.");
det.addProperty(resolved);
```

```
// Add a real property.
IDoubleProperty severity =
propertyManager.createDoubleProperty("Severity", "Percentage
stating how severe the incident is.");
det.addProperty(severity);

// Add a date property.
IDateProperty opened = propertyManager.createDateProperty("Date
Opened", "The date and time the incident was reported.");
det.addProperty(opened);

// Add a text property.
ITextLineProperty customerName =
propertyManager.createTextLineProperty("Customer Name", "The name
 of the customer.");
det.addProperty(customerName);

// Add a long text property.
ITextBlockProperty description =
propertyManager.createTextBlockProperty("Incident Description",
"A summary of the problem the customer is experiencing.");
det.addProperty(description);

// Add an item reference property.
IItemReferenceProperty customerData =
propertyManager.createItemReferenceProperty("Customer Information",
 "Links to information about the customer such as the contact and
 their phone number.");
det.addProperty(customerData);

// Add a selection list property. The selection list must be
persisted before it is added.
String[] listValues = {"ALI", "Search", "Collaboration",
"Publisher", "Analytics"};
ISelectionList selectionList =
selectionListManager.createSelectionList(supportfolder, "Products
 Selection List", listValues);
selectionList.store();
ISelectionListProperty products =
propertyManager.createSelectionListProperty("Product", "The product
 that the support incident is opened against", selectionList);
det.addProperty(products);

// Add a file property.
```

```
IFileProperty log = propertyManager.createFileProperty("Log File",
 "The log file created when the problem occurred.");
det.addProperty(log);

// Add an image property.
IImageProperty screenShot =
propertyManager.createImageProperty("Screen Shot", "A screen shot
 of the problem.");
det.addProperty(screenShot);

// Store the Data Entry Template and all the properties associated
 with it.
det.store();

return det;

...
```

.NET (C#)

```
...

// Get manager objects needed for creating a Data Entry Template.
IDataEntryTemplateManager detManager =
factory.GetDataEntryTemplateManager();
IPropertyManager propertyManager = factory.GetPropertyManager();
ISelectionListManager selectionListManager =
factory.GetSelectionListManager();

// Create a Data Entry Template object.
IDataEntryTemplate det =
detManager.CreateDataEntryTemplate(supportfolder, "Support Incident
 Data Entry Template");

// Add an integer property.
IIntegerProperty incident =
propertyManager.CreateIntegerProperty("Incident ID", "The ticket
 number for this incident.");
det.AddProperty(incident);

// Add a Boolean property
IBooleanProperty resolved =
propertyManager.CreateBooleanProperty("Resolved", "True if the
incident has been closed; false if it is still open.");
det.AddProperty(resolved);
```

```
// Add a real property.
IDoubleProperty severity =
propertyManager.CreateDoubleProperty("Severity", "Percentage
stating how severe the incident is.");
det.AddProperty(severity);

// Add a date property.
IDateProperty opened = propertyManager.CreateDateProperty("Date
Opened", "The date and time the incident was reported.");
det.AddProperty(opened);

// Add a text property.
ITextLineProperty customerName =
propertyManager.CreateTextLineProperty("Customer Name", "The name
 of the customer.");
det.AddProperty(customerName);

// Add a long text property.
ITextBlockProperty description =
propertyManager.CreateTextBlockProperty("Incident Description",
"A summary of the problem the customer is experiencing.");
det.AddProperty(description);

// Add an item reference property.
IItemReferenceProperty customerData =
propertyManager.CreateItemReferenceProperty("Customer Information",
 "Links to information about the customer such as the contact and
 their phone number.");
det.AddProperty(customerData);

// Add a selection list property. The selection list must be
persisted before it is added.
string[] listValues = {"Portal", "Search", "Collaboration",
"Publisher", "Analytics"};
ISelectionList selectionList =
selectionListManager.CreateSelectionList(supportfolder, "Products
 Selection List", listValues);
selectionList.Store();
ISelectionListProperty products =
propertyManager.CreateSelectionListProperty("Product", "The product
 that the support incident is opened against", selectionList);
det.AddProperty(products);

// Add a file property.
IFileProperty log = propertyManager.CreateFileProperty("Log File",
```

```
 "The log file created when the problem occurred.");
det.AddProperty(log);

// Add an image property.
IImageProperty screenShot =
propertyManager.CreateImageProperty("Screen Shot", "A screen shot
 of the problem.");
det.AddProperty(screenShot);

// Store the Data Entry Template and all the properties associated
 with it.
det.Store();

return det;
}

...
```

.NET (VB)

```
...

' Get manager objects needed for creating a Data Entry Template.
Dim detManager As IDataEntryTemplateManager =
factory.GetDataEntryTemplateManager
Dim propertyManager As IPropertyManager =
factory.GetPropertyManager
Dim selectionListManager As ISelectionListManager =
factory.GetSelectionListManager

' Create a Data Entry Template object.
det = detManager.CreateDataEntryTemplate(supportfolder, "Support
 Incident Data Entry Template")

' Add an integer property.
Dim incident As IIntegerProperty =
propertyManager.CreateIntegerProperty("Incident ID", "The ticket
 number for this incident.")
det.AddProperty(incident)

' Add a Boolean property.
Dim resolved As IBooleanProperty =
propertyManager.CreateBooleanProperty("Resolved", "True if the
incident has been closed false if it is still open.")
det.AddProperty(resolved)
```

```
' Add a double property.
Dim severity As IDoubleProperty =
propertyManager.CreateDoubleProperty("Severity", "Percentage
stating how severe the incident is.")
det.AddProperty(severity)

' Add a date property.
Dim opened As IDateProperty =
propertyManager.CreateDateProperty("Date Opened", "The date and
time the incident was reported.")
det.AddProperty(opened)

' Add a text property.
Dim customerName As ITextLineProperty =
propertyManager.CreateTextLineProperty("Customer Name", "The name
 of the customer.")
det.AddProperty(customerName)

' Add a long text property.
Dim description As ITextBlockProperty =
propertyManager.CreateTextBlockProperty("Incident Description",
"A writeup summarizing the problem the customer is experiencing.")
det.AddProperty(description)

' Add an item reference property.
Dim customerData As IItemReferenceProperty =
propertyManager.CreateItemReferenceProperty("Customer Information",
 "Links to information about the customer such as the contact and
 their phone number.")
det.AddProperty(customerData)

' Add a selection list property. The selection list must be
persisted before it is added.
' This example checks to see if the selection list already exists
 before creating it.
Dim selectionList As ISelectionList = Nothing
Dim selectionLists() As ISelectionList =
selectionListManager.GetSelectionLists(supportfolder)
Dim sl As ISelectionList
For Each sl In selectionLists
If (sl.Name.Equals("Products Selection List")) Then
 selectionList = sl
End If
Next
```

```
If (selectionList Is Nothing) Then
 Dim listValues() As String = {"Portal", "Search", "Collaboration",
 "Content Server", "Analytics"}
 selectionList = selectionList.CreateSelectionList(supportfolder,
 "Products Selection List", listValues)
 selectionList.Store()
End If

Dim products As ISelectionListProperty =
propertyManager.CreateSelectionListProperty("Product", "The product
 that the support incident is opened against", SelectionList)
det.AddProperty(products)

' Add a file property.
Dim log As IFileProperty = propertyManager.CreateFileProperty("Log
 File", "The log file created when the problem occurred.")
det.AddProperty(log)

' Add an image property.
Dim screenShot As IImageProperty =
propertyManager.CreateImageProperty("Screen Shot", "A screen shot
 of the problem.")
det.AddProperty(screenShot)

' Store the Data Entry Template and all the properties associated
 with it.
det.Store()

Return det

...
```

About Remote Publisher Presentation Template Operations

Presentation Templates define how content items are formatted and displayed when they are previewed or published. Presentation Templates can be created and managed remotely using the IDK remote Publisher API.

Presentation Templates can include basic formatting, images, standard boilerplate text, and programming logic to display information based on the value of specific content item properties. Presentation Templates include Publisher (PCS) tags that are replaced with values and variables from Publisher. Presentation Templates can output any type of text files, including HTML, JavaScript, CSS, and even Java source files. For more information about PCS tags, refer to the *Administrator Guide for AquaLogic Interaction Publisher*.

You can also create stand-alone Presentation Templates that are publishable without a content item or Data Entry Template. This is useful when the content is static (does not contain variables or properties/PCS tags). The IPresentationTemplateManager interface allows you to preview and publish Presentation Templates using the same methods as IContentItemManager. For details on previewing and publishing, see *About Remote Publisher Content Item Operations* on page 166. For more information about independently publishable Presentation Templates, see the *Administrator Guide for AquaLogic Interaction Publisher* and the *Publisher Templating Specification*.

Only users with Producer or higher roles can create and retrieve Presentation Templates.

For details on using remote Publisher Presentation Template operations, see *Creating Publisher Presentation Templates Using IDK Remote APIs* on page 185.

Creating Publisher Presentation Templates Using IDK Remote APIs

To create and validate Publisher Presentation Templates, use the `IPresentationTemplateManager` interface in the IDK.

Most of the code in a Presentation Template is markup that defines how the content item will be laid out and formatted. Use pcS tags to insert Publisher content. PCS tags are replaced by the referenced property during the publishing operation. All Publisher tags require the following:

- An opening tag and a closing tag (that is, <pcs:…> </pcs:…>).
- The opening tag must include an "expr" parameter that references a property name. The property name must be enclosed in single or double quotation marks (property names are not case-sensitive).

For more details on PCS tags, see Publisher Templating Specification.

After you have created the markup for a Presentation Template, you must create and store it using `IPresentationTemplateManager` so it can be used to publish content items. If published content items should not be included in the search index, set the searchable flag on the Presentation Template object to false.

Note: The `IPresentationTemplateManager.validateTemplateText` method can be used to find any pcS syntax errors. It is a best practice to execute this method after creating a new Presentation Template, before it is used to publish a content item.

These examples create a Presentation Template that displays values in an HTML table. The default file type is HTML; you can use the `IPresentationTemplate` interface to set the file extension to any value that is valid for a basic text file, including .js, .jsp, .asp, .xml, and .txt.

Java

```
...

StringBuffer templateBuffer = new StringBuffer("");
templateBuffer.append(" <table border='1' \n")
.append("<tr> <td> <b> Incident ID </b> </td> <td> <pcs:value
expr='Incident_ID'></pcs:value> </td> </tr>\n")
.append("<tr> <td> <b>Created By </b> </td> <td> <pcs:value
expr='createdBy'></pcs:value> on <pcs:value expr='created'
format='MMMM d, yyyy'> </pcs:value> </td> </tr>\n")
.append("<tr> <td> <b>Resolved </b> </td> <td> <pcs:value
expr='Resolved'></pcs:value> </td> </tr>\n")
.append("<tr> <td> <b>Severity </b> </td> <td> <pcs:value
expr='Severity'> </pcs:value> </td> </tr>\n")
.append("<tr> <td> <b>Date Opened </b></td><td> <pcs:value
expr='Date_Opened' format='MM, dd, yyyy'></pcs:value></td>
</tr>\n")
.append("<tr> <td> <b>Customer Name </b> </td> <td> <pcs:value
expr='Customer_Name'></pcs:value> </td> </tr>\n")
.append("<tr> <td> <b>Customer Information </b> </td> <td>
<pcs:value
expr='Customer_Information.name'></pcs:value></td></tr>\n")
.append("<tr> <td> <b>Product </b> </td> <td> <pcs:value
expr='Product'></pcs:value></td> </tr>\n")
.append(" <tr> <td> <b>Incident Description </b> </td> <td>
<pcs:value expr='Incident_Description'></pcs:value> </td> </tr>\n")
.append("<tr> <td> <b>Log File </b> </td> <td> <pcs:value
expr='Log_File.name'></pcs:value></td> </tr> \n")
.append(" <tr> <td> <b>Screen Shot </b> </td><td> <pcs:value
expr='Screen_Shot.name'></pcs:value> </td> </tr> \n")
.append(" </table>

// Create the Presentation Template.
presentationTemplate =
ptManager.createPresentationTemplate(dataEntryTemplate.getContainingFolder(),
"Customer Support Incident Presentation Template",
templateBuffer.toString());

// Store the Presentation Template.
presentationTemplate.store();

// Validate the template text before using it to publish
// content items.
String[] errorsInText = ptManager.validateTemplateText(pt,
```

```
pt.getTemplateText());
if(errorsInText.length != 0)
{
System.out.println("PCS validation returned the following errors");
for(int i = 0; i < errorsInText.length; i++)
{
System.out.println("error[" + i + "]: " + errorsInText[i]); }
}
}
...
```

.NET (C#)

```
...

StringBuilder templateBuffer = new StringBuilder();
templateBuffer.Append("<table border='1'>\n")
.Append("<tr><td><b>Incident ID</b></td><td><pcs:value
expr='Incident_ID'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Created By</b></td><td><pcs:value
expr='createdBy'></pcs:value> on <pcs:value expr='created'
format='MMMM, dd, yyyy'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Resolved</b></td><td><pcs:value
expr='Resolved'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Severity</b></td><td><pcs:value
expr='Severity'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Date Opened</b></td><td><pcs:value
expr='Date_Opened' format='MM, dd, yyyy'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Customer Name</b></td><td><pcs:value
expr='Customer_Name'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Customer Information</b></td><td><pcs:value
expr='Customer_Information.name'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Product</b></td><td><pcs:value
expr='Product'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Incident Description</b></td><td><pcs:value
expr='Incident_Description'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Log File</b></td><td><pcs:value
expr='Log_File.name'></pcs:value></td></tr>\n")
.Append("<tr><td><b>Screen Shot</b></td><td><pcs:value
expr='Screen_Shot.name'></pcs:value></td></tr>\n")
.Append("</table>\n");

// Create the Presentation Template.
IPresentationTemplate presentationTemplate =
ptManager.CreatePresentationTemplate(dataEntryTemplate.GetContainingFolder(),
 "Customer Support Incident Presentation Template",
```

```
templateBuffer.ToString());

// Store the Presentation Template.
presentationTemplate.Store();

// Validate the template text before using it to publish content
 items.
String[] errorsInText = ptManager.ValidateTemplateText(pt,
pt.TemplateText);
if(errorsInText.Length != 0)
{
Console.WriteLine("PCS validation returned the following errors");
for(int i = 0; i < errorsInText.Length; i++)
{
Console.WriteLine("error[" + i + "]: " + errorsInText[i]);
}
}
...
```

.NET (VB)

```
...
Dim templateBuffer As StringBuilder = New StringBuilder
templateBuffer.Append("<table border='1'>" +
System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Incident
ID</b></td><td><pcs:value
expr='Incident_ID'></pcs:value></td></tr>"
+ System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Created By</b></td><td><pcs:value
 expr='createdBy'></pcs:value>
on <pcs:value expr='created' format='MMMM d,
yyyy'></pcs:value></td></tr>" + System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Resolved</b></td><td><pcs:value
 expr='Resolved'></pcs:value></td></tr>"
+ System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Severity</b></td><td><pcs:value
 expr='Severity'></pcs:value></td></tr>"
+ System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Date
Opened</b></td><td><pcs:value expr='Date_Opened'
format='MMMM d, yyyy'></pcs:value></td></tr>" +
System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Customer
Name</b></td><td><pcs:value
expr='Customer_Name'></pcs:value></td></tr>"
```

```
+ System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Customer
Information</b></td><td><pcs:value
expr='Customer_Information.name'></pcs:value></td></tr>"
+ System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Product</b></td><td><pcs:value
 expr='Product'></pcs:value></td></tr>"
+ System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Incident
Description</b></td><td><pcs:value
expr='Incident_Description'></pcs:value></td></tr>"
+ System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Log File</b></td><td><pcs:value
 expr='Log_File.name'></pcs:value></td></tr>"
+ System.Environment.NewLine)
templateBuffer.Append("<tr><td><b>Screen
Shot</b></td><td><pcs:value
expr='Screen_Shot.name'></pcs:value></td></tr>"
+ System.Environment.NewLine)
templateBuffer.Append("</table>" + System.Environment.NewLine)

' Create the Presentation Template with the template text.
Dim presentationTemplate As IPresentationTemplate =
ptManager.CreatePresentationTemplate(dataEntryTemplate.GetContainingFolder(),
"Customer Support Incident Presentation Template",
templateBuffer.ToString())

' Store the Presentation Template.
presentationTemplate.Store()


...
' Validate the template text before using it to publish items
Dim errorsInText() As String =
ptManager.ValidateTemplateText(presentationTemplate,
presentationTemplate.TemplateText)
If errorsInText.Length <> 0 Then
Console.WriteLine("Validate Template Text returned the following
 errors")
Dim i As Integer
For i = 0 To errorsInText.Length - 1
Console.WriteLine("error[" & i & "]: " & errorsInText(i))
Next
End If
```

. . .

About Publisher Properties and Remote Property Operations

Properties categorize data into types (such as text, integer, or date). Properties are used to represent variables in Data Entry Templates and Presenation Templates that are processed to create content items.

When you create a property, you assign a property type to specify the type of information the property should hold. For a full list of property types, see *Publisher Property Types* on page 193. For details on creating properties, see *Creating Publisher Properties Using IDK Remote APIs* on page 190.

Content items are created from properties defined in a Data Entry Template. The formatting of the properties in the content item is defined by the Presentation Template attached to the Data Entry Template.

Selection lists are a special type of property that limits fields to specific entries and is implemented in Data Entry Templates as a drop-down list. You define the list of static entries, ensuring uniformity and eliminating misspellings. You must create and define a selection list before you can add it to a Data Entry Template. For details on creating selection lists, see *Creating Publisher Selection Lists Using IDK Remote APIs* on page 191.

The remote Publisher API cannot be used to modify properties in an existing Data Entry Template.

Removing properties from a Data Entry Template also deletes any associated property values from the Publisher system. Only published content items will retain the deleted properties. (Existing content items that have not been published will no longer contain the deleted properties.) To remove deleted properties from published content items, you must republish the content items. Selection lists are the exception; a selection list can only be deleted if it is not currently being used by a Data Entry Template.

Creating Publisher Properties Using IDK Remote APIs

To create new Publisher properties from a remote application, use the `IPropertyManager` and associated `IBaseProperty` interfaces in the IDK.

The `IPropertyManager` interface handles the creation of Publisher property interfaces, all of which are sub-types of the `IBaseProperty` super-interface. No store method needs to called to persist a property, but you must add the property to a Data Entry Template using the `IDataEntryTemplate.addProperty` method. Properties that are not added to a Data Entry Template are lost. For a list of property types, see *Publisher Property Types* on page 193. Selection lists are different and require special treatment, for details, see *Creating Publisher Selection Lists Using IDK Remote APIs* on page 191.

All properties have the following naming requirements:

- All property names *including properties of different types* within a Data Entry Template must be unique. Property names are not case sensitive.
- Property names cannot be empty strings.
- Property names cannot have more than 255 characters.

For sample code illustrating how to create properties, see *Creating Publisher Data Entry Templates Using IDK Remote APIs* on page 177.

Creating Publisher Selection Lists Using IDK Remote APIs

To create a new Publisher selection list, use the `ISelectionListManager` and `ISelectionList` interfaces in the IDK.

The `ISelectionListManager` interface can be used to create a new selection list or retrieve existing selection list(s). The `ISelectionList` interface returned by `ISelectionListManager` can be used to add or remove values in a selection list. You must call store to persist the newly created or updated selection list and insert it into the referenced folder. (The folder must exist before the selection list can be stored.) For an example of using a selection list in a Data Entry Template, see *Creating Publisher Data Entry Templates Using IDK Remote APIs* on page 177.

Java

```
...
// Get a selection list manager object.
ISelectionListManager selectionListManager =
factory.getSelectionListManager();

// Create a selection list representing the states in the United
 States.
String[] stateOptionValues = {"CA", "MA"};
ISelectionList stateSelectionList =
selectionListManager.createSelectionList(rootFolder, "State
Selection List", stateOptionValues);
stateSelectionList.store();

// Expand the list options.
String[] expandedStateOptionValues = {"OR", "NY"};
stateSelectionList.addValues(expandedStateOptionValues);
stateSelectionList.store();

// The list now contains "CA", "MA", "OR", and "NY."
// Remove the expanded options.
stateSelectionList.removeValues(expandedStateOptionValues);
```

```
stateSelectionList.store();
...
```

.NET (C#)

```
...
// Get a selection list manager object.
ISelectionListManager selectionListManager =
factory.GetSelectionListManager();

// Create a selection list representing states in the United
States.
String[] stateOptionValues = {"CA", "MA"};
ISelectionList stateSelectionList =
selectionListManager.CreateSelectionList(folder,
"State Selection List", stateOptionValues);
stateSelectionList.Store();

// Expand the list options.
String[] expandedStateOptionValues = {"OR", "NY"};
stateSelectionList.AddValues(expandedStateOptionValues);
stateSelectionList.Store();

// The list now contains "CA", "MA", "OR", and "NY."
// Remove the expanded options
stateSelectionList.RemoveValues(expandedStateOptionValues);
stateSelectionList.Store();
...
```

.NET (VB)

```
...
' Get a selection list manager object.
Dim selectionListManager As ISelectionListManager =
factory.GetSelectionListManager

' Create a selection list representing states in the United States.
Dim stateOptionValues() As String = {"CA", "MA"}
Dim stateSelectionList As ISelectionList =
selectionListManager.CreateSelectionList(folder,
"State Selection List", stateOptionValues)
stateSelectionList.Store()

' Expand the list options.
Dim expandedStateOptionValues() As String = {"OR", "NY"}
stateSelectionList.AddValues(expandedStateOptionValues)
```

```
stateSelectionList.Store()

' The list now contains "CA", "MA", "OR", and "NY."
' Remove the expanded options
stateSelectionList.RemoveValues(expandedStateOptionValues)
stateSelectionList.Store()
...
```

Publisher Property Types

The IDK remote Publisher API provides access to the following property types. Each type enables users to enter or choose a specific type of content in the associated form element in a Data Entry Template.

Property Name	Description	Property Interface
Text	Enables users to enter plain text. The text entry length can be limited to anything below 255 plain-text characters. To enable users to enter longer strings, use the long text property (ITextBlockProperty).	ITextLineProperty
Long Text	Provides a rich text editor that enables users to enter formatted text, images, tables, numbered or bulleted lists, hyperlinks, HTML, and Adaptive Tags. Long text properties store long strings that can span many lines. For short strings that can be limited to only one line, use the text property (ITextLineProperty) instead. You can also specify that the text box allow plain text entry only. The text entry length can be limited only if you use the plain text option.	ITextBlockProperty
Integer	Enables users to enter a whole number. If a user enters text, an “invalid entry” error occurs. It is a best practice to add instructions to help users understand that they must enter a whole number.	IIntegerProperty
Real	Enables users to enter a whole number or a fraction in decimal form. If a user enters text, an “invalid entry” error occurs.	IDoubleProperty
Boolean	Enables users to choose between two options (for example, to answer a yes/no or true/false question).	IBooleanProperty

Property Name	Description	Property Interface
Date	Enables users to enter a date or select it from a popup calendar control and choose a time from a drop-down list.	IDateProperty
Item	Enables users to insert a content item within another content item. The item property is a reference to any existing Publisher content item. The associated content item will appear within the content item. Users can either create a new content item and associate it with an existing content item or choose another existing content item to associate with the content item. The content items do not need to have been created from the same Data Entry Template.	IItemReferenceProperty
List	Enables users to create a list of existing content items. This property is similar to the item property, but creates a list of links to content items instead of displaying the referenced content item.	IItemCollectionProperty
Image	Enables users to upload an image from a local or shared network drive.	IImageProperty
File	Enables users to upload a file from a local or shared network drive.	IFileProperty
Selection List	Enables users to choose from a drop-down list of predefined entries. A selection list ensures uniformity and eliminates the chance of misspellings. Unlike other property types, a selection list can be shared among multiple Data Entry Templates. A selection list must be persisted before it can be added to a Data Entry Template. For more information about selection lists, see *Creating Publisher Selection Lists Using IDK Remote APIs* on page 191.	ISelectionListProperty ISelectionList ISelectionListManager

Publisher Remote API Exceptions

The IDK remote Publisher API (com.plumtree.remote.prc.content) throws six types of exceptions.

Exception	Thown When:
ContentException	A remote invocation results in an error in Publisher.
ContentSecurityException	A user does not have the proper role or access to perform a Publisher operation. For example, only users with Producer or higher roles can create a Data Entry Template; if a user with only a Submitter role were to attempt to create a Data Entry Template, an exception will be thrown.
NameAlreadyInUseException	An object of the same type with the same name already exists in the same folder. For example, a folder cannot have two Presentation Templates called "Design"; however, the folder can have a Presentation Template called "Design" and a content item also called "Design" because the two objects are different types. Publisher ignores capitalization and spacing, so it does not distinguish between "Design" and "design."
IllegalArgumentException	A Publisher object was named improperly. For example, not naming a Data Entry Template at all or using more than 255 characters in a name will result in this exception. All Publisher objects have a 255-character naming limit.
IllegalStateException	An action is performed that cannot be applied to a Publisher object because it does not exist or it is at a state that cannot receive the action. For example, exceptions are thrown if you attempt to attach a Presentation Template to a Data Entry Template that already has an attached Presentation Template, or if you attempt to retrieve the publication date of a Presentation Template that has not been published. An object does not exist until it has been persisted by calling the store method. This exception is also thrown if the containing folder of the object does not exist.
RemoteException	Publisher cannot make a connection with a database.

Publisher Roles

Publisher security is role-based and assigned by folder. The IDK remote Publisher API can be used to query role information but does not provide programmatic access to modify roles.

Roles define the actions that a user is able to perform in a given folder. The actions that each role can perform are cumulative; a higher role can perform certain actions in addition to all the actions lower roles can perform. For example, a Submitter can perform all the actions a Reader can perform, in addition to creating, previewing, editing, and submitting content items. For details on

configuring and assigning Publisher roles, see the *Administrator Guide for AquaLogic Interaction Publisher* or the Publisher online help.

Role	Description
Reader	Can view published content.
Submitter	Can also create, preview, edit, search, and delete content items.
Contributor	Can also view content item notes, version history, publishing information, and publishing schedule.
Editor	Can also publish content items to Publisher and to the portal Knowledge Directory.
Producer	Can also create and edit folders, Data Entry Templates, selection lists, and Presentation Templates. In addition, a Producer can preview and publish content items individually and by folder; edit content items assigned to others; and configure folders to publish and preview.
Folder Administrator	Can also undo other users' check-outs, manage published content portlets, and configure Publisher security on folders.
Portal Administrator	Automatically granted the Publisher Administrator role and has access to every folder and every feature in Publisher. A Portal Administrator can create, edit, copy, and delete top-level folders, and delegate administrative rights at the folder level.

About Adaptive Pagelets

Adaptive pagelets allow you to create a coordinated page with dynamic, interactive functionality comprised of cross-platform services that talk to multiple back-ends.

Adaptive pagelet tools include the following:

- *About the ALI Scripting Framework* on page 260
- *About Adaptive Tags* on page 202

For additional information on adaptive pagelets, see the following topics:

- *About Adaptive Pagelet Design Patterns* on page 197
- *Adaptive Pagelet Development Tips* on page 201

The AquaLogic .NET Portlet Toolkit also provides useful tools for adaptive pagelets. For installation instructions and development information, download the AquaLogic .NET Portlet Toolkit from the ALUI Developer Center on dev2dev.

About Adaptive Pagelet Design Patterns

Adaptive pagelet design patterns provide solutions for broad classes of problems, and provide the base for a range of cross-platform services.

The **Master-Detail** design pattern uses two pagelets; users select an item from a list in one pagelet, and the details for that item are retrieved from the remote server and displayed in another pagelet. For example, a set of customers could be listed by name in the "master" pagelet. When the user clicks a name in this pagelet, the "detail" pagelet presents details about the item. The detail pagelet for a customer list could list all the important information about that customer, such as name, address, and phone number. This pattern assumes a tight coupling between the two pagelets; both the master pagelet and detail pagelet must be displayed on the same page. For a looser coupling, use the Broadcast-Listener pattern. For details and sample code, see *Using Session Preferences* on page 304.

The **Broadcast-Listener** design pattern is similar to the Master-Detail pattern, but assumes a loose coupling between pagelets. Users can select an item or perform some other action in a "broadcast" pagelet, which causes the content in other related "listener" pagelets to be redrawn. The major difference is that the Broadcast-Listener pattern relies on the ALI Scripting Framework to raise an event when an action is performed in the "broadcast" pagelet. One or more "listener" pagelets can respond to this event and update their content accordingly. For details and sample code, see *Using ALI Scripting Framework Event Notification* on page 262.

In Place Refresh allows you to refresh the content in a pagelet without refreshing the page. For details and sample code, see *Using In-Place Refresh* on page 269.

The **Structured Response** design pattern handles structured HTTP responses, typically encoded as XML. In many cases it can be expensive and inefficient to send large amounts of HTML back in response to some HTTP request, if only a small part of the user interface needs to be changed. This is especially true with more complex user interfaces. In these cases, the response can be encoded in XML. The client-side response handler can then parse the XML, and update the user interface (or perform some other action) based on that response. Use the Structured Response design pattern to redraw a small area of the user interface after making an HTTP request, or to access a simple HTTP/URI type web service from a pagelet. The example code below (structuredresponse_portlet.html) accesses an RSS feed from a selection of news sites.

```
<!-- jsxml includes -->
<a id="imgServerHref" href="pt://images/plumtree"
style="display:none"></a>
<script type="text/javascript"
```

```
src="pt://images/plumtree/common/private/js/PTLoader.js"></script>
<script type="text/javascript">
var oImgServer = new Object();
oImgServer.location =
document.getElementById('imgServerHref').href;
var imageServerURL = document.getElementById('imgServerHref').href;
var imageServerConnectionURL = oImgServer.location;
new PTLoader(imageServerURL,
imageServerConnectionURL).include('jsxml','en');
</script>

<!-- jscontrols includes -->
<link rel="stylesheet" type="text/css"
href="/portal-remote-server/js/jscontrols/styles/css/PTMenu.css"/>
<link rel="stylesheet" type="text/css"
href="/portal-remote-server/js/jscontrols/styles/css/PTRichTextEditor.css"/>
<script type="text/javascript"
src="/portal-remote-server/js/jscontrols/strings/PTControls-en.js"></script>
<script type="text/javascript"
src="/portal-remote-server/js/jscontrols/PTControls.js"></script>

<!-- Inline JS helper functions -->
<!-- NOTE: It is standard practice to use namespace tokens (e.g.,
 <pt:nameSpace pt:token="$$TOKEN$$"
xmlns:pt="http://www.plumtree.com/xmlschemas/ptui/"/>) to ensure
 unique global JavaScript function and object names. For
simplicity, we do not do that here.
-->

<script defer type="text/javascript"
id="structured-response-portlet-A-script">
// Function that gets the RSS XML feed found at the specified url
getRSSFeed = function(url)
  {
  // First clear out any existing rows in the table
  channelTable.clearRows();

  // Force the transformer to fix up the url
  var oURL = new Object();
  oURL.location = url;

  // Do the http get
  var get = new PTHTTPGETRequest(oURL.location, handleRSSResponse);

  get.invoke();
```

```
  }

// Function that handles the RSS XML response and updates the
table based on the RSS items
handleRSSResponse = function(response)
  {
  // Get the rss xml
  var xml = response.responseText;
  if (!xml || xml.indexOf('<?xml') == -1) { return; }

  // Parse into a dom, and get the channel node
  var xmlDOM = new PTXMLParser(xml);
  var rssNode = xmlDOM.selectSingleNode('rss');
  var channelNode = rssNode.selectSingleNode('channel');

  // Get the channel title and set the status bar text in the
table
  var channelTitle =
channelNode.selectSingleNode('title').getNodeValue();
  channelTable.statusBarText = '<b>Loaded Channel</b>: ' +
channelTitle;

  // Get channel item nodes
  var itemNodes = channelNode.selectNodes('item');

  // Build table rows
  channelTable.rows = new Array();
  for (var i=0; i<itemNodes.length; i++)

    {
    var itemNode = itemNodes[i];

    // Get channel item properties
    var itemTitle =
itemNode.selectSingleNode('title').getNodeValue();
    var itemLink =
itemNode.selectSingleNode('link').getNodeValue();
    var itemDescription =
itemNode.selectSingleNode('description').getNodeValue();
    if (itemNode.selectSingleNode('author'))
      var itemAuthor =
itemNode.selectSingleNode('author').getNodeValue();
    if (itemNode.selectSingleNode('category'))
      var itemCategory =
itemNode.selectSingleNode('category').getNodeValue();
```

```
    if (itemNode.selectSingleNode('pubDate'))
      var itemPubDate =
itemNode.selectSingleNode('pubDate').getNodeValue();

    // Create a row and add it to the table
    var row = new PTRow();
    row.parent = channelTable;
    row.id = i;
    row.uid = i;
    row.previewText = itemDescription;
    row.link = itemLink;
    row.columnValues[0] = new PTTextColumnValue(itemTitle);
    row.columnValues[1] = new PTTextColumnValue(itemCategory);
    row.columnValues[2] = new PTTextColumnValue(itemAuthor);
    row.columnValues[3] = new PTTextColumnValue(itemPubDate);
    channelTable.rows[channelTable.rows.length] = row;
    }

  // Redraw the table
  channelTable.draw();
  }
</script>

<b>Select RSS Feed:</b>
<a href="#"
onclick="getRSSFeed('http://www.wired.com/news/feeds/rss2/0,2610,,00.xml');
 return false;">Wired News</a>
<a href="#"
onclick="getRSSFeed('http://news.com.com/2547-1_3-0-5.xml'); return
 false;">CNET News.com</a>
<a href="#"
onclick="getRSSFeed('http://partners.userland.com/nytRss/nytHomepage.xml');
 return false;">NY Times</a>
<br><br>

<!-- Set up a table control to display channel items -->
<div id="channelTableContainer"></div>
<script defer type="text/javascript">
  var channelTable = new PTTableControl();
  channelTable.locale = 'en_US';
  channelTable.objName = 'channelTable';
  channelTable.container = 'channelTableContainer';
  channelTable.baseURL =
'/imageserver/plumtree/common/private/portal-remote-server/js/jscontrols/1/';
```

```
  channelTable.statusBarText = 'No RSS Feed Selected';
  channelTable.rowDetailAction = new
PTJavaScriptAction('window.open(\'${ROW.link}\');');
  channelTable.columns[0] = new PTColumn();
  channelTable.columns[0].name = 'Title';
  channelTable.columns[0].width = '40%';
  channelTable.columns[1] = new PTColumn();
  channelTable.columns[1].name = 'Category';
  channelTable.columns[1].width = '20%';
  channelTable.columns[2] = new PTColumn();
  channelTable.columns[2].name = 'Author';
  channelTable.columns[2].width = '20%';
  channelTable.columns[3] = new PTColumn();
  channelTable.columns[3].name = 'Publication Date';
  channelTable.columns[3].width = '20%';
  channelTable.areColumnsResizable = true;
  channelTable.clientSortEnabled = true;
  channelTable.scrollHeight = 250;

  channelTable.init();
  channelTable.draw();
</script>
</div>
```

Adaptive Pagelet Development Tips

These tips apply to most adaptive pagelets.

Gateway all URLs. You cannot make request to a URL whose host/port differs from that of the calling page. All URLs requested through JavaScript must be gatewayed. For details on the gateway, see *About Pagelets and the Gateway* on page 281.

Add all required JavaScript to the page in advance. Browsers might not process script blocks/includes added to the page through the `innerHTML` property.

- IE: Add the defer attribute to the script tag.
- Netscape: Use RegExp to parse the response and look for the script, then eval it.

JavaScript HTTP and gatewayed HTTP must use the same authentication credentials. JavaScript brokered HTTP requests send the same authentication token (cookie) as when you make a normal gatewayed HTTP request.

About Adaptive Tags

AquaLogic Interaction and AquaLogic Ensemble provide a collection of useful XML tags that can be included in the markup returned by any gatewayed page, including pagelets.

Using the attributes defined in the tag, the gateway transforms the XML and replaces it with standard HTML to be displayed in a browser. For example, when used in a banner pagelet in the ALI portal, the following code is replaced with the date and time in the current user's locale.

```
<pt:standard.currenttime
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
```

The adaptive tag libraries provide access to a wide range of components. For details on adaptive tag libraries, see the following topics:

- *About Cross-Platform Adaptive Tag Libraries* on page 203
- *About ALI Adaptive Tag Libraries* on page 202
- *Ensemble Adaptive Tag Library (pt:ensemble)* on page 242
- *Pages Adaptive Tag Library (pt:pages)* on page 246
- *Pathways Search Adaptive Tag Library (pt:pathways)* on page 247

For detailed information on using tags, see *Adaptive Tag Development Tips* on page 204. For information on how the portal processes tags, see *About Adaptive Tag Control Flow* on page 250. You can also create custom tags; for details, see *Creating Custom Adaptive Tags* on page 252.

For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

About ALI Adaptive Tag Libraries

The ALI portal tag libraries include tags to display portal navigation components, portal UI components, and standard UI elements.

Navigation tags in the **plugnav** library are used with data tags in the **ptdata** library to build complete custom navigation solutions for the portal. For details, see *Implementing Custom Navigation Using Adaptive Tags* on page 233. AquaLogic Pages and AquaLogic Pathways also include tags to insert components into portal navigation. For details, see *Adding Links to Pages LiveSpaces and DataSpaces Using Adaptive Tags* on page 246 and *Adding Pathways Search Using Adaptive Tags* on page 248.

The tags in the **ptui** library allow you to add standard ALI user interface components to any pagelet, including search inputs and buttons, login components, access to account settings, error messages, and more. Tags from the standard tag library can be used to display instance-specific

information, including the date and time and the page name and type. For details, see *Implementing Custom UI Elements Using Adaptive Tags* on page 237.

The **standard** tag library includes tags for the following purposes:

- Links: Build links to almost any portal object, community pages, login pages, or any gatewayed page. You can also set Hosted Display Mode for any gatewayed page. For details, see *Building Gatewayed URLs Using Adaptive Tags* on page 217.
- User-Specific Information: Provide user-specific content, leveraging settings and portal permissions. Use conditional statements to secure content based on user or group membership. For details, see *Securing Content Based on User Permissions Using Adaptive Tags* on page 226.
- Tree Controls: Create custom selection trees of portal objects. For details, see *Creating Tree Controls Using Adaptive Tags* on page 218.

This package contains most of the tags available in portal version 5.x, previously called "transformer tags." Legacy tags not included in the standard library are provided in the **transformer** tag library (6.1 and earlier) or the **common** tag library (ALI 6.1MP1 and above).

In addition to the tags above, ALI pagelets can use the cross-platform tag libraries. For details, see *About Cross-Platform Adaptive Tag Libraries* on page 203.

For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

About Cross-Platform Adaptive Tag Libraries

Cross-product adaptive tag libraries provide key functionality for use in pagelets that work in both ALI and Ensemble.

The **core** tag library provides two basic tags to support core tag functionality.

- `pt:core.debugmode` toggles debug mode.
- `pt:core.html` allows you to use HTML tags within JavaScript, and supports attribute replacement.

The tags in the **constants** library provide access to useful URLs, including the stylesheet, Image Service, and the correct return URL for the current user.

Tag	Replaced with	Example
`pt://styles`	the stylesheet URL in hosted pages and pagelets	`<link type="text/css" href="pt://styles" rel="StyleSheet"></link>`

Tag	Replaced with	Example
`pt://images`	the URL to the Image Service	`<img src="pt://images/portal/public/myfolder.gif">`
`pt://return`	a URL that returns users to the page from which they came (the page on which the pagelet that launched the page is hosted)	`<a href="pt://return">Back</a>`

The **common** tag library provides access to useful functionality, including URL transformation and namespace definition. This library also allows you to insert error information in the page, and CSS and JavaScript references in the Head element in a gatewayed HTML page. For details, see *Common Adaptive Tag Library (pt:common)* on page 207.

The tags in the **logic** library handle basic logic, including creating data objects and collections, setting shared variables, and looping over a data collection. For details, see *Logic Adaptive Tag Library (pt:logic)* on page 227.

In addition to the tags above, platform-specific tags are available to access additional information and functionality in ALI and Ensemble. For details, see *About ALI Adaptive Tag Libraries* on page 202 and *Ensemble Adaptive Tag Library (pt:ensemble)* on page 242.

For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

Adaptive Tag Development Tips

These syntax rules and tips apply to all adaptive tags.

- **All tags are XML compliant.** For example, only strings are allowed; you cannot use a tag within an attribute of another tag (`<legal a=<illegal/>/>`).
- **All adaptive tags belong to the namespace http://www.plumtree.com/xmlschemas/ptui/.** The namespace prefix must be "pt" (xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/). To avoid including the namespace in every tag, enclose all tags in a span that defines the namespace.
- **All adaptive tag attributes require the "pt:" prefix.** If you do not include the pt prefix, the portal will not return an error, but will replace the attribute with the default value when the tag is processed.

The adaptive tag framework displays tag errors as HTML comments. If you suspect that a tag error has occurred, simply view the HTML source for the page. If there was a problem, there should be an HTML comment where the adaptive tag would have been.

Adaptive tags adhere to XHTML specifications. These specifications are not handled correctly by all HTML editors and IDEs. Some editors do not display tags correctly because of the required "pt:" prefix before tags and ALI attributes.

Use tag debug mode for additional insight into tag errors. Turning on Debug Mode causes the adaptive tag framework to output HTML comments declaring the start and end of each tag. This can be useful for determining whether a tag ran and did not output the expected result, or did not run at all, for example. Note: Standard HTML tags are not wrapped in HTML comments.

Using Internationalized Strings in Adaptive Tags

Adaptive tag attribute value replacement allows you to display localized content based on the current user's portal locale.

ALI and Ensemble store internationalized strings in localized string files with different files for each supported language. The portal knows the locale of the current user and retrieves strings from the correct language folder automatically. To internationalize a pagelet, move all strings into custom string files and translate them.

To display content in the pagelet, reference the strings using the value tag from the Logic tag library. ALI and Ensemble know the locale of the current user and retrieve the string from the correct language folder automatically. For example, the HTML below retrieves the first string from a XML language file called my_message_file.xml.

```
<span
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>

  <pt:logic.value pt:value="$#1.my_message_file"/>
</span>
```

For details on using shared variables, see *Using Variables in Adaptive Tags* on page 205.

Using Variables in Adaptive Tags

Adaptive tag attribute value replacement allows you to access data stored in memory.

The following simple example uses the variable and value tags from the logic tag library to store a value in memory and then display it in HTML.

```
<span
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>

  <pt:logic.variable pt:key="test" pt:value="example
 text"/>
  <pt:logic.value pt:value="$test"/>
</span>
```

Attribute value replacement can also be used to display more complicated memory structures. Data objects can contain multiple name value pairs. The following example creates a data object with the name and URL of a link, and then displays the name.

```
<span
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>

  <pt:logic.data pt:key="testdata"
url="http://www.myco.com" name="My company"/>
  <pt:logic.value pt:value="$testdata.name"/>
</span>
```

Attribute value replacement cannot be used with tags outside the adaptive tag libraries. However, the pt.core.html tag supports attribute replacement within a tag and allows you to generate any HTML tag. Use the pt:tag attribute to specify the HTML tag and list the necessary HTML attributes as XML attributes. All non-adaptive tag attributes (attributes not prefixed with "pt:") are included automatically in the outputted HTML tag. For example, the following code creates an HTML anchor tag using an in-memory value for the "href" attribute.

```
<span
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<pt:core.html pt:tag="a" href="$myurl" title="My
title">My link</pt:core.html>
</span>
```

This code would be transformed to the following HTML: `<a href="[data stored in the $myurl attribute]" title="My title">My link</a>`.

The example below combines several different techniques and tags to show how to loop over a data collection and output HTML. This code outputs several HTML links with lines in between them.

```
<span
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>

  <pt:logic.collection pt:key="testcollection">
    <pt:logic.data url="http://www.myco.com"
name="My company"/>
    <pt:logic.data url="http://www.otherco.com"
name="Other company"/>
  </pt:logic.collection>
  <pt:logic.foreach pt:data="testcollection"
pt:var="link">
    <pt:core.html pt:tag="a" href="$link.url">
    <pt:logic.value pt:value="$link.name"/>
    </pt:core.html>

<pt:logic.separator><br><br></pt:logic.separator>
  </pt:logic.foreach>
</span>
```

For details on logic tags, see *Logic Adaptive Tag Library (pt:logic)* on page 227. For details on using localized strings in tags, see *Using Internationalized Strings in Adaptive Tags* on page 205.

Common Adaptive Tag Library (pt:common)

The Common tag library (pt:common) provides access to useful functionality, including URL transformation and namespace definition. This library also allows you to insert error information in the page, and CSS and JavaScript references in the Head element in a gatewayed HTML page.

The Common tag library is a cross-platform tag library that can be used in both ALI and Ensemble.

For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

Tag	Function	More Information
`pt:common.namespace`	Defines a token for use in JavaScript functions and HTML elements to ensure unique names in an aggregated page.	*Defining a Unique Namespace Token Using Adaptive Tags* on page 210

Tag	Function	More Information
pt:common.url	Transforms URLs that should be gatewayed.	*Transforming URLs Using Adaptive Tags* on page 211
pt:common.transformer	Disables and enables transformation on a gatewayed page.	*Transforming URLs Using Adaptive Tags* on page 211
pt:common.error	Displays errors on the page so that they can be placed and formatted as desired.	*Displaying Errors Using Adaptive Tags* on page 210
pt:common.errorcode	Stores a collection of the current error codes in memory.	*Displaying Errors Using Adaptive Tags* on page 210
pt:common.errortext	Displays the current error text on the page so that it can be placed and formatted as desired. Only the first error message will be displayed. Used as singleton only (does not display the contents of the tag).	*Displaying Errors Using Adaptive Tags* on page 210
pt:common.headincludes	Allows JavaScript and Style Sheet include information to be added to a specific point in the Head element of an HTML page, as required by the XHTML specification.	*Adding Header Content Using Adaptive Tags* on page 209
pt:common.includeinhead	Marks JavaScript and CSS information to be included in the Head element of the HTML page by the pt:common.headincludes tag.	*Adding Header Content Using Adaptive Tags* on page 209
pt:common.jstransform	Inserts the ALI Scripting Framework constructors into the Head element of the ALI portal page (cannot be used in Ensemble). Used as singleton	*Adding Header Content Using Adaptive Tags* on page 209

Tag	Function	More Information
	only (does not display the contents of the tag).	
pt:common.userinfo	Displays a specific user information setting.	*Accessing User Information Using Adaptive Tags* on page 209

Accessing User Information Using Adaptive Tags

You can use the pt:common.userinfo tag to access specific user information settings.

The `pt:common.userinfo` tag is replaced with the value of the User Information setting specified in the `pt:info` attribute. The name attribute is case sensitive.

Note: In ALI 6.0 and 6.1, this tag is implemented as `pt:userInfo` with the attribute `pt:name`. This syntax is also supported.

```
<pt:common.userinfo pt:info="FullName"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
```

Note: You must configure the Web Service (ALI) or Resource (Ensemble) to send the appropriate user information settings to the pagelet.

Adding Header Content Using Adaptive Tags

The pt:common.includeinhead and headincludes tags allow you to include custom JavaScript and CSS information in the Head element of the HTML page.

The `pt:common.includeinhead` tag marks the JavaScript and CSS information to be included in the Head element of the HTML page by the `pt:common.headincludes` tag. If a .js or .css file is marked for inclusion multiple times, it will only be included once. JavaScript generated by tags will also be included.

Note: This tag will be ignored during automatic in-place refresh requests. Custom in-place refresh solutions must ensure that JavaScript gets included correctly.

```
<pt:common.includeinhead>
<script type="text/javascript"><!-- JavaScript --></script>
<script type="text/javascript"
src="http://test.com/test.js"></script>
<link type="text/css" rel="stylesheet"
href="http://test.com/test.css"></link>
</pt:common.includeinhead>
```

The `pt:common.headincludes` tag adds JavaScript and stylesheet include information defined by the `pt:common.includeinhead` tag to the Head element of the HTML page, as required by the XHTML specification. If no `pt:common.headincludes` tag is present, JavaScript will be included at the bottom of the Head element, and a Head element will be inserted if one does not exist.

```
<head>
<script type="text/javascript"
src="http://test.com/main.js"></script>
</head>
```

The `pt:common.jstransform` tag inserts the ALI Scripting Framework headers into the Head element of the ALI portal page (cannot be used in Ensemble).

Defining a Unique Namespace Token Using Adaptive Tags

It is an established best practice to include the pagelet/portlet ID in the name of any Javascript functions and HTML elements to ensure unique names when the code is combined with markup from other pagelets on an aggregated page.

The `pt:common.namespace` tag allows you to define your own token, which is replaced with the pagelet/portlet ID. The token must follow these specifications:

- Valid values for the token must be in the ASCII range 0x21 to 0x7E, excluding "<" (0x3C).
- The scope of the token runs from the tag defining it to the end of the file; you cannot use a token prior to defining it.
- A second pt:namespace tag with a different token redefines it; two tokens cannot be defined at the same time.

Note: In ALI 6.1 and earlier, this tag is implemented as `pt:namespace`.

```
<pt:common.namespace pt:token="$$TOKEN$$"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
<a onclick="doStuff$$TOKEN$$();" href="#">do
stuff</a>
<script>
function doStuff$$TOKEN$$() {
alert("hello");
}
</script>
```

Displaying Errors Using Adaptive Tags

The error* tags in the Ensemble Common library allow you to insert and format error messages within the page that contains the tag(s).

The pt:common.error tag displays errors on the page, placed and formatted as desired. If the pt:common.errortext tag is included inside this tag, the contents of the tag will only be processed if there is an error. If the child tag is not present, any error messages will be formatted and displayed from this tag in the standard style.

If the pt:common.errortext tag is included, only the first error message will be displayed. Other errors, as well as exception stack traces and extended error messages, will be ignored.

The pt:common.errorcodes tag stores a collection of the current error codes in memory. If the error has already been displayed, no error codes will be available. These error codes can be accessed using the pt:logic.foreach tag as shown below.

Note: If these tags are displayed on a page, errors will no longer be displayed in the normal error location and will not be available after the tag has been displayed.

```
<pt:common.errorcode pt:key="errorcodes"/>
<pt:logic.foreach pt:data="errorcodes"
pt:var="code">
<pt:common.errortext/>
```

Transforming URLs Using Adaptive Tags

The pt:common.url and pt:common.transformer tags allow you to create and manipulate gatewayed URLs.

Note: In ALI portal 6.1 and earlier, these tags are implemented as pt:url and pt:transformer. This syntax is still supported.

The pt:common.url tag is used to transform URLs that should be gatewayed. If the URL in the pt:href attribute is outside the gateway, it will be transformed to an absolute URL. This feature does not generate a link in HTML; it obtains the URL as a string and passes it to a client-side function, as shown in the following example.

```
<script>
function myFunction()
{
document.write("<pt:common.url pt:href="myURL"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>");
}
```

The `pt:common.transformer` tag allows you to turn off JavaScript URL transformation in a gatewayed page. Set the `pt:fixurl` attribute to "off" as shown below.

```
<pt:common.transformer pt:fixurl="off"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
```

The transformer will not insert calls to the JavaScript URL transformation function for the rest of the file, unless you switch the feature back on in a subsequent directive (with a `pt:fixurl` attribute of "on").

Standard Adaptive Tag Library (pt:standard)

Adaptive tags can be used to build links to a variety of ALI portal resources. The Standard tag library (pt:standard) allows you to create links to specific portal objects, the portal login page, or to specific portlets. You can also build gatewayed URLs, disable URL transformation, and enable Hosted Display Mode for gatewayed pages.

The tags in the pt:standard tag library are available for use only in ALI (not in Ensemble).

Note: The transformer copies any attributes not in the PT namespace to the output link tag. These links are platform and version independent, and do not rely on particular ASP/JSP files or query string arguments.

The tables below summarize available standard tags. For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

Page Information: These tags display instance-specific information for use in portal navigation elements, including the date and time and the page name and type.

Tag	Function	More Information

`pt:standard.currenttime`	Writes the current date and time according to the rules of the user's chosen locale. Only the full date and time can be displayed; there is no way to return just the date, just the time, or any other subset of information. This tag is recalculated every time the code is pulled out of the cache.	For example:
		`<pt:standard.currenttime`
		`xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>`

Tag	Function	More Information
pt:standard.realmname	Replaced with the portal page type ("My Pages," "Documents," "Administration," or "Gateway").	For example, the code snippet below creates the portal banner (the pt://images constant is used to reference the portal Image Service).
pt:standard.pagename	Replaced with the name of the current portal page (My Page or Community) or left blank otherwise.	<td align="left" colspan="1" id="pt-header-left"> <!--portal banner --> </td> <td align="right" nowrap="nowrap" colspan="1" id="pt-header-right"> <h1 class="banHeader"> <pt:standard.realmname xmlns:pt="http://www.plumtree.com/xmlschema/ptui/"/> </h1> <h2 class="banSubhead"> <pt:standard.pagename xmlns:pt="http://www.plumtree.com/xmlschema/ptui/"/> </h2> </td>

URLs: These tags provide access to key portal components, including the stylesheet, portal objects, and the portal login pages. Additional tags allow you to create gatewayed links and control hosted display mode for gatewayed pages.

Tag	Function	More Information

`pt:standard.stylesheets`	Allows you to enter the current portal stylesheet in the HEAD of any non-hosted gatewayed HTML page. (In previous versions, this tag was implemented as pt:styleSheets. This syntax is still supported.)	For example,
		`<HTML>`
		`<HEAD>`
		
		`<pt:standard.stylesheets`
		
		`xmlns:pt="http://www.plumtree.com/xmlschema/pt" />`
		
		`...`
		`</HEAD>`
		`<BODY>`
		`...`

`pt:standard.displaymode`	Sets the header that tells the Portal Server to display a page in the style of the portal, with a portal banner. The tag can also set the title and subtitle of the page. The displaymode tag does not display any contents, and should only be used as a singleton. (Note: Pages in hosted display mode should not contain <HTML>, <HEAD>, <META>, <TITLE> or <BODY> tags.)	For example,
		`<pt:standard.displaymode`
		`pt:mode="Hosted"`
		`pt:title="My title"`
		`pt:subtitle="My subtitle"`
		`xmlns:pt="http://www.plumtree.com/xmlschema/pt" />`

`pt:standard.loginlink`	Creates a link to the portal login page. In previous versions, this tag was implemented as pt:loginLink. This syntax is still supported.	For example,
		`<pt:standard.loginlink`
		
		`xmlns:pt="http://www.plumtree.com/xmlschema/pt">Log in`
		`</pt:standard.loginlink>`

Tag	Function	More Information
`pt:standard.openerlink`	Creates a link that can open or view an object or properties of an object that already exists within the portal.	*Accessing ALI Objects Using Adaptive Tags* on page 216
`pt:standard.gatewaylink`	Allows you to buildgatewayed links to remote pages.	*Building Gatewayed URLs Using Adaptive Tags* on page 217

Constants are also available for useful URLs, including the Image Service, current stylesheet, and return URL For details, see *About Cross-Platform Adaptive Tag Libraries* on page 203.

User-Specific Information: These tags allow you to insert content on a page based on conditional statements of user and group membership. For details on implementing these tags, see *Securing Content Based on User Permissions Using Adaptive Tags* on page 226.

Tag	Function
`pt:standard.choose`	Denotes the start of a secured content section.
`pt:standard.when`	Includes a test condition that defines who has access to the enclosed content.
`pt:standard.otherwise`	Includes content that should be displayed as default.

To access user settings stored in the ALI database, use the pt:usersetting tag. The tag is replaced with the value of the user setting specified in the pt:name attribute. This tag will decode %uHHHH encoded values stored in the ALI database. You must configure the Web Service object to send the appropriate settings. For details on ALI settings, see *About ALI Portlet Settings* on page 294.

```
<pt:userSetting pt:name="myUserSetting"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
```

Tree Controls: These tags provide links to a popup window that allows users to select options from a structured list of objects. For details on implementing these tags, see *Creating Tree Controls Using Adaptive Tags* on page 218.

Tag	Function
pt:standard.tree	Creates a form button to a popup window that allows users to select options from a structured list of objects.
pt:standard.treelink	Creates a link to a popup window that allows users to select options from a structured list of objects.
pt:standard.treeurl	Returns a URL to a popup window that allows users to select options from a structured list of objects (can be used in JavaScript). Does not display the contents of the tag and should only be used as a singleton tag (i.e. <pt:standard.treeurl/>), rather than as a tag with both an open and close tag.

Accessing ALI Objects Using Adaptive Tags

To create a link that can open or view an object or properties of an object that already exists within the portal, use the pt:standard.openerlink tag.

You can use the pt:standard.openerlink tag for a variety of purposes, including viewing the User Profile for a user (requires User ID), viewing a community page (requires Community ID), opening a Remote Server object to edit the base URL (requires Remote Server ID), and clicking through to a document in the Knowledge Directory (requires Document ID). The pt:standard.openerlink tag is primarily controlled by three attributes:

Attribute	Value
pt:classid	The portal object type.
pt:objectid	The ID of the portal object referenced in the Class ID attribute (for example, the User or Community ID). To access the object ID, use the PRC. For details, see *Querying ALI Objects Using IDK Remote APIs* on page 60.
pt:mode	The action of the link (open/edit, view and view metadata).

For a full list of class IDs and associated modes, see *ALI Object Type Class IDs and Modes* on page 65.

Note: When you open an object in edit mode from a gatewayed page, clicking Finish or Cancel will close the window, so you should always use a popup window. When you open an object in

edit mode from within a portal page (My Page or Community Page), clicking Finish or Cancel will redirect to the return URI within the same window, so using a popup window might not be necessary. Always test your code in the portal to make sure it functions as expected.

To open a link in a popup window, you must add attributes to the link to control the popup window. All attributes that are not in the PT namespace are passed through to the transformed link. The following example opens a community page in a separate window.

```
<pt:standard.openerlink
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/' pt:objectid='1'
pt:classid='512' pt:mode='2' target='myWindow'
onclick=window.top.open
('','myWindow','height=800,width=700,status=no,toolbar=no,menubar=no,location=no');>View
 my Community.</pt:standard.openerlink>
```

Any time a user's name is displayed on a page, it is a best practice to display a clickable link to the user's profile page. The `pt:standard.openerlink` tag allows you to create links on demand using the User ID. (As noted above, use the PRC to access the object ID.) This example is not displayed in a popup window.

```
<pt:standard.openerlink
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/' pt:objectid='"
 & _ userID & "'
pt:classid='1' pt:mode='2'>"& LocRM.GetString("userName") & _
"</pt:standard.openerlink>
```

Building Gatewayed URLs Using Adaptive Tags

To buildgatewayed links to remote pages, use the pt:standard.gatewaylink tag.

Using attributes, you can include references toassociated portal objects, usually a portlet or community. When the link is executed, the portal sends any preferences associated with the referenced object to the remote server. *Accessing ALI Objects Using Adaptive Tags* on page 216.

The `pt:standard.gatewaylink` tag supports the following attributes:

Attribute	Value
`pt:classid`	The portal object type. The default is portlet (43). The `pt:standard.gatewaylink` tag also supports cards (18), Content Sources (35), and Web Services (47).
`pt:objectid`	The ID of the portal object referenced in the pt:classid attribute (e.g., the Portlet ID). To access the object ID, use the PRC. For details, see *Querying ALI Objects Using IDK Remote APIs* on page 60.
`pt:communityid`	The ID of the associated Community.

Attribute	Value
pt:pageid	The ID of the associated page (can be positive or negative).
pt:href	The URL to the remote page. If you pass in a relative URL, the portal will use the configuration settings for the referenced portal object to create the gatewayed URL.

The sample code below creates a link to a remote page associated with the portlet with ID 201. The arguments in the resulting URL tell the portal to send the preferences associated with the portlet to the remote server.

```
<pt:standard.gatewaylink class="myStyle"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'
pt:objectid='201' pt:href="doStuff.aspx?ID=5">Click
here</pt:standard.gatewaylink>
```

The code below creates a link to a page associated with the Web Service with ID 200, and also sends the community preferences from the community with ID 301 to the remote server.

```
<pt:standard.gatewaylinkpt:href="http://myRemoteServer/myTestPage.jsp"pt:objectid="200"
pt:classid="47"pt:communityid="301"xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
Click here</pt:standard.gatewaylink>
```

You can also use the `pt:standard.gatewayLink` tag to gateway documents that have not been crawled or uploaded to the portal using the ID for the associated WWW Content Source, as shown in the sample code below.

```
<pt:standard.gatewaylinkpt:href="http://myRemoteServer/mydocs/WhitePaper2002.doc"
pt:objectid="202"pt:classid="35"xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
WhitePaper2002</pt:standard.gatewaylink>
```

You can also use the `pt:common.url` tag to transform URLs that should be gatewayed. For details, see *Transforming URLs Using Adaptive Tags* on page 211.

Creating Tree Controls Using Adaptive Tags

To create a form button or link to a popup window that allows users to select options from a structured list of objects, use the pt:standard.tree, pt:standard.treelink or pt:standard.treeurl tag.

The `pt:standard.tree` and `pt:standard.treelink` tags create a form button or link, and the `pt:standard.treeurl` tag returns a URL that can be used in JavaScript. All three tags use a selection of attributes to control the tree display. The first four attributes are required.

Attribute	Description	Default	Syntax
pt:Class	The ID of the types of objects to display in the tree. Community pages are not supported. (REQUIRED)	value required	pt:Class='<classID1>,<classID2>,<classID3>,...'
pt:RootID	The ID of the root folder to be displayed in the tree. Use ID 1 for all folders. (REQUIRED)	value required	pt:RootID='<folderID>'
pt:SubmitMd	The mode in which the tree submits data to the opening page. Use mode 2 (javascript submit for remote development). When the data is submitted, the javascript function defined in pt:Submit is called on the main page. (REQUIRED)	value required (=2)	pt:SubmitMd='2'
pt:Submit	The name of the javascript function in the parent page to call when the tree is submitted (can take in an array of objects with name, Object ID, and Class ID). Do not include parentheses ("()") in the value of this attribute. (REQUIRED)	value required	pt:Submit='<javascriptFunctionName>'

Attribute	Description	Default	Syntax
`pt:AllowEmpty`	Allows users to click finish in a tree window with nothing selected: true=allow no selection; false=must select.	false	`pt:AllowEmpty='true'` or `pt:AllowEmpty='false'`
`pt:Display`	Limits the display to the selected objects, referenced by Class ID and Object ID. Cannot be used to display folders. The Class ID of each object must be included in `pt:Class`. The `pt:RootID` must be specified even though it will be ignored. Note: Do not include any folder Class IDs (17, 20, 515) in the `pt:Class` value or the tree will not display correctly.	n/a	`pt:Display='<classID1>:<objID1>,<classID2>:<objID2>,...'` `pt:Class='<classID1>,<classID2>,...'` `pt:RootID='1'`
`pt:Form` / `pt:Input`	Puts the AActivitySpace ID of the tree space into the target form input (used to reopen the tree after it has been created). The `pt:Form` attribute is the name of the parent form to which data will be passed back. The `pt:Input` attribute is	n/a	`pt:Form='<formName>'` `pt:Input='in_hi_parentSpace'`

Attribute	Description	Default	Syntax
	the name of the target input in the parent form. The AActivitySpace ID of the tree space is placed in this input.		
pt:Hide	Hides the specified objects. (See pt:openerLink for a list of Class IDs.)	n/a	pt:Hide='<ClassID1>,<ObjectID1>,<ClassID2>,<ObjectID2>,...'
pt:Multi	Allows users to select multiple items: true=checkboxes, false=radio buttons.	false	pt:Multi='true' or pt:Multi='false'
pt:Select	The default selected item(s) in the tree, referenced by Class ID and Object ID.	none	pt:Select='<ClassID1>,<ObjectID1>,<ClassID2>,<ObjectID2>,...'
pt:SelectMd	The tree select mode: 1=compositeselect, 2=leafselect, 3=leafcompositeselect (1 = select folders; 2 = select objects; 3 = select folders and objects).	2	pt:SelectMd='<modeID>'
pt:ShowRoot	Allows you to hide the root folder: true=display root folder, false=hide root folder (if false, subfolders are displayed at the top level).	true	pt:ShowRoot='true' or pt:ShowRoot='false'

Attribute	Description	Default	Syntax
pt:SubTitle	Subtitle of the tree, for user instruction (e.g., "Choose a user.").	none	pt:SubTitle='<windowSubtitle>'
pt:Title	Title of the tree popup window.	none	pt:Title='<windowTitle>'
pt:windowFeatures	Allows you to define the features argument for the resulting window.open() function call, specifying the features for a standard browser window.	(see syntax)	pt:windowFeatures='location=no,menubar=no,resizable=yes,height=400,width=400'
pt:windowName	Window name of the popup tree, used in the resulting window.open() function call.	'_blank1'	pt:windowName='<windowName>'
pt:Access	Advanced attribute. Access level for the objects to be displayed: None=0, Read=1, Select =3, Edit=7, Admin=15 Note: For objects in the Knowledge Directory (folders and documents), only two levels of security are available (0 or 1). Use pt:Access='1' to allow users access to Knowledge Directory objects.	3	pt:Access='<accessLevel>'

Attribute	Description	Default	Syntax
pt:CommunityMode / pt:CommunityID	Advanced attribute. Specifies whether to include community objects in the tree: 1=no communities, 2=this community (specified community + all parent communities), 3=all communities. Note: If CommunityMode=2, you must specify the community folder ID (not the community object ID) in the pt:CommunityID attribute.	1	pt:CommunityMode='<communityMode>' pt:CommunityMode='2' pt:CommunityID='<communityFolderID>'
pt:DirMode	Advanced attribute. Specifies which mode to use when selecting objects from the Knowledge Directory: 1=Browse Mode; 2=Edit Mode Note: The default mode is Edit (2); users who do not have edit access to the Knowledge Directory will see an "access denied" error when they access the tree.	2	pt:DirMode='<dirMode>'

For a full list of ALI object type class IDs, see *ALI Object Type Class IDs and Modes* on page 65.

The following code sample produces a button with an "onclick" action that opens a popup window.

```
<pt:standard.tree
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/' value="Button
Name" class="gContentSection" pt:windowName='myWindow'
pt:windowFeatures='location=no,menubar=no,height=500,width=300'
pt:RootID='1' pt:Multi='true' pt:SelectMd='2' pt:SubmitMd='2'
pt:Submit='PickerSubmit'
pt:Title='User' pt:SubTitle='Pick users' pt:Class='1'/>
```

The `pt:treeLink` tag can be used in the same way, except that it generates an anchor tag using the supplied text instead of a form button. In this tree control, the selection is limited to one user.

```
<pt:standard.treeLink
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'
pt:windowName='myWindow'
pt:windowFeatures='location=no,menubar=no,height=500,width=300'
pt:RootID='1' pt:Multi='false'
pt:SelectMd='2' pt:SubmitMd='2' pt:Submit='PickerSubmit'
pt:Title='User' pt:SubTitle='Pick
a user' pt:Class='1'>Pick a user</pt:standard.treeLink>
```

Clicking the link opens a popup window that allows the user to browse and choose the referenced object type, as shown in the image below. (If the popup was opened using the first code sample,

the tree would display checkboxes instead of radio buttons to allow multiple choices.)

As noted above, tree tags require a JavaScript function (named in the `pt:Submit` attribute) to handle the submission from the tree. The following sample code takes in an array with name, Object ID, and Class ID. When the `pt:Multi` attribute is set to false (single selections only), only the first set of declarations is necessary.

```
function PickerSubmit (myInput)
{
item0Name = myInput[0].Name;
item0ObjectID = myInput[0].ObjectID;
item0ClassID = myInput[0].ClassID;

item1Name = myInput[1].Name;
item1ObjectID = myInput[1].ObjectID;
item1ClassID = myInput[1].ClassID;
```

```
...
}
```

For optimum usability, the return array can be placed into hidden form elements and posted back to the source page so that the transformer link can specify which items should be selected if the user opens the dialog box again.

```
function returnFromFolderSelection(arrIn){
var tmpObject;
var iLength;

iLength = arrIn.length;


if (iLength > 0) {
tmpObject = arrIn[0];
document.Form1.HiddenSelectedFolderName.value = tmpObject.Name;
document.Form1.HiddenSelectedFolderObjectID.value =
tmpObject.ObjectID;
document.Form1.HiddenSelectedFolderClassID.value =
tmpObject.ClassID;
}
document.Form1.submit();
}
```

Securing Content Based on User Permissions Using Adaptive Tags

To insert content on a page based on conditional statements of user and group membership, use the pt:standard.choose, pt:standard.when and pt:standard.otherwise tags.

The `pt:standard.choose` tag denotes the start of a secured content section. The `pt:standard.when` tag includes a test condition (pt:test) that defines who has access to the enclosed content. The `pt:standard.otherwise` tag includes content that should be displayed by default.

Note: In ALI 6.1 and earlier, these tags are implemented as `pt:choose`, `pt:when` and `pt:otherwise`. This syntax is still supported.

The value for the `pt:test` attribute is case-sensitive. Multiple users or groups should be separated by commas, with semicolons separating user values from group values. The syntax is as follows:

```
<pt:standard.choose
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<pt:standard.when
pt:test='stringToACLGroup('user=userid1,userid2,...;group=groupid1,groupid2,groupid3;').isMember($currentuser)
 xmlns:pt='http://www.Plumtree.com/xmlschemas/ptui/'>
... content ...
```

```
</pt:standard.when>
<pt:standard.otherwise
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
... default content ...
</pt:standard.otherwise>
</pt:standard.choose>
```

For example:

```
<pt:standard.choose
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<pt:when
pt:test="stringToACLGroup('user=1;').isMember($currentuser)"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<title>welcome administrator</title></head>
... secret administrator content ...
</pt:standard.when>
<pt:standard.when
pt:test="stringToACLGroup('user=200,201;group=200;').isMember($currentuser)"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
 <title>the 200 club</title></head>
... content only group 200 or users 200 and 201 can see ...
</pt:standard.when>
<pt:standard.otherwise
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<title>everyone else</title></head>
... content any user can see ...
</pt:standard.otherwise>
</pt:standard.choose>
```

You can also test if the current user is a guest user (not logged in). Since there can be multiple guest users in the portal, simply testing for default guest user ID 2 does not work.

```
<pt:standard.choose>
  <pt:standard.when pt:test="isGuest($currentuser)">
  ... guest user content ...
  </pt:standard.when>
  <pt:standard.otherwise>
  ... logged in user content ...
  </pt:standard.otherwise>
</pt:standard.choose>
```

Logic Adaptive Tag Library (pt:logic)

Logic tags handle basic logic, including creating data objects and collections, setting shared variables, evaluating expressions, and looping over a data collection.

The pt:logic tag library is a cross-platform tag library that can be used in both ALI and Ensemble.

Note: Many logic tags have a `pt:scope` attribute. The valid scope values are: tag, portlet request, http request, session, persistent session, and application. The default is portlet request scope.

For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

Tag	Function	More Information
`pt:logic.data`	Creates a data object (collection of name=value pairs) and stores it in a shared variable using the key supplied.	*Using Shared Variables in Adaptive Tags* on page 231
`pt:logic.concat`	Concatenates two values into one and sets the new value in a variable with a specified name.	*Using Shared Variables in Adaptive Tags* on page 231
`pt:logic.variable`	Stores a shared variable using the key and value supplied. Designed for use with attribute replacement or with the `pt:logic.value` tag.	*Using Shared Variables in Adaptive Tags* on page 231
`pt:logic.collection`	Creates a collection of data objects and stores it in a shared variable using the key supplied.	*Using Shared Variables in Adaptive Tags* on page 231
`pt:logic.collectionlength`	Evaluates the length of a collection and stores the result in memory.	*Using Shared Variables in Adaptive Tags* on page 231
`pt:logic.value`	Evaluates an attribute and displays the referenced value. Used as singleton only (does not display the contents of the tag).	*Using Shared Variables in Adaptive Tags* on page 231

Tag	Function	More Information
`pt:logic.boolexpr`	Evaluates a boolean expression and stores the result as a boolean in memory. Designed to work with the `pt:logic.if` tag.	*Evaluating Expressions Using Adaptive Tags* on page 230
`pt:logic.intexpr`	Evaluates an integer expression and stores the result as a boolean in memory. Designed to work with the `pt:logic.if` tag.	*Evaluating Expressions Using Adaptive Tags* on page 230
`pt:logic.stringexpr`	Evaluates whether or not two strings are equal and stores the result as a boolean in memory. The case must match. Designed to work with the `pt:logic.if` tag.	*Evaluating Expressions Using Adaptive Tags* on page 230
`pt:logic.containsexpr`	Checks if a collection contains a specific data element and sets a specified variable to true or false. Designed to work with the `pt:logic.if` tag.	*Evaluating Expressions Using Adaptive Tags* on page 230
`pt:logic.if`	Evaluates an expression and displays either the `pt:logic.iftrue` or `pt:logic.iffalse` tag contents.	*Evaluating Expressions Using Adaptive Tags* on page 230
`pt:logic:iffalse`	Displayed if the surrounding `pt:logic.if` tag evaluates to false.	*Evaluating Expressions Using Adaptive Tags* on page 230
`pt:logic:iftrue`	Displayed if the surrounding `pt:logic.if` tag evaluates to true.	*Evaluating Expressions Using Adaptive Tags* on page 230

Tag	Function	More Information
pt:logic.foreach	Allows looping over a data collection. Supports tag and portlet request scope only.	*Looping Over Data Collections Using Adaptive Tags* on page 230
pt:logic.separator	Inserts a separator between the elements of a foreach loop.	*Looping Over Data Collections Using Adaptive Tags* on page 230

Evaluating Expressions Using Adaptive Tags

The pt:logic.boolexpr, intexpr, stringexpr and containsexpr tags work with the pt:logic.if tag to evaluate a range of expressions.

The sample code below determines whether the current value for the variable "title" is set to "Administrator". Variables can be set using the `pt:logic.data` or `pt:logic.variable` tags.

```
<pt:logic.stringexpr pt:expr="($title) ==
Administrator" pt:key="boolvalue"/>
<pt:logic.if pt:expr="$boolvalue">
<pt:logic.iftrue>
This is displayed if expr evaluates to true.
</pt:logic.iftrue>
<pt:logic.iffalse>
This is displayed if expr evaluates to false.
</pt:logic.iffalse>
</pt:logic.if>
```

For details on using shared variables, see *Using Shared Variables in Adaptive Tags* on page 231.

Looping Over Data Collections Using Adaptive Tags

The pt:logic.foreach tag allows you to loop over collections of data.

The sample code below creates a table to store links for a navigation menu.

```
<span
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<table cellpadding="5" cellspacing="0" width="100%"
 border="0">
<!-- loop starts here -->
<pt:logic.foreach pt:data="directorymenu"
```

```
pt:var="temp">
<tr>
<td height="25" colspan="3" class="navSidebarText">
<pt:core.html pt:tag="img" src="$temp.img" alt=""
border="0" align="absmiddle" height="20" width="20"
 />
<pt:core.html pt:tag="a" href="$temp.url">
<pt:logic.value pt:value="$temp.title" />
</pt:core.html>
</td>
</tr>
</pt:logic.foreach>
</table>
</span>
```

This table can then be populated with links using navigation tags. For details on navigation tags, see *Navigation Adaptive Tag Library (pt:plugnav)* on page 232.

Using Shared Variables in Adaptive Tags

The pt:logic.data, variable, and collection tags allow you to store editable shared variables, which can be used in attribute value replacement or with the pt:logic.value tag.

The `pt:logic.data` tag stores a data object (a name=value pair) as an editable shared variable using the key passed in. The `pt:logic.variable` tag stores an editable shared variable using the key and value passed in. If either tag is used inside the `pt:logic.collection` tag, the variables are stored directly in the parent collection. If the tag is used alone, the key attribute is required. The variable is only stored after the tag is finished processing all its contents. A collection can only contain a single type of variable, such as string variables or data objects.

Note: If a variable or collection with the same name already exists, it will be overwritten. If the preexisting variable is not editable, the tag will fail. Variable names cannot contain the reserved character '.'.

```
<pt:logic.variable pt:key="title" pt:value="Administrator"/>

<pt:logic.data pt:key="myurl" name="Home"
url="http://edocs.bea.com"/>

<pt:logic.collection pt:key="testcollection">
<pt:logic.data url="http://www.myco.com" name="My company"/>
<pt:logic.data url="http://www.otherco.com" name="Other company"/>
</pt:logic.collection>
```

```
<pt:logic.collection pt:key="teststringcollection">
<pt:logic.variable pt:value="my string data"/>
<pt:logic.variable pt:value="my other string data"/>
</pt:logic.collection>
```

The `pt:logic.value` tag displays the value of the variable referenced by the `pt:value` attribute. Variables can be set using the `pt:logic.data` or `pt:logic.variable` tags as explained in the previous section. This tag can be used to reference localized strings in message files.

```
<pt:logic.value pt:value="$title"/>
<pt:logic.value pt:value="$myurl.Home"/>
```

For details on referencing localized strings using tags, see *Using Internationalized Strings in Adaptive Tags* on page 205.

Navigation Adaptive Tag Library (pt:plugnav)

In AquaLogic Interaction 6.0 and above, customizing navigation can be implemented without coding against the portal UI. The Navigation tag library (pt:plugnav) is used to manage display of navigation elements.

The tags in the pt:plugnav tag library must be used with tags from the pt:ptata tag library. For details on implementing custom navigation using tags, see *Implementing Custom Navigation Using Adaptive Tags* on page 233. These tags are available for use only in ALI.

For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

Tag	Function
`pt:plugnav.ddmenurowcontainer`	Manages the display and positioning of navigation tabs that activate dropdown menus. (Only accepts `ddmenutab` or `ddmenusimpletabs` or equivalent as data.)
`pt:plugnav.ddmenusimpletabs`	Defines a list of simple tabs using the data provided. (Must be used with `ddmenurowcontainer` or equivalent.)
`pt:plugnav.ddmenutab`	Defines a tab that activates a dropdown menu with the data provided. (Must be used with `ddmenurowcontainer` or equivalent.)
`pt:plugnav.horizontalrowcontainermenu`	Generates and displays HTML for dynamic horizontal menus. (Only accepts `horizontalrowtab` or equivalent as data.)

Tag	Function
pt:plugnav.horizontalrowtab	Defines a horizontal menu tab that displays a row of links using the data provided. (Must be used with horizontalrowcontainermenu or equivalent.)

Implementing Custom Navigation Using Adaptive Tags

Navigation tags are used with Data tags to build complete navigation solutions for AquaLogic Interaction.

The first step is coding the portlet.

Initialize the menus by retrieving the navigation links using data tags. To create a collection, set the same ID on multiple data tags. For details on the ptdata tag library, see *Data Adaptive Tag Library (pt:ptdata)* on page 238.

```
<span xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<html>

<!-- Links to my pages are stored in mypagemenu -->
<pt:ptdata.mypageactionsdata pt:id='mypagemenu' />
<pt:ptdata.mypagesdata pt:id='mypagemenu' />

<!-- Links to my communities are stored in commmenu -->
<pt:ptdata.communityactionsdata pt:id='commmenu' />
<pt:ptdata.mycommunitiesdata pt:id='commmenu' />

<!-- Links to directory are stored in directorymenu -->
<pt:ptdata.directorybrowsedata pt:id='directorymenu' />
<pt:ptdata.directoryeditdata pt:id='directorymenu'/>

<!-- Mandatory links are stored in mandlinks-->
<pt:ptdata.mandatorylinksdata pt:id='mandlinks' />
<pt:ptdata.mandatorylinknamedata pt:key='mandlinksname'/>

<!--Links to administration and mandatory communites are stored
in menutabs -->
<pt:ptdata.administrationdata pt:id='menutabs' />
<pt:ptdata.mandatorycommunitiesdata pt:id='menutabs'/>
```

Next, create the structure to display the menus. To replace standard ALI navigation using a header portlet, use navigation tags to handle display as shown in the code sample below.

```
<!-- Dropdown menus section begin -->
<pt:plugnav.ddmenurowcontainer pt:menuvar='midrowmenu'
```

```
pt:hideemptymenus='true' >
<pt:plugnav.ddmenutab pt:containervar='midrowmenu'
pt:datavar='mypagemenu' pt:text='$#1840.ptmsgs_portalbrowsingmsgs'
 />
<pt:plugnav.ddmenutab pt:containervar='midrowmenu'
pt:datavar='commmenu' pt:text='$#1841.ptmsgs_portalbrowsingmsgs'
 />
<pt:plugnav.ddmenutab
pt:containervar='midrowmenu'pt:datavar='directorymenu'pt:text='$#1842.ptmsgs_portalbrowsingmsgs'
 />
<pt:plugnav.ddmenutab pt:containervar='midrowmenu'
pt:datavar='mandlinks' pt:text='$mandlinksname' />
<pt:plugnav.ddmenusimpletabs pt:containervar='midrowmenu'
pt:datavar='menutabs' />
</pt:plugnav.ddmenurowcontainer>
<!-- Dropdown menus section end -->
```

You can also display navigation links within a portlet, as shown in the sample code below.

```
<table cellpadding='0' cellspacing='0' width='200' border='0'>
  <tr>
    <td height='2' colspan='3'>
    </td>
  </tr>
  <tr class='menuHeaderBg'>
    <td align='left' valign='middle' height='20' colspan='3'
class='navSidebarSectionHeader'>
       My Communities
    </td>
  </tr>

<!-- links to communities are entered here -->
<pt:logic.foreach pt:data='commmenu' pt:var='temp'>
  <tr class='navMidtabBg'>
    <td height='16' colspan='2' class='navMidtabBtn'>
    <table cellpadding='0' cellspacing='0' width='100%'>
      <tr>
        <td height='20' width='100%' nowrap='nowrap' colspan='1'
 class='objectBtn'>
        <span class='actionbarBanText'>
          <pt:core.html pt:tag='img' src='$temp.img' alt=''
border='0' align='absmiddle' height='20' width='20' />
          <pt:core.html pt:tag='a' href='$temp.url'>
            <pt:logic.value pt:value='$temp.title' />
          </pt:core.html>
        </span>
```

```
        </td>
      </tr>
    </table>
    </td>
  </tr>
</pt:logic.foreach>
</table>
```

You can also add portal UI elements to custom navigation using UI tags. For details on UI tags, see *UI Adaptive Tag Library (pt:ptui)* on page 235.

To deploy a custom navigation header portlet in ALI (to replace standard navigation), follow the steps below.

1. Register the portlet in the portal.
2. Create an Experience Definition that uses the custom navigation header portlet you registered in step 1.
3. Create an Experience Rule to direct users to the new Experience Definition. For details on Experience Definitions and Experience Rules, see the *Administrator Guide for AquaLogic Interaction*.

UI Adaptive Tag Library (pt:ptui)

UI tags allow you to add standard portal UI components to any portlet in the ALI portal, including search inputs and buttons, login components, access to account settings, and more.

The tags in the pt:ptui tag library are available for use only in ALI (not in Ensemble). Additional tags from the pt:standard tag library can be used to display instance-specific information, including the date and time and the page name and type. For details, see *Standard Adaptive Tag Library (pt:standard)* on page 212.

For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

For an example of implementing UI customization using tags, see *Implementing Custom UI Elements Using Adaptive Tags* on page 237.

Tag	Function
`pt:ptui.welcome`	Displays the user's personalized welcome message. Used as singleton only (does not display the contents of the tag).

`pt:ptui.myhome`	Displays a link to the user's home page (MyPage or community). Can be used as singleton or wrapper for HTML.
`pt:ptui.myaccount`	Displays a link to the user's My Account page. Can be used as singleton or wrapper for HTML.
`pt:ptui.createaccount`	Displays a link to the Create Account page. Can be used as singleton or wrapper for HTML.
`pt:ptui.searchform`	Displays the basic search form without any buttons or links.
`pt:ptui.basicsearchbutton`	Displays the basic search button. Can be used as singleton or wrapper for HTML.
`pt:ptui.advancedsearchbutton`	Displays the advanced search button. Can be used as singleton or wrapper for HTML.
`pt:ptui.federatedsearchbutton`	Displays the federated search button. Can be used as singleton or wrapper for HTML.
`pt:ptui.topbestbetsearchbutton`	Displays the top best bet button. Can be used as singleton or wrapper for HTML. (Must be used with pt:ptui.searchform.)
`pt:ptui.help`	Displays the help image and link. Can be used as singleton or wrapper for HTML.
`pt:ptui.login`	Displays a login/logoff link based on the current state of the user. (If the user is logged in, the URL executes logoff; if the user is not logged in, the URL executes login.)
`pt:ptui.loginform`	Outputs the basic login form without any buttons or links.
`pt:ptui.loginusername`	Displays the user name text box for the login form.
`pt:ptui.loginpassword`	Displays the password text box for the login form.
`pt:ptui.loginbutton`	Displays the login button.

pt:ptui.loginauthsource	Displays the authentication source input. Note: This tag is string- and case-sensitive. The name of the authentication source must match the entry in portalconfig.xml.
pt:ptui.loginrememberme	Displays the "Remember My Password" checkbox for the login form.
pt:ptui.loginoptionsenabled	Conditionally processes content based on the parameters specified (e.g., remembermypassword).

pt:ptui.error and pt.ptui.errortext	Displays portal error messages. Can be used as singleton or wrapper for formatted error text. The ptui.errortext tag is used to reformat or modify error message text.
	`<pt:ptui.error>`
	`<p style="msg1">`
	`<pt:ptui.errortext pt:text="Call support at 555-1212"/>`
	`</p>`
	`</pt:ptui.error>`

pt:ptui.include	Used to include JSComponent scripts, string files and css files.
pt:ptui.rulesdebug	Displays a debug button to display experience rules debugging messages in a popup window. Can be used as singleton or wrapper for HTML.

Implementing Custom UI Elements Using Adaptive Tags

UI tags can be used to insert ALI UI elements in portlets to create custom UI layouts.

The sample code below implements standard portal header elements using tags. You can also add navigation elements to any portlet using Navigation Tags. For details, see *Navigation Adaptive Tag Library (pt:plugnav)* on page 232. Additional tags from the Standard tag library can be used to display instance-specific information, including the date and time and the page name and type. For details, see *Standard Adaptive Tag Library (pt:standard)* on page 212 .

```
<span xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>

<!-- Topbar -->
<table cellpadding="0" cellspacing="0" width="100%" border="0"
```

```
class="banTopbarBg" id="pt-topbar">
  <tr>
    <td align="left" valign="middle" nowrap="nowrap">
      <pt:ptui.myhome pt:usespan="true"/>
      <span class="banGreetingText banText" id="pt-user-nav">
      <pt:ptui.welcome pt:usespan="true" />
      <span class="spacer" style="padding-left:8px;"></span>
      <pt:ptui.myaccount pt:usespan="true" />
      <span class="spacer" style="padding-left:8px;"></span>
      <pt:ptui.login pt:usespan="true"/>
      </span>
    </td>
    <td align="right" valign="middle" nowrap="nowrap">
      <pt:ptui.rulesdebug/>
      <pt:ptui.help/>
      <pt:ptui.searchform pt:usespan="true">
      <pt:ptui.basicsearchbutton/>
      <pt:ptui.advancedsearchbutton/>
      <pt:ptui.federatedsearchbutton/>
      </pt:ptui.searchform>
    </td>
  </tr>
</table>
<!-- Topbar section end -->
</span>
```

This code creates the following header:

Data Adaptive Tag Library (pt:ptdata)

The Data tag library (pt:ptdata) provides access to URLs for most navigation-related components, such as a user's my pages, my communities, subcommunities, my account page or administration.

The tags in the pt:ptdata tag library are available for use only in ALI (not in Ensemble). Data tags return URL attributes as data; they must be used in conjunction with a display tag (navigation tags or pt:core.html).

Each data tag requires an ID that is set with an attribute and returns a single URL, a collection of URLs, or nothing. Data tags might return no URL at all if a user does not have access to the

referenced page. You can also create a collection of data tags by setting the same ID on multiple data tags.

```
<pt:ptdata.mypageactionsdata pt:id="mypagemenu" />
<pt:ptdata.mypagesdata pt:id="mypagemenu" />
```

In addition to the URL, each navigation data tag also provides additional information, such as the title of the URL and the icon associated with the URL. Certain types of URLs also contain objectIDs, classIDs, or a current page flag. It is also possible to get values for individual query string parameters from an URL. The URL and all other data is stored as a dataobject (DO) component. Each DO component can be accessed through a text replacement syntax. Data tags take in the following URL attributes: title, url, uri, img, imgwidth, imgheight, and params. For example, the following code gets the title and URL component from the mydata URL.

```
<pt:ptdata.administrationdata pt:id="mydata" />
<pt:logic.value pt:value="$mydata.title"/>
<pt:logic.value pt:value="$mydata.url"/>
```

After transformation, this code generates the following data: "Administration http://servername/portal/server.pt?open=space&name=ObjMgr&parentid=7&parentname=ObjMgr&control=AdminRedirect&in_hi_userid=1&cached=true

The tables below summarize available data tags. For an example of implementing custom navigation using data tags, see *Implementing Custom Navigation Using Adaptive Tags* on page 233. For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

Basic Portal Components: These tags provide URLs to access standard portal components, including login/logoff, Administration, Knowledge Directory, search, and online help.

Tag	Function
`pt:ptdata.loginlogoffdata`	Returns URL to Login/Logoff action based on the current state of the user. (If the user is logged in, the URL executes logoff; if the user is not logged in, the URL executes login.)
`pt:ptdata.myaccountdata`	Returns URL to current user's My Account page.
`pt:ptdata.administrationdata`	Returns URL to portal Administration. The URL is only returned if the user has permission to access Administration.
`pt:ptdata.directorybrowsedata`	Returns URL to the portal Knowledge Directory in browse mode.

Tag	Function
pt:ptdata.directoryeditdata	Returns URL to the portal Knowledge Directory in edit mode.
pt:ptdata.advancedsearchdata	Returns URL to the Advanced Search page.
pt:ptdata.federatedsearchdata	Returns URL to the Federated Search page.
pt:ptdata.helppagedata	Returns URL to the portal online help.
pt:ptdata.genericurl	Returns URL based on parameters set in tag attributes.

MyPages: These tags provide URLs to MyPage components, including editors.

Tag	Function
pt:ptdata.mypagesdata	Returns a list of URLs to the user's MyPages.
pt:ptdata.mypageactionsdata	Returns a list of URLs to the user's MyPage-related actions.
pt:ptdata.editmypageactionsdata	Returns URL to launch the Edit MyPage editor.
pt:ptdata.editmypageportletprefsdata	Returns URL to launch the Edit MyPage Portlet Preferences editor.
pt:ptdata.createnewmypagedata	Returns URL to launch the Create New MyPage editor. The URL is not returned if the user already has the maximum number of MyPages.
pt:ptdata.addmypageportletsdata	Returns URL to launch the Add Portlets to MyPages editor.
pt:ptdata.deletemypagedata	Returns URL to the Delete MyPage action. The URL is not returned if the user is on the main MyPage.

Experience Definitions: These tags provide URLs to experience-specific components as specified in the experience definition associated with the current user.

Tag	Function
pt:ptdata.myhomedata	Returns URL to current user's Home page as specified in the associated experience definition.

Tag	Function
pt:ptdata.mandatorylinksdata	Returns a list of URLs to the user's Mandatory Links as specified in the associated experience definition.
pt:ptdata.mandatorylinksnamedata	Returns the name of the Mandatory Links folder as a string.

Communities: These tags provide URLs to community components, and lists of URLs for community pages that meet specific conditions, including subcommunities, related communities, and a user's current communities.

Tag	Function
pt:ptdata.mycommunitiesdata	Returns a list of URLs to the communities in the user's My Communities list.
pt:ptdata.communitypagesdata	Returns a list of URLs to the pages in the specified community.
pt:ptdata.currcommunitypagesdata	Returns a list of URLs to the pages in the current community.
pt:ptdata.subcommunitiesdata	Returns a list of URLs to the subcommunities for the specified community.
pt:ptdata.currsubcommunitiesdata	Returns a list of URLs to the subcommunities for the current community.
pt:ptdata.relatedcommunitiesdata	Returns a list of URLs to the related communities for the specified community.
pt:ptdata.currrelatedcommunitiesdata	Returns a list of URLs to the related communities for the current community.
pt:ptdata.communitykddata	Returns the URL to the community Knowledge Directory
pt:ptdata.communityactionsdata	Returns a list of URLs to the user's community-related actions.
pt:ptdata.editcommunitydata	Returns URL to launch the Community Editor for the current community.

Tag	Function
pt:ptdata.createnewcommpagedata	Returns URL to launch the Create New Community Page page of the Community Editor. The URL is returned only if the user has permission to edit the community.
pt:ptdata.addcommunityportletsdata	Returns URL to launch the Add Portlets page of the Community Editor. The URL is returned only if the user has permission to edit the community.
pt:ptdata.joincommunitiesdata	Returns URL to launch Join Communities editor.
pt:ptdata.joinparentcommunitydata	Returns URL to launch Join Communities editor for the parent Community of the current Community.
pt:ptdata.joincurrcommunitydata	Returns URL to the Join Current Community action.
pt:ptdata.joincurrparentcommunitydata	Returns URL to the Join Current Community action for the parent Community of the current Community.
pt:ptdata.unsubscribecommunitiesdata	Returns URL to the Unsubscribe Communities editor.
pt:ptdata.navsettingvalue	Returns a list of URLs to the communities listed in the NavigationSettings.xml file, specified by the commID attribute.

You can also access Pages LiveSpace and DataSpace URLs using tags; *Pages Adaptive Tag Library (pt:pages)* on page 246.

Ensemble Adaptive Tag Library (pt:ensemble)

The Ensemble tag library (pt:ensemble) provides tags to insert pagelets in Ensemble consumer pages and access authentication and role information for Ensemble resources.

In addition to the tags below, Ensemble pagelets can use cross-platform libraries to access constants and implement logic. For details, see *About Cross-Platform Adaptive Tag Libraries* on page 203.

For a full list of tags and attributes, see the tagdocs. Tagdocs for all platforms are available on the *API Libraries* on page 519 page.

Tag	Function	More Information
`pt:ensemble.inject`	Injects the output of the specified pagelet into the page.	*Inserting Pagelets Using Ensemble Adaptive Tags* on page 245
`pt:ensemble.resourcedata`	Stores data for a specific Ensemble resource, if available, in memory as a data object containing information about the resource.	*Accessing Resource Data Using Ensemble Adaptive Tags* on page 244
`pt:ensemble.authsourcedata`	Stores a list of available authentication sources for the currently requested resource in memory.	*Accessing Authentication Data Using Ensemble Adaptive Tags* on page 244
`pt:ensemble.loginlink`	Stores the URL prefix for the login page in memory using the given key and scope.	*Accessing the Login URL Using Ensemble Adaptive Tags* on page 244
`pt:ensemble.roleexpr`	Evaluates a role expression and stores the result as a boolean in memory. Designed to work with the pt:logic.if tag.	*Accessing User Roles Using Ensemble Adaptive Tags* on page 245
`pt:ensemble.rolelist`	Stores a list of the roles for the current user in memory.	*Accessing User Roles Using Ensemble Adaptive Tags* on page 245
`pt:ensemble.ssologout`	Notifies Ensemble that the current user should be logged out of all resources in Ensemble. Used as singleton only (does not display the contents of the tag).	

Accessing Authentication Data Using Ensemble Adaptive Tags

The pt:ensemble.authsourcedata tag stores a list of available authentication sources for the currently requested resource in memory.

The data is stored as a collection, and each item in the collection is a data object containing information about the authentication source (prefix, name, description) accessible through the data object dot notation (`$curauth.name`).

```
<pt:ensemble.authsourcedata pt:key="sources"/>
<pt:logic.foreach pt:data="sources" pt:var="source">
<pt:logic.value pt:value="$source.prefix"/>
<pt:logic.value pt:value="$source.name"/>
<pt:logic.value pt:value="$source.description"/>
<pt:logic.separator><br>-----<br></pt:logic.separator>
</pt:logic.foreach>
```

This example uses logic tags to display information about each authentication source. For details on logic tags, see *Logic Adaptive Tag Library (pt:logic)* on page 227.

Accessing the Login URL Using Ensemble Adaptive Tags

The pt:ensemble.loginlink tag stores the URL prefix for the login page in memory using the given key and scope.

The login prefix will end with a forward slash. This login prefix should be followed by the page suffix for the page that should be displayed after login. For example, if the external URL prefix of the resource is http://www.ensemble.com/app/ and the desired page after login is http://www.ensemble.com/app/pages/mainpage.html, then the full login link would be made by adding pages/mainpage.html to the login link prefix.

```
<pt:ensemble.loginlink pt:level="4"
pt:key="loginurlprefix"/>
var loginLink = "<pt:logic.value
pt:value="$loginurlprefix"/>" +
"pages/mainpage.html";
```

Accessing Resource Data Using Ensemble Adaptive Tags

The pt:ensemble.resourcedata tag stores data for a specific Ensemble resource, if available, in memory as a data object.

The data object contains information about the resource (name, description, urlprefix, secureurlprefix) accessible through the data object dot notation (`$resource.name`). If the resource does not have a description, urlprefix, or secureurlprefix, the data will not be available.

```
<pt:ensemble.resourcedata pt:name="Welcome Resource"
 pt:key="resource"/>
<pt:logic.value pt:value="$resource.name"/>
<pt:logic.value pt:value="$resource.description"/>
<pt:logic.value pt:value="$resource.urlprefix"/>
<pt:logic.value
pt:value="$resource.secureurlprefix"/>
```

Accessing User Roles Using Ensemble Adaptive Tags

The role* tags in the Ensemble tag library allow pagelets to modify content based on the role of the current user.

For details on roles, see *About Ensemble Security* on page 333.

The `pt:ensemble.roleexpr` tag evaluates a role expression and stores the result as a boolean in memory. This tag is designed to work with the `pt:logic.if` tag as shown below.

```
<pt:ensemble.roleexpr pt:expr="hasRole Default Role from Seed
State" pt:key="boolvalue"/>
<pt:logic.if pt:expr="$boolvalue">
<pt:logic.iftrue>
  <!-- This is displayed if expr evaluates to true. -->
</pt:logic.iftrue>
<pt:logic.iffalse>
  <!-- This is displayed if expr evaluates to false. -->
</pt:logic.iffalse>
</pt:logic.if>
```

The `pt:ensemble.rolelist` tag stores a list of the roles for the current user in memory. The data is stored as a collection, and each item in the collection is a variable containing the role name. This can be used with the `logic.foreach` tag to iterate over role data as shown below.

```
<pt:ensemble.rolelist pt:key="roles"/>
<pt:logic.foreach pt:data="roles" pt:var="role">
<pt:logic.value pt:value="$role"/>
<pt:logic.separator></pt:logic.separator>
</pt:logic.foreach>
```

For details on logic tags, see *Logic Adaptive Tag Library (pt:logic)* on page 227.

Inserting Pagelets Using Ensemble Adaptive Tags

The pt:ensemble.inject tag injects the output of the specified pagelet into the page.

The pagelet is referenced by the fully qualified name of the pagelet as defined in Ensemble, in the form `libraryname:pageletname`.

Any non-ALI attributes (not prefixed with "pt:") will be passed on to the pagelet. Any HTML content inside the pagelet tag will be passed to the pagelet as an XML payload.

```
<pt:ensemble.inject pt:name="mylibrary:mypagelet"
pagelet-attribute="A pagelet attribute">
<?xml version="1.0" encoding="utf-8"?>
<doc>This is an XML payload.</doc>
</pt:ensemble.inject>
```

For an example of using this code, see *Creating a Custom Pagelet with the Java IDK Proxy API* on page 329 or *Creating a Custom Pagelet with the .NET IDK Proxy API* on page 330.

Pages Adaptive Tag Library (pt:pages)

AquaLogic Pages is delivered with a set of adaptive tags to add links to Pages LiveSpaces and DataSpaces to pagelets or portal navigation, allowing users to navigate to Pages components from within an ALI or Ensemble page.

These tags are data tags; they return URL attributes as data, and must be used in conjunction with a display tag. For details on data tags, see *Data Adaptive Tag Library (pt:ptdata)* on page 238. For an example of using these tags, see *Adding Links to Pages LiveSpaces and DataSpaces Using Adaptive Tags* on page 246.

Tag	Function
`pt:pages.livespacesdata`	Returns URLs to available Pages LiveSpaces.
`pt:pages.dataspacesdata`	Returns URLs to available Pages DataSpaces.

Adding Links to Pages LiveSpaces and DataSpaces Using Adaptive Tags

To add links to Pages LiveSpaces and DataSpaces to pagelets or portal navigation, use the tags in the Pages adaptive tag library.

Pages tags retrieve LiveSpaces and DataSpaces by making requests to web services on the Pages server. The results are cached for 5 minutes or until the user logs out. The tags return only the LiveSpaces and DataSpaces to which the current user has access.

You can add `pt:pages` data tags to the navigation in any custom header portlet using the syntax shown below.

```
<pt:pages.livespacesdata pt:id="livespacemenu" />
<pt:pages.dataspacesdata pt:id="dataspacemenu" />

<pt:plugnav.ddmenurowcontainer pt:menuvar="midrowmenu"
pt:hideemptymenus="true" >
<pt:plugnav.ddmenutab pt:containervar="midrowmenu"
pt:datavar="mypagemenu" pt:text="$#1840.ptmsgs_portalbrowsingmsgs"
 />
<pt:plugnav.ddmenutab pt:containervar="midrowmenu"
pt:datavar="commmenu" pt:text="$#1841.ptmsgs_portalbrowsingmsgs"
 />
<pt:plugnav.ddmenutab pt:containervar="midrowmenu"
pt:datavar="directorymenu"
pt:text="$#1842.ptmsgs_portalbrowsingmsgs" />
<pt:plugnav.ddmenutab pt:containervar="midrowmenu"
pt:datavar="mandlinks" pt:text="$mandlinksname" />
<pt:plugnav.ddmenutab pt:containervar="midrowmenu"
pt:datavar="livespacemenu pt:text="Livespaces" />
<pt:plugnav.ddmenutab pt:containervar="midrowmenu"
pt:datavar="dataspacemenu" pt:text="Dataspaces" />
<pt:plugnav.ddmenusimpletabs pt:containervar="midrowmenu"
pt:datavar="menutabs" />
</pt:plugnav.ddmenurowcontainer>
```

For information and instructions on deploying custom navigation elements, see *Implementing Custom Navigation Using Adaptive Tags* on page 233.

You can also add LiveSpaces, DataSpaces and other Pages components to a portal page using custom portlets. For details, see *Creating a Portlet from a Page Component* in the *Administrator Guide for AquaLogic Pages*.

Pathways Search Adaptive Tag Library (pt:pathways)

AquaLogic Pathways is delivered with a set of adaptive tags to add Pathways UI elements to the ALI portal UI, allowing users to access Pathways search from a portal page.

For all Pathways tags, the `pt:text` parameter defines the text to be displayed in the UI element. If the `pt:text` parameter is not provided, no text will be displayed. For details on implementing Pathways tags, see *Adding Pathways Search Using Adaptive Tags* on page 248.

Tag	Function
pt:pathways.pathwayssearchform	Adds a Pathways search box. In Pathways 1.5, two optional attributes are available: • pt:searchtabtype: Either "u" or "d" indicating whether searches submitted from the banner search form should be presented with the user tab or documents tab initially displayed. • pt:subquery: A string to be appended to the search terms provided by the end user in the search box. For example, if this attribute is set to "and user" and the user entered the search term "Knowledge Directory" the resulting search presented in the Pathways UI would be executed as "Knowledge Directory and user".
pt:pathways.pathwayssearchbutton	Adds a Pathways search button.
pt:pathways.pathwayshome pt:id='menutabs'	Returns the URL to the Pathways home page. This tag is a data tag and must be used in conjunction with a display tag . For details on data tags, see *Data Adaptive Tag Library (pt:ptdata)* on page 238.

Adding Pathways Search Using Adaptive Tags

Pathways tags allow you to add Pathways search elements to pagelets and page components in ALI portal and Ensemble.

This example adds a Pathways search box and button to the top bar in the portal banner, a tab to the main portal menu, and a link to the portal Directory tab. This example includes tags from the UI Elements (pt:ui) and Navigation (pt:plugnav) Adaptive Tag libraries. For information on deploying custom navigation elements, see *Implementing Custom Navigation Using Adaptive Tags* on page 233.

```
<span xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<!-- Topbar -->
<table cellpadding='0' cellspacing='0' width='100%' border='0'
class='banTopbarBg' id='pt-topbar'>
 <tr>
```

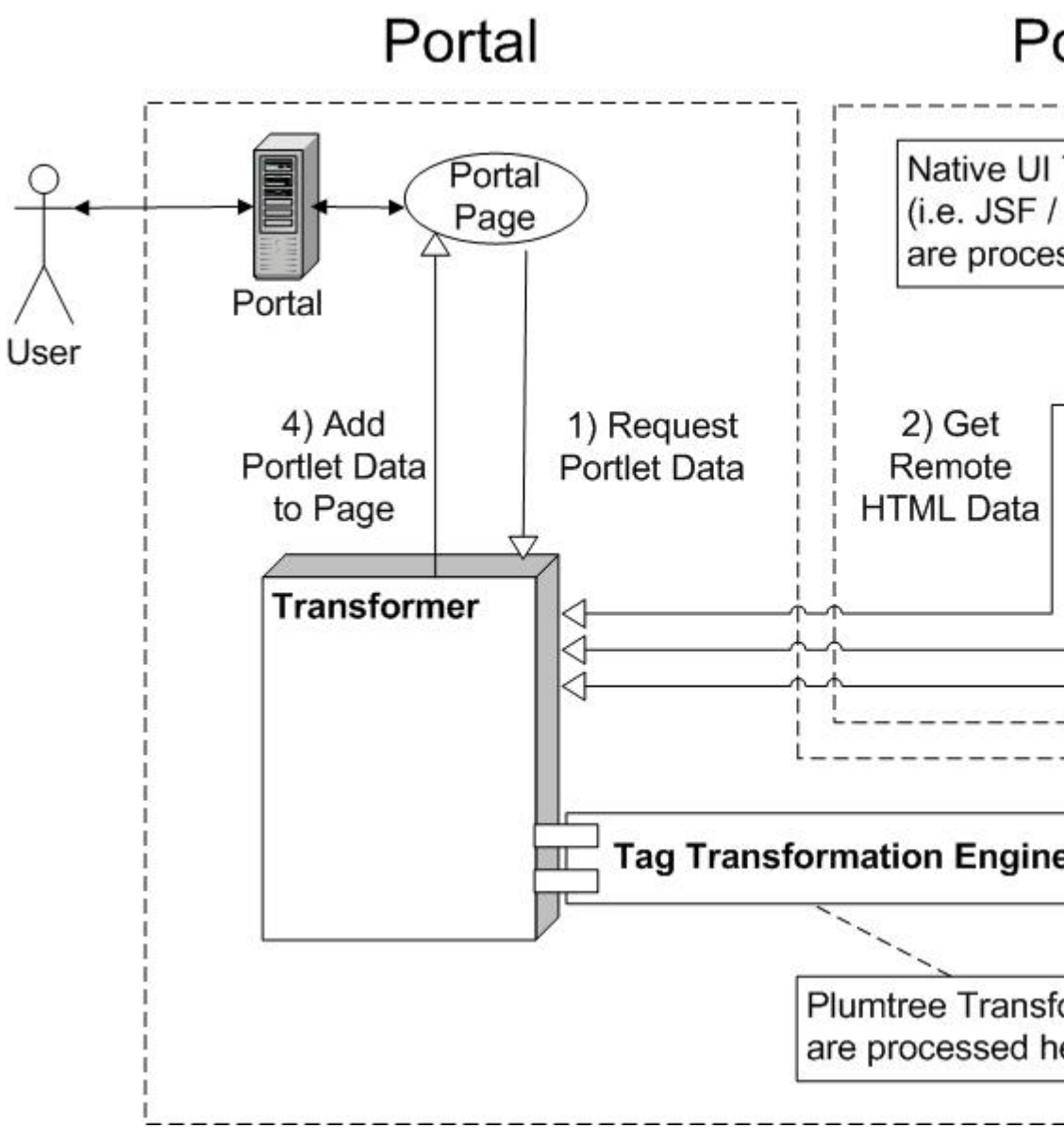
```
   <td align='left' valign='middle' nowrap='nowrap'>
   <pt:ptui.myhome pt:usespan='true'/>
   <span class='banGreetingText banText' id='pt-user-nav'>
      <pt:ptui.welcome pt:usespan='true' />
     <span class='spacer' style='padding-left:8px;'></span>
     <pt:ptui.myaccount pt:usespan='true' />
     <span class='spacer' style='padding-left:8px;'></span>
     <pt:ptui.login pt:usespan='true'/>
   </span>
   </td>
   <td align='right' valign='middle' nowrap='nowrap'>
   <pt:ptui.rulesdebug/>
   <pt:ptui.help/>
   <pt:ptui.searchform pt:usespan='true'>
   <pt:ptui.basicsearchbutton/>
   <pt:ptui.advancedsearchbutton/>
   <pt:ptui.federatedsearchbutton/>
   </pt:ptui.searchform>
   <!-- Add the Pathways Banner Search Elements -->
   <pt:pathways.pathwayssearchform pt:usespan='true'
pt:text='Pathways:' pt:searchtabtype="u" pt:subquery="and user">>

     <pt:pathways.pathwayssearchbutton/>
   </pt:pathways.pathwayssearchform>
   </td>
 </tr>
</table>
<!-- Topbar section end -->

<!-- Dropdown menus section begin -->
<pt:ptdata.communityactionsdata pt:id='commmenu' />
<pt:ptdata.mycommunitiesdata pt:id='commmenu' />
<pt:ptdata.mandatorylinksdata pt:id='mandlinks' />
<pt:ptdata.mandtabcommsdata pt:id='menutabs' />
<pt:ptdata.administrationdata pt:id='menutabs' />
 <!-- Add Pathways Home link as a menu tab. -->
<pt:pathways.pathwayshome pt:id='menutabs' pt:text='Pathways
Home'/>
<pt:ptdata.mypageactionsdata pt:id='mypagemenu' />
<pt:ptdata.mypagesdata pt:id='mypagemenu' />
<pt:ptdata.currcommunitypagesdata pt:id='commpages' />
<pt:ptdata.currsubcommunitiesdata pt:id='subcomms' />
<pt:ptdata.currrelatedcommunitiesdata pt:id='relcomms' />
<pt:ptdata.directorybrowsedata pt:id='directorymenu' />
<pt:ptdata.directoryeditdata pt:id='directorymenu' />
```

```
 <!-- Add Pathways Home link to the directory menu. -->
<pt:pathways.pathwayshome pt:id='directorymenu' pt:text='Pathways
 Home'/>
<pt:ptdata.mandatorylinknamedata pt:key='mandlinksname' />
<pt:plugnav.ddmenurowcontainer pt:menuvar='midrowmenu'
pt:hideemptymenus='true' >
<pt:plugnav.ddmenutab pt:containervar='midrowmenu'
pt:datavar='mypagemenu'
pt:text='$#1840.ptmsgs_portalbrowsingmsgs'/>
<pt:plugnav.ddmenutab pt:containervar='midrowmenu'
pt:datavar='commmenu' pt:text='$#1841.ptmsgs_portalbrowsingmsgs'/>
<pt:plugnav.ddmenutab pt:containervar='midrowmenu'
pt:datavar='directorymenu'
pt:text='$#1842.ptmsgs_portalbrowsingmsgs' />
<pt:plugnav.ddmenutab pt:containervar='midrowmenu'
pt:datavar='mandlinks' pt:text='$mandlinksname' />
<pt:plugnav.ddmenusimpletabs pt:containervar='midrowmenu'
pt:datavar='menutabs' />
</pt:plugnav.ddmenurowcontainer>
<!-- Dropdown menus section end -->
...
</span>
```

About Adaptive Tag Control Flow

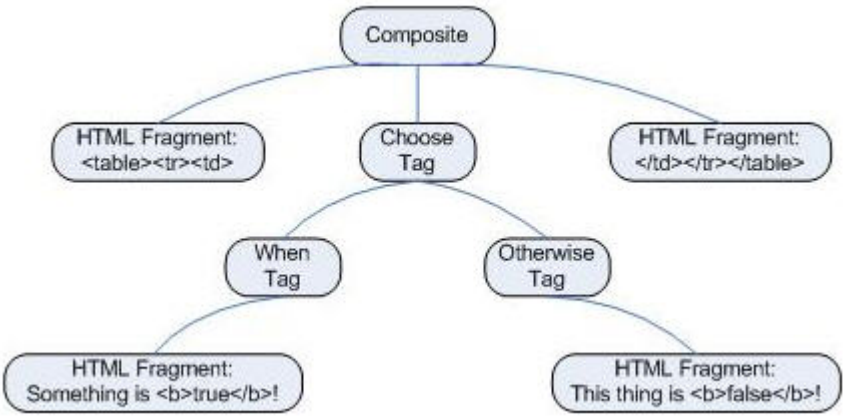
This page describes the control flow of a typical portal request that makes use of Adaptive Tags.

1. First, the portal page requests portlet data from the Transformer.
2. The Transformer retrieves the requested portlets from the remote portlet servers. Native UI tags, such as JSP Tags or .NET web controls, are processed on the remote server before the HTML is returned to the Transformer.
3. The Transformer converts the HTML into markup data including both HTML and Adaptive Tags. This markup data is passed to the Tag Transformation Engine, which processes the tags and converts them into standard HTML.
4. Finally, the HTML is returned to the portal page where it is displayed to the end user.



The **Tag Transformation Engine** converts markup data from the Transformer into a tree of HTML and Adaptive Tags. The Engine moves through the tree and outputs HTML and processes the tags. When a tag is processed, it can cause all of its child nodes to be processed, or it can skip that entire section of the tree.

The figure below shows an example of a tree. In this example, when the `choose` tag is executed, it determines whether or not the current user matches the conditions in the choose clause. If it does, the When tag will display the HTML inside the tag. If not, the `otherwise` tag will display its HTML.



For details on these tags, see *Logic Adaptive Tag Library (pt:logic)* on page 227.

Creating Custom Adaptive Tags

The Adaptive Tag Framework allows you to create custom tags for use in pagelets and gatewayed pages.

The `ATag` class is the base class used to write custom tags

To implement a custom tag, follow the steps below.

1. . To implement a new tag, you must have a tag library. A tag library is simply a .jar or .dll file with exactly one class that implements `ITagLibraryMetaData`.

   Java

   ```
   public static final TagLibraryMetaData LIBRARY = new
   TagLibraryMetaData
   ("Sample Tags", "sample", "This library provides sample tags.",
    1.0);
   ```

.NET

```
public static readonly TagLibraryMetaData LIBRARY = new
TagLibraryMetaData
("Sample Tags",  "sample", "This library provides sample tags.",
 1.0);
```

2. Create one public static final `ITagMetaData` member variable that provides the name and description of the tag. Create a public static final `RequiredTagAttribute` or `OptionalTagAttribute` member variable for every attribute that the tag supports. You can also use standard HTML and XML attributes; see *Accessing Attributes in Custom Adaptive Tags* on page 255.

Java

```
public static final ITagMetaData TAG;
public static final RequiredTagAttribute MESSAGEATTRIBUTE;
public static final OptionalTagAttribute LOCATIONATTRIBUTE;

static
{
TAG = new TagMetaData("hellolocation", "This tag displays a
hello message for the given location.");
MESSAGEATTRIBUTE = new RequiredTagAttribute( "message", "The
message to display for hellolocation tag",
AttributeType.STRING);
LOCATIONATTRIBUTE = new OptionalTagAttribute("location", "The
sample location attribute for hellolocation tag",
AttributeType.STRING, "World");
}
```

.NET

```
public static readonly ITagMetaData TAG;
public static readonly RequiredTagAttribute MESSAGEATTRIBUTE;
public static readonly OptionalTagAttribute LOCATIONATTRIBUTE;

static HelloLocationTag()
{
TAG = new TagMetaData("hellolocation", "This tag displays a
hello message for the given location.");
MESSAGEATTRIBUTE = new RequiredTagAttribute( "message", "The
message to display for hellolocation tag",
AttributeType.STRING);
LOCATIONATTRIBUTE = new OptionalTagAttribute("location", "The
sample location attribute for hellolocation tag",
```

```
AttributeType.STRING, "World");
}
```

Type validation is performed by the tag framework automatically. If an optional attribute is not present in the HTML, the tag framework will use the default value. In the same code below, the optional attribute has a default value of "World.".

3. Implement the `DisplayTag` abstract method. Use this method to create and display HTML. To display any HTML and tags defined within the tag, call `ProcessTagBody` and return the resulting HTML. The sample code below adds the "Hello" string with a user-specified location to an `HTMLElement` and returns it to be displayed.

   Java

   ```
   public HTMLElement DisplayTag()
   {
   String strLocation = GetTagAttributeAsString(LOCATIONATTRIBUTE);
   String strMessage = GetTagAttributeAsString(MESSAGEATTRIBUTE);
   HTMLElementCollection result = new HTMLElementCollection();
   result.AddInnerHTMLString(strMessage + strLocation + "!");
   return result;
   }
   ```

   .NET

   ```
   public override HTMLElement DisplayTag()
   {
   String strLocation = GetTagAttributeAsString(LOCATIONATTRIBUTE);
   String strMessage = GetTagAttributeAsString(MESSAGEATTRIBUTE);
   HTMLElementCollection result = new HTMLElementCollection();
   result.ddInnerHTMLString(strMessage + strLocation + "!");
   return result;
   }
   ```

4. If the tag should not display any HTML contained within the tag, use the `GetTagType` method to return TagType.NO_BODY.

   Java

   ```
   public TagType GetTagType()
   {
   return TagType.NO_BODY;
   }
   ```

   .NET

   ```
   public override TagType GetTagType()
   {
   ```

```
return TagType.NO_BODY;
}
```

5. Implement the Create abstract method to return a new instance of the tag.

Java

```
public ATag Create()
{
return new HelloLocationTag();
}
```

.NET

```
public override ATag Create()
{
return new HelloLocationTag();
}
```

The ATag class allows you to include a wide range of functionality in custom tags. For a full list of interfaces and methods, see the tagdocs. For links to all tagdocs, see *API Libraries* on page 519. For details on deploying your custom tag, see *Deploying Custom Adaptive Tags* on page 259.

Accessing Browser Session Information in Custom Adaptive Tags

To access browser session information from a custom adaptive tag, use the `IEnvironment` class.

The `IEnvironment` class provides access to information about the current request and user, including the following:

- **HTTP Request and Response**: Retrieve the Request or Response objects, or the Request URL. For example: `IXPRequest request = GetEnvironment().GetCurrentHTTPRequest();`
- **User information**: Retrieve the user's session, or key information including language, locale, time zone, and access style (standard, 508, or low bandwidth). For example: `String strTZ = GetEnvironment().GetTimeZone();`
- **VarPacks**: Retrieve any VarPacks associated with the application in which the tag is executed.

Accessing Attributes in Custom Adaptive Tags

To access attributes used in a custom tag, use one of the `GetTagAttribute*` methods.

All basic data types are supported as attributes (defined in the `AttributeType` class), including boolean, char, double, int, long and string. The "pt:" attributes specify the logic for the tag, while any non-pt attributes specify the behavior of the resulting HTML tag. Non-pt attributes are only applicable in tags that output a simple HTML tag.

- To access pt attributes, use the appropriate `GetTagAttributeAs*` method using the attribute name. A method is provided for each supported attribute type, e.g., `GetTagAttributeAsLong`. The `GetTagAttribute` method is provided for backwards compatibility and should not be used.
  a) First, define the attribute: `MODE = new OptionalTagAttribute("mode", "Turns debug mode on and off.", AttributeType.BOOLEAN, "true");`
  b) Then, access the attribute in the `DisplayTag` method:`boolean bNewDebugMode = GetTagAttributeAsBoolean(MODE);`
- To access non-pt (XML/HTML) attributes, use the `GetXMLTagAttribute` method using the attribute name, or `GetXMLTagAttributesAsString` to retrieve all non-pt attributes. `result.AddInnerHTMLElement(new HTMLGenericElement("<a href=\"" + GetHREF() + "\" " + GetXMLTagAttributesAsString() + ">"));`

The `ITagMetaData`, `RequiredTagAttribute`, and `OptionalTagAttribute` objects pre-process tag attributes (presence, correct type, and default values). If the required attributes are not correct, an error is logged and the tag and its children are skipped. An HTML comment describing the tag and error is displayed instead.

Storing and Accessing Custom Data in Custom Adaptive Tags

To store custom data as member variables using a custom tag, use the `SetStateVariable` or `SetStateSharedVariable` methods. To retrieve it, use `GetStateVariable` or `GetStateSharedVariable`.

Standard variables (stored with `SetStateVariable`) can only be accessed by tags in the same library. Shared variables (stored with `SetStateSharedVariable`) can be accessed by tags from any library. To prevent tags from other libraries from editing a shared variable, set `bOwnerEditOnly` to true when the shared variable is stored (tags in other libraries will still be able to read the variable).

The `Scope` parameter determines who can see the data and how long it stays in memory. The following options are defined in the `Scope` class:

Application Scope	Data is visible to all tags and all users, and is only removed when the application is restarted. Therefore, care should be used when storing data on the application to make sure it does not become cluttered with large amounts of data.
HTTP Request Scope	Data will be visible to all tags in the same HTTP Request as the current tag, and is removed from memory when the HTTP Request is finished.

Session Scope	Data is visible to all tags for the current user, and is cleared from memory when a user logs out and logs in again.
Persistent Session Scope	Data is visible to all tags in the same HTTP session, and is only removed from memory when the browser is closed or the browser session times out. Note: Data is not cleared on user logout, so do not cache anything on this scope that could be considered a security risk if it was leaked to another user. Most tags should use Session Scope for HTTP Session data storage (as described above).
Portlet Request Scope	Data is visible to all tags in the same portlet as the current tag, and is removed from memory when the portlet is finished displaying. Tags in other portlets on the same page will not be able to see the data.
Tag Scope	Data can only be seen by children of the current tag and is removed from memory when the tag is finished. (For example, in the following tags: <pt:atag><pt:btag/></pt:atag><pt:ctag/>, data stored in Tag Scope by "atag" would be visible to "btag" but not to "ctag.")

If data is stored directly in the tag in member variables (not recommended), override the `ReleaseTag` method to release the data stored on the tag.

```
/**
* @see com.plumtree.portaluiinfrastructure.tags.ATag#ReleaseTag()

*/
public void ReleaseTag()
{
// Release all member variables.
m_strPreviousRequestURL = null;
}
```

Note: Displaying an `HTMLElement` in a tag and caching it so another tag can add more HTML is not supported. `HTMLElement` trees can be generated and stored for later use as long as they are self-contained trees and used in a read-only way. It is safest to clone a cached `HTMLElement` tree before trying to display it again to make sure there are no threading problems.

Note: It is a best practice not to use static fields for data storage in tags. Each tag instance is guaranteed to be accessed by only a single thread at a time, but there may be multiple threads accessing different instances of the same tag class at the same time, either from the same user or a different user. This means that any static fields must be accessed using synchronized methods. Since there can be multiple instances of the same tag running at the same time, state variables set in shared scopes (Session, Persistent Session and Application) could change values during the execution of a single tag.

Including JavaScript in Custom Adaptive Tags

To include JavaScript in a tag, use the `AddJavaScript` method inside the `DisplayTag` method.

For example:

```
HTMLScriptCollection scriptCollection = new HTMLScriptCollection();

HTMLScript script = new HTMLScript("text/javascript");
scriptCollection.AddInnerHTMLElement(script);
script.AddInnerHTMLString("function myTest() { alert('test'); }");

AddJavascript(scriptCollection);
```

To include common JavaScript that can be shared between multiple instances of a tag (i.e. JavaScript that is displayed once per page, regardless of how many tags of a certain type there are), override the `DisplaySharedJavascript` method. `DisplaySharedJavaScript` is called automatically by the framework.

```
/**
* Adds the PTIncluder object to the client.  This object is used
 for
* retrieving JSComponent client classes from a page.
*/
public HTMLScriptCollection DisplaySharedJavascript()
{
HTMLScriptCollection result = new HTMLScriptCollection();
HTMLScript script = new HTMLScript("text/javascript");
result.AddInnerHTMLElement(script); script.SetSrc("/myjsfile.js");

return result;
}
```

If there are errors in the tag and the JavaScript cannot be displayed properly, the tag should throw an `XPException` with an error message, and the tag framework will log the error and add the message and stack trace to the HTML as an HTML comment. The message contents will be HTML encoded before being added to the comment.

Note: JavaScript is not displayed in 508 mode for either method, since section 508 compliant browsers do not support JavaScript.

Using Nested Tags in Custom Adaptive Tags

Tags can be used within other tags. To implement nested tags, use the `RequiredParentTag`, `RequiredChildTag` and `RelatedChildTag` member variables.

The outer tag is referred to as the "parent" tag. Any tags within a parent tag are referred to as "child" tags of that tag.

If the tag is only intended for use within a particular parent tag, create a public static final `RequiredParentTag` member variable. If there are multiple `RequiredParentTag` members, at least one of the parent tags must be present for the child tag to function.

If the tag must include a particular child tag to function, create a public static final `RequiredChildTag` member variable for each tag that is required inside the parent tag. If the child tag is not required for the parent tag to function, but is still related to that tag, create a public static final `RelatedChildTag` member variable instead.

```
public static final RequiredChildTag DATA_OBJECT;
static
{
... DATA_OBJECT = new RequiredChildTag(DataObjectTag.TAG);
}
```

Note: If required parent or child tags are missing when a tag is displayed, the tag framework will not process the incorrect tag and will add an error message to the HTML as an HTML comment.

Implementing Non-Standard Custom Adaptive Tag Types

To implement non-standard tag types in custom adaptive tags, including 508-accessible, looping or singleton tags, override the associated method.

- To display a custom tag in non-standard access styles (508 or low bandwidth), override the `SupportsAccessStyle` method. The default implementation of the `SupportsAccessStyle` method will cause the tag to be skipped in 508 and low-bandwidth mode. Make sure that tags that support 508 mode can function without JavaScript, since JavaScript will not be displayed in 508 mode.
- If the tag displays the tag body more than once (looping tag), override the `GetTagType()` method and return TagType.LOOPING.
- If the tag never displays the tag body (singleton tag), override `GetTagType()` and return TagType.NO_BODY.

Deploying Custom Adaptive Tags

To deploy custom adaptive tags, follow these steps.

1. Navigate to PORTAL_HOME\settings\portal and open CustomTags.xml in a text editor (you might need to make the file writeable).

2. Find the <AppLibFiles> tag and add a new entry using the name of the .jar/.dll file used to define the custom tag library (e.g., mytags).

```
<AppLibFiles>
<libfile name="sampletags"/>
</AppLibFiles>
```

3. Add the custom implementation (.jar/.dll) to the portal hierarchy:
   - Java: Copy the custom .jar file to PORTAL_HOME\lib\java and add it to the portal.war file in PORTAL_HOME\webapp. (You must stop the portal while modifying portal.war because it will be locked while the portal is running.)
   - .NET: Copy the custom .dll file to PORTAL_HOME\webapp\portal\bin.
4. Run a clean build of the portal to refresh all the jar files associated with the portal.
5. Once you have deployed your code, create a portlet that contains the tag. Custom Adaptive Tags must either include the correct XML namespace or be contained within another tag that does. The simplest way is to put the HTML inside a span. Custom adaptive tags must use the pt:libraryname.tagname and pt:attributename format. The sample code below references the custom tag from *Creating Custom Adaptive Tags* on page 252.

```
<span xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<pt:sample.hellolocation pt:message="Hello" pt:location="San
Francisco"/>
</span>
```

6. Add the portlet to a portal page and view the page. Test all custom tags.

About the ALI Scripting Framework

The ALI Scripting Framework is a client-side JavaScript library that provides services to pagelets, portlets and hosted gatewayed pages. The Portlet Communication Component (PCC) is contained within the Scripting Framework.

The ALI Scripting Framework allows pagelets to:

- **Store and share session state through browser level variables.** Browser-level variables can be stored and shared among pagelets, even pagelets that are not on the same page. For example, a value entered by the user in one pagelet can be retrieved by another. The Scripting Framework acts as an intermediary, allowing all pagelets in the portal access to all values stored in a common session. For details, see *Using Session Preferences* on page 304.

- **Leverage page-level events.** A portlet can respond when specific events happen, such as when the page loads or when the browser focus changes. For details, see *Using ALI Scripting Framework Event Notification* on page 262.
- **Refresh portlet content without reloading the portal page.** Portlets can reload their internal content without refreshing the portal page. For details, see *Using In-Place Refresh* on page 269.
- **Open any ALI object from anywhere within the portal.** The CommonOpener_OpenObject function is included in every page generated by the ALI application, and can be called from within a portlet through the PRC. For details, see *Retrieving ALI Object Managers Using IDK Remote APIs* on page 60. You can also reference ALI objects in portlets and UI components using Adaptive Tags. For details, see *Accessing ALI Objects Using Adaptive Tags* on page 216.

For a full list of classes and methods, see the *JSPortlet API documentation*.

ALI Scripting Framework Development Tips

These tips and best practices apply to all code that utilizes the ALI Scripting Framework.

Use unique names for all forms and functions. Use the GUID of a portlet to form unique names and values to avoid name collisions with other code on the page. You can append the portlet ID using the `pt:namespace` and `pt:token` tags, as shown in the code below.

```
<pt:namespace pt:token="$$TOKEN$$"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
<a onclick="doStuff$$TOKEN$$();" href="#">do stuffa
onclick="doStuff$$TOKEN$$();" href="#">do stuff</a>
<script>
function doStuff$$TOKEN$$() {
alert("hello");
}
</script>
```

Valid values for the token are in the ASCII range 0x21 to 0x7E, excluding "<" (0x3C). The scope of the token runs from the tag defining it to the end of the file; you cannot use a token prior to defining it. A second `pt:namespace` tag with a different token redefines it; two tokens cannot be defined at the same time. For details on adaptive tags, see *About Adaptive Tags* on page 202.

Gateway all URLs. You cannot make a request to a URL whose host/port differs from that of the calling page. All URLs requested through JavaScript must be

gatewayed. For details on the gateway, see *About Pagelets and the Gateway* on page 281.

Check for ALI Scripting Framework support. The Scripting Framework is a standard feature starting with version ALI 6.0 and Ensemble 1.0. It is good practice to include code that determines whether or not the component is present. Ideally, your portlet should be able to handle either situation. The simplest solution is to precede your code with an If statement that alerts the user if the ALI Scripting Framework is not supported.

```
<script>
if (PTPortlet == null)
  {
  if (document.PCC == null)
    {
    alert("This portlet only works in portals that support
 the ALI JSPortlet API or Portlet
    Communication Component (PCC). The portlet will be
displayed with severely reduced
    functionality. Contact your BEA Administrator.");
    }
  }
else
  {
  [scripting code here]
  }
</script>
```

Close all popup windows opened by a portlet when the portal window closes. The ALI Scripting Framework can be used to close popup windows using the `onunload` event.

Do not assume that browsers will process script blocks/includes added through the innerHTML property. Add all required JavaScript to the page in advance:

- IE: Add the defer attribute to the script tag.
- Netscape: Use RegExp to parse the response and look for the script, then eval it.

Using ALI Scripting Framework Event Notification

The ALI Scripting Framework allows pagelets to respond to both page-level events and custom events raised by other pagelets.

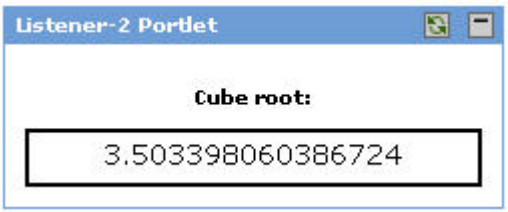
The `registerForWindowEvent` and `registerOnceForWindowEvent` methods in the ALI Scripting Framework provide pagelets with access to page-level events. For a complete list, see *Page-Level Events for Use with the ALI Scripting Framework* on page 268. To register for notification of these events, pass in the name of the event and the name of the method that should be called when it occurs. When a page-level event is raised, the JavaScript event object is passed to the event handler as an argument.

The ALI Scripting Framework also allows pagelets to raise and respond to custom events using `raiseEvent` and `registerForEvent`. The Broadcast-Listener design pattern illustrates an important example of using notification services with session preferences. Users can select an item or perform some other action in a "broadcast" portlet, which causes the content in other related "listener" portlets to be redrawn.

In the following example, the broadcast portlet displays a form that allows you to enter a number in a text box.

When the user enters a number in the text box, the values in the listener portlets change. The first listener portlet displays the square root of the number entered in the broadcast portlet.

The second listener portlet displays the cube root of the number entered in the broadcast portlet.



The following steps summarize how the portlets work:

- On load, each listener portlet calls its own instance method (registerForEvent) to register for events of type 'onBroadcastUpdate'.
- On each onkeyup event that occurs in the "Enter number" text box, the broadcast portlet sets a session preference to the value entered in the text box, and calls its own instance method (raiseEvent) to raise an event of type 'onBroadcastUpdate'.
- When the 'onBroadcastUpdate' event is raised or the page is reloaded, each listener portlet retrieves the session preference set by the broadcast portlet and computes a new value to display based on the value of the preference.

Broadcast Pagelet

```
<div style="padding:10px;" align="center">
<p><b>Enter number:</b>
 <input type="text"
style="font-size:22px;font-weight:bold;text-align:center;"
id="broadcast_prefName" value="4" size="7"
onkeyup="broadcast_setPrefs(this.value)"></p>
<br>
</div>

<script type="text/javascript">

function broadcast_setPrefs(val)
{
 var prefName = 'broadcastNumber';
 var prefValue = val;
 PTPortlet.setSessionPref(prefName,prefValue);

 var broadcastPortlet =
PTPortlet.getPortletByGUID('{D9DFF3F4-EAE7-5478-0F4C-2DBD94444000}');


   if (!broadcastPortlet)
 {
```

```
  broadcast_debug('Could not locate PTPortlet object which
corresponds to <b>Broadcast Portlet</b> on page.');
  return;
  }

   broadcast_debug('<b>Broadcast Portlet</b> raising
onBroadcastUpdate event.');
 broadcastPortlet.raiseEvent('onBroadcastUpdate',false);

}

function broadcast_debug(str)
{
 if (window.PTDebugUtil)
 {
  PTDebugUtil.debug(str);
 }
}
</script>
```

Listener Pagelet #1

```
<div style="padding:10px;" align="center">
<p><b>Square root:</b>
<div style="height:21px;border:2px solid
black;padding:2px;overflow:visible;font-size:14px;"id="listener1-swatch">
</div>
</div>

<script>

function listener1_update()
{
 var broadcastNumber =
parseFloat(PTPortlet.getSessionPref('broadcastNumber'));
 if (isNaN(broadcastNumber))
 {
  listener1_error('<b>Listener-1 Portlet</b> cannot parse number
 from session pref broadcastNumber');
  return;
 }

   listener1_debug('<b>Listener-1 Portlet</b> computing square
root of ' + broadcastNumber);
   var swatch = document.getElementById('listener1-swatch');
 swatch.innerHTML = Math.sqrt(broadcastNumber);
```

```
}

function listener1_debug(str)
{
 if (window.PTDebugUtil)
 {
  PTDebugUtil.debug(str);
 }
}

function listener1_error(str)
{
 if (window.PTDebugUtil)
 {
  PTDebugUtil.error(str);
 }
}

function listener1_getPortlet()
{
 var portletGUID = '{D9DFF3F4-EAE7-5478-0F4C-2DBDB4F4A000}';
 var listener1Portlet = PTPortlet.getPortletByGUID(portletGUID);
 return listener1Portlet;
}

var listener1Portlet = listener1_getPortlet();
if (listener1Portlet)
{

listener1Portlet.registerForEvent('onBroadcastUpdate','listener1_update');

 listener1_debug('<b>Listener-1 Portlet</b> registered
refreshOnEvent for event onBroadcastUpdate');
 listener1Portlet.registerForEvent('onload','listener1_update');
}

</script>
```

Listener Pagelet #2

```
<div style="padding:10px;" align="center">
<p><b>Cube root:</b>
<div style="height:21px;border:2px solid
black;padding:2px;overflow:visible;font-size:14px;"id="listener2-swatch">
</div>
</div>
```

```
<script>
var listener2_oneThird = (1/3);

function listener2_update()
{
 var broadcastNumber =
parseFloat(PTPortlet.getSessionPref('broadcastNumber'));
 if (isNaN(broadcastNumber))
 {
  listener2_error('<b>Listener-2 Portlet</b> cannot parse number
 from session pref broadcastNumber');
  return;
 }

 listener2_debug('<b>Listener-2 Portlet</b> computing square root
 of ' + broadcastNumber);

 var swatch = document.getElementById('listener2-swatch');
 swatch.innerHTML = Math.pow(broadcastNumber,listener2_oneThird);
}

function listener2_debug(str)
{
 if (window.PTDebugUtil)
 {
  PTDebugUtil.debug(str);
 }
}

function listener2_error(str)
{
 if (window.PTDebugUtil)
 {
  PTDebugUtil.error(str);
 }
}

function listener2_getPortlet()
{
 var portletGUID = '{D9DFF3F4-EAE7-5478-0F4C-2DBDCA1C7000}';
 var listener2Portlet = PTPortlet.getPortletByGUID(portletGUID);
 return listener2Portlet;
}
```

```
var listener2Portlet = listener2_getPortlet();
if (listener2Portlet)
{

listener2Portlet.registerForEvent('onBroadcastUpdate','listener2_update');

 listener2_debug('<b>Listener-2 Portlet</b> registered
refreshOnEvent for event onBroadcastUpdate');
 listener2Portlet.registerForEvent('onload','listener2_update');
}

</script>
```

Page-Level Events for Use with the ALI Scripting Framework

The ALI Scripting Framework automatically has access to the following page-level events.

Event	Triggered:
onload	immediately after the browser loads the page
onbeforeunload	prior to a page being unloaded (browser window closes or navigates to different location)
onunload	immediately before the page is unloaded (browser window closes or navigates to different location)
onactivate	the page is set as the active element (receives focus)
onbeforeactivate	immediately before the page is set as the active element (receives focus)
ondeactivate	when the active element is changed from the current page to another page in the parent document
onfocus	when the page receives focus
onblur	when the page loses focus
oncontrolselect	when the user is about to make a control selection of the page
onresize	when the size of the page is about to change
onresizestart	when the user begins to change the dimensions of the page in a control selection
onresizeend	when the user finishes changing the dimensions of the page in a control selection
onhelp	when the user presses the F1 key while the browser is the active window
onerror	when an error occurs during page loading

onafterprint	immediately after an associated document prints or previews for printing

Using In-Place Refresh

To refresh pagelet content in place, without affecting other content on the page, use the ALI Scripting Framework to implement in-place refresh.

Many pagelets display data that is time sensitive. In some cases, users should be able to navigate across links within a portlet without changing or refreshing the rest of the portal page. You can refresh portlet content on command, associate the refresh action with an event (`refreshOnEvent`), or program the portlet to refresh at a set interval (`setRefreshInterval`). The ALI Scripting Framework also contains methods for expanding and collapsing portlets.

In the simplified example below, the refresh portlet displays a "Refresh Portlet" button. Clicking the button updates the date and time displayed in the portlet. (The refresh button in the portlet header is an optional feature available in ALI portal only, configured on the Advanced Settings page of the Web Service Editor.)

The in-place refresh is executed by calling the `refresh()` method on the portlet object instance. The portlet reference can be retrieved by GUID, ID or name, available via the IDK IPortletRequest interface. You can also set a new URL to be displayed within the portlet upon refresh by using `setRefreshURL` or passing in a new URL when you call `refresh`. (The title bar cannot be altered on refresh.)

```
<div style="padding:10px;" align="center">
<p><button onclick="refresh_portlet()">Refresh Portlet</button></p>
<p><b>Current time is:</b><br> <span
id="refreshTimeSpan"></span></p>
</div>
<pt:namespace pt:token="$PORTLET_ID$"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
```

```
<script type="text/javascript">
function refresh_portlet()
{
var refreshPortlet = PTPortlet.getPortletByID($PORTLET_ID$);
if (!refreshPortlet)
  {
  refresh_debug('Could not locate PTPortlet object which
corresponds to <b>Refresh Portlet</b> on page.');
  return;
  }
refresh_debug('<b>Refresh Portlet</b> calling refresh() method.');
refreshPortlet.refresh();
}

function refresh_debug(str)
{
if (window.PTDebugUtil)
  {
  PTDebugUtil.debug(str);
  }
}

var t = new Date();
document.getElementById('refreshTimeSpan').innerHTML = t;
</script>
```

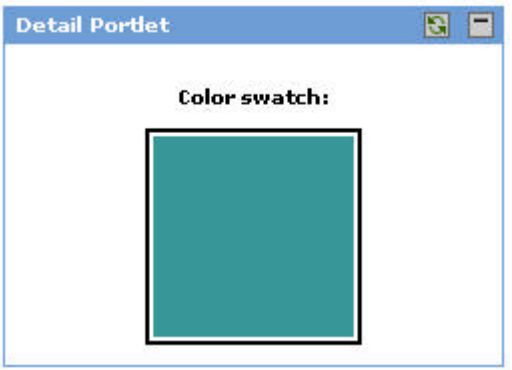
Using Session Preferences

To store and share settings within the client browser, use session preferences.

Pagelets can use preferences to communicate with each other, but accessing preferences usually requires a round trip to the portal database. Session preferences provide a way to store and share settings in the user's session within the client browser.

The Master-Detail design pattern illustrates the most basic usage of session preferences. This design pattern splits control and display between two portlets. For example, the "master" portlet could summarize data in list form, and the "detail" portlet could display details on each data item in response to user selection. In the example below, the master portlet displays a form that allows you to enter a color code in a text box.

When the user enters a color code in the text box, the color in the detail portlet changes.



For each onkeyup event that occurs in the "Enter color" text box in the master portlet, the following steps are executed:

1. The master portlet sets the session preference using the current value of the text box.
2. The master portlet calls an update method on the detail portlet.
3. The detail portlet retrieves the session preference to get the color value.
4. The detail portlet redraws its color swatch area to reflect the new color value.

Pagelets can manipulate session preferences using the ALI Scripting Framework or the IDK. Sample code for both options is provided below.

Note: Shared session preferences must be specified by name on the Preferences page of the associated Web Service Editor or they will not be sent to the pagelet.

ALI Scripting Framework Methods

This example is oversimplified; the master portlet makes a direct call to a JavaScript method of the detail portlet. Unless the master portlet takes extra measures to ensure that the detail portlet is actually present on the same page, calls from master to detail could generate errors. The ALI Scripting Framework provides an easy way to detach the relationship between portlets and use a

common event interface for communication. See *Using ALI Scripting Framework Event Notification* on page 262 for more information on inter-portlet communication.

Master Portlet

```
<div style="padding:10px;" align="center">
<p><b>Enter color:</b>  
<input type="text"
style="font-size:22px;font-weight:bold;text-align:center;"
id="master_prefName"
value="#FFFFFF" size="8"
onkeyup="master_setPrefs(this.value)"></p><br>
</div>

<script type="text/javascript">
function master_setPrefs(val)
{
var prefName = 'masterColor';
var prefValue = val;
PTPortlet.setSessionPref(prefName,prefValue);

master_debug('<b>Master Portlet</b> called
PTPortlet.setSessionPref(\'masterColor\',\'' + prefValue + '\').');

if (window.detail_update)
  {
  master_debug('<b>Master Portlet</b> calling detail_update().');

  detail_update();
  }
else
  {
  master_debug('Could not locate portlet <b>Detail Portlet</b> on
 page.');
  }
}
function master_debug(str)
{
if (window.PTDebugUtil)
  {
  PTDebugUtil.debug(str);
  }
}
</script>
```

Detail Portlet

```
<div style="padding:10px;" align="center">
<p><b>Color swatch</b>  
<div style="width:100px;height:100px;border:2px solid
black;padding:2px;"id="detail-swatch"></div>
<script>
function detail_update()
{
var color = PTPortlet.getSessionPref('masterColor');
detail_debug('<b>Detail Portlet</b> received value="' + color +
'" for PTPortlet.getSessionPref(\'masterColor\')');

var swatch = document.getElementById('detail-swatch');
if (swatch)
  {
  swatch.innerHTML = '<div style="background-color:' + color +
';width:100%;height:100%;"></div>';
  }
else
  {
  detail_debug('<b>Detail Portlet</b> cannot find \'detail-swatch\'
 DIV element.');
  }
}

function detail_debug(str)
{
if (window.PTDebugUtil)
  {
  PTDebugUtil.debug(str);
  }
}
</script>
```

IDK Methods

In most cases, reading session preferences via the ALI Scripting Framework is inefficient and insecure. **Always use the IDK to read session preferences**, as shown in the example code below.

Java

```
<%@ page language="java"
import="com.plumtree.remote.portlet.*,java.util.Date" %>

IPortletContext portletContext =
PortletContextFactory.createPortletContext(request,response);
```

```
IPortletResponse portletResponse = portletContext.getResponse();

IPortletUser portletUser = portletContext.getUser();
IPortletRequest portletRequest = portletContext.getRequest();

masterColor = portletRequest.getSettingValue(SettingType.Session,
 "masterColor");
```

.NET

```
...
Dim portletContext As IPortletContext
portletContext =
PortletContextFactory.CreatePortletContext(Request, Response)

Dim portletRequest As IPortletRequest
portletRequest = PortletContext.GetRequest

Dim portletUser As IPortletUser
portletUser = PortletContext.GetUser

Dim portletResponse As IPortletResponse
portletResponse = PortletContext.GetResponse

Dim masterColor As String
masterColor = portletRequest.GetSettingValue(SettingType.Session
 "masterColor")
...
```

About Pagelet Caching

Caching is the functionality that allows ALI and Ensemble to request pagelet content, save the content, and return the saved content to users when appropriate. The importance of caching cannot be overstated.

Efficient caching makes every web application faster and less expensive. The only time content should not be cached is if the data must be continuously updated. If every pagelet had to be freshly generated for each request, performance could become unacceptably slow. ALI and Ensemble rely on caching to improve performance. Pagelet content is cached and returned when later requests match the cache's existing settings.

Caching is indexed on the settings sent by the pagelet. When the ALI or Ensemble gateway server processes a request for a page, it looks individually at each pagelet on the page and checks it against the cache. The process can be summarized as follows:

1. The gateway server assembles a cache key used to uniquely identify each pagelet in the cache.
2. The gateway server checks the cache for a matching cache key entry:
   - If the gateway server finds a match that is not expired, it returns the content in the cache and does not make a request to the remote server.
   - If there is no matching cache key for the pagelet or if the cache key has expired, the gateway server makes a request to the remote server. If the matching cache entry uses ETag or Last-Modified caching, it also sends the appropriate caching header to the remote server in the request.
3. The response comes back from the remote server; the gateway server checks for caching headers:
   - If the headers include an Expires header, the gateway server stores the new pagelet content (along with a new expiration date) in its cache.
   - If the headers use ETag or Last-Modified caching, the existing cache entry might be revalidated (in the case of '304-Not Modified') or new pagelet content might be stored in the cache.

ALI and Ensemble cache gatewayed content to complement, not replace, browser caching. Public content is accessible to multiple users without any user-specific information (based on HTTP headers). The gateway server calculates the cache headers sent to the browser to ensure that the content is properly cached on the client side.

ALI and Ensemble cache all text (i.e., nonbinary) content returned by GET requests. Even if gateway caching is disabled (via PTSpy), pagelet caching still takes place. Gatewayed content can be cached by a proxy server or by the user's browser. Beware browser caching of gatewayed content; it is a good idea to clear your browser cache often during development. An incorrectly set Expires header can cause browsers to cache gatewayed content.

The pagelet cache contains sections of finished markup and sections of markup that require further transformation. Post-cache processing means content can be more timely and personalized. Adaptive tags enable certain pagelets (for example, Community banners) to be cached publicly for extended periods of time and yet contain user specific and page-specific information, as well as the current date and time.

For details, see the following topics:

- *About Pagelet Caching Strategies* on page 319
- *Pagelet/Portlet Cache Key* on page 320 *Setting HTTP Caching Headers - Cache-Control* on page 322
- *Setting HTTP Caching Headers - Expires* on page 324

- *Setting HTTP Caching Headers - Last-Modified and ETag* on page 325 *Implementing Portlet Caching* on page 322
- *Configuring ALI Portlet Caching Settings* on page 325

For a full explanation of HTTP caching, see RFC 2616 (*http://www.w3.org/Protocols/rfc2616/rfc2616.html*).

About Pagelet Caching Strategies

Pagelet caching is controlled both by the programmer and by the administrator who registers the pagelet in ALI or Ensemble. Each and every pagelet needs a tailored caching strategy to fit its specific functionality.

A pagelet's caching strategy should take all possibilities into account and use the most efficient combination for its specific functionality. A pagelet that takes too long to generate can degrade the performance of every page that displays it. These questions can help you determine the appropriate caching strategy:

- Will the content accessed by the pagelet change? How often?
- How time-critical is the content?
- What processes are involved in producing pagelet content? How expensive are they in terms of server time and impact?
- Is the pagelet the only client with access to the back-end application?
- Is the content different for specific users?
- Can users share cached content?

Determine how often pagelet content must be updated, dependent on data update frequency and business needs. Find the longest time interval between data refreshes that will not negatively affect the validity of the content or the business goals of the pagelet.

Since caching is indexed on the settings used by a pagelet, new content is always requested when settings change (assuming that no cached content exists for that combination of settings).

There are two common situations in which you might mistakenly decide that a pagelet cannot be cached:

- **In-place refresh**: You might think that caching would "break" a pagelet that uses in-place refresh because the pagelet would be redirected to the original (cached) content. This can be avoided if a unique setting is updated on every action that causes a redraw, effectively "flushing" the cache. (In-place refresh renews the portlet display by causing the browser to refresh the portal page at a set interval.)
- **Invisible preferences**: If the content of the pagelet is dependent on something other than preferences (for example, the pagelet keys off the User ID to display a name or uses ALI

security to filter a list), caching can still be implemented with “invisible preferences” (in this case, User ID). As with in-place refresh, invisible preferences are set solely for the purpose of creating a different cache entry. They are set programmatically, without the user’s knowledge.

For details on implementing caching, see the following topics:

- *Setting HTTP Caching Headers - Cache-Control* on page 322
- *Setting HTTP Caching Headers - Expires* on page 324
- *Setting HTTP Caching Headers - Last-Modified and ETag* on page 325
- *Implementing Portlet Caching* on page 322

Pagelet/Portlet Cache Key

The cache key for a pagelet/portlet entry in ALI consists of these values.

Portlet ID	The unique ID for the portlet, defined by the portal.
Content Mode	The content mode of the portlet.
Portal Settings	All seven types of settings stored in the ALI database: Portlet settings, User settings, Community settings, CommunityPortlet settings, Administrative settings, Session preferences, and User Information.
User Interface	The type of device used to access the portlet.
CanSet values	All three values for the CanSet header: CanSetPersonal, CanSetCommunity, CanSetAdmin
LocaleID	The ID for the portal-defined locale associated with the current user.
UserID	The unique ID for the current user. Included only if private caching is used.
URI	The URL to the portlet page on the remote server.
Community ID	Included only if the portlet is displayed on a community page.
Last-modified date	The last modified date of the portlet.

The data below can be added to the cache key by setting options in the Web Service editor on the Advanced Settings page.

Community ACL	The ACL for the community in which the portlet is displayed.
Page ID	The ID for the portal page on which the portlet is displayed.
TimeZone	The time zone for the portal in which the portlet is displayed.
Experience Definition ID	The ID for the Experience Definition in which the portlet is displayed.
Portlet Alignment	The alignment of the portlet in the current page.
Activity Rights	Only the Activity Rights configured in the Web Service editor are included in the cache key.

The data below is deliberately not included in the cache key:

StyleSheetURI	Portal stylesheets are applied at runtime, depending on the user preference. Portlet content does not depend on the particular stylesheet that the user has selected.
HostpageURI	All parts of the Hostpage URI value are covered separately. The cache key includes Community ID, so it already distinguishes between My Pages and Community pages. The User ID is added if private caching is used.

Setting HTTP Caching Headers - Cache-Control

The Cache-Control header can be used to expire content immediately or disable caching altogether. The value of this header determines whether cached portlet content can be shared among different users.

The Cache-Control header can contain the following values:

public	Allows any cached content to be shared across users with identical sets of preferences using the same Portal Server. This value should be used whenever possible.
private	Tells the Portal Server not to share cached content. The User ID is added to the cache key so that a separate copy is retained in the cache for each individual user. This value should only

	be used to protect sensitive information, for example, an e-mail inbox portlet. (User settings can also make public content effectively private.)
max-age=[seconds]	Specifies the maximum amount of time that an object is considered fresh. Similar to the Expires header, this directive allows more flexibility. [seconds] is the number of seconds from the time of the request that the object should remain fresh.
must-revalidate	Tells the cache that it must obey any freshness information it receives about an object. HTTP allows caches to take liberties with the freshness of objects; specifying this header tells the cache to strictly follow your rules.
no-cache	Disables caching completely and overrides Web Service editor settings. Neither the client nor the Portal Server responds to subsequent requests with a cached version.

In JSP, use the setHeader method to configure the Cache-Control header:

```
<%
response.setHeader("Cache-Control","public");
%>
```

The JSP example below expires the content immediately using the maximum age header.

```
<%
response.setHeader("Cache-Control","max-age=0");
%>
```

In the .NET Framework, the Cache-Control header is accessed through the System.Web.HttpCachePolicy class. To set the header to public, private or no-cache, use the Response.Cache.SetCacheability method.

```
Response.Cache.SetCacheability(HttpCacheability.Public);
```

To set a maximum age for content in .NET, use the Response.Cache.SetMaxAge method. The example below expires the content immediately.

```
TimeSpan ts = new TimeSpan(0,0,0);
Response.Cache.SetMaxAge(ts);
```

To set the header to must-revalidate in .NET, use the Response.Cache.SetRevalidation method.

```
Response.Cache.SetRevalidation(HttpCacheRevalidation.AllCaches);
```

Setting HTTP Caching Headers - Expires

The Expires header specifies when content will expire, or how long content is "fresh." After this time, the Portal Server will always check back with the remote server to see if the content has changed.

Most Web servers allow setting an absolute time to expire, a time based on the last time that the client saw the object (last access time), or a time based on the last time the document changed on your server (last modification time).

In JSP, setting caching to forever using the Expires header is as simple as using the code that follows:

```
<%
response.setDateHeader("Expires",Long.MAX_VALUE);
%>
```

The .NET Framework's System.Web.HttpCachePolicy class provides a range of methods to handle caching, but it can also be used to set HTTP headers explicitly (see MSDN for API documentation: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemwebhttpcachepolicyclasssetexpirestopic.asp*). The Response.Cache.SetExpires method allows you to set the Expires header in a number of ways. The following code snippet sets it to forever:

```
Response.Cache.SetExpires(DateTime.Now.AddYears(100000000));
```

In .NET, the Web Form page (.aspx) can also use standard ASP methods to set HTTP headers.

Note: Never use Expires = 0 to prevent caching. The Expires header is sent by the remote server and passed through to the browser by the Portal Server. Unless the time on all three machines is synchronized, an Expires=0 header can mistakenly return cached content. To solve this problem, set the Expires header to a fixed date that is definitely in the past.

Setting HTTP Caching Headers - Last-Modified and ETag

The Last-Modified response header specifies the last time a change was made in the returned content, in the form of a time stamp. ETag values are unique identifiers generated by the server

and changed every time the object is modified. Either can be used to determine if cached content is up to date.

When an object stored in the cache includes a Last-Modified or ETag header, the Portal Server can use this value to ask the remote server if the object has changed since the last time it was seen.

- The Portal Server sends the value from the Last-Modified header to the remote server in the If-Modified-Since Request header.
- The remote server sends the ETag header to the Portal Server with portlet content. When another request is made for the same content, the Portal Server sends the value in the ETag header back to the remote server in the If-None-Match header.

The portlet code on the remote server uses the header value to determine if the content being requested has changed since the last request, and responds with either fresh content or a 304 Not Modified Response. If the Portal Server receives the latter, it displays the cached content.

JSP portlets can access the value in the If-Modified-Since request header using the getLastModified(HttpServletRequest req) method provided by the Java class HttpServlet.

In .NET, the Response.Cache.SetLastModified method allows you to set the Last-Modified header to the date of your choice. Alternately, the SetLastModifiedFromFileDependencies method sets the header based on the time stamps of the handler's file dependencies.

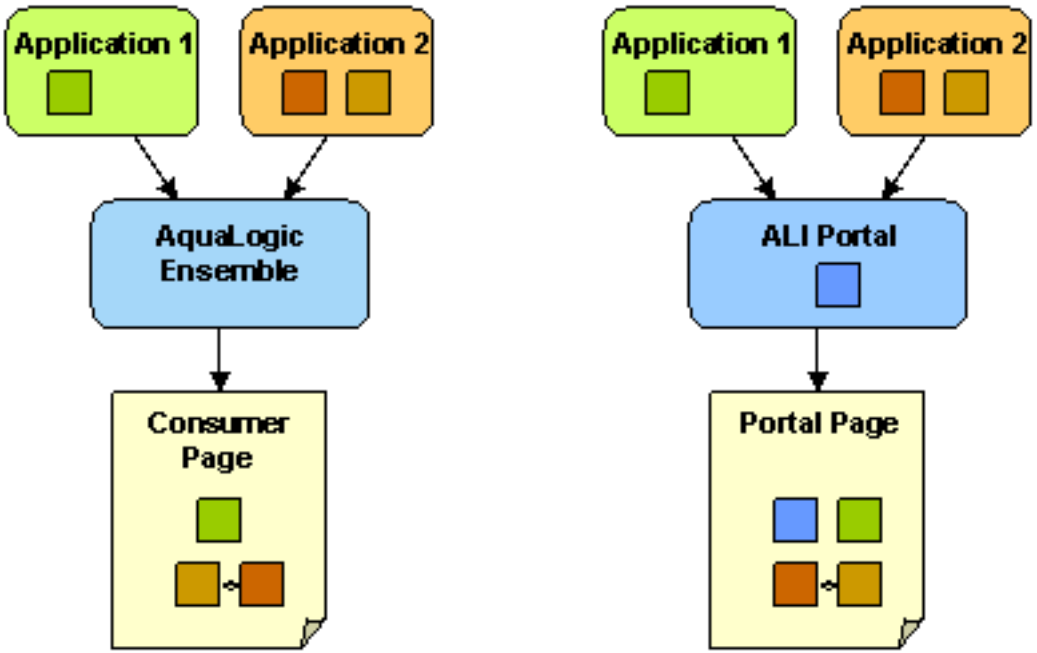
```
Response.Cache.SetLastModified(DateTime.Now);
```

To use ETag in .NET, use the Response.Cache.SetETag method to pass in the string to be used as the ETag. The SetETagFromFileDependencies method creates an ETag by combining the file names and last modified timestamps for all files on which the handler is dependent.

About Pagelets and the Gateway

All pagelets are designed to be displayed with other pagelets. ALI and Ensemble both act as a gateway, processing and combining pagelets from multiple applications to create a single, unified page with a range of functionality.

The code returned by a pagelet is parsed by the gateway server and inserted into the appropriate cell in the HTML table that makes up the mashup page. Pagelets from the same back-end application can interact with each other within the page.



The same pagelets can be used in both Ensemble and ALI portal, but a different process is used to implement the mashup page.

- Modeled after the “window” metaphor from desktop user interfaces, the **ALI portal** displays pagelets as a series of windows or boxes arranged in columns on a page with borders, title bars, buttons, headers and footers rendered by the portal framework. The page layout is defined through the portal's administrative UI. Each My Page or community page is made up of many pagelets, selected and arranged using portal editors. Portlets are pagelets specifically designed for use in the AquaLogic Interaction portal.
- In **Ensemble**, a consumer page defines the layout and includes specific pagelets in the page using adaptive tags. The portal header is not included by default, but portal header navigation can be added using tags.

For an introduction to the gateway, see *About Server Communication and the Gateway* on page 40.

About Pagelet Internationalization

These tips and best practices apply to all pagelets that will be translated into multiple languages.

Identify ALL culturally dependent data. Text messages are the most obvious example of locale-specific data, but there are many other parts of a service that can vary with language or location. These include: images, UI labels and buttons, icons, sounds, graphics, dates, times, measurements, honorifics and titles, phone numbers, and postal addresses. In ALI portal, the title bar for the portlet must also be localized; for details, see *Modifying the Portlet Title Bar* on page 317.

Do not use compound messages (concatenated strings) to create text. Compound messages contain variable data. For example, in the text string "You have XX credits," only the integer "XX" will vary. However, the position of the integer in the sentence is not the same in all languages. If the message is coded as a concatenated string, it cannot be translated without rewriting the code.

Use the IDK to avoid encoding issues. All content is stored in the database in Unicode. The IDK handles encoding for international characters.

For details on implementing internationalization, see *Using Internationalized Strings in Adaptive Tags* on page 205.

About ALI Portlet Development

The following topics provide general information about ALI portlet development and configuration.

- *About the IDK ALI Portlet API* on page 284: The plumtree.remote.portlet package provides portal-specific support for portlet development, including manipulating settings in the ALI database, accessing user information, and managing communication with the portal. For details on creating portlets with the IDK Proxy API, see *Creating a Custom ALI Portlet with the Java IDK Portlet API* on page 285 and *Creating a Custom ALI Portlet with the .NET IDK Portlet API* on page 287.
- *About Adaptive Pagelets* on page 196: Adaptive pagelets allow you to create a coordinated page with dynamic, interactive functionality comprised of cross-platform services that talk to multiple back-ends.

  - *About Adaptive Tags* on page 202: Adaptive Tags are used to display contextual data and control Ensemble and ALI portal functionality from remote pagelets. Unlike the IDK, Adaptive Tags use XML in pagelet content instead of code, which avoids a network round trip. Tags can be included in the markup returned by any gatewayed page (HTML, JSP or ASP.Net). For details on tags available for use in ALI, see *About ALI Adaptive Tag Libraries* on page 202.
  - *About the ALI Scripting Framework* on page 260: The ALI Scripting Framework is a set of client-side JavaScript libraries that provide services to pagelets and gatewayed pages.

- *About Programmable Remote Client (PRC) Remote APIs* on page 54: The plumtree.remote.prc package includes a collection of APIs that provide access to functionality within the ALI portal, Collaboration, Publisher, and Search Service. These APIs can be used by any pagelet deployed in an environment with access to these applications.
- *About ALI Portlet Alignment* on page 291
- *About CSS Customization for ALI Portlets* on page 294
- *About ALI Portlet Settings* on page 294
- *About Pagelet Internationalization* on page 317
- *About ALI Portlet Security* on page 310
- *About Pagelet Caching* on page 318
- *About ALI Logging Utilities* on page 16
- All cross-platform pagelet development concepts also apply to ALI portlets. For details, see *About Cross-Platform Pagelet Development* on page 48.

For details on configuring portlets in the ALI portal, see the ALI portal online help.

About the IDK ALI Portlet API

The IDK Portlet API provides ALI-specific support for portlet development, including manipulating settings in the ALI database, accessing user information, and managing communication with the portal.

This page provides an introduction to the IDK Portlet API. For more details on objects and methods, see the IDK API documentation.

Note: Note: The IDK was formerly called the Plumtree Development Kit (EDK); some package names retain old naming conventions.

The plumtree.remote.portlet package/namespace includes the following interfaces:

- `IPortletContext`
- `IPortletRequest`

- `IPortletResponse`
- `IPortletUser`

In general, these interfaces are called in the following order:

1. A portlet uses `PortletContextFactory.createPortletContext` to initiate a connection for communicating with Ensemble or the portal.
2. The `IPortletContext` object returned allows the portlet to access information about the request and response, the current user, and the session. The portlet uses this information as needed, in arbitrary order, to generate a proper response. Using `IPortletContext`, the portlet can access `IPortletRequest`, `IPortletUser`, `IRemoteSession` and `IPortletResponse`.
3. The portlet retrieves parameters from the request using `IPortletRequest`.
4. The portlet retrieves user information and preferences from `IPortletUser`.
5. The portlet can access functionality in AquaLogic applications using `IRemoteSession`. For details, see *About Programmable Remote Client (PRC) Remote APIs* on page 54.
6. The portlet constructs a response using `IPortletResponse`. The response includes content to be displayed and any settings to be stored or removed.

For examples of using IPortlet interfaces in a portlet, see *Creating a Custom ALI Portlet with the Java IDK Portlet API* on page 285 and *Creating a Custom ALI Portlet with the .NET IDK Portlet API* on page 287.

Creating a Custom ALI Portlet with the Java IDK Portlet API

This simplified Hello World portlet example allows a user to set the message that is displayed within a portlet.

Before writing any code, create a new IDK project as described in *Setting Up a Custom Java IDK Project in Eclipse Stand-Alone (without WTP)* on page 12 or *Setting Up a Custom Java IDK Project in Eclipse with WTP* on page 10.

This example uses two pages: a portlet that displays the current setting value and a form for changing the value, and a page that sets the value in the ALI database and redirects to the portal page.

In the new IDK project, create a new JSP page for the portlet (portlet.jsp). The portlet code shown below instantiates the IDK and uses the IDK Portlet API `IPortletRequest` object to check for a Portlet setting called "PortletEntry." If the setting has an associated value, the portlet displays it. The portlet also displays a form that allows the user to enter a value. When the user clicks Submit, the portlet code sends the value from the form in a request to the setPrefs.jsp page, shown next.

Note: There is no need to include html, head and body tags; the portlet is displayed as part of the HTML table that makes up the portal page.

```
<%@ page language="java"
import="com.plumtree.remote.portlet.*,java.util.Date" %>
You refreshed at <%= new Date().toString()%><br/>

<%
//get the idk
IPortletContext portletContext =
PortletContextFactory.createPortletContext(request, response);
IPortletRequest portletRequest = portletContext.getRequest();
String settingKey = "PortletEntry";

String settingValue =
portletRequest.getSettingValue(SettingType.Portlet, settingKey);

//if the entry has already been set, display it here
if (null != settingValue)
{
%>
  <br/><b> Preference value is <%=settingValue%></b>!<br/>
<%
}

//form to enter the preference
%>
<P>Enter your preference:
<form METHOD="post" ACTION="setPrefs.jsp" name="form1">
<input type="text" name="<%=settingKey%>">
<br/>
<input type="submit" name="Submit" value="Submit">
</form>
```

Next, create the Set Preferences page (setPrefs.jsp). The code shown below gets the value for the PortletEntry Portlet setting from the request, then uses the IDK Portlet API `IPortletResponse` object to add the setting to the database and redirect to the portal. The redirect causes the portal page to refresh and display the updated setting in the portlet.

```
<%@ page language="java" import="com.plumtree.remote.portlet.*"
%>

<%
//set the cache control so we don't get a cached page
response.setHeader("Cache-control", "max-age=0");
```

```
//get the idk
IPortletContext portletContext =
PortletContextFactory.createPortletContext(request, response);

//get IPortletResponse to set preferences and redirect back to
the portal
IPortletResponse portletResponse = portletContext.getResponse();

//get the setting value from the servlet request
String settingKey = "PortletEntry";
String settingValue = request.getParameter(settingKey);

//set the setting value
portletResponse.setSettingValue(SettingType.Portlet, settingKey,
 settingValue);

//redirect back to the portal
portletResponse.returnToPortal();

%>
```

After you have completed the JSP pages, deploy your custom project as described in *Deploying a Custom Java IDK Project in Eclipse Stand-Alone (without WTP)* on page 12 or *Deploying a Custom Java IDK Project in Eclipse with WTP* on page 10.

Creating a Custom ALI Portlet with the .NET IDK Portlet API

This simplified Hello World portlet example allows a user to set the message that is displayed within a portlet.

Before writing any code, create a new IDK project as described in *Setting Up a Custom .NET IDK Project* on page 14.

This example creates a portlet that displays the current setting value and a form for changing the value. .NET portlets use a code-behind page to manipulate settings and redirect to the portal. The web form that makes up the portlet (portlet.asp) simply initiates the code-behind page (portlet.aspx.cs) and displays a form that prompts the user to enter a message. The sample code below is for VisualStudio 2005.

Note: There is no need to include html, head and body tags; the portlet is displayed as part of the HTML table that makes up the portal page.

```
<%@ Page language="c#" CodeFile="portlet.aspx.cs"
AutoEventWireup="false" Inherits="HelloWorld.WebForm1"
ResponseEncoding="UTF-8"%>
```

```
<form id="Form1" method="post" runat="server">
<br>
<asp:label runat="server" id="settingsDisplay"></asp:label>
<br><br>
Enter your message:<br>   
<asp:textbox id="PrefName" runat="server"></asp:textbox>
<br><br>
<asp:Button id="AddButton" runat="server" Text="Submit"
></asp:Button>
<br>
</form>
```

The code-behind page instantiates the IDK and uses the IDK Portlet API `IPortletRequest` object to check for a Portlet setting called "MyPref". If the setting has an associated value, the page sends it back for display. If the user entered a new value in the form, the portlet sends it back for display and uses the `IPortletResponse` object to store it in the portal database. When the setting value and portlet display are updated, the portal page is refreshed.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Plumtree.Remote.Portlet;

namespace HelloWorld

{

      /// <summary>
/// Summary description for WebForm1.
/// </summary>

      public class WebForm1 : System.Web.UI.Page

      {

            //put member variables in protected scope as that is
 the most
```

```
//limited scope that the aspx page can see.

protected System.Web.UI.WebControls.Label settingsDisplay;
protected System.Web.UI.WebControls.TextBox PrefName;
protected System.Web.UI.WebControls.Button AddButton;
protected Plumtree.Remote.Portlet.IPortletRequest portletRequest;
protected Plumtree.Remote.Portlet.IPortletResponse portletResponse;
protected string settingKey = "MyPref";
protected string settingValue;


           private void Page_Load(object sender, System.EventArgs
 e)

             {
       }

             #region Web Form Designer generated code

             override protected void OnInit(EventArgs e)

             {

                   //
// CODEGEN: This call is required by the ASP.NET Web Form
// Designer.
//

                   InitializeComponent();
              base.OnInit(e);

             }

             /// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>

             private void InitializeComponent()

             {

                   this.AddButton.Click += new
System.EventHandler(this.AddButton_Click);
this.Load += new System.EventHandler(this.Page_Load);
```

```
            }

            #endregion

            private void AddButton_Click(object sender,
System.EventArgs e)

            {

                  //get the setting value

                  IPortletContext portletContext =
PortletContextFactory.CreatePortletContext(Request,Response);

                  portletRequest = portletContext.GetRequest();
             portletResponse = portletContext.GetResponse();

                  settingValue =
portletRequest.GetSettingValue(SettingType.Portlet,settingKey);

                  //set the label with the retrieved value
//(if it has already been set)

                  if(null != settingValue)

                  {
settingsDisplay.Text = "Old preference value is " +

Server.HtmlEncode(settingValue) + "!";

                  }

                  if (PrefName.Text != "")

                  {

                        settingsDisplay.Text += "\New preference
 value is " +
Server.HtmlEncode(PrefName.Text)+ "!";


portletResponse.SetSettingValue(SettingType.Portlet,
settingKey,PrefName.Text);
```

```
                                }
}
}
}
```

After you have completed the portlet code, deploy your custom project as described in *Deploying a Custom .NET IDK Project in IIS* on page 15.

About ALI Portlet Alignment

Where a portlet is displayed on the ALI portal page defines its size.

The ALI portal page is made up of columns. In a two-column configuration, narrow portlets are displayed on the left, wide portlets on the right. In a three-column configuration, wide portlets are displayed in the middle. In the Low Bandwidth version of the portal or the Portal for People with Disabilities, width is irrelevant; portlets are displayed in a single column.

When you configure a portlet object in the portal, you may choose from the following alignments:

- **Narrow** portlets are displayed in a narrow side column on the portal page. Narrow portlets must fit in a column that is fewer than 255 pixels wide.
- **Wide** portlets are displayed in the middle or widest side column on the portal page. Wide portlets fit in a column fewer than 500 pixels wide.
- **Header** portlets override the portal header at the top of the page, allowing you to add custom branding to a Community page.

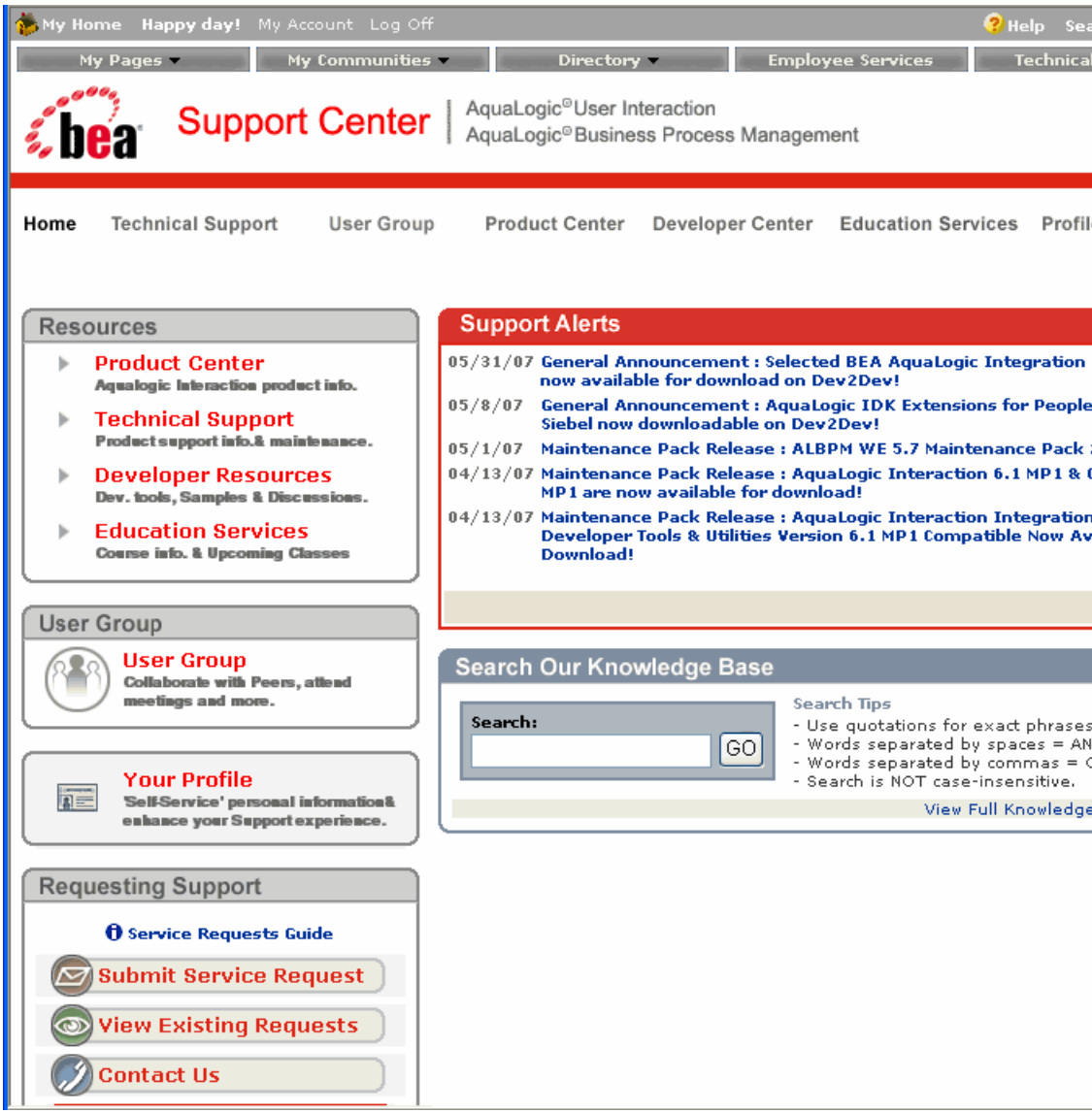
  Note: In header portlets, make sure to include the `pt:pageName` and `pt:realmName` markup tags. The "realm" name is either the name of the current Community or one of the following: "My Pages," "Documents," "Administration," or "Gateway." The page name is the name of the current page in a Community or My Page or blank otherwise. The localized name will be used if available. For details, see *About ALI Adaptive Tag Libraries* on page 202 .

- **Footer** portlets override the portal footer at the bottom of the page, allowing you to add custom branding to a Community page.
- **Content Canvas** portlets span across all rows and columns of a Community page, taking up all space between the header and footer.

Each My Page or community page is made up of many portlets, selected and arranged based on alignment. For example, the portlets on the Support Center community page shown below includes a selection of portlets:

- The Support Center header and custom navigation are implemented using portlets.

- The left column includes a portlet that provides navigation links to the subsections (Resources) within the Support Center site, portlets with links to profile configuration pages, and another that provides access to support contact information.
- The center column includes a portlet that lists recent Support Alerts and a Knowledge Base search portlet.
- The right column includes portlets with links to product downloads and information.



About CSS Customization for ALI Portlets

The CSS template provided with ALI version 6.0 and above allows you to customize portlet content and design in a variety of ways.

CSS customization allows you to customize specific portlets using the unique portlet ID, or modify the design of a group of portlets (for example, those in the first column of a two-column page). You can also set constraints for portlets, including limiting a specific portlet to a three-column layout or preventing users from collapsing portlets.

The portal CSS template file follows standard CSS syntax rules. For details on CSS, see *http://www.w3.org/Style/CSS/*.

For details on using CSS in portlets, see *About ALI 6.5 Adaptive Styles (CSS Customization)* on page 445.

About ALI Portlet Settings

Most portlets use settings. In some cases, a portlet can access settings stored by another portlet or service.

For details on ALI portlet settings, see the following topics:

- *ALI Portlet Setting Types* on page 296
- *Portlet Settings Development Tips* on page 294
- *About Portlet Preferences Pages* on page 300
- *About Community Preferences Pages* on page 299
- *About Administrative Preferences and Portlet Template Preferences Pages* on page 298

Portlet Settings Development Tips

These tips and best practices apply to all portlets that access settings.

Enter all preference and configuration pages used by a portlet in the Web Service editor. You must enter the URL to any preference pages in the Web Service editor on the Preferences page. You must enter the URL to any configuration pages in the Web Service editor on the Advanced URLs page.

Enter all User settings and Community Settings required by a portlet in the Preference list in the Web Service editor. If a shared setting is not entered in this list, it will not be available to the portlet.

Gateway all pages that store settings in the portal. To store settings on the portal, preference pages must be included in the Gateway Prefixes List, configured in the Web Service editor on the HTTP Configuration page. For instructions on entering gateway prefixes, see the portal online help.

Never send query string or form variables directly to the My Page. Always use a separate page to set and remove settings. Sending query string or form variables directly to the portal page is unreliable and can result in seemingly random errors. Your code will not have access to these variables in most cases, because it might be preceded by other code on the portal page.

Do not use session preferences for shared settings; portlets on the same portal page and the same remote server cannot use the same session. To implement shared settings, you must use the Application object, store User settings in the ALI database, or use the Portlet Communication Component (PCC).

Return the user to the location of the portlet on the page when redirecting from a preferences page to a portal page. This can be done using the IDK `IPortletResponse.ReturnToPortal` method.

Always include a link to the correct settings page within the portlet display. It might not be clear to users where they should enter settings for a portlet. If the portlet requires configuration, include a link to the appropriate preference page or configuration page within the portlet display.

Always use popup windows for preference pages. Use the following guidelines for all popup windows.

- **The portal window must remain open.** Do not redirect the portal window to the destination URL.
- **The popup window should regain focus on any successive click-through.** If the user leaves the window open and then clicks the same link, the same popup window should regain focus (instead of opening a new window).
- **The popup window should close if the portal window is closed.** If the user closes the portal window, any associated popup windows should close automatically.
- **The popup window should appear in the style of the portal.** You can reference the stylesheet in any gatewayed page by using the pt:Stylesheets tag as shown in the code snippet below. For details on adaptive tags, see *About Adaptive Tags* on page 202.

```
<%@ page language="java"
import="com.plumtree.remote.portlet.*" %>
<pt:styleSheets
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>
```

```
<div class=platportletHeaderBg>The background here is
<i>platportletHeaderBg</i>. <span
class=platportletWideHeader>This adds the font style
<i>platportletWideHeader</i> from the portal
stylesheet.</span></div>
<p><input type=button class=inputBox value="This button
uses the inputBox style">
...
```

There are some additional considerations if the data stored by a portlet should be secure; for details, see *About ALI Portlet Security* on page 310.

ALI Portlet Setting Types

Portlets can use seven types of settings (aka preferences) to provide personalized functionality. Each type of setting is intended for a specific range of use, and each is handled differently by the IDK and the portal.

Setting Type	User	Portlet	Preference Page Type	Notes
Administrative Setting	All	1 specific	Administrative Preferences page or Portlet Template Preferences page	Administrative settings can be modified only by users with administrative access to the portal and read/write access to the object.
User Setting	1 specific	All	Portlet Preferences page or User Configuration page	User settings must be uniquely named. A portlet has access to the User settings entered by name in the Web Service editor.

Session Preference	1 specific	All	n/a (not stored in portal database) Session preferences can be set using the ALI Scripting Framework. For details, see *Using Session Preferences* on page 304.	ALI 6.0 and above. Session preferences are persisted for the duration of the user's session. A portlet has access to the session preferences entered by name in the Web Service editor. Note: Portlets on the same portal page and the same remote server cannot use the same session.
Portlet Setting	1 specific	1 specific	Portlet Preferences page	
CommunityPortlet Setting	All in a specific community	1 specific	Community Preferences page	Only available when a portlet is displayed in a community (not on a My Page), and can be set only by a community owner.
Community Setting	All in a specific community	All in a specific community	Community Preferences page	Only available when a portlet is displayed in a community (not on a My Page), and can be set only by a community

				owner. A portlet has access to the Community settings entered by name in the Web Service editor.
User Information Setting	1 specific	All	n/a (not stored in portal database)	User Information settings required by a portlet must be configured in the Web Service editor on the User Information page.

About Administrative Preferences and Portlet Template Preferences Pages

Administrative Preference pages and Portlet Template Preference pages are used to manipulate Administrative settings for specific portlets.

Administrative settings affect all users of a specific portlet and can only be defined by administrative users with read/write access to the associated object.

- Administrative Preferences pages are accessible only from within the associated Portlet editor, and only to users with administrative rights in the portal and read/write access to the portlet object.
- Portlet Template Preferences pages are accessible only from within the associated Portlet Template editor, and only to users with administrative rights in the portal and read/write access to the Portlet Template. Administrative settings that are set via a Portlet Template Preferences page apply to all portlet objects created from that Portlet Template.

For details, see the following topics:

- *About ALI Portlet Settings* on page 294
- *Portlet Settings Development Tips* on page 294
- *Creating an Administrative Preferences Page* on page 298

Creating an Administrative Preferences Page

To create an Administrative Preferences page or Portlet Template Preference page, deploy the page to the remote server that hosts the portlet and enter the page URL in the Web Service editor.

The URLs to the Administrative Preferences page and/or Portlet Template Preference page are specified in the Web Service editor on the Preferences page. The base URL is defined by the associated Remote Server.

A Portlet Template Preferences page is structured exactly like a standard Administrative Preferences page; the only difference is that the settings on the page apply to all portlets created from the associated portlet template. The IDK provides interfaces to manipulate settings in the portal database. For an example of setting and accessing settings, see *Creating a Portlet Preferences Page* on page 301.

About Community Preferences Pages

Community Preferences pages are used to manipulate Community and CommunityPortlet settings, allowing Community Owners to modify content for all users without leaving the community.

A Community Preference page is used to define settings that apply only within the current community, either to all portlets and users (Community Preference) or a single portlet for all users (CommunityPortlet Preference). Community Preference pages are accessed from the Community Editor, accessible only to Community Owners. The Community Editor also allows you to disable the portlet title bar within the community.

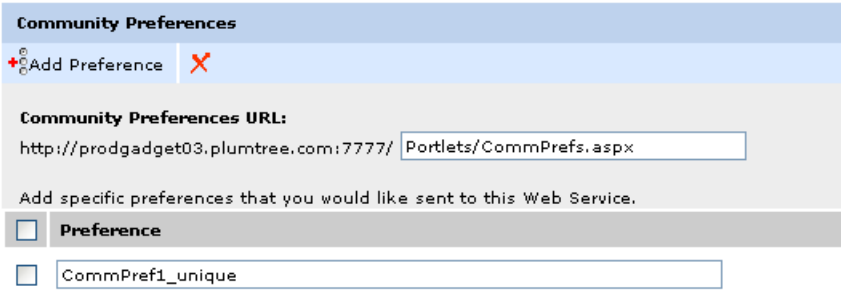
For details, see the following topics:

- *About ALI Portlet Settings* on page 294
- *Portlet Settings Development Tips* on page 294
- *Creating a Community Preferences Page* on page 299

Creating a Community Preferences Page

To create a Community Preferences page, deploy the page to the remote server that hosts the portlet and enter the page URL in the Web Service editor.

The URL to the Community Preferences page is specified in the Web Service editor on the Preferences page. The base URL is defined by the associated Remote Server. All Community settings required by the portlet must be entered by name in the Preference list on this page. If a setting is not entered in this list, it will not be available to the portlet. CommunityPortlet settings do not need to be entered. Any pages that access settings must be gatewayed.

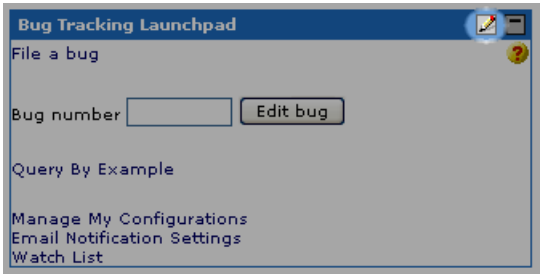


> The IDK provides interfaces to manipulate settings in the portal database. For an example of setting and accessing settings, see *Creating a Portlet Preferences Page* on page 301.

About Portlet Preferences Pages

Portlet Preferences pages are accessible from the portal page and can be used to manipulate both Portlet and User settings.

A Portlet Preferences page is used to define settings that affect a single user. Portlet settings apply to one specific portlet object and one particular user. User settings apply to one specific user but can be used by multiple portlets and services.

If a portlet has been configured with a Portlet Preferences page, an edit icon appears in the portlet title bar. 

When a user first adds the portlet to a page, it might not be obvious where to enter necessary settings. If a portlet requires configuration, always enter a link to the preferences page in the Web Service editor.

For details, see the following topics:

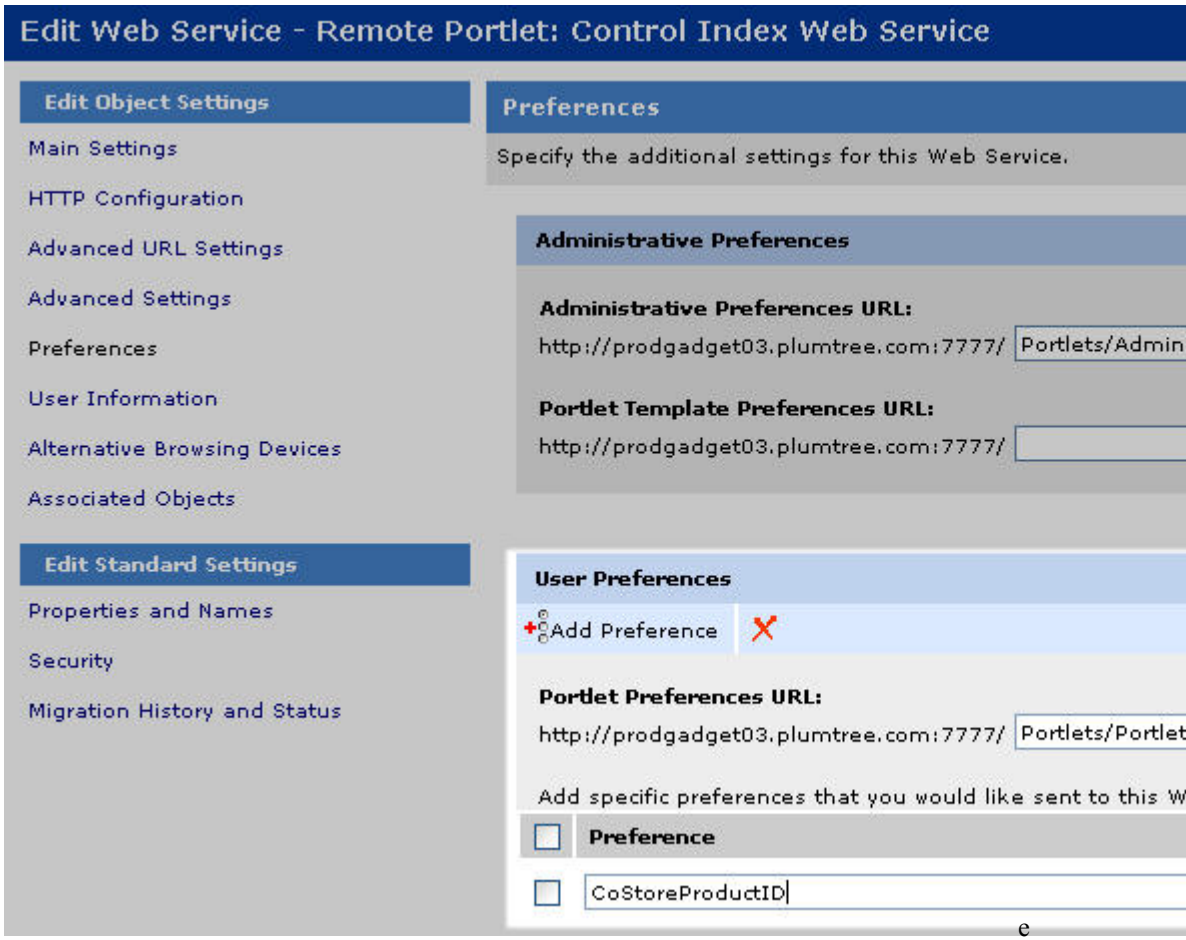
- *About ALI Portlet Settings* on page 294
- *Portlet Settings Development Tips* on page 294
- *Creating a Portlet Preferences Page* on page 301

Creating a Portlet Preferences Page

To create a Portlet Preferences page, deploy the page to the remote server that hosts the portlet and enter the page URL in the Web Service editor.

The URL to the Portlet Preferences page is specified in the Web Service editor on the Preferences page. The base URL is defined by the associated Remote Server. All User settings required by the portlet must be entered by name in the Preference list on this page. If a setting is not entered in this list, it will not be available to the portlet. Portlet settings do not need to be entered.

Note: User settings are shared among all services, so you must use a unique setting name. For example, "Exchange55Password" is less likely to result in a name collision than "password".



IDK provides interfaces to manipulate settings in the portal database. In the example code below, two portlets share the User setting CoStoreProductID. Note: The setting name must be entered in the Web Service editor for both portlets. All portlet files must be gatewayed.

The first portlet provides a form for the user to enter the product ID, and stores this information as a User setting. **Portlet 1 - Java**

```
<%@ page language="java"
import="com.plumtree.remote.portlet.*,java.util.Date" %>

IPortletContext portletContext =
PortletContextFactory.createPortletContext(request,
```

```
response); IPortletResponse portletResponse =
portletContext.getResponse();
IPortletUser portletUser = portletContext.getUser();
IPortletRequest portletRequest = portletContext.getRequest();

// Get the incoming Product ID from the query string
String currProduct = request.getParameter("ID");

if (null == currProduct)
  {
  currProduct = "";
  }
portletResponse.setSettingValue(SettingType.User,
"CoStoreProductID",
sCurrProduct);
// Redirect to the Company Store Community
portletResponse.returnToPortal();
...
```

Portlet 1 - .NET

```
...
Dim portletContext As IPortletContext
portletContext =
PortletContextFactory.CreatePortletContext(Request, Response)

Dim portletRequest As IPortletRequest
portletRequest = PortletContext.GetRequest

Dim portletUser As IPortletUser
portletUser = PortletContext.GetUser

Dim portletResponse As IPortletResponse
portletResponse = PortletContext.GetResponse

portletResponse.SetSettingValue(SettingType.User,
"CoStoreProductID",  Request.QueryString("ID"))
...
```

The second portlet checks for the User setting before building its display. (The portlet then retrieves the stored User setting from the ALI database and displays the product.)

Portlet 2 - Java

```
...
currentProductID = portletRequest.getSettingValue(SettingType.User,
```

```
 "CoStoreProductID");
...
```

Portlet 2 - .NET

```
...
Dim currentProductID As String
currentProductID = portletRequest.GetSettingValue(SettingType.User,
 "CoStoreProductID")
...
```

User settings can also be entered on the User Configuration page, accessible from the My Account page in the portal. For details, see the portal online help.

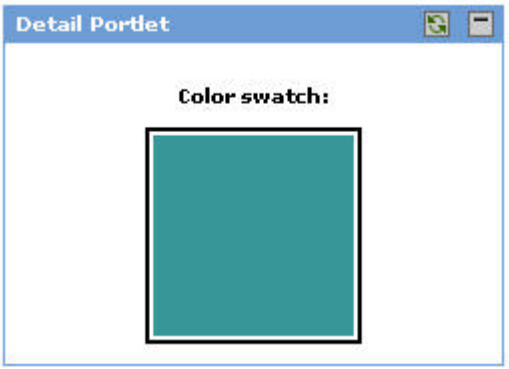
Using Session Preferences

To store and share settings within the client browser, use session preferences.

Pagelets can use preferences to communicate with each other, but accessing preferences usually requires a round trip to the portal database. Session preferences provide a way to store and share settings in the user's session within the client browser.

The Master-Detail design pattern illustrates the most basic usage of session preferences. This design pattern splits control and display between two portlets. For example, the "master" portlet could summarize data in list form, and the "detail" portlet could display details on each data item in response to user selection. In the example below, the master portlet displays a form that allows you to enter a color code in a text box.

When the user enters a color code in the text box, the color in the detail portlet changes.



For each onkeyup event that occurs in the "Enter color" text box in the master portlet, the following steps are executed:

1. The master portlet sets the session preference using the current value of the text box.
2. The master portlet calls an update method on the detail portlet.
3. The detail portlet retrieves the session preference to get the color value.
4. The detail portlet redraws its color swatch area to reflect the new color value.

Pagelets can manipulate session preferences using the ALI Scripting Framework or the IDK. Sample code for both options is provided below.

Note: Shared session preferences must be specified by name on the Preferences page of the associated Web Service Editor or they will not be sent to the pagelet.

ALI Scripting Framework Methods

This example is oversimplified; the master portlet makes a direct call to a JavaScript method of the detail portlet. Unless the master portlet takes extra measures to ensure that the detail portlet is actually present on the same page, calls from master to detail could generate errors. The ALI Scripting Framework provides an easy way to detach the relationship between portlets and use a common event interface for communication. See *Using ALI Scripting Framework Event Notification* on page 262 for more information on inter-portlet communication.

Master Portlet

```
<div style="padding:10px;" align="center">
<p><b>Enter color:</b>  
<input type="text"
style="font-size:22px;font-weight:bold;text-align:center;"
id="master_prefName"
value="#FFFFFF" size="8"
```

```
onkeyup="master_setPrefs(this.value)"></p><br>
</div>

<script type="text/javascript">
function master_setPrefs(val)
{
var prefName = 'masterColor';
var prefValue = val;
PTPortlet.setSessionPref(prefName,prefValue);

master_debug('<b>Master Portlet</b> called
PTPortlet.setSessionPref(\'masterColor\',\'' + prefValue + '\').');

if (window.detail_update)
  {
  master_debug('<b>Master Portlet</b> calling detail_update().');

  detail_update();
  }
else
  {
  master_debug('Could not locate portlet <b>Detail Portlet</b> on
 page.');
  }
}
function master_debug(str)
{
if (window.PTDebugUtil)
  {
  PTDebugUtil.debug(str);
  }
}
</script>
```

Detail Portlet

```
<div style="padding:10px;" align="center">
<p><b>Color swatch</b>  
<div style="width:100px;height:100px;border:2px solid
black;padding:2px;"id="detail-swatch"></div>
<script>
function detail_update()
{
var color = PTPortlet.getSessionPref('masterColor');
detail_debug('<b>Detail Portlet</b> received value="' + color +
'" for PTPortlet.getSessionPref(\'masterColor\')');
```

```
var swatch = document.getElementById('detail-swatch');
if (swatch)
  {
  swatch.innerHTML = '<div style="background-color:' + color +
';width:100%;height:100%;"></div>';
  }
else
  {
  detail_debug('<b>Detail Portlet</b> cannot find \'detail-swatch\'
 DIV element.');
  }
}

function detail_debug(str)
{
if (window.PTDebugUtil)
  {
  PTDebugUtil.debug(str);
  }
}
</script>
```

IDK Methods

In most cases, reading session preferences via the ALI Scripting Framework is inefficient and insecure. **Always use the IDK to read session preferences**, as shown in the example code below.

Java

```
<%@ page language="java"
import="com.plumtree.remote.portlet.*,java.util.Date" %>

IPortletContext portletContext =
PortletContextFactory.createPortletContext(request,response);
IPortletResponse portletResponse = portletContext.getResponse();

IPortletUser portletUser = portletContext.getUser();
IPortletRequest portletRequest = portletContext.getRequest();

masterColor = portletRequest.getSettingValue(SettingType.Session,
 "masterColor");
```

.NET

```
...
Dim portletContext As IPortletContext
```

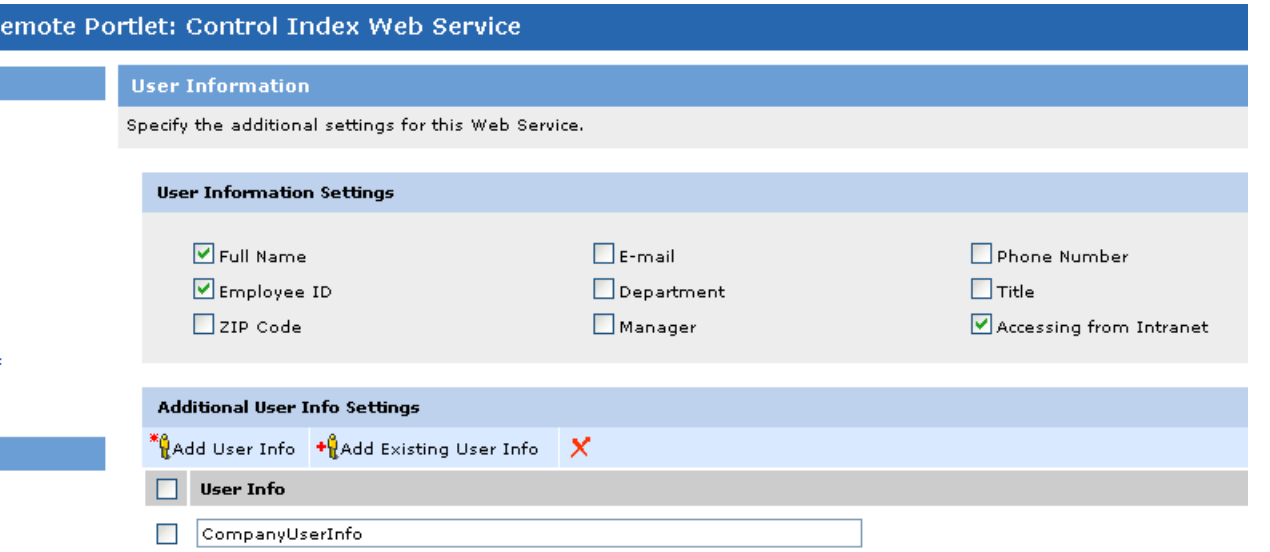
```
portletContext =
PortletContextFactory.CreatePortletContext(Request, Response)

Dim portletRequest As IPortletRequest
portletRequest = PortletContext.GetRequest

Dim portletUser As IPortletUser
portletUser = PortletContext.GetUser

Dim portletResponse As IPortletResponse
portletResponse = PortletContext.GetResponse

Dim masterColor As String
masterColor = portletRequest.GetSettingValue(SettingType.Session
 "masterColor")
...
```

Accessing User Information

Portlets can use the IDK to access read-only User Information settings that have been entered by users or imported into the portal using a Profile Source Identity Service.

The User Information settings required by a portlet must be selected in the Web Service editor on the User Information page. The standard settings appear at the top of the page; select any settings that should be sent to the portlet. You can select additional settings that already exist in the portal by clicking Add Existing User Info. You can also enter setting names manually; to add

rows to the list, click Add User



Once the portlet object has been configured as described above, portlet code can access the User Information settings sent from the portal using the IDK. In the following example code, the portlet retrieves the Company information property (CompanyUserInfo).

Java

```
...
// Get the user's company info setting
String companyID;
companyID = portletRequest.getSettingValue(SettingType.User,
"CompanyUserInfo");

// if the user's company info does not exist, retrieve it from
User Info properties
if (null == companyID) {
companyID = portletRequest.getSettingValue(SettingType.UserInfo,
 "CompanyUserInfo");
}
...
```

.NET

```
...
' Get the user's company info setting
```

```
Dim companyID As String
companyID = portletRequest.GetSettingValue(SettingType.User,
"CompanyUserInfo")

' if the user's company info does not exist, retrieve it from User
 Info properties
If companyID Is Nothing Then
companyID = portletRequest.GetSettingValue(SettingType.UserInfo,
 "CompanyUserInfo")
End If
...
```

About ALI Portlet Security

Portlets can be used to manipulate secure content. AquaLogic Interaction provides a variety of ways to control access to specific functionality.

- **Portal Roles (settings rights)** control whether or not a user has the right to change settings in the portal database. Administrative settings can only be changed by a portal administrator. Community settings can only be changed by a community Owner. To check which types of settings the current user has rights to change, use the IDK methods `IPortletUser.GetSettingsRights` and `IPortletUser.HasSettingsRight`. For details on portal roles, see the *Administrator Guide for AquaLogic Interaction* or the portal online help.
- **Activity Rights** confer system-wide privileges in the portal, such as the right to create new portal objects, including portlets, communities and folders. While ACLs control access to a specific object, activity rights confer a general, global privilege. You can create new activity rights to correspond to user privileges.

  Note: Activity rights apply to groups, and cannot be assigned directly to users. If a group is given an activity right, every member of the group inherits that activity right. Users' rights in the portal are the sum of the activity rights of all of the groups to which they belong.

  To access the current user's activity rights, configure the portal to send activity rights to the portlet on the Advanced Settings page of the Web Service Editor, and use the IDK methods `IPortletUser.GetActivityRights` and `IPortletUser.HasActivityRight`.
- **Access Control Lists (ACL)** govern which users can see each object in the portal and what they can do with it. An ACL is different from activity rights because it applies to a specific object. For details, see *Access Control List (ACL) Privileges* on page 74.

ACLs can be used to control access to content or functionality in community portlets. To determine the CommunityAccessLevel (in the Community ACL) for the current user in the current community, configure the portal to send the community ACL to the portlet on the Advanced Settings page of the Web Service Editor, and use the IDK method `IPortletUser.GetCurrentCommunityAccessLevel`. (This method can be used only if the portlet is on a community page.)

- **Encrypted credentials** should be used for all authentication credentials used by a portlet. The IDK provides encryption methods for use in portlets. For details, see *Using IDK Encryption* on page 313. Portlets can use four types of encryption:
  - **Advanced Encryption Standard (AES)** is private key encryption using 128-bit keys.
  - **RC2** is private key encryption using 64-bit keys.
  - **Base64** converts binary data into ASCII text and vice versa. Base64 does not require a key for decryption. Base64 is used by the credential vault if no RSA key is provided.
  - **RSA** is a public key/private key encryption type. In ALI 6.0 and above, the credential vault provides a central repository that securely stores and manages all credentials. Portlets that need credentials to access back-end applications can securely retrieve the appropriate user credentials from a central location. To use RSA encryption with IDK methods, you must use the credential vault. For details, see *Using the ALI Credential Vault* on page 311.
- All portlets should obey **SSL** rules because ALI can be configured to run under SSL. When you are testing against SSL (https://), make sure all images come through and do not pop up an "Unsecure items" dialog. Any portlet that uses a password that is not encrypted should follow the rules below:
  - Do not store any passwords in the database in clear text.
  - Do not expose passwords on every request. Only send the password when it is required (usually in the finalize method).

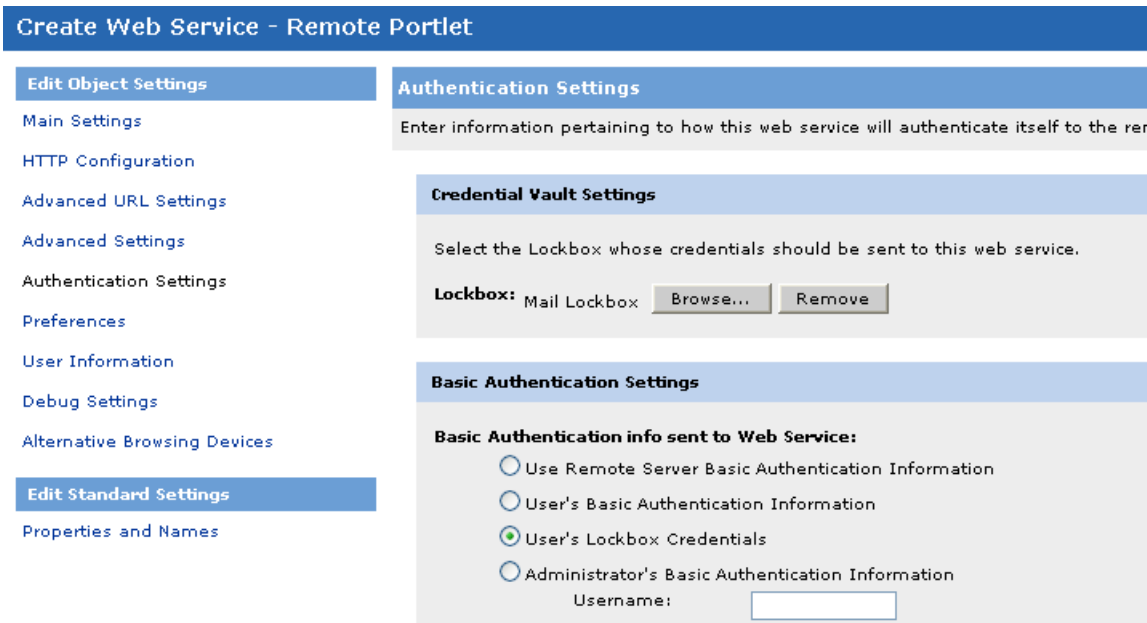
Using the ALI Credential Vault

The credential vault (ALI 6.0 and above) provides a central repository that securely stores and manages all credentials. Portlets that need login information to access a back-end application can securely retrieve the appropriate user credentials from a central location. Users enter their credentials once in their account settings and have seamless access to every application they interact with throughout the portal session.

Credentials are sent in portlet headers, using RSA public key/private key encryption. The IDK ICredentialProvider interface allows portlets to access user credentials stored in the central credential vault.

To use the credential vault, there must be a Lockbox in the portal associated with the authentication source. To create or configure a Lockbox, go to portal administration and click **Choose Utility ➤ Credential Vault Manager** . For details, see the portal online help. To configure the credential vault for use with your portlet, three steps are required:

1. In the Remote Server Editor associated with the portlet, enter the Public Encryption Key.
2. In the Web Service Editor on the Authentication Settings page, choose the appropriate Lockbox and set the Basic Authentication Settings to User's Lockbox Credentials.



3. Provide the private key for RSA encryption in one of two ways:
   - Enter the private key in the RSAPrivateKey parameter in the IDK web.xml/Web.config file on the remote server.
   - Set the private key programmatically using the ICredentialProvider.setPrivateKey method as shown in the example below.

   If you do not enter a key, the credential vault will use Base64 encryption.

The IDK ICredentialProvider interface lets you retrieve the user name and password from portlet headers with a few lines of code.

Note: If the private key for RSA encryption is set in the web.xml/Web.config file, the setPrivateKey method is not required. The values in the configuration file override any value set through the setPrivateKey method.

Java

```
// get an ICredentialProvider instance from IPortletContext
IPortletContext portletContext =
PortletContextFactory.createPortletContext(req, resp);
ICredentialProvider cProvider =
CredentialManager.getProviderInstance(req);

// set the private key used to decrypt the password
cProvider.setPrivateKey(rsaPrivateKeyString);

// get the username and password
String username = cProvider.getUsername();
String password = cProvider.getPassword();
```

.NET

```
// get an ICredentialProvider instance from IPortletContext
IPortletContext portletContext =
PortletContextFactory.CreatePortletContext(req, resp);
ICredentialProvider cProvider =
portletContext.GetCredentialProvider();

// set the private key used to decrypt the password
cProvider.SetPrivateKey(rsaPrivateKeyString);

// get the username and password
String username = cProvider.GetUsername();
String password = cProvider.GetPassword();
```

You can also use ICredentialProvider to access settings encrypted in RC2, AES and Base64 that are stored in the ALI database. For details, see *Using IDK Encryption* on page 313.

Using IDK Encryption

The IDK provides standard methods for encrypting and decrypting credentials stored in the ALI database.

In ALI 6.0 and above, you can use the IDK to access credentials from the credential vault. If you are not using the credential vault, you must set the encryption type and associated key, and the setting type and setting names. You can enter these parameters in the IDK web.xml/Web.config file, or set them programmatically. Both options are detailed below.

- To configure encryption in the web.xml/Web.config file, enter values for the following parameters:

Parameter	Accepted Values
CredentialSettingType	Portal setting type: • GADGET: Portlet Preference • COMMUNITYGADGET: CommunityPortlet Preference • COMMUNITY: Community Preference • ADMIN: Administrative Preference • SESSION: Session Preference • USER: User Preference • USERINFO: User Information Setting
UsernameParameterName	The setting name for the user name setting (for example, MyAppUserName).
PasswordParameterName	The setting name for the password setting (e.g., MyAppPassword).
CredentialEncryptionType	Encryption type: • BASE64 • RC2 • AES • NONE (RSA encryption is only available with the credential vault.)
RC2PrivateKey	String of private key for RC2 encryption.
AESPrivateKey	String of private key for AES encryption.

Note: The encryption settings in the configuration file will override any values set programmatically. If you do not include encryption settings in the configuration file, you must set them programmatically as shown below.

- To encrypt and store credentials in the portal database, use ICredentialSetter.

Java

```
// get an ICredentialSetter instance from IPortletContext
IPortletContext portletContext =
PortletContextFactory.createPortletContext(req, resp);
ICredentialSetter cSetter =
portletContext.getCredentialSetter();

// set the header type and parameter names
cSetter.setCredentialSettingType(SettingType.User);
cSetter.setUsernameParameterName("MyAppUserName");
cSetter.setPasswordParameterName("MyAppPassword");

// set the encryption type and key
cSetter.setCredentialEncryptionType(EncryptionType.RC2);
cSetter.setPrivateKey("skiroblbpauwyryrhfvnmsl");

// set the user name and password
cSetter.setUsername(username);
cSetter.setPassword(password);
```

.NET

```
// get an ICredentialSetter instance from IPortletContext
IPortletContext portletContext =
PortletContextFactory.CreatePortletContext(req, resp);
ICredentialSetter cSetter =
portletContext.GetCredentialSetter();

// set the header type and parameter names
cSetter.SetCredentialSettingType(SettingType.User);
cSetter.SetUsernameParameterName("MyAppUserName");
cSetter.SetPasswordParameterName("MyAppPassword");

// set the encryption type and key
cSetter.SetCredentialEncryptionType(EncryptionType.RC2);
cSetter.SetPrivateKey("skiroblbpauwyryrhfvnmsl");

// set the user name and password
cSetter.SetUsername(username);
cSetter.SetPassword(password);
```

- To decrypt credentials stored in the portal database, use ICredentialProvider.

Java

```
// get an ICredentialProvider instance from IPortletContext
IPortletContext portletContext =
PortletContextFactory.createPortletContext(req, resp);
ICredentialProvider cProvider =
portletContext.getCredentialProvider();

// set the header type and parameter names
cProvider.setCredentialSettingType(SettingType.User);
cProvider.setUsernameParameterName("MyAppUsername");
cProvider.setPasswordParameterName("MyAppPassword");

// set the encryption type and key
cProvider.setCredentialEncryptionType(EncryptionType.RC2);
cProvider.setPrivateKey("skiroblbpauwyryrhfvnmsl");

// get the username and password
String username = cProvider.getUsername();
String password = cProvider.getPassword();
```

.NET

```
// get an ICredentialProvider instance from IPortletContext
IPortletContext portletContext =
PortletContextFactory.CreatePortletContext(req, resp);
ICredentialProvider cProvider =
portletContext.GetCredentialProvider();

// set the header type and parameter names
cProvider.SetCredentialSettingType(SettingType.User);
cProvider.SetUsernameParameterName("DCTMUsername");
cProvider.SetPasswordParameterName("DCTMPassword");

// set the encryption type and key
cProvider.SetCredentialEncryptionType(EncryptionType.RC2);
cProvider.SetPrivateKey("skiroblbpauwyryrhfvnmsl");

// get the username and password
String username = cProvider.GetUsername();
String password = cProvider.GetPassword();
```

About Pagelet Internationalization

These tips and best practices apply to all pagelets that will be translated into multiple languages.

Identify ALL culturally dependent data. Text messages are the most obvious example of locale-specific data, but there are many other parts of a service that can vary with language or location. These include: images, UI labels and buttons, icons, sounds, graphics, dates, times, measurements, honorifics and titles, phone numbers, and postal addresses. In ALI portal, the title bar for the portlet must also be localized; for details, see *Modifying the Portlet Title Bar* on page 317.

Do not use compound messages (concatenated strings) to create text. Compound messages contain variable data. For example, in the text string "You have XX credits," only the integer "XX" will vary. However, the position of the integer in the sentence is not the same in all languages. If the message is coded as a concatenated string, it cannot be translated without rewriting the code.

Use the IDK to avoid encoding issues. All content is stored in the database in Unicode. The IDK handles encoding for international characters.

For details on implementing internationalization, see *Using Internationalized Strings in Adaptive Tags* on page 205.

Modifying the Portlet Title Bar

The portlet title bar is the solid colored bar that displays the portlet name at the top of each portlet on an ALI portal page. The portlet code has full control over the text and functionality displayed in the title bar.

The default title for a portlet is entered in the Portlet Editor. In internationalized portlets, the portlet title bar should be localized.

To override the default title, use the IDK method `PortletResponse.setTitle` as shown in the sample VB code below.

```
<%
Dim portletContext As IPortletContext
portletContext =
PortletContextFactory.CreatePortletContext(Request, Response)

Dim portletResponse As IPortletResponse
portletResponse = PortletContext.GetResponse()
```

```
Dim portletRequest As IPortletRequest
portletRequest = PortletContext.GetRequest()

portletResponse.SetTitle("New Title")
...
```

This code can be combined with logic to determine the locale of the user and display the title in the appropriate language. For details on internationalizing portlet content, see *Using Internationalized Strings in Adaptive Tags* on page 205.

About Pagelet Caching

Caching is the functionality that allows ALI and Ensemble to request pagelet content, save the content, and return the saved content to users when appropriate. The importance of caching cannot be overstated.

Efficient caching makes every web application faster and less expensive. The only time content should not be cached is if the data must be continuously updated. If every pagelet had to be freshly generated for each request, performance could become unacceptably slow. ALI and Ensemble rely on caching to improve performance. Pagelet content is cached and returned when later requests match the cache's existing settings.

Caching is indexed on the settings sent by the pagelet. When the ALI or Ensemble gateway server processes a request for a page, it looks individually at each pagelet on the page and checks it against the cache. The process can be summarized as follows:

1. The gateway server assembles a cache key used to uniquely identify each pagelet in the cache.
2. The gateway server checks the cache for a matching cache key entry:
   - If the gateway server finds a match that is not expired, it returns the content in the cache and does not make a request to the remote server.
   - If there is no matching cache key for the pagelet or if the cache key has expired, the gateway server makes a request to the remote server. If the matching cache entry uses ETag or Last-Modified caching, it also sends the appropriate caching header to the remote server in the request.
3. The response comes back from the remote server; the gateway server checks for caching headers:
   - If the headers include an Expires header, the gateway server stores the new pagelet content (along with a new expiration date) in its cache.

- If the headers use ETag or Last-Modified caching, the existing cache entry might be revalidated (in the case of '304-Not Modified') or new pagelet content might be stored in the cache.

ALI and Ensemble cache gatewayed content to complement, not replace, browser caching. Public content is accessible to multiple users without any user-specific information (based on HTTP headers). The gateway server calculates the cache headers sent to the browser to ensure that the content is properly cached on the client side.

ALI and Ensemble cache all text (i.e., nonbinary) content returned by GET requests. Even if gateway caching is disabled (via PTSpy), pagelet caching still takes place. Gatewayed content can be cached by a proxy server or by the user's browser. Beware browser caching of gatewayed content; it is a good idea to clear your browser cache often during development. An incorrectly set Expires header can cause browsers to cache gatewayed content.

The pagelet cache contains sections of finished markup and sections of markup that require further transformation. Post-cache processing means content can be more timely and personalized. Adaptive tags enable certain pagelets (for example, Community banners) to be cached publicly for extended periods of time and yet contain user specific and page-specific information, as well as the current date and time.

For details, see the following topics:

- *About Pagelet Caching Strategies* on page 319
- *Pagelet/Portlet Cache Key* on page 320 *Setting HTTP Caching Headers - Cache-Control* on page 322
- *Setting HTTP Caching Headers - Expires* on page 324
- *Setting HTTP Caching Headers - Last-Modified and ETag* on page 325 *Implementing Portlet Caching* on page 322
- *Configuring ALI Portlet Caching Settings* on page 325

For a full explanation of HTTP caching, see RFC 2616 (*http://www.w3.org/Protocols/rfc2616/rfc2616.html*).

About Pagelet Caching Strategies

Pagelet caching is controlled both by the programmer and by the administrator who registers the pagelet in ALI or Ensemble. Each and every pagelet needs a tailored caching strategy to fit its specific functionality.

A pagelet's caching strategy should take all possibilities into account and use the most efficient combination for its specific functionality. A pagelet that takes too long to generate can degrade

the performance of every page that displays it. These questions can help you determine the appropriate caching strategy:

- Will the content accessed by the pagelet change? How often?
- How time-critical is the content?
- What processes are involved in producing pagelet content? How expensive are they in terms of server time and impact?
- Is the pagelet the only client with access to the back-end application?
- Is the content different for specific users?
- Can users share cached content?

Determine how often pagelet content must be updated, dependent on data update frequency and business needs. Find the longest time interval between data refreshes that will not negatively affect the validity of the content or the business goals of the pagelet.

Since caching is indexed on the settings used by a pagelet, new content is always requested when settings change (assuming that no cached content exists for that combination of settings).

There are two common situations in which you might mistakenly decide that a pagelet cannot be cached:

- **In-place refresh**: You might think that caching would "break" a pagelet that uses in-place refresh because the pagelet would be redirected to the original (cached) content. This can be avoided if a unique setting is updated on every action that causes a redraw, effectively "flushing" the cache. (In-place refresh renews the portlet display by causing the browser to refresh the portal page at a set interval.)
- **Invisible preferences**: If the content of the pagelet is dependent on something other than preferences (for example, the pagelet keys off the User ID to display a name or uses ALI security to filter a list), caching can still be implemented with “invisible preferences” (in this case, User ID). As with in-place refresh, invisible preferences are set solely for the purpose of creating a different cache entry. They are set programmatically, without the user’s knowledge.

For details on implementing caching, see the following topics:

- *Setting HTTP Caching Headers - Cache-Control* on page 322
- *Setting HTTP Caching Headers - Expires* on page 324
- *Setting HTTP Caching Headers - Last-Modified and ETag* on page 325
- *Implementing Portlet Caching* on page 322

Pagelet/Portlet Cache Key

The cache key for a pagelet/portlet entry in ALI consists of these values.

Portlet ID	The unique ID for the portlet, defined by the portal.
Content Mode	The content mode of the portlet.
Portal Settings	All seven types of settings stored in the ALI database: Portlet settings, User settings, Community settings, CommunityPortlet settings, Administrative settings, Session preferences, and User Information.
User Interface	The type of device used to access the portlet.
CanSet values	All three values for the CanSet header: CanSetPersonal, CanSetCommunity, CanSetAdmin
LocaleID	The ID for the portal-defined locale associated with the current user.
UserID	The unique ID for the current user. Included only if private caching is used.
URI	The URL to the portlet page on the remote server.
Community ID	Included only if the portlet is displayed on a community page.
Last-modified date	The last modified date of the portlet.

The data below can be added to the cache key by setting options in the Web Service editor on the Advanced Settings page.

Community ACL	The ACL for the community in which the portlet is displayed.
Page ID	The ID for the portal page on which the portlet is displayed.
TimeZone	The time zone for the portal in which the portlet is displayed.
Experience Definition ID	The ID for the Experience Definition in which the portlet is displayed.
Portlet Alignment	The alignment of the portlet in the current page.
Activity Rights	Only the Activity Rights configured in the Web Service editor are included in the cache key.

The data below is deliberately not included in the cache key:

StyleSheetURI	Portal stylesheets are applied at runtime, depending on the user preference. Portlet content does not depend on the particular stylesheet that the user has selected.
HostpageURI	All parts of the Hostpage URI value are covered separately. The cache key includes Community ID, so it already distinguishes between My Pages and Community pages. The User ID is added if private caching is used.

Implementing Portlet Caching

Caching on the Portal Server can be set in two ways: programmatically through HTTP headers and/or using the administrative settings in the Portlet Web Service Editor. You should always implement caching programmatically, although the administrator can still choose to override caching through administrative settings.

For details on implementing caching, see the following topics:

- *Setting HTTP Caching Headers - Cache-Control* on page 322 *Setting HTTP Caching Headers - Expires* on page 324
- *Setting HTTP Caching Headers - Last-Modified and ETag* on page 325
- *Configuring ALI Portlet Caching Settings* on page 325

While caching is an integral and necessary part of portlet design, it is helpful to disable it while developing and debugging. Otherwise, it can be very difficult to view the results of any modifications you have made. To disable the caching implemented by the Portal Server, go to the HTTP Configuration page of the Portlet Web Service editor (shown under Portlet Settings above) and set the minimum and maximum caching times to 0. Clear the checkbox marked “Suppress errors where possible (use cached content instead).”

Note: After the code has been developed and debugged, make sure to turn caching on and test the performance of your portlet. For details on troubleshooting portlets, see Portlet Debugging. If you using the ALI Logging Utilities to debug caching, turn on all types of tracing for the OpenKernel.OpenHttp.Cache component.

Setting HTTP Caching Headers - Cache-Control

The Cache-Control header can be used to expire content immediately or disable caching altogether. The value of this header determines whether cached portlet content can be shared among different users.

The Cache-Control header can contain the following values:

public	Allows any cached content to be shared across users with identical sets of preferences using the same Portal Server. This value should be used whenever possible.
private	Tells the Portal Server not to share cached content. The User ID is added to the cache key so that a separate copy is retained in the cache for each individual user. This value should only be used to protect sensitive information, for example, an e-mail inbox portlet. (User settings can also make public content effectively private.)
max-age=[seconds]	Specifies the maximum amount of time that an object is considered fresh. Similar to the Expires header, this directive allows more flexibility. [seconds] is the number of seconds from the time of the request that the object should remain fresh.
must-revalidate	Tells the cache that it must obey any freshness information it receives about an object. HTTP allows caches to take liberties with the freshness of objects; specifying this header tells the cache to strictly follow your rules.
no-cache	Disables caching completely and overrides Web Service editor settings. Neither the client nor the Portal Server responds to subsequent requests with a cached version.

In JSP, use the setHeader method to configure the Cache-Control header:

```
<%
response.setHeader("Cache-Control","public");
%>
```

The JSP example below expires the content immediately using the maximum age header.

```
<%
response.setHeader("Cache-Control","max-age=0");
%>
```

In the .NET Framework, the Cache-Control header is accessed through the System.Web.HttpCachePolicy class. To set the header to public, private or no-cache, use the Response.Cache.SetCacheability method.

```
Response.Cache.SetCacheability(HttpCacheability.Public);
```

To set a maximum age for content in .NET, use the Response.Cache.SetMaxAge method. The example below expires the content immediately.

```
TimeSpan ts = new TimeSpan(0,0,0);
Response.Cache.SetMaxAge(ts);
```

To set the header to must-revalidate in .NET, use the Response.Cache.SetRevalidation method.

```
Response.Cache.SetRevalidation(HttpCacheRevalidation.AllCaches);
```

Setting HTTP Caching Headers - Expires

The Expires header specifies when content will expire, or how long content is "fresh." After this time, the Portal Server will always check back with the remote server to see if the content has changed.

Most Web servers allow setting an absolute time to expire, a time based on the last time that the client saw the object (last access time), or a time based on the last time the document changed on your server (last modification time).

In JSP, setting caching to forever using the Expires header is as simple as using the code that follows:

```
<%
response.setDateHeader("Expires",Long.MAX_VALUE);
%>
```

The .NET Framework's System.Web.HttpCachePolicy class provides a range of methods to handle caching, but it can also be used to set HTTP headers explicitly (see MSDN for API documentation: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemwebhttpcachepolicyclasssetexpirestopic.asp*). The Response.Cache.SetExpires method allows you to set the Expires header in a number of ways. The following code snippet sets it to forever:

```
Response.Cache.SetExpires(DateTime.Now.AddYears(100000000));
```

In .NET, the Web Form page (.aspx) can also use standard ASP methods to set HTTP headers.

Note: Never use Expires = 0 to prevent caching. The Expires header is sent by the remote server and passed through to the browser by the Portal Server. Unless the time on all three machines is synchronized, an Expires=0 header can mistakenly return cached content. To solve this problem, set the Expires header to a fixed date that is definitely in the past.

Setting HTTP Caching Headers - Last-Modified and ETag

The Last-Modified response header specifies the last time a change was made in the returned content, in the form of a time stamp. ETag values are unique identifiers generated by the server and changed every time the object is modified. Either can be used to determine if cached content is up to date.

When an object stored in the cache includes a Last-Modified or ETag header, the Portal Server can use this value to ask the remote server if the object has changed since the last time it was seen.

- The Portal Server sends the value from the Last-Modified header to the remote server in the If-Modified-Since Request header.
- The remote server sends the ETag header to the Portal Server with portlet content. When another request is made for the same content, the Portal Server sends the value in the ETag header back to the remote server in the If-None-Match header.

The portlet code on the remote server uses the header value to determine if the content being requested has changed since the last request, and responds with either fresh content or a 304 Not Modified Response. If the Portal Server receives the latter, it displays the cached content.

JSP portlets can access the value in the If-Modified-Since request header using the getLastModified(HttpServletRequest req) method provided by the Java class HttpServlet.

In .NET, the Response.Cache.SetLastModified method allows you to set the Last-Modified header to the date of your choice. Alternately, the SetLastModifiedFromFileDependencies method sets the header based on the time stamps of the handler's file dependencies.

```
Response.Cache.SetLastModified(DateTime.Now);
```

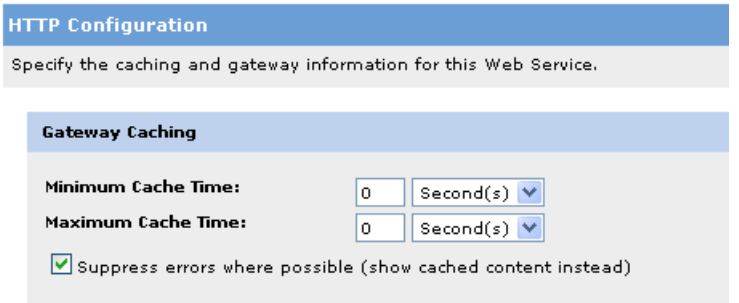
To use ETag in .NET, use the Response.Cache.SetETag method to pass in the string to be used as the ETag. The SetETagFromFileDependencies method creates an ETag by combining the file names and last modified timestamps for all files on which the handler is dependent.

Configuring ALI Portlet Caching Settings

In ALI , the HTTP Configuration page of the Web Service editor allows portal administrators to set minimum and maximum validation times for cached portlet content.

Note: Using HTTP headers to control caching is always preferable. Administrators can override some programmatic caching, but they cannot be relied upon to set caching correctly. If your portlet requires specific editor settings for its caching strategy, you must include this information in your Installation Guide.

The default cache settings are a minimum of 0 seconds and a maximum of 20 days.



The minimum and maximum caching settings in the Portlet editor affect caching as follows.

- The Portal Server **never** makes a request to the remote server **before the Minimum Cache Time** if there is content in the cache. (In version 6.0, the portlet cache is limited to 15 minutes, so a request will always be made after 15 minutes.) Multiple requests made for the same portlet with identical cachekeys within this minimum time always receive cached content. As noted earlier, setting the Cache-Control header to "no-cache" overrides editor caching settings; content will not be cached.
- The Portal Server **always** makes a request to the remote server **after the Maximum Cache Time**. Cached content might or might not be returned, based on other information (for example, the Last-Modified header).
- The Portal Server **might or might not** make a request to the remote server if content has been cached **in between the Minimum and Maximum Cache Time**. The Portal Server observes programmatic caching (for example, the Expires header) in the window between the minimum and maximum times.

Setting the Cache-Control header to "no-cache" overrides editor settings; content will never be cached.

For example, the minimum caching time for a particular portlet is set to ten minutes, and the maximum caching time is set to one hour. Client A requests the portlet content. Five minutes later, Client B, with an identical set of preferences, requests the same content. Five minutes is under the minimum caching time set in the Portlet editor, so cached content is returned, no matter what type of programmatic caching has been implemented by the portlet. (Remember, the Portal Server only abides by headers if cached content was generated between the minimum and maximum caching times set in the editor. An Expires header set to two minutes does not refresh the cache in this example.) If no copies of the content existed for Client B's particular collection of settings or no content was cached, the remote server would be called to generate content that matched that group of settings.

To continue the example, Client A requests the portlet content again, and there is a matching copy of the content in the cache that is 15 minutes old. This is over the minimum caching time and under the maximum. In this case, whether or not new content is generated depends on the HTTP headers sent by the portlet. If the portlet has not specified any caching programmatically, the Portal Server asks the remote server for fresh content. If the portlet set the Expires header to 30 minutes, new content is not generated. If ETag or Last-Modified caching was implemented, new content is only returned if content has changed.

Finally, Client A requests the same content two hours later, and the matching copy was generated more than an hour before. Since this is over the maximum caching time set in the Portlet editor, the Portal Server requests new content from the remote server, regardless of the caching specified programmatically by the portlet. Of course, if the portlet has implemented ETag or Last-Modified caching, new content is only returned if content has changed.

About Ensemble Pagelet Development

The following topics provide general information about Ensemble pagelet development and configuration.

- *About the IDK Proxy API* on page 328: The bea.alui.proxy package supports cross-product pagelet development. The interfaces in this package provide access to information about the environment in which the pagelet is displayed and the user currently accessing the pagelet, including session preferences associated with that user. This package also includes Ensemble-specific methods to implement security and access XML payloads. For details on creating pagelets with the IDK Proxy API, see *Creating a Custom Pagelet with the Java IDK Proxy API* on page 329 and *Creating a Custom Pagelet with the .NET IDK Proxy API* on page 330.
- *About Programmable Remote Client (PRC) Remote APIs* on page 54: The plumtree.remote.prc package includes a collection of APIs that provide access to functionality within the ALI portal, Collaboration, Publisher, and Search Service. These APIs are supported by Ensemble and ALI portal, and can be used by any pagelet deployed in an environment with access to these applications.
- *About Ensemble Pagelet Development* on page 327: Adaptive Tags are used to display contextual data and control Ensemble and ALI portal functionality from remote pagelets. Unlike the IDK, Adaptive Tags use XML in pagelet content instead of code, which avoids a network round trip. Tags can be included in the markup returned by any proxied page (HTML, JSP or ASP.Net). Using the attributes defined in the tag, the Ensemble or ALI portal gateway

transforms the XML and replaces it with standard HTML and/or executes the relevant operations. The Adaptive Tag collection currently includes libraries for use in both ALI portal and Ensemble, as well as libraries that are specific to each environment. For details on Ensemble-specific tags, see *Ensemble Adaptive Tag Library (pt:ensemble)* on page 242.

- *About the ALI Scripting Framework* on page 260: The ALI Scripting Framework is a set of client-side JavaScript libraries that provide services to pagelets and proxied/gatewayed pages.
- *About Ensemble Security* on page 333

For details on Ensemble pagelet configuration, see the Ensemble online help.

About the IDK Proxy API

The IDK Proxy API supports cross-product pagelet development. The interfaces in this package provide access to information about the environment in which the pagelet is displayed and the user currently accessing the pagelet, including session preferences associated with that user. This package also includes Ensemble-specific methods to implement security and access XML payloads.

This page provides an introduction to the IDK Proxy API. For more details on objects and methods, see the IDK API documentation.

The bea.alui.proxy package/namespace includes the following interfaces:

- `IProxyContext`
- `IProxyRequest`
- `IProxyResponse`
- `IProxyUser`

In general, these interfaces are called in the following order:

1. A pagelet uses `ProxyContextFactory.getInstance().createProxyContext` to initiate a connection for communicating with Ensemble or the portal.
2. The `IProxyContext` object returned allows the pagelet to access information about the request and response, the current user, and the session. The pagelet uses this information as needed, in arbitrary order, to generate a proper response. Using `IProxyContext`, the pagelet can access `IProxyRequest`, `IProxyUser`, `IRemoteSession` and `IProxyResponse`.
3. The pagelet retrieves parameters from the request using `IProxyRequest`.
4. The pagelet retrieves user information and preferences from `IProxyUser`.
5. The pagelet can access functionality in AquaLogic applications using `IRemoteSession`. For details, see *About Programmable Remote Client (PRC) Remote APIs* on page 54.
6. The pagelet constructs a response using `IProxyResponse`. The response includes content to be displayed and any settings to be stored or removed.

For examples of using IProxy interfaces in a pagelet, see *Creating a Custom Pagelet with the Java IDK Proxy API* on page 329 and *Creating a Custom Pagelet with the .NET IDK Proxy API* on page 330.

Creating a Custom Pagelet with the Java IDK Proxy API

This example creates a simple pagelet that displays information from the proxy, including setting values.

1. Before writing any code, create a new IDK project as described in *Setting Up a Custom Java IDK Project in Eclipse Stand-Alone (without WTP)* on page 12 or *Setting Up a Custom Java IDK Project in Eclipse with WTP* on page 10.
2. In the new project, create a new JSP page for the pagelet (pagelet.jsp).
3. Implement your code. The pagelet code shown below instantiates the IDK and uses the `IProxyContext` interface to retrieve `IProxyRequest` and `IProxyUser` objects to access information about the user and the settings associated with the pagelet.

   Note: There is no need to include html, head and body tags; the display is handled by the Consumer resource.

```
<%@ page language='java' import='com.bea.alui.proxy.*' %>
<%
String Att1 = 'no setting';
String Att2 = 'no setting';
String sessionVariable = 'no setting';

//get the idk
IProxyContext proxyContext =
ProxyContextFactory.getInstance().createProxyContext(request,
response);
IProxyRequest proxyRequest = proxyContext.getProxyRequest()

IProxyUser proxyUser = proxyRequest.getUser();
String userName = proxyUser.getUserName();
int userID = proxyUser.getUserID();

Att1 = proxyRequest.getSetting('Att1')
Att2 = proxyRequest.getSetting('Att2');
sessionVariable = proxyRequest.getSetting('sessionVar');

byte[] payload = proxyRequest.getPayload().getText();
String payloadStr = new String(payload)
%>
```

```
<p>User name: <%=userName%><br/>
User ID: <%=userID%><br/>
Attribute 1: <%=Att1%><br/>
Attribute 2: <%=Att2%><br/>
Session variable: <%=sessionVariable%><br/>
Payload: <textarea name=xml cols=80 rows=6> <%=payloadStr%>
</textarea>
</p>
```

Creating a Custom Pagelet with the .NET IDK Proxy API

This example creates a simple pagelet that displays information from the proxy, including setting values. .NET pagelets use a code-behind page (.aspx.cs) to retrieve settings and a web form (.aspx) to display the pagelet content.

1. Before writing any code, create a new IDK project as described in *Setting Up a Custom .NET IDK Project* on page 14.
2. In the new project, implement your code. The example below uses a code-behind page and a web form.

The code-behind page (IDKPagelet.aspx.cs) instantiates the IDK and uses the `IProxyContext` interface to retrieve `IProxyRequest` and `IProxyUser` objects to access information about the user and the settings associated with the pagelet.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Plumtree.Remote.Portlet;
using System.Xml;
using System.Text;
using Bea.Alui.Proxy;

namespace IDKProxyWS
{
/// <summary>
/// Hello World Pagelet
/// </summary>
   public class IDKPagelet : System.Web.UI.Page
```

```
   {
      public String name;
      public bool isGuest;
      public int userID;
      public String envType;
      public String payload;
      public String Att1,Att2;
      public String SessionVar;
      private void Page_Load(object sender, System.EventArgs e
      {
         // Put user code to initialize the page here
         InitializeCSP();
      }
      private void InitializeCSP()
      {
         IProxyRequest proxyRequest;
         IProxyResponse proxyResponse;
         IProxyUser proxyUser;
         IProxyContext proxyContext;
         ProxyContextFactory factory;
         HttpRequest request = HttpContext.Current.Request;
         HttpResponse response = HttpContext.Current.Response;

         try
         {
            factory = ProxyContextFactory.getInstance();
            proxyContext = factory.CreateProxyContext(request,
response);
            proxyRequest = proxyContext.GetProxyRequest();
            proxyResponse = proxyContext.GetProxyResponse();
            envType =
proxyRequest.GetEnvironment().GetType().ToString();
            proxyUser = proxyRequest.GetUser();
            isGuest = proxyUser.IsAnonymous();
            name= proxyUser.GetUserName();
            userID = proxyUser.GetUserID();

            Att1 = (String)proxyRequest.GetSetting('attr1');
            Att2 = (String)proxyRequest.GetSetting('attr2');
           Att2 = (String)proxyRequest.GetSetting('SessionVar');


           byte[] bpayload = proxyRequest.GetPayload().GetText()

            System.Text.ASCIIEncoding enc = new
```

```
System.Text.ASCIIEncoding()
            payload = enc.GetString(bpayload)
         }
         catch(Bea.Alui.Proxy.NotGatewayedException e)
         {
         }
      }
   }
#region Web Form Designer generated code
...
#endregion
}
```

The web form that displays the pagelet (IDKPagelet.aspx) displays the information retrieved by the code-behind page above.

```
<%@ Page Language='c#' runat='server'
CodeBehind='IDKPagelet.aspx.cs' AutoEventWireup='false'
inherits='IDKProxyWS.IDKPagelet' %>
<%@ import Namespace='System.Collections' %>
<%@ import Namespace='System.Web' %>
<%@ import Namespace='System.Web.UI' %>

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN' >
<html>
<head>
<title>IDKPagelet</title>
<meta name='GENERATOR' Content='Microsoft Visual Studio .NET 7.1'>
<meta name='CODE_LANGUAGE' Content='C#'>
<meta name='vs_defaultClientScript' content='JavaScript'>
<meta name='vs_targetSchema'
content='http://schemas.microsoft.com/intellisense/ie5'>
</head>

<body MS_POSITIONING='GridLayout'>
<span xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>

Proxy Pagelet <BR>
<%
   Response.Write('IDK Proxy Pagelet<BR>');
   Response.Write('Environment Type ' + envType + '<BR>');
   Response.Write('Guest User? ' + isGuest + '<BR>');
   Response.Write('User Name: ' + name + '<BR>');
   Response.Write('User ID: ' + userID + '<BR>');
   Response.Write('<P>');
```

```
   Response.Write('Pagelet Attributes:<BR>');
   Response.Write('Attribute1: ' + Att1 + '<BR>');
   Response.Write('Attribute2: ' + Att2 + '<BR>')
   Response.Write('SessionVar: ' + SessionVar + '<BR>')
   Response.Write('<P>')

   Response.Write('Pagelet XML Payload:<BR>');
   Response.Write('<textarea name=xml cols=80 rows=6>' + payload
 + '</textarea>');
   Response.Write('<P>');
%>
</span>
</body>
</html>
```

About Ensemble Security

There are two factors that control access to a resource in Ensemble: authentication and policies.

Ensemble manages authentication with each proxied application based on the settings defined for the associated resource. Each resource is protected by a policy set, which describes the conditions under which a user may be granted access to the resource, and the roles associated with those conditions. For details on using roles, see *Using Ensemble Roles in Pagelets and Proxied Applications* on page 333.

Using Ensemble Roles in Pagelets and Proxied Applications

Pagelets and proxied applications can use Ensemble roles to control access to content and functionality.

Each incoming request to Ensemble is evaluated against the policies for the requested resource. If the user is found to be in one or more roles, access is granted and the set of matching roles is passed on to the proxied application, allowing the application to determine the correct access level for the user. This is called **Role-Based Access Control (RBAC)**.

Roles are sent in an HTTP header and can be accessed using the Proxy IDK and adaptive tags.

Adaptive tags can be included in the markup returned by any proxied page, including pagelets. Using the attributes defined in the tag, Ensemble transforms the XML and replaces it with standard HTML to be displayed in a browser. For details, see *Ensemble Adaptive Tag Library (pt:ensemble)* on page 242.

- The `pt:ensemble.rolelist` tag creates a collection of the user's roles in the current context and stores it in memory using the name in the pt:key attribute. Each item in the

collection is a variable containing the role name. The example below displays a list of the user's roles by iterating over the collection using the pt:logic.foreach tag.

```
<pt:ensemble.rolelist pt:key='roles'/>
<pt:logic.foreach pt:data='roles' pt:var='role'>
<pt:logic.value pt:value='$role'/>
 <pt:logic.separator><br></pt:logic.separator>
</pt:logic.foreach><BR>
```

- The `pt:ensemble.roleexpr` tag evaluates an expression and stores the result as a boolean in memory using the name in the pt:key attribute. The example below checks if the user has the Admin role and displays a message based on the result using the pt:logic.if tag.

```
<pt:ensemble.roleexpr pt:expr='hasRole Admin' pt:key='hasrole'/>
<pt:logic.if pt:expr='$hasrole'>
   <pt:logic.iftrue>
      This user has the Admin role.
   </pt:logic.iftrue>
   <pt:logic.iffalse>
      Warning: This user DOES NOT have the Admin role.
   </pt:logic.iffalse>
</pt:logic.if>
```

The IDK `bea.alui.proxy.IProxyUser` interface also allows you to get a list of the user's roles in the current context, or determine whether the user has a specific role.

- The `IProxyUser.getRoles` method returns an iterator of the user's roles as strings.
- The `IProxyUser.isUserInRole` method determines whether the user is in the role passed in the role parameter and returns true if the user has the role (false otherwise).
- The `IProxyUser.isAnonymous` method determines whether the user is an Anonymous user.
- The `IProxyUser.isUserInRole` method determines whether the user is in the role passed in the role parameter and returns true if the user has the role (false otherwise).

The simplified example below (roleconsumer.jsp) retrieves role information for the current user. The associated Ensemble resource has three roles defined: AdminRole, MgrRole, and UserRole. (The associated policy set assigns these roles to groups or users.) In this example, the associated Ensemble pagelet is named 'rolePagelet'. For more details on the IDK proxy API, see the IDK API documentation.

```
 <%@ page language='java' import='com.plumtree.remote.portlet.*,
 java.util.Date, java.util.*, com.bea.alui.proxy.*' %>

You refreshed at <%= new Date().toString()%><br/>
```

```
<%
response.setHeader('Cache-Control','no-cache');  //HTTP 1.1
response.setHeader('Pragma','no-cache');  //HTTP 1.0
response.setDateHeader ('Expires', 0);  //prevents caching at the
 proxy server

IProxyContext ctx =
ProxyContextFactory.getInstance().createProxyContext(request,response);

IProxyRequest req = ctx.getProxyRequest();
IProxyResponse res = ctx.getProxyResponse();

Enumeration roles = req.getUser().getRoles();
boolean isAdmin = req.getUser().isUserInRole('AdminRole');
boolean isMgr = req.getUser().isUserInRole('MgrRole');
boolean isUser = req.getUser().isUserInRole('UserRole')
%>

<html>
<head>
<meta http-equiv='Content-Type' content='text/html;
charset=ISO-8859-1'>
<META HTTP-EQUIV='PRAGMA' CONTENT='NO-CACHE'>
<title>Preferences</title>
</head>

<body>
<br/> CONSUMER SETTINGS <br/>
<% while (roles.hasMoreElements()) {
   String role = (String)roles.nextElement();  %>
   <br/>User has role: <%=role%><br/>
<% } %>
<br/>User is admin? <%=isAdmin%><br/>
<br/>User is manager? <%=isMgr%><br/>
<br/>User is standard user? <%=isUser%><br/>

<pt:ensemble.inject
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'
pt:name='idkLib:rolePagelet'/>

</body>
</html>
```

Customizing the Ensemble Login Process

To display custom pages to the user at different steps during Ensemble login and logout, you can customize a range of steps in the login process.

You can create custom login, logout, error and interstitial pages and configure Ensemble to display them at specific points in the login/logout process. (An interstitial page is a page that appears before the expected content page.) Custom pages are hosted on a proxied application server, called the **login resource**.

For details, see the following topics:

Creating a Custom Ensemble Pre-Login Page

The pre-login page is an interstitial page displayed before the login form.

The pre-login page could display an important message about availability or new functionality. The pre-login page can also be used to display a custom message to users who are part of an experience definition that is blocked from accessing the requested resource.

In the example below, the pre-login page (preinterstitialpage.jsp) displays a message about server maintenance. This page uses the `pt:common.error` tag to display any errors within the page, and the `pt:core.html` tag to display the submit button.

```
 <%@page contentType="text/html;charset=UTF-8"%>
<HTML>
<BODY>
<SPAN xlmns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<FORM action="./processpreinterstitialpage.jsp" method="POST">
<P>
<TABLE>
<TR><TD>
<CENTER><B>Maintenance Updates</B></CENTER>
</TD></TR>
<TR><TD>
```

```
 This server will be going down for maintenance on 5/28/07 at
6:37PM and 24 seconds.<BR>
<pt:common.error>
<P><B><FONT color="red"><pt:logic.value
pt:value="$#10.ptmsgs_login"/></FONT>:</B>
<pt:common.errortext/>
</P>
</pt:common.error>
</TD></TR>
<TR><TD>
<CENTER>
<pt:core.html pt:tag="input" type="submit"
value="$#2.ptmsgs_samples"/>
</CENTER>
</TD></TR>
</TABLE>
</FORM>
</SPAN>
</BODY>
</HTML>
```

When the user clicks the submit button, the processing page shown below (processpreinterstitialpage.jsp) sets the `runner_pre_interstitial_complete` header to **true**, which directs the browser to the login page. It also provides error text for the error page in case processing fails.

```
 <%@page contentType="text/html;charset=UTF-8"%>
<HTML>
<BODY>
<%
out.println( "<P>Ensemble pre-login interstitial page processing
 login completion error.
Contact your system administrator.</P>");
response.addHeader( "runner_pre_interstitial_complete", "true")
%>
</BODY>
</HTML>
```

Creating a Custom Ensemble Login Page

The login page allows you to customize the Ensemble login form display and functionality.

The `pt:ensemble.authsourcedata` tag provides a collection of the authentication sources available for the resource. The data is stored as a collection, and each item in the collection is a data object containing information about the authentication source (prefix, name, description)

accessible through the data object dot notation (`$authsource.name`). You can use additional adaptive tags to iterate through the collection and allow the user to select the appropriate choice, as shown in the example that follows.

The example below (loginpage.jsp) displays a banner and a login form. The login form posts back to the page, which sets the appropriate headers to authenticate with the resource (`runner_username`, `runner_password`, `runner_authentication_provider`, and `runner_portal_authentication_source`). This page uses the `pt:ensemble.authsourcedata` tag, as well as several other adaptive tags to handle logic and display.

```
 <%@ page import="java.net.*"%>
 <%@page contentType="text/html;charset=UTF-8"%>
<%
String username = request.getParameter( "username" );
String password = request.getParameter( "password" );
if(username != null && password != null)
{
   response.addHeader( "runner_username", username);
   response.addHeader( "runner_password", password);
}
String authsource = request.getParameter( "authsource" );
if ( authsource != null
{
   response.addHeader( "runner_authentication_provider", "portal");

   response.addHeader( "runner_portal_authentication_source",
authsource );
}
%>
<html>
<head>
<link rel='stylesheet' type='text/css' href='css/main.css'/>
</head>
<body>
<span xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<table class="banner" cellpadding="0" cellspacing="0">
<tr>
   <td class="appLogo"></htm:td>
   <td class="liquid"></htm:td>
 </tr>
</table>
<br/>

<!-- Welcome message: -->
```

```
<font face="arial" size=2> <pt:logic.value
pt:value="$#3.ptmsgs_login"/><br> <pt:logic.value
pt:value="$#7.ptmsgs_login"/> </font>
<br/>
<%
String errorValue =
request.getHeader("runner_error_last_error_message" );
if(errorValue != null)
{
   out.println("<br>");
   out.println("<font face=\"arial\" color=\"0000AA\" size=2>");
   out.println("<pt:logic.value
pt:value=\"$#8.ptmsgs_login\"/><br>");
   out.println("</font>");
   out.println("<font face=\"arial\" color=\"AA0000\" size=2>");
   out.println("<b>");
   out.println(errorValue);
   out.println("</b>");
   out.println("</font>");
   out.println("<br>");
}
%>
<br/>
<pt:common.error>
<P><B><FONT color="red"><pt:logic.value
pt:value="$#10.ptmsgs_login"/></FONT>:</B>
<pt:common.errortext/>
 </pt:common.error>

<!-- Login form: -->
<FORM name="loginform" action="loginpage.jsp" method="POST">
<table>
<tr><!-- Username -->
    <td align="right"><font face="arial" size=2>
    <pt:logic.value pt:value="$#0.ptmsgs_login"/>
    </font></td>
    <td align="left"><font face="arial" size=2>
    <!-- inputs and password inputs are of slightly different
length in IE.  This can be fixed with CSS.-->
    <pt:core.html pt:tag="input" type="text" name="username"
alt="$#0.ptmsgs_login" size="30" value="${param["username"]}"/>
    </font> </td> </tr>
<tr><!-- Password -->
    <td align="right"><font face="arial" size=2>
    <pt:logic.value pt:value="$#1.ptmsgs_login"/>
```

```
    </font></td>
    <td align="left"><font face="arial" size=2>
    <pt:core.html pt:tag="input" onkeypress="return
submitform(event)"  type="password"
name="password" size="30" alt="$#1.ptmsgs_login"
value="${param["password"]}"/>
    </font> </td> </tr>
 <!-- Auth sources -->
<pt:ensemble.authsourcedata pt:key="authsources"/>
<pt:logic.collectionlength pt:data="authsources"
pt:key="authsourceslength"/>
<pt:logic.intexpr pt:expr="($authsourceslength)>0"
pt:key="hasvalues"/>
<pt:logic.if pt:expr="$hasvalues">
<pt:logic.iftrue>
   <pt:logic.intexpr pt:expr="($authsourceslength)>1"
pt:key="hasmultvalues"/>
   <pt:logic.if pt:expr="$hasmultvalues">
   <pt:logic.iftrue>
       <tr><!--Authentication Source:-->
          <td align="right" width="40%" colspan="1"><font
face="arial" size=2>
          <pt:logic.value pt:value="$#5.ptmsgs_login"/>
          </font></td>
          <td align="left" width="40%" colspan="1"><font
face="arial" size=2>
          <select name="authsource" onkeypress="return
submitform(event)" lang="en">
           <pt:logic.foreach pt:data="authsources" pt:var="auth">

              <pt:core.html pt:tag="option" value="$auth.prefix"
 alt="$auth.description"> <pt:logic.value
pt:value="$auth.name"/></pt:core.html>
             </pt:logic.foreach>
          </select>
          </td> </tr>
</pt:logic.iftrue>
<pt:logic.iffalse>
   <!-- Hidden input for single auth source. -->
   <pt:logic.foreach pt:data="authsources" pt:var="auth">
     <pt:core.html pt:tag="input" type="hidden" name="authsource"
 alt="" value="$auth.prefix"/>
   </pt:logic.foreach>
</pt:logic.iffalse>
</pt:logic.if>
```

```
</pt:logic.iftrue>
<pt:logic.iffalse><!-- Otherwise no auth sources for this resource.
 --></pt:logic.iffalse>
</pt:logic.if>
 <tr><!-- Login -->
   <td align="right"></td>
   <td align="left">
   <pt:core.html pt:tag="input" type="submit"
value="$#2.ptmsgs_login"/>
   </td>
</tr></table>
</FORM>
<br>

<SCRIPT language="JavaScript">
function submitform(evt)
{
   evt = (evt) ? evt : event;
   var charCode = (evt.charCode) ? evt.charCode : ((evt.which) ?
 evt.which : evt.keyCode);
   if (charCode == 13 || charCode == 3)
   {
   document.loginform.submit();
   }
return true;
}
</SCRIPT
```

The login page can use the `pt:ensemble.loginlink` tag to retrieve the external URL prefix defined for the resource. For example, if the external URL prefix of the resource is http://www.ensemble.com/app/ and the desired page after login is http://www.ensemble.com/app/pages/mainpage.html, then the full login link would be made by adding pages/mainpage.html to the login link prefix as shown in the sample code below.

```
 <pt:ensemble.loginlink pt:level="4" pt:key="loginurlprefix"/>
var loginLink = "<pt:logic.value pt:value="$loginurlprefix"/>" +
 "pages/mainpage.html";
```

Creating a Custom Ensemble Error Page

A custom error page can be displayed if there is an error in the Ensemble login process.

The `pt:common.error`, `errortext` and `errorcodes` tags allow you to insert Ensemble error information into a custom error page. Note: If these tags are included on a page, errors will

no longer be displayed in the normal error location and will not be available after the page has been displayed.

By itself, the `pt:common.errortext` tag displays only the first error message, or the custom error message defined in the pt:text attribute. Other errors, as well as exception stack traces and extended error messages, will be ignored.

Combined with the `pt:common.errorcodes` tag and `pt:logic` tags, the `pt:common.errortext` tag can be used to display all error codes in memory. (If the errors have already been displayed, no error codes will be available.)

The example below (errorpage.jsp) illustrates how to retrieve and display a collection of errors and how to replace system errors with a custom error message. This example uses `pt:logic` tags to display the error collection.

```
 <%@page contentType="text/html;charset=UTF-8"%>
<HTML> <head> <link rel='stylesheet' type='text/css'
href='css/main.css'/> </head>
<BODY>
<SPAN xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<table class="banner" cellpadding="0" cellspacing="0"> <tr>
   <td class="appLogo"></htm:td>
   <td class="liquid"></htm:td>
</tr> </table>
<br/>
<P><pt:logic.value pt:value="$#11.ptmsgs_login"/> </P>
<P>
<pt:common.errorcode pt:key="errorcodes"/>
<pt:logic.foreach pt:data="errorcodes" pt:var="code">
   <pt:common.errortext/>
   <br>
   <pt:logic.value pt:value="$#12.ptmsgs_login"/><pt:logic.value
 pt:value="$code"/>
   <br>

<!-- This is how you would override a specific error with a new
message. -->
<!--
<pt:logic.intexpr pt:expr="($code)==2010"
pt:key="isrequestedresource"/>
<pt:logic.if pt:expr="$isrequestedresource">
<pt:logic.iftrue>
   <pt:common.errortext pt:text="The requested resource is not in
 the resource map. This is a custom error message."/>
</pt:logic.iftrue>
```

```
<pt:logic.iffalse>
   <pt:common.errortext/>
</pt:logic.iffalse>
</pt:logic.if>
-->

</pt:logic.foreach>
</P>
<P><pt:logic.value pt:value="$#13.ptmsgs_login"/></P>
</SPAN> </BODY> </HTML>
```

Creating a Custom Ensemble Post-Login Page

The post-login page is an interstitial page that can be used to display messages or gather input from the user after the login form is submitted.

The example below (interstitialpage.jsp) displays a user agreement that requires approval. This page uses adaptive tags to display errors and form elements.

```
 <%@page contentType="text/html;charset=UTF-8"%>
<HTML>
<BODY>
<SPAN xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<FORM action="./processinterstitialpage.jsp" method="POST">
<P>
<TABLE>
<TR>
<TD><CENTER><B><pt:logic.value
pt:value="$#3.ptmsgs_samples"/></B></CENTER>
</TD></TR>
<TR>
   <TD>
   <TEXTAREA name="thetext" rows="15" cols="80">
   Owner: This web site belongs to Sample Company United States,
 Inc. Sample Company may change or terminate this web site or any
 parts thereof.
  Agreement: Your use of this web site constitutes your agreement
 with Sample Company to operate under the auspices of, and to act
 in
   concordance with, these Terms and Conditions of use.  By
clicking "Agree" below, you are confirming your agreement, which
 will also be confirmed
   by merely accessing this web site beyond this page.
  Continuing Agreement: Sample Company may modify or alter these
 Usage Terms at any time.  Further usage of this web site after
the terms have
```

```
   been modified confirms your agreement with the modified Usage
 Terms.
   Use of Materials: Sample Company owns all of the data on this
 web site or has secured permission from a third party to use the
 material. Usage of
   this material in any way outside this web site is strictly
prohibited.
   </TEXTAREA>

    <pt:common.error>
    <P>
    <B><FONT color="red"><pt:logic.value
pt:value="$#10.ptmsgs_login"/></FONT>:</B>
    <pt:common.errortext/>
    </P>
    </pt:common.error>

    <P>
   <pt:core.html pt:tag="input" type="checkbox" name="agreement"
 alt="$#0.ptmsgs_samples" value="agree"/>
    <pt:logic.value pt:value="$#1.ptmsgs_samples"/>
    
    <pt:core.html pt:tag="input" type="submit"
value="$#2.ptmsgs_samples"/>
    </P>
    </TD>
</TR>
</TABLE></SPAN></BODY></HTML>
```

When the user clicks the submit button, the processing page (processinterstitialpage.jsp) sets the `runner_post_interstitial_complete header` to **true**, which directs the browser to the resource. If the user did not select Accept on the post-login page, a message is displayed including a link back to the agreement. This file also provides error text for the error page in case processing fails.

```
<%@page contentType="text/html;charset=UTF-8"%>
<HTML>
<BODY>
<SPAN xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
<pt:common.error>
<P>
<B><FONT color="red"><pt:logic.value
pt:value="$#10.ptmsgs_login"/></FONT>:</B>
<pt:common.errortext/>
</P>
```

```
</pt:common.error>

<%
if (request.getParameter( "agreement" ) != null)
{
   out.println( "<P>Ensemble post-login interstitial page
processing login completion error.
   Contact your system administrator.</P>");
   response.addHeader( "runner_post_interstitial_complete",
"true");
   session.invalidate();
}
else
{
    out.println( "<P>If you do not consent to the web site usage
 agreement, you will not be able to access any content.</P>");
    out.println( "<P>Click <A
href=\"./interstitialpage.jsp\">here</a> to view the usage
agreement again, or ");
    out.println( "you can use <A
href=\"http://www.google.com\">Google</a> to find another web
site.</P>");
}
%>
</SPAN></BODY></HTML>
```

Creating a Custom Ensemble Post-Logout Page

The post-logout page is displayed when a user logs out of the resource.

In most cases, this page simply displays a message that informs users that they have been logged out of the system. The `pt:ensemble.ssologout` tag can be added to any proxied page to display a link that logs out of all resources and directs the browser to the post-logout page associated with the user's experience definition.

Configuring Custom Ensemble Login Pages

Custom login resources and pages are configured through the associated experience definition.

To define the login resource, create a resource and select **Is login resource** on the General tab. (If the application server is already registered as a resource in Ensemble, confirm that the login resource setting is enabled.)

To deploy custom pages in Ensemble, edit the associated experience definition. On the **Log In Settings** page, define the login resource and any of the following custom pages:

Pre-login page	Displayed before attempting to authenticate the user.
Login page	Displayed only when form authentication is being used; provides the form for login.
Post-login page	Displayed to the user after successful authentication and before the resource is accessed.
Error page	Displayed if there is an error in the login process.
Post-logout page	Displayed after the user logs out of the resource.

NOTE: The settings in the experience definition are used regardless of the authenticator used to access a resource. If the required authenticator uses a login page and there is no login page configured in the experience definition, the user will be presented with a blank page and will be unable to authenticate. For details, see the *AquaLogic Ensemble Administrator Guide* Chapter 8, 'Experience Definitions'.

Ensemble Login Headers

Communication between login resource pages and Ensemble is done using HTTP headers

The following table describes the available headers, including how and when they are used. Error and Post-logout pages are considered terminal pages and do not communicate with Ensemble using headers.

Property Type	Property Value	Property Description
Pre-login	runner_pre_interstitial_complete	true indicates that the pre-login page has completed successfully. Ensemble proceeds to the login page. false (or no header) means the page has not completed successfully. The pre-login page is displayed again.
Login	runner_username	The user name Ensemble will use to authenticate the user.

Property Type	Property Value	Property Description
Login	runner_password	The password Ensemble will use to authenticate the user.
Login	runner_authentication_provider	The provider for authentication. For AquaLogic Ensemble 1.0, the only valid value is portal. If the header is not present, the provider defaults to portal.
Login	runner_portal_authentication_source	The authentication source against which to authenticate the user. This is the same as the authentication source the user would use to log in to the portal.
Post-login	runner_post_interstitial_complete	true indicates that the post-login page has completed successfully. Ensemble proceeds to the resource. false (or no header) means the page has not completed successfully. The post-login page is displayed again.

About Content Service Development

Content services allow you to search external repositories through the portal and index external content in the portal Knowledge Directory. These services allow users to access documents and other resources from multiple repositories without leaving the portal workspace.

- **Content crawlers** access content from an external repository and index it in the portal. Portal users can search for and open crawled files through the portal Knowledge Directory. Content Crawlers can be used to provide access to files on protected back-end systems without violating access restrictions. Content Crawlers are implemented as remote web services. For details, see *About Content Crawlers* on page 349.
- **Federated search services** are remote web services that search external repositories, including the web, internal company databases and document repositories. For details, see *About ALI Federated Search Services* on page 385. For additional search customization options, see *Customizing ALI Search* on page 384.

About Content Crawlers

Content Crawlers are extensible components used to import documents into the portal Knowledge Directory from a back-end document repository, including Lotus Notes, Microsoft Exchange, Documentum and Novell. Portal users can search for and open crawled files on protected back-end systems through the portal without violating access restrictions.

The IDK allows you to create remote content crawlers and related configuration pages without parsing SOAP or accessing the portal API; you simply implement four object interfaces to access the back-end repository and retrieve files. UDDI servers are not required.

The purposes of a Content Crawler are two-fold:

1. Iterate over and catalog a hierarchical data repository. Retrieve metadata and index documents in the data repository and include them in the portal Knowledge Directory and search index. Files are indexed based on metadata and full-text content.
2. Retrieve individual documents on demand through the portal Knowledge Directory, enforcing any user-level access restrictions.

Content Crawlers are run asynchronously by the ALI Automation Service. The associated Content Crawler completes step 1. The Content Crawler Job can be run on a regular schedule to refresh any updated or added files. The portal creates a Document object for each crawled file and indexes it in the Knowledge Directory. Each object includes basic file information, security information, and a URL that opens the file from the back-end content repository. (No crawled files are stored on the portal server.) If the content is not contained within a file or cannot be indexed for another reason, you must implement a servlet/aspx page to return files that can be indexed to the portal.

Step 2 occurs when a user browses the Knowledge Directory and opens to a previously crawled document. After a file is crawled into the portal, users must be able to access the file from within the portal by clicking a link. This step is called click-through. If files are publicly accessible, click-through is simple. In many cases, you must provide access to documents that are behind a firewall or are otherwise inaccessible from the portal interface.

For details, see the following topics:

- *IDK Interfaces for Content Crawler Development* on page 351: The IDK provides object interfaces to implement custom content crawlers. This page introduces the IDK's crawler interfaces and lists useful warnings and best practices.
- *Content Crawler Development Tips* on page 358: These best practices and development tips apply to all content crawler development.
- *About Content Crawler Indexing* on page 361: Content crawlers must return an indexable version of each crawled file to be included in the portal Knowledge Directory. This page provides an introduction to indexing.
- *About Content Crawler Click-Through* on page 368: The crawl is just the first step. This page explains how content crawlers can provide access to secured files that have been indexed in the portal. For instructions, see *Implementing Content Crawler Click-Through* on page 369.
- *Deploying a Custom Content Crawler (Java)* on page 372 and *Deploying a Custom Content Crawler (.NET)* on page 375: After coding your Content Crawler, you must deploy your code. These pages provide detailed instructions.

- *Configuring Content Crawlers* on page 379: Implementing a successful Content Crawler in the portal requires specific configuration.
- *Debugging Custom Content Crawlers* on page 378: Logging is a key component of any successful crawl. This page introduces logging options.
- *Testing Custom Content Crawlers* on page 378: This checklist summarizes key tests that should be performed on every content crawler.

IDK Interfaces for Content Crawler Development

The IDK plumtree.remote.crawler package/namespace includes four interfaces to support content crawler development: `IContainerProvider`, `IContainer`, `IDocumentProvider` and `IDocument`.

When the ALI Automation Service initiates a crawl, it issues a SOAP request to return a list of folders. It iterates over the list of folders and retrieves lists of documents with metadata. In general, the portal calls IDK interfaces in the following order. See the definitions that follow for more information.

1. `IContainerProvider.Initialize` once per thread. Use `DataSourceInfo` and `CrawlerInfo` to initialize the Container Provider (make a connection to the back-end system and create a new session). Note: This is not a true HTTP session, and sessions can get dropped. Keep a variable that can be used to ensure the session is still initialized; if it is not, throw `NotInitializedException`. Store the Content Source in a member variable in Initialize. Do not use direct access to the member variable; instead use a method that checks if it is null and throws a `NotInitializedException`.
2. `IContainerProvider.AttachToContainer`, using the starting location in the key CrawlerConstants.TAG_PATH. The key should be populated using a Service Configuration page in the Content Crawler editor. The string in TAG_PATH is service-specific; a file Content Crawler could use the UNC path to a folder, while a database Content Crawler could use the full name of a table. The following methods are not called in any specific order.
   - `IContainer.GetUsers` and `IContainer.GetGroups` on that container as required. (IContainer.GetMetaData is deprecated.)
   - `IContainer.GetChildContainers` up to the number specified in CrawlerConstants.TAG_DEPTH. (This key must be set via a Service Configuration page.)
   - `IContainerProvider.AttachToContainer` for each ChildContainer returned.
   - `IContainer.GetChildDocuments`, then `IDocumentProvider.AttachToDocument` for each ChildDocument returned.

3. `IContainerProvider.Shutdown` (this call is optional and could be blocked by exceptions or network failure).
4. `IDocumentProvider.Initialize` once per thread. Note: Sessions can get dropped. Keep a variable that can be used to ensure the session is still initialized; if it is not, throw NotInitializedException.
5. `IDocumentProvider.AttachToDocument` for each ChildDocument, then `IDocument.GetDocumentSignature` to see if the document has changed. If the document is new or has been modified, the following methods are called (not in any specific order).
   - `IDocument.GetUsers` and `IDocument.GetGroups` on that document as required.
   - `IDocument.GetMetaData` to get the file name, description, content type, URL, etc.
   - `IDocument.GetDocument` to index the document (only if DocFetch is used).
6. `IDocumentProvider.Shutdown` (this call is optional and could be blocked by exceptions or network failure).

The sections below provide helpful information on the interfaces used to implement a Content Crawler. For a complete listing of interfaces, classes, and methods, see the IDK API documentation.

IContainerProvider

The `IContainerProvider` interface allows the portal to iterate over a back-end directory structure. The portal calls `IContainerProvider` first in most cases. This interface provides the following methods:

- `Initialize` allows the remote server to initialize a session and create a connection to the back-end document repository. The IDK passes in a `DataSourceInfo` object that contains the necessary settings associated with a Content Source object (the name of a directory in the repository and the credentials of a system user). The `CrawlInfo` object contains the settings for the associated Content Crawler object in the portal. The start location of the crawl is the value stored in the key **CrawlerConstants.TAG_PATH** , set using a Service Configuration page.
- `AttachToContainer` is always the next call after Initialize; the order of the remaining calls is not defined. It associates the session with the container specified in the `sContainerLocation` parameter; subsequent calls refer to this container until the next `AttachToContainer` call. The value in the `sContainerLocation` parameter will be the CrawlerConstants.TAG_PATH key for the initial attach, and the value specified in ChildContainer.GetLocation for subsequent attaches. Each time `AttachToContainer` is called, discard any state created during the previous `AttachToContainer` call. If multiple

translations of the container are available, select the most appropriate using the Locale parameter, which can be sent as a full locale (e.g., "en-us") or in the abbreviated language-only format (e.g., "en"). Note: If the container specified does not exist, you must throw a new `NoLongerExistsException` to avoid an infinite loop. If the Content Crawler is configured to delete missing files, all files in the container will be removed from the portal index.

- `Shutdown` allows the portal to clean up any unused sessions that have not yet expired. Content Crawlers are implemented on top of standard cookie-based session mechanisms, so sessions expire and resources and connections are released after an inactivity period, typically around 20 minutes. As a performance optimization, the portal might send a `Shutdown` message notifying the remote server to end the session immediately. No parameters are received and none are returned. Do not assume that `Shutdown` will be called; the call could be blocked by an exception or network failure. Remote servers must terminate sessions after an inactivity timeout but can choose to ignore the `Shutdown` message and keep the session alive until it times out.

IContainer

The portal uses the `IContainer` interface to query information about back-end resource directories. This interface provides the following methods:

- `GetGroups` and `GetUsers` return a list of the portal groups or users that have read access to the container. These calls are made only if the Web Service and Content Crawler objects are configured to import security. The portal batches these calls; the Content Crawler code should return all groups or users at once.
- `GetChildContainers` returns the containers inside the current container (i.e., subfolders of a folder). The value stored in the key **CrawlerContants.TAG_DEPTH** is used to determine how many times `GetChildContainers` is called (crawl depth). This value must be set via a Service Configuration page. If no value is stored with this key, `GetChildContainers` is never called; only the documents in the folder specified for the start location are crawled into the portal. Note: Setting CrawlerConstants.TAG_DEPTH to -1 could result in an infinite loop.
- `GetChildDocuments` returns the documents inside the current container (folder). The portal batches this call; the Content Crawler code should return all documents at once. The **TypeNamespace** and **TypeID** parameters define the Content Type for the document. `TypeNamespace` associates the document with a row in the Global Content Type Map, and the TypeID associates it with a particular Content Type. The value in `ChildDocument.GetLocation` is used in `IDocumentProvider.AttachToDocument`, so any information required by

`AttachToDocument` must be included in the location string. You can describe the document using file or MIME, as shown in the example below.

```
ChildDocument doc=new ChildDocument();
String filename = WordDoc.doc;

//Location is a crawler-specific string to retrieve doc, e.g.,
 file name
doc.setLocation(filename);

//TypeNameSpace is either FILE or MIME unless using a custom
namespace (Notes, Exchange)
//NOTE: example uses getCode because setTypeNameSpace expects
a String
doc.setTypeNameSpace(TypeNamespace.MIME.getCode()):

//For file descriptions, TypeID is simply the document name
with extension (i.e., filename)
//For MIME descriptions, set the document type or map multiple
 file extensions to MIME types
doc.setTypeID("application/msword");

//DisplayName is the name to display in the KD, usually
overridden in IDocument.getMetaData();
doc.setDisplayName(filename);
```

- `GetMetaData` (DEPRECATED) returns all metadata available in the repository about the container. The name and location are used in mirrored crawls to mirror the structure of the source repository. In most cases, the container metadata is only the name and description.

IDocumentProvider

The `IDocumentProvider` interface allows the portal to specify back-end documents for retrieval. In most cases, the portal calls `IContainerProvider` first. However, in some cases, the service is used to refresh existing documents and `IDocumentProvider` might be called first.

- `Initialize` allows the remote server to initialize a session and create a connection to the back-end document repository. (For details on parameters and session state, see `IContainerProvider.Initialize` above.) `IDocumentProvider.Initialize` will be called once per thread as long as the session does not time out or get interrupted for other reasons, and `AttachToDocument` will be called next.

- `AttachToDocument` is always the next call made after Initialize; the order of the remaining calls is not defined. This method 'attaches' a session to the document specified in the **sDocumentLocation** parameter; subsequent calls refer to this document until the next AttachToDocument call. The sDocumentLocation string is the value specified in ChildDocument.GetLocation (ChildDocument is returned by IContainer.GetChildDocuments). If multiple translations of the document are available, select the most appropriate by using the Locale parameter, which can be sent as a full locale (e.g., 'en-us') or in the abbreviated language only format (e.g., 'en'). When implementing this method, you can throw the following exceptions:

Exception	Description
NoLongerExistsException	The document has been moved or deleted. (The refresh agent will delete documents from the portal index only if this exception has been thrown.)
NotAvailableException	The document is temporarily unavailable.
NotInitializedException	The IDocumentProvider is in an uninitialized state.
AccessDeniedException	Access to this document is denied.
ServiceException	Propagates the exception to the portal and adds an entry to ALI Logging Spy.

- `Shutdown` allows the portal to clean up any unused sessions that have not yet expired. (For details, see IContainerProvider.Shutdown above.)

IDocument

The `IDocument` interface allows the portal to query information about and retrieve documents. This interface provides the following methods:

- `GetDocumentSignature` allows the portal to determine if the document has changed and should be re-indexed and flagged as updated. It can be a version number, a last-modified date, or the CRC of the document. The IDK does not enforce any restrictions on what to use for the document signature, or provide any utilities to get the CRC of the document. This is

always the first call made to IDocument; on re-crawls, if the documentSignature has not changed, no additional calls will be made.

- `GetMetadata` returns all metadata available in the repository about the document. The portal maps this data to properties based on the mappings defined for the appropriate Content Type, along with metadata returned by the associated accessor. The following field names are reserved. Additional properties can be added using the portal's Global Document Property Map; for details, see Configuring Custom Content Crawlers: Properties and Metadata. (Any properties that are not in the Global Document Property Map will be discarded.)

Field Name	Description
Name	REQUIRED. The name of the link to be displayed in the portal Knowledge Directory. Note: By default, the portal uses the name from the crawled file properties as the name of the card. To set the portal to use the Name property returned by GetMetadata, you must set the CrawlerConstants.TAG_PROPERTIES to REMOTE using the Service Configuration Interface.
Description	The description of the link to be displayed in the portal Knowledge Directory.
UseDocFetch	Whether or not to use DocFetch to retrieve the file. The default is **False**. If you use DocFetch, the value in the File Name field is used to retrieve the file during both indexing and click-through. If you do not use DocFetch, you must provide values for Indexing URL and Click-Through URL.
File Name (required for DocFetch)	The name of the click-through file, used for DocFetch.
Content Type (required for DocFetch)	The content type of the click-through file, used to associated the crawled document with the Global Content Type Map.
Indexing URL (public URL)	(Required if not using DocFetch.) The URL to the file that can be indexed in the portal. URLs can be relative to the Remote Server. If a file is publicly accessible via a URL, that URL can be used to access the document for both indexing and click-through. Documents that cannot be indexed must provide an additional URL at crawl-time for indexing purposes. For details on crawling secured content, see Accessing Secured Content .

Field Name	Description
Click-Through URL (public URL)	(Required if not using DocFetch.) The URL to the click-through file. URLs can be relative to the Remote Server. For details on crawling secured content, see Accessing Secured Content.
Image UUID (optional)	This parameter is only required for custom Content Types. For standard Content Types, the accessor will assign the correct image UUID.

- `GetDocument` returns the path to the file if it was not provided by `GetMetaData`. (For public URLs, you do not need to implement `GetDocument`, but you *must* provide values for `IndexingURL` and `ClickThroughURL` in `GetMetaData`.) During crawl-time indexing, this file is copied to the web-accessible `IndexFilePath` location specified in your deployment descriptor and returned to the portal via a URL to that location. If the file is not supported for indexing by the portal, implement `GetDocument` to convert the document into a supported file format for indexing (e.g., text-only) and return that file during indexing. Note: To create a custom implementation of `GetDocument`, you must set `UseDocFetch` to True. When a user clicks through to the document, the display file is streamed back via the DocFetch servlet to the browser. Any necessary cleanup due to temporary file usage should be done on subsequent calls to `IDocumentProvider.AttachToDocument` or `IDocumentProvider.Shutdown`. For details on accessing secured content and files that are not accessible via a public URL, see *About Content Crawler Click-Through* on page 368.
- `GetGroups` and `GetUsers` return a list of the groups or users with read access to the document. Each entry is an ACLEntry with a domain and group name. The portal batches these calls; the Content Crawler code should return all groups or users at once. This call is made only if the *Supports importing security with each document* option is checked on the Advanced Settings page of the Web Service editor.

SCI Variables for Content Crawler Properties

Content crawler properties are configured using a defined set of variables.

The Content Crawler object should include the following properties. These properties can be hard-coded or configured using a Service Configuration (SCI) page. For details on SCI pages, see *Creating Service Configuration Pages for Content Crawlers* on page 381.

Variable	Property Value
TAG_PATH	The path to the container to crawl. Depending on the type of container, this could be a URL, a UNC path, information for a table in a database, information for a view in Notes, etc.
CRAWL_DEPTH	If the variable TAG_DEPTH has not been included, the content crawler only crawls documents in the first directory. This works for resources with no subdirectories, such as a database. For a file system, it is usually best to use a SCISelectElement to let users select the crawl depth (where -1 means until subcontainers return no child containers). If you do not want users to set this option, use a SCIHiddenElement for the same field. Note: The SCISelectElement must call `SetStorageType(TypeStorage.STORAGE_INTEGER)` to be stored correctly; otherwise the portal will return the message "wrong property type."
TAG_PROPERTIES	(optional) Represents whether properties from GetMetaData or the local accessor should be used. Setting this variable to TAG_PROPERTIES_LOCAL causes the local accessor properties used to retrieve a file to override the properties returned by the content crawler. Setting the variable to TAG_PROPERTIES_REMOTE causes the properties from GetMetaData to override properties from local accessors.

Content Crawler Development Tips

These best practices and development tips apply to all content crawler development.

Use logging extensively to provide feedback during a crawl. In some cases, the portal reports a successful crawl when there were minor errors. Use Log4J or Log4Net to track progress. For more information, see Logging and Troubleshooting.

Use relative URLs in your code to allow migration to another remote server. Note: These URLs might be relative to different base URL endpoints. The click-through URL is relative to the remote server base URL, and the indexing URL is relative to the SOAP URL. Depending on whether you have implemented your Content Crawler using Java or .NET, the base URL endpoint for the remote server might differ from the base URL endpoint for SOAP. For example, the Java IDK uses Axis, which implements programs as services. In Axis, the SOAP URL is the remote server base URL with '/services' attached to the end. Given the remote server base URL http://server:port/sitename, the

SOAP URL would be http://server:port/sitename/services. If both click-through and indexing URLs point to the same servlet (http://server:port/sitename/customdocfetch?docId=12345), the relative URLs would be different. The relative URL for indexing would be "../customdocfetch?docId=12345" and the relative URL for click-through would be "customdocfetch?docId=12345". (Since the indexing URL is relative to the SOAP URL, the '../' reorients the path from http://server:port/sitename/services to http://server:port/sitename, yielding the correct URL to http://server:port/sitename/customdocfetch?docId=12345.)

Do your initial implementation of IDocumentProvider and IDocFetchProvider in separate classes, but factor out some code to allow reuse of the GetDocument and GetMetaData methods. See the Viewer sample application included with the IDK for sample code.

Do not make your calls order-dependent. The portal can make the above calls in any order, so your code cannot be dependent on order.

If a document or container does not exist, always throw a new NoLongerExistsException. This is the only way the portal can determine if the file or folder has been deleted. Not throwing the exception could result in an infinite loop.

If there are no results, return a zero-length array. If your intention is to return no results, use a zero-length array, not an array with empty strings. (For example, `return new ChildContainer[0]; )`

Check the SOAP timeout for the back-end server and calibrate your response accordingly. In version 5.0 and above, the SOAP timeout is set in the Web Service editor. In version 4.5, the SOAP timeout must be set via a Service Configuration page.

Pages that are not publicly accessible must be gatewayed. Gateway settings are configured in the Web Service editor on the HTTP Configuration page, and in the Content Source editor. You can gateway all URLs relative to the remote server or enter individual URLs and add paths to other servers to gateway additional pages. For details, see Deploying Custom Content Crawlers.

You must define mappings for any associated Content Types before a Content Crawler is run. The portal uses the mappings in the Content Type definition to map the data returned by the Content Crawler to portal properties. Properties are only stored if you configure the Content Type mapping before running the Content Crawler. (Properties that apply to all documents are configured in the Global Document Property Map.)

To import security settings, the backend repository must have an associated Authentication Source. Content Crawlers that import security need the user and category

(domain) defined by an Authentication Source. You must configure the Authentication Source before the Content Crawler is run. Many repositories use the networks NT or LDAP security store; if an associated Authentication Source already exists, there is no need to create one. For details on Authentication Sources, see the portal online help. For details on security, see Configuring Custom Content Crawlers : Importing File Security.

If you use a mirrored crawl, only run it when you first import documents. Always check every directory after a mirrored crawl. After you have imported documents into the portal, it is safer to refresh your portal directory using a regular crawl with filters.

For mirrored crawls, make crawl depth as shallow as possible. Portal users want to access documents quickly, so folder structure is important. Also, the deeper the crawl, the more extensive your QA process will be.

Use filters to sort crawled documents into portal folders. Mirrored crawls can return inappropriate content and create unnecessary directory structures. Filters are a more efficient way to sort crawled documents. To use filters, choose Apply Filter of Destination Folder in the Content Crawler editor. For details on filters, see the portal online help.

Do not use automatic approval unless you have tested a Content Crawler. It is dangerous to use automatic approval without first testing the structure, metadata and logs for a Content Crawler.

To clear the deletion history, you must re-open the Content Crawler editor. To re-crawl documents that have been deleted from the portal, you must re-open the Content Crawler editor and configure the Importing Documents settings on the Advanced Settings page as explained in Deploying Custom Content Crawlers.

You can also import access restrictions during a crawl; for details, see *Configuring Content Crawlers* on page 379.

About Content Crawler Security Options

A crawler can use a range of credential types to access a secure file.

If you need to apply credentials to access a file, you can use any of the following options:

SSO	SSO must be configured in the portal and on the remote server, using the instructions of your SSO vendor.
Basic Authentication	Set the remote server to pass the user's basic authentication headers to the remote resource. Both sources must be using the same directory. For example, if a user

	logs in using an IPlanet directory, it is unlikely they will be able to access an Exchange resource.
Content Source credentials	Content Source credentials are generally valid only for crawling a database. Most other use cases require user-specific credentials.
User preferences via form-based authentication	Preferences stored in the ALI database can be used to create a cookie if the resource accepts session-based authentication. User preferences generally cannot be used if the resource expects basic authentication. For example, the Content Service for Notes uses this approach when Notes is using session-based (cookie) authentication. You must enter all User settings and User Information required by a content crawler on the Preferences page of the Content Crawler editor.
Force users to log in	If the required credentials are not available, redirect the user to the appropriate page and/or provide an intelligible error message. For example, the Content Service for Notes uses this approach when Notes is using basic authentication.

About Content Crawler Indexing

A content crawler must return an indexable version of each crawled file to be included in the portal Knowledge Directory.

The crawler's servlet/aspx page must return content in a indexable format and set the content type and file name using the appropriate headers. Any information required to retrieve the document must be included in the query string of the index URL, including credentials (if necessary).

Note: The request from the portal to the indexing servlet is a simple HTTP GET. This call is not gatewayed, so the content crawler code does not have access to the Content Source settings, user credentials and preferences, or anything other information through the IDK.

For files, content can be streamed directly from the source directory. If the content is not in a file, the crawler code should create a temporary file that includes the content with as little extraneous information as possible.

For details, see the following topics:

- *Indexing Streaming Content* on page 361
- *Creating Temporary Files for Indexing* on page 366

Indexing Streaming Content

If the content being crawled is in a file, the file can be streamed directly from the source directory.

The following steps describe a typical custom mechanism to return files in a indexable format and set the content type and file name using the appropriate headers.

1. In `IDocument`, get all the variables needed to access the document and add them to the query string of the indexing servlet. This could be as simple as a UNC path for a file crawler or as complicated as server name, database name, schema, table, primary key(s) and primary key value(s) for a database record. It depends entirely on the content crawler and the document being crawled. Make sure all values are URLEncoded.
2. Add the content type to the query string.
3. In `IDocument`, add URLEncoded credentials to the query string. Keep in mind that URLEncoding the credentials will turn a '+' to a space, which must be turned back into a space in the indexing servlet.
4. Pass back URLs via the IDK's `DocumentMetadata` class that point to the servlet(s).
   - **UseDocFetch**: Set UseDocFetch to **False**.
   - **IndexingURL**: Set the IndexingURL to the endpoint/servlet that provides the indexable version of the file, including the query string arguments defined in steps 1-3 above.
   - **ClickThroughURL**: Set the ClickThroughURL to the endpoint/servlet that provides the path to be used when a user clicks through to view the file. During the crawl, the ClickThroughURL value is stored in the associated Knowledge Directory document.
5. In the indexing servlet, get the location string and content type from the query string and parse the location string to get the path to the resource.
6. Obtain the resource.
7. Set the ContentType header and the Content-Disposition header.
8. Stream the file (binary or text) or write out the file (text) in a try-catch block.

Creating Temporary Files for Indexing

If crawled content cannot be indexed as-is, the crawler code must create a temporary file for indexing.

The following steps describe a typical custom mechanism to create a temporary indexable file with as little extraneous information as possible and set the content type and file name using the appropriate headers. In most cases, the resource has already been accessed in AttachToDocument, so there is no need to call the back-end system again. This example does not use credentials. If you do not want to create temporary files, you can implement an indexing servlet that returns indexable content.

1. In `IDocument`, write a temporary file to a publicly accessible location (usually the root directory of the Web application as shown in the code snippet below).

```
MessageContext context = MessageContext.getCurrentContext();
   HttpServletRequest req =
(HttpServletRequest)context.getProperty(HTTPConstants.MC_HTTP_SERVLETREQUEST)

   StringBuffer buff = new StringBuffer();

buff.append(req.getScheme()).append('://').append(req.getServerName())


.append(':').append(req.getServerPort()).append(req.getContextPath());

      String indexRoot = buff.toString();
```

2. Pass back URLs via the IDK's `DocumentMetadata` class that point to the servlet(s).
   - **UseDocFetch**: Set UseDocFetch to **False**.
   - **IndexingURL**: Set the IndexingURL to the endpoint/servlet that provides the indexable version of the file, including the query string arguments defined in steps 1-3 above.
   - **ClickThroughURL**: Set the ClickThroughURL to the endpoint/servlet that provides the path to be used when a user clicks through to view the file. During the crawl, the ClickThroughURL value is stored in the associated Knowledge Directory document.
3. Add the temporary file path to the query string, along with the content type. Make sure to URLEncode both.
4. In the indexing servlet, get the file path and content type from the query string. Get the file name from the file path.
5. Set the ContentType header and the Content-Disposition header.
6. Stream the file (binary or text) or write out the file (text) in a try-catch block.
7. In the finally block, delete the file.

The following sample code indexes a text file.

```
logger.Debug('Entering Index.Page_Load()');

// try to get the .tmp filename from the Content Crawler
string indexFileName = Request[Constants.INDEX_FILE];
if (indexFileName != null)
{
    StreamReader sr = null;
```

```
    string filePath = ''; try
    {
        filePath = HttpUtility.UrlDecode(indexFileName);
        string shortFileName =
filePath.Substring(filePath.LastIndexOf('\\') + 1);

        // set the proper response headers
        Response.ContentType = 'text/plain';
        Response.AddHeader('Content-Disposition', 'inline;
filename=' + shortFileName);

        // open the file
        sr = new StreamReader(filePath);

        // stream out the information into the response
        string line = sr.ReadLine();

         while (line != null)
         {
               Response.Output.WriteLine(line);
               line = sr.ReadLine();
         }
    }
    catch (Exception ex)
    {
    logger.Error('Exception while trying to write index file: '
+ ex.Message, ex);
    }
    finally
    {
    // close and delete the temporary index file even if there is
 an error
    if(sr != null){sr.Close();}
    if(!filePath.Equals('')){File.Delete(filePath);}
    }
//done
return;
}
...
```

About Content Crawler DocFetch

The IDK's DocFetch mechanism is one way for a content crawler to retrieve files that are not accessible via a public URL.

If a content crawler implements DocFetch, the IDK manages the process of creating temporary files for indexing and click-through. DocFetch also allows you to implement user-level access control. You can pass user preferences or User Information to the content crawler, and this information can be used by DocFetch to authenticate with the back-end system or limit access to specific users.

Note: DocFetch does not allow you to use multiple methods of authentication or implement custom error handling. If you cannot use public URLs and are not using DocFetch, you must implement a custom document fetching mechanism (i.e., servlet or aspx page). If necessary, you can implement separate servlets for indexing and click-through.

For detailed instructions, see *Implementing Content Crawler DocFetch* on page 365.

Implementing Content Crawler DocFetch

Content crawler code can use DocFetch to access files that are not available via a public URL.

To use DocFetch, there are three relevant fields in the DocumentMetaData object returned in the portal's call to `IDocument.GetMetaData`:

- **UseDocFetch**: Set UseDocFetch to **True**.
- **File Name**: Set the File Name to the name of the file in the repository (must be unique).
- **Content Type**: Set the Content Type to the content type for the file. The content type must be mapped to a supported Content Type in the portal.

When UseDocFetch is set to True, the IDK sets the ClickThroughURL stored in the Knowledge Directory to the URL of the DocFetch servlet, and calls IDocument.GetDocument to retrieve the file path to the indexable version of the document. When a user subsequently clicks on a link to the crawled document in the Knowledge Directory, the request to the DocFetch servlet makes several calls to the already-implemented content crawler code. GetDocument is called again, but this time as part of the IDocFetch interface. The file path returned is opened by the servlet and streamed back in the response.

As explained above, the content crawler must implement the GetDocument method in both the Crawler.IDocument and DocFetch.IDocFetch interfaces to return the appropriate file path(s). If the repository cannot access files directly, you must serialize the binary representation to a temporary disk file and return that path. The IDocument and IDocFetch interfaces can use the same process. The IDK provides a cleanup call to delete any temporary files later.

Note: If `GetDocument` returns a path to a file (not a URL to a publicly accessible file), the file name must be unique. Otherwise, all copies of the file are removed during cleanup, including copies that are currently in use by other users.

To use user preferences or User Information, you must configure the settings to be used in the Content Crawler editor.

DocFetch interfaces are called in the following order. For a complete listing of interfaces, classes, and methods, see the IDK API documentation

1. **IDocFetchProvider.Initialize** using the DataSourceInfo, UserPrefs and UserInfo returned from the portal to make a connection to the backend system and create a new session. The implementation should initialize in a similar manner to IDocumentProvider.Initialize. IDocFetchProvider can use UserInfo and UserPrefs to perform additional authentication. The ICrawlerLog object is not available. Note: Sessions can get dropped. Keep a variable that can be used to ensure the session is still initialized; if it is not, throw NotInitializedException.
2. **IDocFetchProvider.AttachToDocument** using the authentication information provided (including UserPrefs and UserInfo).
   a) **IDocFetch.GetMetaData**: The only DocumentMetadata required for click-through is the file name and content type.
   b) **IDocFetch.GetDocument:** As noted above, IDocFetch.GetDocument method should reuse as much code as possible from the IDocument.GetDocument method. The IDK looks in web.config/*.wsdd to get the file path and URL to the directory for creating temporary files.
3. **IDocFetchProvider.Shutdown** (optional).

Creating Temporary Files for Indexing

If crawled content cannot be indexed as-is, the crawler code must create a temporary file for indexing.

The following steps describe a typical custom mechanism to create a temporary indexable file with as little extraneous information as possible and set the content type and file name using the appropriate headers. In most cases, the resource has already been accessed in AttachToDocument, so there is no need to call the back-end system again. This example does not use credentials. If you do not want to create temporary files, you can implement an indexing servlet that returns indexable content.

1. In `IDocument`, write a temporary file to a publicly accessible location (usually the root directory of the Web application as shown in the code snippet below).

```
MessageContext context = MessageContext.getCurrentContext();
   HttpServletRequest req =
(HttpServletRequest)context.getProperty(HTTPConstants.MC_HTTP_SERVLETREQUEST)

   StringBuffer buff = new StringBuffer();
```

```
buff.append(req.getScheme()).append('://').append(req.getServerName())

.append(':').append(req.getServerPort()).append(req.getContextPath());

        String indexRoot = buff.toString();
```

2. Pass back URLs via the IDK's `DocumentMetadata` class that point to the servlet(s).
   - **UseDocFetch**: Set UseDocFetch to **False**.
   - **IndexingURL**: Set the IndexingURL to the endpoint/servlet that provides the indexable version of the file, including the query string arguments defined in steps 1-3 above.
   - **ClickThroughURL**: Set the ClickThroughURL to the endpoint/servlet that provides the path to be used when a user clicks through to view the file. During the crawl, the ClickThroughURL value is stored in the associated Knowledge Directory document.
3. Add the temporary file path to the query string, along with the content type. Make sure to URLEncode both.
4. In the indexing servlet, get the file path and content type from the query string. Get the file name from the file path.
5. Set the ContentType header and the Content-Disposition header.
6. Stream the file (binary or text) or write out the file (text) in a try-catch block.
7. In the finally block, delete the file.

The following sample code indexes a text file.

```
logger.Debug('Entering Index.Page_Load()');

// try to get the .tmp filename from the Content Crawler
string indexFileName = Request[Constants.INDEX_FILE];
if (indexFileName != null)
{
    StreamReader sr = null;
    string filePath = ''; try
    {
        filePath = HttpUtility.UrlDecode(indexFileName);
        string shortFileName =
filePath.Substring(filePath.LastIndexOf('\\') + 1);

        // set the proper response headers
        Response.ContentType = 'text/plain';
```

```
        Response.AddHeader('Content-Disposition', 'inline;
filename=' + shortFileName);

        // open the file
        sr = new StreamReader(filePath);

        // stream out the information into the response
        string line = sr.ReadLine();

         while (line != null)
         {
               Response.Output.WriteLine(line);
               line = sr.ReadLine();
         }
    }
    catch (Exception ex)
    {
    logger.Error('Exception while trying to write index file: '
+ ex.Message, ex);
    }
    finally
    {
    // close and delete the temporary index file even if there is
 an error
    if(sr != null){sr.Close();}
    if(!filePath.Equals('')){File.Delete(filePath);}
    }
//done
return;
}
...
```

About Content Crawler Click-Through

After a repository is crawled and files are indexed in the portal, users must be able to access the file from within the portal by clicking a link; this is the 'click-through' step.

Click-through retrieves a crawled file over HTTP to be displayed to the user. To retrieve documents that are not available via a public URL, you can write your own code or use the DocFetch mechanism in the IDK. If you handle document retrieval, you can also implement custom caching or error handling. Click-through links are gatewayed, so the content crawler can leverage user credentials and other preferences.

For details, see the following topics:

- *Implementing Content Crawler Click-Through* on page 369
- *About Content Crawler DocFetch* on page 364
- *Implementing Content Crawler DocFetch* on page 365
- *About Content Crawler Security Options* on page 360

Implementing Content Crawler Click-Through

The content crawler's click-through implementation must return content in a readable format and set the content type and file name using the appropriate headers.

The following example uses a file, but the crawled resource could be any type of content. If the content is not in a file, the click-through servlet should create a representation with as little extraneous information as possible in a temporary file (for example, for a database, you would retrieve the record and transform it to HTML). See *Creating Temporary Files for Indexing* on page 366. You can also use the IDK's DocFetch mechanism to handle indexing and click-through; see *Implementing Content Crawler DocFetch* on page 365.

1. Create the clickThroughServlet, and add a mapping in web.xml.
2. Complete the implementation of IDocument.GetMetaData. Set the ClickThoughURL value to an URL constructed using the following steps:
   a) Construct the base URL of the application using the same approach as in the index servlet.
   b) Add the servlet mapping to the clickThroughServlet.
   c) Add any query string parameters required to access the document from the clickThroughServlet (or aspx page). Remember: The click-through page will have access to Content Source parameters (as administrative preferences), but no access to Content Crawler settings.
3. To authenticate to the back-end resource, you can use Basic Auth, User Preferences, User Info, or credentials from the Content Source. Below are suggestions for each; security will need to be tailored to your content crawler

   - Use **Basic Auth** to use the same credentials used to log in to the portal. For example, if the portal uses AD credentials, Basic Auth could be used to access NT files.
   - Use (encrypted) **User Preferences** if the authentication source is different from the one used to log in to the portal. For example, if the portal log in uses IPlanet, but you need to access an NT or Documentum file.
   - Use (encrypted) **User Info** if the encrypted credentials are stored in another profile source and imported using a profile job.

- Use **Content Source credentials** when there a limited connections, for example with a database.

4. Extract the parameters from the query string as required.
5. Display the page.
   - If there is already an HTML representation of the page, authenticate to the page. If the site is using Basic Auth and you are using Basic Auth headers, simply redirect to that page. If the site is using Basic Auth and you are not using Basic Auth, users must log in unless the site and the portal are using the same SSO solution. If the site is using form-based authentication, post to the site and follow the redirect.
   - If there is not an HTML representation of the page, retrieve the resource and stream it out to the client as shown in the sample code below (Java). If you use a temporary file, put the code in a try-catch-finally block, and delete the file in the finally block.

```
//get the content type, passed as a query string parameter
String contentType = request.getParameter('contentType')

//if this is a file, get the file name
String filename = request.getParameter('filename');

//set the content type on the response
response.setContentType(contentType);

//set the content disposition header to tell the browser the
file name
response.setHeader('Content-Disposition', 'inline; filename='
+ filename);

//set the header that tells the gateway to stream this through
 the gateway
response.setHeader('PTGW-Streaming', 'Yes');

//get the content - for a file, get a file input stream based
on the path (shown below)
//other repositories may simply provide an input stream
//NOTE: this code contains no error checking
String filePath = request.getParameter('filePath');
File file = new File(filePath);
FileInputStream fileStream = new FileInputStream(file);

//create a byte buffer for reading the file in 40k chunks
int BUFFER_SIZE = 40 * 1024;
```

```
byte[] buf = new byte[BUFFER_SIZE];

//start reading the file
int bytesRead = fileStream.read(buf);
ServletOutputStream out = response.getOutputStream();

//start writing out the body
out.write(buf, 0, bytesRead);

//continue writing until the input stream returns -1
while ((bytesRead = fileStream.read(buf)) != -1
{
    out.write(buf, 0, bytesRead);
}
```

Handling Exceptions in Custom Content Crawlers

Content crawler code should handle exceptions.

Most calls should be put into a try-catch block. The scope of the try-catch block should be small enough to diagnose errors easily.

In the catch block, log the error in both Log4j/Log4net as well as `ICrawlerLog` and then re-throw the exception as a ServiceException. This will result in the error displaying in the job log. However, only the error message shows up in the log; look at the log from Log4j/Log4net to get the full stack trace. The following exceptions have special meaning:

- **NotInitializedException** means to re-initialize.
- **NoLongerExistsException** means that the folder or document no longer exists, and tells the portal to delete that resource.

If any exception is thrown during the initial `AttachToContainer`, the crawl aborts. If `NotInitializedException` is thrown, the content crawler re-initializes. If `NoLongerExistsException` is thrown, the resource is removed from the Knowledge Directory, and the content crawler continues to the next resource. If other exceptions are thrown, the error is logged, and the content crawler continues to the next resource.

To use `ICrawlerLog`, store the member variable in your implementation of `IContainerProvider.Initialize`. To send a log message, simply add the following line: `m_logger.Log('enter logging message here')`

Note: The container provider log reads the logs only after AttachToContainer and after exceptions. The document provider log reads only after exceptions. For more information and the best visibility, use Log4j/Log4net.

For details on logging, see *About ALI Logging Utilities* on page 16.

Deploying a Custom Content Crawler (Java)

After implementing a custom content crawler, you must deploy your code.

Follow the instructions below to deploy a Java content crawler.

1. Compile the class that implements the IDK interface and copy the entire package structure to the appropriate location in your Web application (usually the \WEB-INF\classes directory).
2. Update the web.xml file in the WEB-INF directory by adding the class to the appropriate *Impl keys. For a content crawler, add your class to ContainerProviderImpl and DocumentProviderImpl as shown below. Note:The *Impl key in web.xml must reference the fully-qualified name of both provider classes required by the service. If the service uses SCI, you must also enter the fully-qualified name of the appropriate implementation of the IAdminEditor interface.

```
...
<env-entry>
<env-entry-name>ContainerProviderImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.crawler.helloworld.CrawlContainer</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
</env-entry>

<env-entry>
<env-entry-name>DocumentProviderImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.crawler.helloworld.CrawlDocument</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
</env-entry>
...
```

3. Start your application server. (In most cases, you must restart your application server after copying a file.)
4. Test the directory by opening the following page in a Web browser: http://<hostname:port>/edk/services/<servicetype>ProviderSoapBinding (for example, http://localhost:8080/edk/ContainerProviderSoapBinding and http://localhost:8080/edk/DocumentProviderSoapBinding). The browser should display the following message: "Hi there, this is an AXIS service! Perhaps there will be a form for invoking

the service here..." When you configure the Web Service for the content crawler in the portal, enter this path as the Service Provider URL.

5. If the content crawler uses DocFetch, you must also deploy your DocFetch code. Open the WEB-INF\web.xml file and add the fully-qualified name of your class in the DocFetchProvider initialization parameter, as shown in the code that follows.

```
...
<servlet>
<servlet-name>DocFetch</servlet-name>
<servlet-class>com.plumtree.remote.docfetch.DocFetch</servlet-class>

<!-- Modify the param-value below to reference your class -->
<init-param>
<param-name>DocFetchProvider</param-name>
<param-value>com.mycompany.MyDocFetchProvider</param-value>
</init-param>

</servlet>
...
```

For details on configuring your crawler, see *Configuring Content Crawlers* on page 379.

Web Service Class Names (*Impl)

To deploy a content service, identity service or SCI page, you must enter the qualified class name in the web.xml (Java) or Web.config (.NET) file for your project.

For Java services, add your class to the web.xml for your project. The `*Impl` key in the web.xml file must reference the fully-qualified name of the class. For content services and identity services, you must enter the name of both provider classes required by the service. If the service uses SCI, enter the fully-qualified name of the appropriate implementation of the `IAdminEditor` interface. For example, for an authentication service, add your class to `AuthProviderImpl` as shown below.

```
...
<env-entry>
<env-entry-name>AuthProviderImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.auth.helloworld.Auth</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
```

```
</env-entry>
...
```

For .NET services, add your class to the Web.config for your project. Some services also require an entry for the assembly. For example, for an authentication service, add your assembly to `AuthProviderAssembly` and your class to `AuthProviderImpl` as shown below.

Note: When you create a new project, Visual Studio sets the root namespace to the name of the project.This name is prepended to any namespace you define in the project.The `*Impl` key in the web.config file must reference the full namespace. (To view the root namespace, right-click the project name in Solution Explorer and click Properties.If you clear the root namespace field, the change will not be reflected until the project is rebuilt.)

```
...
<appSettings>
<add key="AuthProviderAssembly" value="dotnetauthsample"/>
<add key="AuthProviderImpl" value="dotnetauthsample.AuthSample"/>
</appSettings>
...
```

The table that follows provides the required `*Impl` parameters and associated service types.

Service Type	***Impl Parameters**
Authentication Service	AuthProviderImpl, SyncProviderImpl
Content Crawler	ContainerProviderImplDocumentProviderImplDocFetchImpl (.NET only) Note: Deploying DocFetch requires additional parameters.
Profile Service	ProfileProviderImpl
Federated Search	SearchImpl
Service Configuration Interface (SCI)	SciImpl

.NET services also require an entry for the associated assembly. The table below provides the required `*Assembly` parameters and associated service types.

Service Type	***Assembly Parameters**
Authentication Service	AuthProviderAssembly SyncProviderAssembly

Service Type	*Assembly Parameters
Content Crawler	ContainerProviderAssemblyDocumentProviderAssemblyDocFetchAssembly
Profile Service	ProfileProviderAssembly
Federated Search	SearchAssembly
Service Configuration Interface (SCI)	AdminEditorAssembly

Deploying a Custom Content Crawler (.NET)

After implementing a custom content crawler, you must deploy your code.

To deploy a .NET content crawler, add a line to the deployment file (web.config) that specifies the fully qualified name of the class. For a content crawler, enter values for the following parameters, as shown in the code that follows.

- **ContainerProviderImpl**
- **DocumentProviderImpl**
- **ContainerProviderAssembly**
- **DocumentProviderAssembly**

```
...
<appSettings>
<add key='ContainerProviderAssembly' value='CompanyStoreCWS'/>
<add key='ContainerProviderImpl'
value='Plumtree.CompanyStore.CWS.CompanyStoreContainer'/>
<add key='DocumentProviderAssembly' value='CompanyStoreCWS'/>
<add key='DocumentProviderImpl'
value='Plumtree.CompanyStore.CWS.CompanyStoreDocument'/>
...
```

If the content crawler uses DocFetch, you must also deploy your DocFetch code. Add a line to the deployment file (web.config) that specifies the fully qualified name of your class and the associated assembly (`DocFetchImpl` and `DocFetchAssembly`). You must also add three additional parameters to the web.config deployment descriptor:

- **DocFetchURL**: The URL to the DocFetch servlet or server page. This URL should be relative to the Remote Server object URL configured for the Content Crawler object in the portal to facilitate migration to another portal.

- **IndexFilePath**: A writable, web-accessible directory to which the IDK can write temporary files. During crawl-time, the IDK calls IDocument.GetDocument and copies the file path returned to this temporary file location, which is returned to the portal. These temporary files should be deleted upon completion of the crawl. (The DocFetch mechanism will clean up its own resources, but you must delete the temporary file you return to GetDocument.)
- **IndexURLPrefix**: The public Web address of the IndexFilePath directory. IndexURLPrefix must be an URL accessible from the portal server.

The code below is an example of deploying DocFetch in web.config.

```
...
<appSettings>
<add key='DocFetchAssembly' value='MyDocFetch' />
<add key='DocFetchImpl' value='com.mycompany.MyDocFetchProvider'
 />
<add key='DocFetchURL' value='iis/docfetch.aspx'/>
<add key='IndexFilePath' value='D:\\root\\config\\mydomain'/>
<add key='IndexURLPrefix'
value='http://yourhost/IISVirtualDirectory'/>
...
```

For details on configuring your crawler, see *Configuring Content Crawlers* on page 379.

Web Service Class Names (*Impl)

To deploy a content service, identity service or SCI page, you must enter the qualified class name in the web.xml (Java) or Web.config (.NET) file for your project.

For Java services, add your class to the web.xml for your project. The `*Impl` key in the web.xml file must reference the fully-qualified name of the class. For content services and identity services, you must enter the name of both provider classes required by the service. If the service uses SCI, enter the fully-qualified name of the appropriate implementation of the `IAdminEditor` interface. For example, for an authentication service, add your class to `AuthProviderImpl` as shown below.

```
...
<env-entry>
<env-entry-name>AuthProviderImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.auth.helloworld.Auth</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
```

```
</env-entry>
...
```

For .NET services, add your class to the Web.config for your project. Some services also require an entry for the assembly. For example, for an authentication service, add your assembly to `AuthProviderAssembly` and your class to `AuthProviderImpl` as shown below.

Note: When you create a new project, Visual Studio sets the root namespace to the name of the project.This name is prepended to any namespace you define in the project.The `*Impl` key in the web.config file must reference the full namespace. (To view the root namespace, right-click the project name in Solution Explorer and click Properties.If you clear the root namespace field, the change will not be reflected until the project is rebuilt.)

```
...
<appSettings>
<add key="AuthProviderAssembly" value="dotnetauthsample"/>
<add key="AuthProviderImpl" value="dotnetauthsample.AuthSample"/>
</appSettings>
...
```

The table that follows provides the required `*Impl` parameters and associated service types.

Service Type	***Impl Parameters**
Authentication Service	AuthProviderImpl, SyncProviderImpl
Content Crawler	ContainerProviderImplDocumentProviderImplDocFetchImpl (.NET only) Note: Deploying DocFetch requires additional parameters.
Profile Service	ProfileProviderImpl
Federated Search	SearchImpl
Service Configuration Interface (SCI)	SciImpl

.NET services also require an entry for the associated assembly. The table below provides the required `*Assembly` parameters and associated service types.

Service Type	***Assembly Parameters**
Authentication Service	AuthProviderAssembly SyncProviderAssembly

Service Type	*Assembly Parameters
Content Crawler	ContainerProviderAssemblyDocumentProviderAssemblyDocFetchAssembly
Profile Service	ProfileProviderAssembly
Federated Search	SearchAssembly
Service Configuration Interface (SCI)	AdminEditorAssembly

Testing Custom Content Crawlers

These key tests should be performed on every content crawler.

All the following tests should be performed in multiple implementations of the portal.

- **Test the entire crawl depth.** Confirm that documents are structured correctly in every level. Crawl depth should be as shallow as possible. If there are problems, check the filters on the target folders. If nothing is returned, check the authentication settings in the associated Content Source and Web Service - Content objects.
- **Check the document metadata.** Is it stored in the appropriate properties? Does it match the metadata in the source repository? If there are problems, check the Content Type settings in the Content Crawler editor, and check the mappings for each associated Content Type.
- **Click through to crawled documents from each crawled directory.** If there are problems, check the gateway settings in the Web Service - Content editor.
- **Test refreshing documents to confirm that they reflect modifications.** If there are problems, make sure you are providing the correct document signature.
- **Check logs after every crawl.** The log can reveal problems even if the portal reports a successful crawl.

Debugging Custom Content Crawlers

To debug custom content crawlers, use logging.

Logging is an important component of any successful content crawler. Logging allows you to track progress and find problems.

In most implementations, using Log4J or Log4Net for logging is the best approach. The IDK `ICrawlerLog` object is more efficient and useful than ALI Logging Spy or a SOAP trace, but

it only includes standard exceptions and messages from `ContainerProvider.AttachToContainer`.

If you are viewing the `ICrawlerLog`, do not assume that the every card was imported if the job is successful. Successful means no catastrophic failures, such as ALI Search not started, or unable to attach to the start node. Individual document failures will not fail a job.

If you are viewing logs created by Log4net or Log4j, see the associated documentation for logging configuration options. Both products allow you to specify a file location and a rollover log with a specified file size. If you know the location of the file, it is not difficult to create a servlet/aspx page that streams the file from the log to the browser.

For more information, see the following topics:

- *Handling Exceptions in Custom Content Crawlers* on page 371
- *About ALI Logging Utilities* on page 16
- *About the IDK Logging API* on page 16

Configuring Content Crawlers

Implementing a successful Content Crawler in the portal requires specific configuration.

To register a Content Crawler in the portal, you must create the following administrative objects and portal components:

- **Remote Server**: The Remote Server defines the base URL for the Content Crawler. Content Crawlers can use a Remote Server object or hard-coded URLs. Multiple services can share a single Remote Server object. If you will be using a Remote Server object, you must register it before registering any related Web Service objects.
- **Web Service - Content**: The Web Service object includes basic configuration settings, including the SOAP endpoints for the ContainerProvider and DocumentProvider, and Preference page URLs. Multiple Content Source or Content Crawler objects can use the same Web Service object. All remote Content Crawlers require an associated Web Service object. For information on specific settings, see the portal online help.
- **Content Source - Remote**: The Content Source defines the location and access restrictions for the back-end repository. Each Web Service - Content object has one or more associated Content Source objects. The Content Source editor can include Service Configuration pages created for the Content Crawler. Multiple Content Crawler objects can use the same Remote Content Source, allowing you to crawl multiple locations of the same content repository without having to repeatedly specify all the settings. For details on specific settings, see the portal

online help. For details on Service Configuration pages, see *Creating Service Configuration Pages for Content Crawlers* on page 381.

- **Content Crawler - Remote**: Each Content Crawler has an associated Content Crawler object that defines basic settings, including destination folder and Content Type. The Content Crawler editor can include Service Configuration pages created for the Content Crawler. Refresh settings are also entered in the Content Crawler editor. For details on specific settings, see the portal online help. For details on Service Configuration pages, see *Creating Service Configuration Pages for Content Crawlers* on page 381.
- **Job**: To run the Content Crawler, you must schedule a Job or add the Content Crawler object to an existing Job. The Content Crawler editor allows you to set a Job. For details on configuring Jobs, see the portal online help.
- **Global Content Type Map**: If you are importing a proprietary file format, you might need to create a new Content Type. Content Types are used to determine the type of accessor used to index a file. You can create new Content Types, or map additional file extensions to an existing Content Type using the Global Content Type Map. Most standard file formats are supported for indexing by the portal. In most cases, the same document is returned during a crawl (for indexing) as for click-through (for display). You can also map additional file extensions to Content Types through the Global Content Type Map. For detailed instructions, see the portal online help or the *AquaLogic Interaction Administrator Guide*.
- **Global Document Property Map**: To map document attributes to portal Properties, you must update the Global Document Property Map before running a Content Crawler. During a crawl, file attributes are imported into the portal and stored as Properties. The relationship between file attributes and portal Properties can be defined in two places: the Content Type editor or the Global Document Property Map.

  Two types of metadata are returned during a crawl.

  - The **crawler** (aka provider) iterates over documents in a repository and retrieves the file name, path, size, and usually nothing else.
  - During the indexing step, the file is copied to ALI Search, where the appropriate **accessor** executes full-text extraction and metadata extraction. For example, a for a Microsoft Office document, the portal uses the MS Office accessor to obtain additional properties, such as author, title, manager, category, etc.

  If there are conflicts between the two sets of metadata, the setting in CrawlerConstants.TAG_PROPERTIES determines which is stored in the database (for details, see Service Configuration Pages above).

  Note: If any properties returned by the crawler or accessor are not included in the Global Document Property map, they are discarded. Mappings for the specific Content Type have

precedence over mappings in the Global Document Property Map. The Object Created property is set by the portal and cannot be modified by code inside a Content Crawler.

- **Global ACL Sync Map**: Content Crawlers can import security settings based the Global ACL Sync Map, which defines how the Access Control List (ACL) of the source document corresponds with ALI's authentication groups. (An ACL consists of a list of names or groups. For each name or group, there is a corresponding list of possible permissions. The ACL returned to the portal is for read rights only.) For detailed instructions, see the portal online help or the *AquaLogic Interaction Administrator Guide*.

  In most cases, the Global ACL Sync Map is automatically maintained by Authentication Sources. The Authentication Source is the first step in ALI security. To import security settings in a crawl, the back-end repository must have an associated Authentication Source. Content Crawlers that import security need the user and category (domain) defined by an Authentication Source. You must configure the Authentication Source before the Content Crawler is run. Many repositories use the network's NT or LDAP security store; if an associated Authentication Source already exists, there is no need to create one.

  Note: Two settings are required to import security settings:

  - In the **Web Service - Content** editor on the Advanced Settings page, check Supports importing security with each document.
  - In the **Content Crawler** editor on the Main Settings page, check Import security with each document.

Creating Service Configuration Pages for Content Crawlers

Service Configuration (SCI) pages are integrated with portal editors and used to define settings used by a Content Crawler.

Content Crawlers must provide SCI pages for the Content Source and/or Content Crawler editors to build the preferences used by the Content Crawler. The URL to any associated SCI page(s) must be entered on the Advanced URLs page of the Web Service - Content editor.

All optional settings are in the class CrawlerConstants. For a list, see *SCI Variables for Content Crawler Properties* on page 357.

SCI provides an easy way to write configuration pages that are integrated with portal editors. SCI wraps the portal's XUI XML and allows you to create controls without XUI. For a complete listing of classes and methods in the plumtree.remote.sci namespace, see the IDK API documentation. The following methods must be implemented: .

- `Initialize` passes the namespace, whether Content Source or Content Crawler, settings (NamedValueMap). Dependent objects supply data.
- `GetPages` returns fixed-length array of the number of custom pages.
- `GetContent` returns the XML content for a page. The API provides a collection of helper classes to build the page (textbox, select box, tree element, etc.)

The example below is a SCI page for a Content Source editor that gets credentials for a database Content Crawler.

```
Imports System
Imports Plumtree.Remote.Sci
Imports Plumtree.Remote.Util
Imports System.Security.Cryptography

Namespace Plumtree.Remote.Crawler.DRV
'Page to enter name and password- first page for DataSourceEditor
Public Class AuthPage
Inherits AbstractPage
#Region "Constructors"
Public Sub New(ByVal editor As AbstractEditor)
MyBase.New(editor)
End Sub
#End Region

#Region "Functions"
'Gets the content for the page in string form.
'One textElement for name, one PasswordElement for password
'Note the way that the password is stored & the encryption used
Public Overrides Function GetContent(ByVal errorCode As Integer,
 ByVal pageInfo As NamedValueMap) As String
Dim page As New SciPage
Dim userElement As New SciTextElement(DRVConstants.USER_NAME,
"Enter the user name to authenticate to SQL Server")
Dim userName As String = pageInfo.Get(DRVConstants.USER_NAME)
If Not userName Is Nothing Then
  userElement.SetValue(userName)
End If
userElement.SetMandatoryValidation("User name is mandatory")

            Dim passElement As New
SciPasswordElement(DRVConstants.PASSWORD, "Enter the password to
 authenticate to SQL Server", "Confirm", "Passwords do not match")
'deal with asterisks and the like- for now, just show password
Dim password As String = pageInfo.Get(DRVConstants.ENC_PASSWORD)
```

```
'save the initial password?
Dim settings As NamedValueMap = Me.Editor.Settings
settings.Put(DRVConstants.ENC_PASSWORD, password)
Editor.Settings = settings
'set asterisks for the value
passElement.SetValue(DRVConstants.ASTERISKS)

            page.Add(userElement)
page.Add(passElement)

            Return page.ToString
End Function

        'Gets the help page URI for the page.
Public Overrides Function GetHelpURI() As String
Return ""
End Function

        'Gets the image (icon) URI for the page. (This setting is
 for backward compatibility; no icon is displayed in version 5.0.)
Public Overrides Function GetImageURI() As String
Return ""
End Function

        'Gets the instructions for the page, displayed below the
 title in the editor.
Public Overrides Function GetInstructions() As String
Return "Enter SQL Server authentication information"
End Function

        'Gets the title for the page.
Public Overrides Function GetTitle() As String
Return "SQL Server Authentication"
End Function

        'Validates the current page and throws a
ValidationException to report an error. Returns a NamedValueMap
array of the settings entered on the editor page.
Public Overrides Sub ValidatePage(ByVal pageInfo As NamedValueMap)
'if the password is not asterisks, then put it into settings
Dim password As String = pageInfo.Get(DRVConstants.PASSWORD)
If Not password.Equals(DRVConstants.ASTERISKS) Then
  Dim settings As NamedValueMap = Me.Editor.Settings
  'encrypt this
  Dim encPassword As String = Utilities.EncryptPassword(password,
```

```
 Me.Editor.Locale)
  settings.Put(DRVConstants.ENC_PASSWORD, encPassword)
  Editor.Settings = settings
End If

        End Sub
#End Region

    End Class
End Namespace
```

Customizing ALI Search

ALI search can be extended in a number of ways, including adding to the portal search index, implementing web services to access content in other repositories, customizing the search UI, and adding portal search to remote services.

The most common customizations are implemented using web services.

- To import content and index it in the portal Knowledge Directory, use **Content Crawlers**. For details, see *About Content Crawlers* on page 349.
- To access external repositories from the portal without importing content into the Knowledge Directory, use **Federated Search Services**. For details, see *About ALI Federated Search Services* on page 385.
- To display search results, provide a customized search form, add constraints to search, and more, use **Search Portlets**.
    - **To display the results of a common query for a specific audience**, you might be able to use a **Snapshot Query** and **Content Snapshot Portlet**. This approach does not require any coding. The search query is defined in the portal UI and automatically generate a portlet that displays current results. The results are automatically cached to improve performance and decrease the load on ALI Search. There are some minor drawbacks: only a few styles are available for the result list, and not all constraints are available. (For example, you can search for portlets, but you cannot restrict to community portlets or to portlets that fit in a narrow column.) For more information and detailed instructions, see the portal online help the *Administrator Guide for AquaLogic Interaction*.

- **To display basic search functionality in a portlet**, use **Adaptive Tags**. Search tags provide access to a basic search form, as well as advanced search, federated search, and top best bet functionality. For details see *About Adaptive Tags* on page 202.
- **To implement custom search functionality in a portlet**, use the **PRC Search API**. The PRC is a SOAP API used to run search queries against the ALI system. You can run virtually any search supported by the portal and use any technology to generate the HTML. Simply create an HTML form with hidden inputs specifying constraints on the search. Search queries can be simple text strings (like banner search) or complex trees of filter clauses, and can be restricted by Knowledge Directory or administrative folder, or by community. For details, see *About Remote Search Operations* on page 90.

About ALI Federated Search Services

Federated Search provides access to external repositories without adding documents to the portal Knowledge Directory. Federated Search is especially useful for content that is updated frequently or is only accessed by a small number of portal users

When the portal requests a federated search service, the remote service accesses the content repository and sends information about each file to the portal. The returned information is displayed to users in search results. The results include a URL that opens the file from the back-end content repository.

For details on implementing federated search services, see the following topics:

- *Creating a Federated Search Service* on page 385
- *IDK Interfaces for Federated Search Service Development* on page 386
- *Deploying a Federated Search Service (Java)* on page 388
- *Deploying a Federated Search Service (.NET)* on page 389

Creating a Federated Search Service

The AquaLogic Interaction Development Kit (IDK) allows you to create remote Federated Search services and related configuration pages without parsing SOAP or accessing the portal API. The IDK Search API provides an abstraction from the necessary SOAP calls; you simply implement an object interface.

The following best practices apply to every federated search service:

- Know what to expect in response to a query. You must be ready to handle pagination and authentication if necessary.
- Check the SOAP timeout for the back-end server and calibrate your response accordingly.

- Use relative URLs in your code to allow migration to another remote server.

For details on implementing Federated Search Services using the IDK Search API, see *IDK Interfaces for Federated Search Service Development* on page 386.

IDK Interfaces for Federated Search Service Development

The IDK's Plumtree.Remote.Search package/namespace includes a set of interfaces to support federated search service development.

The IDK's Plumtree.Remote.Search package/namespace includes the following interfaces:

- `IRemoteSearch`
- `ISearchQuery`
- `ISearchUser`
- `ISearchContext`
- `ISearchRecord`
- `ISearchResult`

In general, the portal calls these interfaces in the following order. See the definitions that follow for more information.

1. `IRemoteSearch.BasicSearch`, using `ISearchQuery`, `ISearchUser` and `ISearchContext` as parameters.
2. The `ISearchResult` object returned allows the federated search service to iterate through the search results and return them to the user. The service calls `ISearchResult.GetSearchResultList` to retrieve an `ISearchRecord` for each record returned. `ISearchRecord` allows you to retrieve the title, description, file URL and image URL and set the title, description, file URL and image URL to be returned to the portal.

The sections below provide helpful information on the interfaces used to implement a federated search service For a complete listing of interfaces, classes, and methods, see the IDK API documentation.

IRemoteSearch

The `IRemoteSearch` interface allows the portal to initiate a query over a back-end directory structure. `BasicSearch` allows you to pass in an `ISearchQuery` that defines the query to be performed. You can also pass in a `ISearchUser` and `ISearchContext` for access to the PRC.

ISearchQuery

The `ISearchQuery` interface defines the search query to be performed by the portal. Using `ISearchQuery`, you can define the scope of the query and provide user preferences and user information to be used for authentication or user-level access control. `SearchException` allows you to provide useful error messages (for example, the specific preference type that was not found). For details, see the IDK API documentation. This interface provides the following methods:

- `GetMaxReturn` determines the maximum number of records to return per page.
- `GetNumberToSkip` returns the number of records that will be skipped: where the search will start. For example, the search could start at record 30.
- `GetSearchInfo` returns any related administrative preferences set for the associated Federated Search object in the portal.
- `GetSearchResult` returns an `ISearchResult` object that allows the federated search service to access the results returned by IRemoteSearch.
- `GetSearchString` returns the query string passed to the portal.
- `GetUserInfo` returns any User Information settings sent to the federated search service. To access User Information, you must configure the specific settings you need in the Web Service editor on the User Information page.
- `GetUserPrefs` returns any user settings sent to the federated search service. To access user settings, you must configure the specific settings you need in the Web Service editor on the Preferences page.

ISearchUser

The `ISearchUser` interface can be used to access the current user's portal object ID and locale, and to obtain the login token for the current session with the portal to access the PRC.

ISearchContext

The `ISearchContext` interface can be used to access the portal UUID and SOAP service endpoint URI to implement the PRC.

ISearchResult

The `ISearchResult` interface allows you to retrieve the results returned from a search query and return the results to the portal. The federated search service code must handle pagination; the methods in the `ISearchResult` facilitate iteration over large numbers of search records.

- `Get/SetNumberSkipped` returns the number of records that were skipped: where the search started. For example, the search could start at record 30.
- `Get/SetSearchResultList` returns a `SearchRecord` array of search results.
- `Get/SetTotalNumberofHits` returns the total number of search records.
- `Is/SetDescriptionEncoded` determines whether or not the description for the search results is HTMLencoded.

ISearchRecord

The `ISearchRecord` interface allows you to manipulate the metadata for each search record. Only the title is required.

- `Get/SetTitle` returns the title for the search record (required).
- `Get/SetDescription` returns the description for the search record. If the description should be HTMLencoded, use `ISearchResult.SetDescriptionEncoded`.
- `Get/SetOpenDocumentURL` returns the URL that will retrieve the document. This URL must be accessible over the web or through the gateway. If the document is gatewayed, make sure to configure the Web Service object with the appropriate gateway URLs.
- `Get/SetImageURL` returns the URL to the image that will be displayed with the search record.

Deploying a Federated Search Service (Java)

After implementing a federated search service, you must deploy your code.

Follow the instructions below to deploy a Java federated search service:

1. Compile the class that implements the IDK interface and copy the entire package structure to the appropriate location in your web application (usually the \WEB-INF\classes directory).
2. Update the web.xml file in the WEB-INF directory by adding the class to the appropriate *Impl keys. For example, add your class to `SearchImpl` as shown below. Note: The *Impl key in the web.xml file must reference the fully-qualified name of the class. If the service uses SCI, you must also enter the fully-qualified name of the appropriate implementation of the `IAdminEditor` interface. For a list of keys, see *Web Service Class Names (*Impl)* on page 421.

```
...
<env-entry>
<env-entry-name>SearchImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.search.helloworld.Search</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
```

```
</env-entry>
...
```

3. Start your application server. (In most cases, you must restart your application server after copying a file.)
4. Test the directory by opening the following page in a web browser: http://<hostname:port>/idk/services/<servicetype>ProviderSoapBinding (for example, http://localhost:8080/idk/SearchSoapBinding). The browser should display the following message: "Hi there, this is an AXIS service! Perhaps there will be a form for invoking the service here..." When you configure the Web Service for the federated search service in the portal, enter this path as the Service Provider URL.
5. If the federated search service uses a SCI page to define settings, you must also deploy the SCI code. For details on using SCI pages, see *Creating Service Configuration Pages for Content Crawlers* on page 381.

Deploying a Federated Search Service (.NET)

After implementing a federated search service, you must deploy your code.

To deploy a .NET federated search service, add a line to the deployment file (web.config) that specifies the fully qualified name of the class used to implement federated search. For a federated search service, you must enter values for the following parameters, as shown in the code that follows.

- **SearchImpl**
- **SearchAssembly**

For a list of keys, see *Web Service Class Names (*Impl)* on page 421.

```
...
<appSettings>
<add key='SearchAssembly' value='CompanyStoreSWS'/>
<add key='SearchImpl'
value='Plumtree.CompanyStore.SWS.CompanyStoreSWS'/>
...
```

If the federated search service uses a SCI page to define settings, you must also deploy the SCI code. For details on using SCI pages, see *Creating Service Configuration Pages for Content Crawlers* on page 381.

CHAPTER 5

About Identity Services (IDS)

Identity Services allow you to integrate established repositories of user information into your portal.

Users, groups, and group membership configuration can be imported into the portal. Users logging into the portal can be authenticated against the existing system of record. Information about users can be imported from any number of external sources and mapped to portal properties, which can then be made available to the portal or other services.

- Authentication services are used to import users into the portal and authenticate them against a back-end system. For details, see *About Authentication Services (AWS)* on page 392
- Profile services are used to import information about existing portal users from external systems and map that information to portal properties. For details, see *About Profile Services (PWS)* on page 410

In addition to the authentication and profile services, the following functionality is available to control portal users in your implementation:

- Experience definitions let you tailor portal experiences for different groups of users. For details, see the portal online help.
- Remote User Operations allow you to access and manage portal users from remote applications. For details, see *About Remote User Operations* on page 86.

About Authentication Services (AWS)

Authentication services are comprised of two parts: synchronization and authentication. Together, these components import new users and allow them to authenticate against the external system of record.

Synchronization

The synchronization component of an authentication service imports users from an external system into the portal so that the users can be categorized in the portal's group hierarchy. The synchronization process is handled by the portal Automation Server, as scheduled in the Job associated with the Authentication Source object in the portal.

Synchronization does not store users' passwords in the portal database. Authentication is handled by the authentication component and the system of record.

Note: Creating an Authentication Source object with a synchronization component creates an associated option in the Authentication Source drop-down list on the portal login page. The name that appears in the drop-down list is the Description of the Authentication Source object. Enter a description that all users will recognize.

Authentication

The authentication component of an authentication service handles real-time authentication of portal users against an external system. Since the portal cannot change an externally managed password, a user's login must be compared against the system of record. The remote authentication service must maintain state and handle the communication between the portal and the back-end system. The user name and password can be captured in the session at login to be used later for basic authentication.

Development

The following topics provide detailed instructions on developing custom authentication services:

- *Authentication Service Internals* on page 393 describes the interfaces that must be implemented when creating an authentication service, and how the interfaces will be called by the portal.
- *Implementing an Authentication Service* on page 394 provides step by step instructions on implementing the required interfaces, with example code.

- *Deploying a Java Authentication Service* on page 403 and*Deploying a .NET Authentication Service* on page 405 describe how to deploy an authentication service to a Java or .NET application server.
- *Configuring an Authentication Service* on page 409 describes how to configure the authentication service in the portal.

Authentication Service Internals

The following interfaces must be implemented when creating an authentication service.

The AquaLogic Interaction Development Kit (IDK), formerly called the EDK, allows you to create remote authentication services and related configuration pages without parsing SOAP or accessing the portal API. The IDK Authentication API provides an abstraction from the necessary SOAP calls; you simply implement an object interface. For a complete listing of interfaces, classes, and methods, see the IDK API documentation.

Note: The differences between the Java and .NET versions of the IDK are platform-specific. In this guide, method names are listed using the .NET standard of initial capitalization. Java methods are identical, except begin with a lowercase letter. The `ISyncProvider.Initialize` method in the .NET IDK provides the same functionality as the `ISyncProvider.initialize` method in the Java IDK.

Plumtree.Remote.Auth

The `Plumtree.Remote.Auth` namespace (the com.plumtree.remote.auth package in Java), provides interfaces for creating authentication and synchronization services for users and groups in the portal. There are three interfaces provided:

- `ISyncProvider`
- `IGroup`
- `IAuthProvider`

To provide synchronization with an external source, implement `ISyncProvider` and `IGroup`. To provide authentication against an external source, implement `IAuthProvider`. In most cases, all three interfaces should be implemented.

Synchronization

User and group synchronization takes place when the associated synchronization Job is run by the portal Automation Service. The synchronization service must maintain state between the portal, the remote server, and the back-end system until synchronization is complete. Users are imported

on each run via `ISyncProvider`. Imported users are put into groups based on information from `IGroup` object(s). The portal typically calls the methods of the authentication service interfaces in the following order:

1. `ISyncProvider.Initialize`
2. `ISyncProvider.GetGroups`
3. `ISyncProvider.Initialize`
4. `ISyncProvider.GetUsers`
5. `ISyncProvider.Initialize`
6. `ISyncProvider.AttachToGroup` for each group returned in `ISyncProvider.GetGroups`
   a. `IGroup.GetChildGroups`
   b. `IGroup.GetChildUsers`

Note: The portal may take a long time between calls to `GetGroups()`, `GetUsers()`, and `AttachToGroup()`. Because of this, the Java or .NET session on the remote server may time out, so `Initialize()` is called more than once.

Authentication

When a user logs into the portal, the authentication service is called to authenticate against the back-end system. This is done through a single call to `IAuthProvider.Authenticate`.

Once logged in, each user is associated with a portal User object; authentication services do not need to maintain state.

Implementing an Authentication Service

To implement an authentication service, follow these step by step instructions.

This section describes the details of how to create an authentication service. This authentication service is very simple, and is intended only as an example. It should not be used in a production environment. The functional requirements for this authentication service are as follows:

- A single group will be synchronized into the portal: BASEGROUP
- Ten users will be synchronized into the portal: TESTUSER0 - TESTUSER9
- The ten users will be members of BASEGROUP
- The ten users will all authenticate with the same password: TESTUSER

This authentication source is created by implementing the following interfaces:

- `ISyncProvider`
- `IGroup`
- `IAuthProvider`

These interfaces are in the `Plumtree.Remote.Auth` namespace (C#) or the `plumtree.remote.auth` package (Java). Any exceptions thrown in this code can be found in `Plumtree.Remote` (C#) or `plumtree.remote` (Java).

For the purposes of this authentication service, a simple class, `Constants`, is created in the example namespace/package. Constants has two public members, the String constants GROUPNAME and USERNAME. These are set as follows:

- `Constants.GROUPNAME = "BASEGROUP"`
- `Constants.USERNAME = "TESTUSER"`

These constants are used to provide the group name and a base for the user names to the authentication source code.

For the complete code of this example, see the sample code on the Developer Center.

For details of the classes described in this section, see the IDK (EDK) API documentation.

1. Implement the `ISyncProvider` Interface

The `ISyncProvider` interface provides methods used by the portal to synchronize users and groups with a back-end repository. The methods of `ISyncProvider` are typically called in this order:

1. `ISyncProvider.Initialize()`
2. `ISyncProvider.GetGroups()`
3. `ISyncProvider.Initialize()`
4. `ISyncProvider.GetUsers()`
5. `ISyncProvider.Initialize()`
6. `ISyncProvider.AttachToGroup()`

Because the portal may take a long time between calls to `GetGroups()`, `GetUsers()` and `AttachToGroup()`, `Initialize()` is called more than once. This ensures that any configuration information passed to the synchronization service is available, even if the session has timed out.

a) Implement `ISyncProvider.Initialize`

The `Initialize()` method is passed a `SyncInfo` object from the portal. The `SyncInfo` object is a set of name-value pairs, populated with information entered in the

portal **Service Configuration Interface (SCI)** editor by a portal administrator. Typically, this is information such as credentials for connecting to the back-end system.

Java:

```
public boolean initialize(SyncInfo info)
  throws ServiceException
{
    return true;
}
```

C#:

```
public bool Initialize(SyncInfo syncInfo)
{
    return true;
}
```

For this example authentication service, there is no need to perform any initialization, so the method simply returns `true`.

In a production implementation, `false` should be returned if initialization fails. For example, if the authentication service cannot make a connection with the back-end repository. If `false` is returned, the synchronization job will stop.

b) Implement `ISyncProvider.GetGroups`

The `GetGroups()` method is responsible for returning all of the groups from the back-end system. A `SyncObject` should be created for each group, using the static `SyncObject.CreateGroup()` method. The return value, a `SyncObjectList`, is then constructed using an array of `SyncObject` objects and a boolean flag, `isDone`.

The `isDone` flag determines whether or not the `GetGroups()` method will be called again. When you have a large number of groups in your back-end system, you can return groups to the portal in smaller batches. The size of each batch should be based on network bandwidth, the SOAP timeout set in the **Authentication Web Service**, and the speed of the back-end system. As a general rule, return no more than 1000 groups per batch.

If `GetUsers()` returns `SyncObjects` in batches, it must maintain state and set `isDone` to false until the last batch. Otherwise, `isDone` should be set to true.

Java:

```
public SyncObjectList getGroups()
  throws ServiceException
{
  SyncObject[] groups = new SyncObject[1];
```

```
  groups[0] = SyncObject.createGroup(
      Constants.GROUPNAME,
      Constants.GROUPNAME);
  return new SyncObjectList(groups, true);
}
```

C#:

```
public SyncObjectList GetGroups()
{
  SyncObject[] groups = new SyncObject[1];
  groups[0] = SyncObject.CreateGroup(
      Constants.GROUPNAME,
      Constants.GROUPNAME);
  return new SyncObjectList(groups, true);
}
```

For this authentication service, `GetGroups()` returns a single group, *BASEGROUP*. As stated above, *Constants.GROUPNAME* is a String set to "*BASEGROUP*". `SyncObject.CreateGroup()` accepts two String arguments. The first argument is the name the imported group will have in the portal. The second argument is the name of the group in the back-end system. In this case, we set both to "*BASEGROUP*". The returned `SyncObjectList` is then constructed with the single item `SyncObject` array and the `isDone` flag set to true, signifying there are no more groups for the portal to synchronize.

c) Implement `ISyncProvider.GetUsers`

The `GetUsers()` method is responsible for returning all of the users from the back-end system. Similar to `GetGroups()`, a *SyncObject* should be created for each group. For `GetUsers()`, use the static `SyncObject.CreateUser()` method to create each new *SyncObject*. The return value, a *SyncObjectList*, is then constructed using an array of *SyncObject* objects and a boolean flag, `isDone`.

As with `GetGroups()`, the `isDone` flag tells the portal whether it should call `GetUsers()` again or not, allowing you to break retrieval of users into batches. The size of each batch should be based on network bandwidth, the SOAP timeout set in the **Authentication Web Service**, and the speed of the back-end system. As a general rule, return no more than 1000 users per batch.

If `GetUsers()` returns `SyncObjects` in batches, it must maintain state and set `isDone` to false until the last batch. Otherwise, `isDone` should be set to true.

Java:

```
public SyncObjectList getUsers()
  throws ServiceException
```

```
{
    SyncObject[] users = new SyncObject[10];
    for (int i = 0; i < 10; i++)
    {
      String userName = Constants.USERNAME + i;
      users[i] = SyncObject.createUser(
          userName, userName, userName);
    }
    return new SyncObjectList(users, true);
\}
```

C#

```
public SyncObjectList GetUsers()
{
    SyncObject[] users = new SyncObject[10];
    for (int i = 0; i < 10; i++)
    {
      String userName = Constants.USERNAME + i;
      users[i] = SyncObject.CreateUser(
                  userName, userName, userName);
    }
    return new SyncObjectList(users, true);
```

For this authentication service, `GetUsers()` returns ten users, TESTUSER0 - TESTUSER9. An array of ten *SyncObject* objects is created, and then populated using a for loop. The `SyncObject.CreateUser()` method takes three String arguments: The first is the name of the user in the portal, the second is the user name that will be passed for authentication, and the third is the name of the user in the back-end system. In this case, all are set to the same String, *Constants.USERNAME* + i.

The returned `SyncObjectList` is then constructed with the ten item `SyncObject` array and the `isDone` flag set to true, signifying there are no more users for the portal to synchronize.

d) Implement `ISyncProvider.AttachToGroup`

`AttachToGroup()` returns an `IGroup` object that allows the portal to query for users and groups contained within a given group. For details on implementing the `IGroup` interface, see Implementing the `IGroup` Interface, below. `AttachToGroup()` is passed a String, a group identifier on the back-end system. This is the same as the second argument passed to `SyncObject.CreateGroups()` in `ISyncProvider.GetGroups()`. `AttachToGroup()` should return an instance of an implementation of the `IGroup` interface.

Java:

```
public IGroup attachToGroup(String groupID)
  throws ServiceException
{
    if (groupID.equals(Constants.GROUPNAME))
    {
      return new Group();
    }
    else
    {
      return null;
    }
}
```

C#:

```
public IGroup AttachToGroup(String groupID)
{
    if (groupID.Equals(Constants.GROUPNAME))
      return new Group();
    else
      return null;
}
```

For this authentication service, there is only one group, *BASEGROUP*. If the group ID passed to `AttachToGroup()` is *BASEGROUP*, a Group object is returned. Otherwise, null is returned. This is highly simplified; in a production implementation, `AttachToGroup()` would query a back-end system and return a group object with specific information for the given group.

2. Implement the `IGroup` Interface

The `IGroup` interface provides methods that allow the portal to determine relationships between users and groups. The portal takes the `IGroup` object returned from each call to `ISyncProvider.AttachToGroup()` and calls two `IGroup` methods:

1. `IGroup.GetChildGroups()`
2. `IGroup.GetChildUsers()`

Similar to the `ISyncProvider.GetGroups()` and `ISyncProvider.GetUsers()` methods, the `GetChildGroups()` and `GetChildUsers()` methods return objects that contain an array of either groups or users. In both cases, the `isDone` flag can be used to send results back to the portal in batches. The size of each batch should be based on network

bandwidth, the SOAP timeout set in the **Authentication Web Service**, and the speed of the back-end system. As a general rule, return no more than 1000 groups or users per batch.

a) Implement `IGroup.GetChildGroups`

The `GetChildGroups()` method defines which child groups (subgroups) each group contains. The child groups are returned as `ChildGroup` objects in a `ChildGroupList` object. The `ChildGroup` constructor takes a single String argument, the unique name that identifies the group in the authentication service. The `ChildGroupList` constructor should be passed the array of `ChildGroup` objects and the `isDone` flag. If you want to return child groups in batches, your implementation of IGroup must maintain state internally and the `isDone` flag must be set to false until the final batch.

Java:

```
public ChildGroupList getChildGroups()
  throws ServiceException
{
    ChildGroup[] children = new ChildGroup[0]);
    return new ChildGroupList(children, true);
}
```

C#:

```
public ChildGroupList GetChildGroups()
{
    ChildGroup[] children = new ChildGroup[0];
    return new ChildGroupList(children, true);
}
```

In this example authentication service, there are no child groups, so an empty array is returned.

b) Implement `IGroup.GetChildUsers`

The `GetChildUsers()` method returns the user membership of the group. The users are returned as ChildUser objects, which are constructed with the same arguments as `ISyncProvider.CreateUser()`. The `ChildUserList` constructor should be passed the array of `ChildGroup` objects and the `isDone` flag. If you want to return child users in batches, your implementation of `IGroup` must maintain state internally and the `isDone` flag must be set to false until the final batch.

Java:

```
public ChildUserList getChildUsers()
  throws ServiceException
{
```

```
        ChildUser[] users = new ChildUser[10];
        for (int i = 0; i < 10; i++)
        {
          String userName = Constants.USERNAME + i;
         users[i] = new ChildUser(userName, userName, userName);

        }
\        return new ChildUserList(users, true);
}
```

C#:

```
public ChildUserList GetChildUsers()
{
        ChildUser[] users = new ChildUser[10];
        for (int i = 0; i < 10; i++)
        {
          String userName = Constants.USERNAME + i;
         users[i] = new ChildUser(userName, userName, userName);

        }
        return new ChildUserList(users, true);
}
```

For this authentication service, `GetChildUsers()` returns the same ten users as `ISyncProvider.GetUsers()`, TESTUSER0 - TESTUSER9. An array of ten `ChildUser` objects is created, and then populated using a for loop. The `ChildUser` constructor takes three String arguments: the first is the name of the user in the portal, the second is the user name that will be passed for authentication, and the third is the name of the user in the back-end system. In this case, all are set to the same String, *Constants.USERNAME* + *i*.

The returned `ChildUserList` is then constructed with the ten item `ChildUser` array and the `isDone` flag set to true, signifying there are no more users associated with this group.

3. Implement the IAuthProvider Interface

The `IAuthProvider` interface validates credentials from a portal login against a back-end repository. There is a single method to implement,`Authenticate()`.

The `Authenticate()` method is passed three arguments: two Strings for username and password, and an `AuthInfo` object. The `AuthInfo` object, like the `SyncInfo` object passed to `ISyncProvider.Initialize()`, is a set of name-value pairs, populated with

information entered in the portal SCI editor by a portal administrator. Typically, this is information such as credentials for connecting to the back-end system.

If the credentials passed are valid, `Authenticate()` should return normally; however, if the credentials are invalid, an exception of type `ServiceException` must be thrown. See the IDK API documentation for a complete description of the exceptions derived from `ServiceException`.

Java:

```
public void authenticate(String username,
                String password, AuthInfo authinfo)
  throws ServiceException
{
    if (username.startsWith(Constants.USERNAME_BASE) &&
        password.startsWith(Constants.USERNAME_BASE))
    {
      //do nothing- authenticated
    }
    else
    {
      throw new AccessDeniedException();
    }
}
```

C#:

```
public void Authenticate(String username,
                String password, AuthInfo authInfo)
{
    if (username.StartsWith(Constants.USERNAME) &&
        password.StartsWith(Constants.USERNAME))
    {
       //do nothing- authenticated
    }
    else
    {
      throw new AccessDeniedException();
    }
}
```

For this authentication service, if the username and password passed in start with *Constants.USERNAME* ("TESTUSER"), the user is authenticated. Otherwise, an `AccessDeniedException` is thrown.

Note: If a message is provided in the exception, the message will not be displayed to the user in the UI. The message will be caught by the IDK and sent to the ALI Logging Utilities.

Deploying a Java Authentication Service

To deploy an authentication service to a Java application server, follow these instructions.

- IDK must be installed on the server to which you intend to deploy.
- You must have implemented ISyncProvider and IGroup (for synchronization), IAuthProvider (for authentication), or both. For details, see *Implementing an Authentication Service* on page 394.

This topic describes how to deploy an Authentication Service implemented in Java to a Java application server.

To deploy an authentication service to a supported Java application server:

1. Access the IDK deployment servlet (DeployServlet) in a browser.

   The IDK deployment servlet is located at `http://app server:port/edk/DeployServlet`
2. Choose **Auth** and wait for the page to reload.
3. Enter a prefix to identify this authentication service and the fully qualified name of the implementation of IAuthProvider, ISyncProvider, or both.
4. If this service uses SCI, check **Use Service Configuration Interface (SCI)** and enter the fully qualified name of the appropriate implementation of `IAdminEditor`.
5. Copy and paste the URLs displayed on the results page to a text file; these are the URLs that should be used when you configure the service in the portal.

Web Service Class Names (*Impl)

To deploy a content service, identity service or SCI page, you must enter the qualified class name in the web.xml (Java) or Web.config (.NET) file for your project.

For Java services, add your class to the web.xml for your project. The `*Impl` key in the web.xml file must reference the fully-qualified name of the class. For content services and identity services, you must enter the name of both provider classes required by the service. If the service uses SCI, enter the fully-qualified name of the appropriate implementation of the `IAdminEditor` interface.

For example, for an authentication service, add your class to `AuthProviderImpl` as shown below.

```
...
<env-entry>
<env-entry-name>AuthProviderImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.auth.helloworld.Auth</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
</env-entry>
...
```

For .NET services, add your class to the Web.config for your project. Some services also require an entry for the assembly. For example, for an authentication service, add your assembly to `AuthProviderAssembly` and your class to `AuthProviderImpl` as shown below.

Note: When you create a new project, Visual Studio sets the root namespace to the name of the project.This name is prepended to any namespace you define in the project.The `*Impl` key in the web.config file must reference the full namespace. (To view the root namespace, right-click the project name in Solution Explorer and click Properties.If you clear the root namespace field, the change will not be reflected until the project is rebuilt.)

```
...
<appSettings>
<add key="AuthProviderAssembly" value="dotnetauthsample"/>
<add key="AuthProviderImpl" value="dotnetauthsample.AuthSample"/>
</appSettings>
...
```

The table that follows provides the required `*Impl` parameters and associated service types.

Service Type	***Impl Parameters**
Authentication Service	AuthProviderImpl, SyncProviderImpl
Content Crawler	ContainerProviderImplDocumentProviderImplDocFetchImpl (.NET only) Note: Deploying DocFetch requires additional parameters.
Profile Service	ProfileProviderImpl
Federated Search	SearchImpl
Service Configuration Interface (SCI)	SciImpl

.NET services also require an entry for the associated assembly. The table below provides the required `*Assembly` parameters and associated service types.

Service Type	*Assembly Parameters
Authentication Service	AuthProviderAssembly SyncProviderAssembly
Content Crawler	ContainerProviderAssemblyDocumentProviderAssemblyDocFetchAssembly
Profile Service	ProfileProviderAssembly
Federated Search	SearchAssembly
Service Configuration Interface (SCI)	AdminEditorAssembly

Deploying a .NET Authentication Service

To deploy an authentication service to IIS and .NET, follow these instructions.

- The IDK must be installed on the server to which you intend to deploy.
- You must have implemented ISyncProvider and IGroup (for synchronization), IAuthProvider (for authentication), or both. For details, see *Implementing an Authentication Service* on page 394.

This topic describes how to deploy an Authentication Service implemented in Java to a Java application server.

To deploy an authentication service to IIS and .NET:

1. Ensure that you have built your project with the AuthProviderSoapBinding.asmx and/or SyncProviderSoapBinding.asmx SOAP endpoints.

   If this service uses SCI, also include SCIProviderBinding.asmx. These files can be found in your IDK installation.

   The IDK installation is typically installed to `C:\Program Files\plumtree\ptedk\5.3\devkit\`

2. Update `Web.config` for services that provide authentication.

   If this service provides authentication, add the following nodes to `<appSettings>`:

- `<add key="AuthProviderAssembly" value="`*`assembly name`*`" />`

  Where *assembly name* is the name of the assembly containing your `IAuthProvider` implementation.

- `<add key="AuthProviderImpl" value="`*`fully qualified path`*`" />`

  Where *fully qualified path* is the fully qualified path to the class implementing `IAuthProvider.`

For example:

```
<appSettings>

    <add key="AuthProviderAssembly" value="Helloworld" />

    <add key="AuthProviderImpl"
value="Plumtree.Remote.Auth.Helloworld.Auth" />

...
```

3. Update `Web.config` for services that provide synchronization.

   If this service provides synchronization, add the following nodes to `<appSettings>`:

   - `<add key="SyncProviderAssembly" value="`*`assembly name`*`" />`

     Where *assembly name* is the name of the assembly containing your `ISyncProvider` implementation.

   - `<add key="SyncProviderImpl" value="`*`fully qualified path`*`" />`

     Where *fully qualified path* is the fully qualified path to the class implementing `ISyncProvider.`

   For example:

```
<appSettings>

...

    <add key="SyncProviderAssembly" value="Helloworld" />

    <add key="SyncProviderImpl"
value="Plumtree.Remote.Auth.Helloworld.Sync" />

...
```

4. Update `Web.config` for services that use SCI.

   If this service uses SCI, add the following nodes to `<appSettings>`:

   - `<add key="AdminEditorAssembly" value="`*`assembly name`*`" />`

     Where *assembly name* is the name of the assembly containing your `IAdminEditor` implementation.

   - `<add key="AdminEditorImpl" value="`*`fully qualified path`*`" />`

     Where *fully qualified path* is the fully qualified path to the class implementing `IAdminEditor`.

Web Service Class Names (*Impl)

To deploy a content service, identity service or SCI page, you must enter the qualified class name in the web.xml (Java) or Web.config (.NET) file for your project.

For Java services, add your class to the web.xml for your project. The `*Impl` key in the web.xml file must reference the fully-qualified name of the class. For content services and identity services, you must enter the name of both provider classes required by the service. If the service uses SCI, enter the fully-qualified name of the appropriate implementation of the `IAdminEditor` interface. For example, for an authentication service, add your class to `AuthProviderImpl` as shown below.

```
...
<env-entry>
<env-entry-name>AuthProviderImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.auth.helloworld.Auth</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
</env-entry>
...
```

For .NET services, add your class to the Web.config for your project. Some services also require an entry for the assembly. For example, for an authentication service, add your assembly to `AuthProviderAssembly` and your class to `AuthProviderImpl` as shown below.

Note: When you create a new project, Visual Studio sets the root namespace to the name of the project.This name is prepended to any namespace you define in the project.The `*Impl` key in the web.config file must reference the full namespace. (To view the root namespace, right-click

the project name in Solution Explorer and click Properties.If you clear the root namespace field, the change will not be reflected until the project is rebuilt.)

```
...
<appSettings>
<add key="AuthProviderAssembly" value="dotnetauthsample"/>
<add key="AuthProviderImpl" value="dotnetauthsample.AuthSample"/>
</appSettings>
...
```

The table that follows provides the required `*Impl` parameters and associated service types.

Service Type	***Impl Parameters**
Authentication Service	AuthProviderImpl, SyncProviderImpl
Content Crawler	ContainerProviderImplDocumentProviderImplDocFetchImpl (.NET only) Note: Deploying DocFetch requires additional parameters.
Profile Service	ProfileProviderImpl
Federated Search	SearchImpl
Service Configuration Interface (SCI)	SciImpl

.NET services also require an entry for the associated assembly. The table below provides the required `*Assembly` parameters and associated service types.

Service Type	***Assembly Parameters**
Authentication Service	AuthProviderAssembly SyncProviderAssembly
Content Crawler	ContainerProviderAssemblyDocumentProviderAssemblyDocFetchAssembly
Profile Service	ProfileProviderAssembly
Federated Search	SearchAssembly
Service Configuration Interface (SCI)	AdminEditorAssembly

Configuring an Authentication Service

To deploy an authentication service in the portal, you must configure a set of portal objects.

This topic describes how to configure portal objects in order to register your authentication service in the portal. The steps are specific to the needs of authentication services and assume that you are familiar with creating and configuring portal objects. For more details on portal objects, see the portal online help.

1. Create and configure a **Remote Service** object.

   This is optional. Configuring a Remote Service object allows multiple services to share a single remote service configuration. Authentication Web Services can use either a Remote Service object or hard-coded URLs.

2. Create and configure a **Web Service — Authentication**.

   Each remote authentication service must have an associated Authentication Web Service object. The Authentication Web Service editor allows you to specify general settings for the back-end system.

   The following settings are necessary for Authentication Services:

   - The **encoding style** must reflect the service implementation (.NET vs Java). The encoding style is set on the **Advanced Settings page**. For .NET, you must set the encoding to `Document/Literal`. Java uses the default, `RPC/Encoded`.
   - All configuration pages must be entered on the **Advanced URLs** page. You can add configuration pages to the **Authentication Source** editor. These URLs must be entered on the **Advanced URLs** page.

3. Create and configure an **Authentication Source — Remote**

   Each Authentication Web Service has one or more associated Remote Authentication Source objects that define basic settings.

   Keep the following in mind when configuring the Authentication Source:

   - **Users imported by a synchronization service must be unique by name and Authentication Source.** The portal identifies users first by their category, then by username; this combination must be unique per user. It is a best practice to use the source domain for the category name. The category is entered in the Authentication Source editor. You can use the same category for multiple back-end systems, but the systems must not have users or groups with the same name.

- **The description of the Authentication Source object is displayed on the portal login page.** Creating an Authentication Source object with a synchronization component creates an option in the authentication source drop-down list on the portal login page. The name that appears in the drop-down list is the description of the Authentication Source object. Enter a description that users will recognize.
- **By default, the portal performs partial users synchronization.** Confirm that the synchronization settings are correct for the service. The default of Partial User Synchronization may not perform the synchronization you desire.

4. Create a configure a **Job**.

   To run the authentication service, you must schedule a job or add the Authentication Source to an existing job. The **Remote Authentication Source editor** allows you to set a job.

About Profile Services (PWS)

Profile services are used to import information about existing portal users from external systems. This information is mapped to portal properties and made available to other services.

Synchronization

The purpose of a profile service is to import information about portal users from an external system into the portal so that the information can be used by the portal and other services. The first step is to synchronize the user information in the external system with existing users in the portal; this is the process that must be handled by the remote service. As with authentication services, the synchronization process is handled by the portal Automation Server, as scheduled in the Job associated with the Profile Source object in the portal.

Property Mapping: User Information

The profile information imported by the profile service must be associated with portal properties so that it can be accessed by portal objects and other remote services.

Development

The following topics provide detailed instructions on developing custom profile services:

- *Profile Service Internals* on page 411 describes the interfaces that must be implemented when creating a profile service, and how the interfaces will be called by the portal.
- *Implementing an Authentication Service* on page 394 provides step by step instructions on implementing the required interfaces, with example code.
- *Deploying a Java Authentication Service* on page 403 and *Deploying a .NET Authentication Service* on page 405 describe how to deploy the profile service to a Java or .NET application server.
- *Configuring an Profile Service* on page 423 describes how to configure the profile service in the portal.

Profile Service Internals

The following interfaces must be implemented when creating a profile service.

The AquaLogic Interaction Development Kit (IDK) allows you to create remote profile services and related configuration pages without parsing SOAP or accessing the portal API. The IDK Profile API provides an abstraction from the necessary SOAP calls; you simply implement an object interface. For a complete listing of interfaces, classes, and methods, see the IDK API documentation.

Note: The differences between the Java and .NET versions of the IDK are platform-specific. In this guide, method names are listed using the .NET standard of initial capitalization. Java methods are identical, except begin with a lowercase letter. The `IProfileProvider.Initialize` method in the .NET IDK provides the same functionality as the `IProfileProvider.initialize` method in the Java IDK.

Plumtree.Remote.Profile

The `Plumtree.Remote.Profile` namespace (`plumtree.remote.profile` package in Java) provides the following interfaces:

- IProfileProvider
- IUser

To import information from an external source into portal user properties, you must implement both interfaces.

Profile Synchronization

The portal accesses a remote profile service when the associated job is run by the portal Automation Service. The portal calls the methods of the profile service interfaces in the following order:

1. `IProfileProvider.Initialize()`
2. `IProfileProvider.GetGlobalSignature()`
3. `IProfileProvider.AttachToUser()`
   a. `IUser.GetUserSignature()`
   b. `IUser.GetUserProperties()`
4. `IProfileProvider.Shutdown()`

Step 3 is called by the portal once for every user in the group or groups to be synchronized. Step 4, the `IProfileProvider.Shutdown()` method is optional. It can be used to clean up resources used by the profile service; however, it may or may not be called by the portal, so it should not be relied upon.

Implementing a Profile Service

To implement a profile service, follow these step by step instructions.

This section describes the details of how to create a profile service by taking you step by step through the implementation of a sample profile service. This is a very simple profile service, and is not intended for use in a production environment.

The functional requirement for this profile service is simply:

- For any request, return the user's profile property "REGION" set to "WEST".

The following interfaces will be implemented to create this profile service:

- IProfileProvider
- IUser

These interfaces are in the `Plumtree.Remote.Profile` namespace (C#) or the `plumtree.remote.profile` package (Java). Any exceptions thrown in this code can be found in `Plumtree.Remote` (C#) or `plumtree.remote` (Java).

For the complete code of this example, see the sample code on the Developer Center.

For details of the classes described in this section, see the IDK API documentation for .NET or Java.

Except for in the example Java code, this section uses C# method names. In the IDK, methods are named the same in C# or Java, except for leading letter capitalization. For example, `IProfileProvider.GetGlobalSignature()` in the C# API is `IProfileProvider.getGlobalSignature()` in the Java API.

1. Implement the `IProfileProvider` Interface

   The `IProfileProvider` interface is used by the portal to initiate access to, and obtain user profile information from, the back-end repository. The portal calls the methods of `IProfileProvider` in the following order:

   1. `IProfileProvider.Initialize()`
   2. `IProfileProvider.GetGlobalSignature()`
   3. `IProfileProvider.AttachToUser()`
   4. `IProfileProvider.Shutdown()`

   To implement the `IProfileProvider` interface:

   a) Implement `IProfileProvider.Initialize`

      `Initialize()` allows the profile service to initialize a session and create a connection to the back-end repository. The method is passed two objects from the portal: `PropertyList` and `ProfileInfo`. `PropertyList` is the list of attributes mapped to properties on the **Property Map** page of the Profile Source object in the portal. `ProfileInfo` is a set of name-value pairs, populated with information entered in the portal **Service Configuration Interface (SCI)** editor by a portal administrator. Typically, this is information such as credentials for connecting to the back-end system.

      Java:

```
private String[] m_propertyList;
protected String[] getPropertyList()
{
    return m_propertyList;
}
public void initialize(String[] propertyList,
                ProfileInfo profileInfo)
  throws ServiceException
{
    this.m_propertyList = propertyList;
}
```

      C#:

```
private string[] m_propertyList;
internal string[] GetPropertyList()
{
   return m_propertyList;
}
public void Initialize(string[] PropertyList,
                ProfileInfo ProfileSourceInfo)
```

```
{
    this.m_propertyList = PropertyList;
}
```

In this example profile service, the `PropertyList` is stored in the *m_propertyList* member variable, where it can later be accessed by the `IUser` implementation. The `IUser` implementation requires the `PropertyList` to determine which properties to retrieve.

b) Implement `IProfileProvider.GetGlobalSignature`

`GetGlobalSignature()` allows the portal to determine whether profile information for any of the users has changed. The portal compares the string returned from `GetGlobalSignature()` with the string returned from `GetGlobalSignature()` during the previous run of the profile service job. If the two strings match, the job stops. The IDK does not enforce any restrictions on the string used for the global signature; it can be a last-modified date, a random number, or another identifier.

Java:

```
public String getGlobalSignature()
  throws ServiceException
{
    return new Date().toString();
}
```

C#:

```
public string GetGlobalSignature()
{
    return System.DateTime.Now.Ticks.ToString();
}
```

This profile service returns a string representation of the current date and time. This ensures the job will continue to `AttachToUser()`.

c) Implement `IProfileProvider.AttachToUser`

`AttachToUser()` is called for each user within the group or groups configured in the Profile Source editor in the portal. The first three parameters passed to `AttachToUser()` identify which user the profile service should retrieve from the back-end system:

- UserID: The portal user ID. Can be used via the PRC to look up other user attributes.
- LoginName: The portal login name. If this user was added using an authentication service, this value corresponds to ChildUser.UserName.

- UniqueName: Usually the name used to look up the user in the back-end system. If the user was added using an authentication service, this value corresponds to ChildUser.UserUniqueName.

If these parameters do not identify a valid user, a `NoSuchUserException` should be thrown.

The final parameter, LastSignature, is the signature returned by `IUser.GetUserSignature()` during the previous job.

Java:

```
public IUser attachToUser(int userId, String loginName,
                String uniqueName, String lastSignature)
  throws ServiceException
{
    return new User(this);
}
```

C#:

```
public IUser AttachToUser(int UserID, string LoginName,
                string UniqueName, string LastSignature)
{
    return new User(this);
}
```

For this profile service, `AttachToUser()` simply returns an instance of `User`, the `IUser` implementation. This is appropriate for this implementation because the profile service updates the "Region" property to "WEST" regardless of the portal user being queried.

d) Implement `IProfileProvider.Shutdown`

As a performance optimization, the portal might call the `Shutdown()` method. No parameters are received or returned. This method is optional on both ends; the profile service might not receive the Shutdown message, and, if received, the profile service can ignore the call to `Shutdown()`.

`Shutdown()` can be used to clean up resources used by the profile service; however, you should not rely on it being called.

2. Implement the `IUser` Interface

Returned by `IProfileProvider.AttachToUser()`, the `IUser` interface is used by the portal to synchronize profile property information for a specific user. The portal calls the methods of `IUser` in the following order:

1. `IUser.GetUserSignature()`
2. `IUser.GetUserProperties()`

Note: For this profile service, the `IUser` interface implementation, `User`, has a constructor that accepts a parameter of type `Profile` (the `IProfileProvider` implementation), which is stored in the private member *m_profile*. This is variable is used in `GetUserProperties()` to access the `PropertyList`.

a) Implement `IUser.GetUserSignature`

`GetUserSignature()` is similar to `IProfileProvider.GetGlobalSignature()`, except it allows the portal to determine if a specific user's profile information has changed. If the returned string matches the string returned from `GetUserSignature()` during the previous job, `GetUserProperties()` is not called. The IDK does not enforce any restrictions on the string used for the global signature; it can be a last-modified date, a random number, or another identifier.

Java:

```
public String getUserSignature()
  throws ServiceException
{
    return new Date().toString();
}
```

C#:

```
public string GetUserSignature()
{
    return System.DateTime.Now.Ticks.ToString();
}
```

This profile service returns a string representation of the current date and time. This ensures `GetUserProperties()` is called each time the job runs.

b) Implement `IUser.GetUserProperties`

Typically, `GetUserProperties()` will access the `PropertyList` object from the `IProfileProvider` implementation, retrieving values for each property in the `PropertyList` from the back-end system. `GetUserProperties` then builds a

`UserPropertyInfo` object to return to the portal. The portal maps the back-end property names with portal properties and updates the portal property values.

Java:

```
public UserPropertyInfo getUserProperties()
  throws ServiceException
{
    String prop;
    UserPropertyInfo info = new UserPropertyInfo();
    for (int i=0; i < m_profile.getPropertyList().length;
i++)
    {
      prop = m_profile.getPropertyList()[i];
      if (prop.equalsIgnoreCase("REGION"))
      {
        info.put(prop, "WEST");
      }
    }
    return info;
}
```

C#:

```
public UserPropertyInfo GetUserProperties()
{
    String prop;
    UserPropertyInfo info = new UserPropertyInfo();
    for (int i=0; i < m_profile.GetPropertyList().Length;
i++)
    {
       prop = m_profile.GetPropertyList()[i];
       if (prop.ToUpper().Equals("REGION"))
       {
         info.Put(prop, "WEST");
       }
    }
    return info;
}
```

In this profile service, a for loop checks each property in the `PropertyList` against the string "REGION". If "REGION" is found, a `UserPropertyInfo` object is updated with name "REGION" and value "WEST" and returned to the portal.

Deploying a Java Profile Service

To deploy an authentication service to a Java application server, follow these steps.

- The IDK must be installed on the server to which you intend to deploy.
- You must have implemented IProfileProvider and IUser. For details, see *Implementing a Profile Service* on page 412.

This topic describes how to deploy an Authentication Service implemented in Java to a Java application server.

To deploy an authentication service to a supported Java application server:

1. Access the IDK deployment servlet (DeployServlet) in a browser.

   The IDK deployment servlet is located at `http://app server:port/edk/DeployServlet`
2. Choose **Profile** and wait for the page to reload.
3. Enter a prefix to identify this authentication service and the fully qualified name of the implementation of IAuthProvider, ISyncProvider, or both.
4. If this service uses SCI, check **Use Service Configuration Interface (SCI)** and enter the fully qualified name of the appropriate implementation of `IAdminEditor`.
5. Copy and paste the URLs displayed on the results page to a text file; these are the URLs that should be used when you configure the service in the portal.

Web Service Class Names (*Impl)

To deploy a content service, identity service or SCI page, you must enter the qualified class name in the web.xml (Java) or Web.config (.NET) file for your project.

For Java services, add your class to the web.xml for your project. The `*Impl` key in the web.xml file must reference the fully-qualified name of the class. For content services and identity services, you must enter the name of both provider classes required by the service. If the service uses SCI, enter the fully-qualified name of the appropriate implementation of the `IAdminEditor` interface. For example, for an authentication service, add your class to `AuthProviderImpl` as shown below.

```
...
<env-entry>
<env-entry-name>AuthProviderImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.auth.helloworld.Auth</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
```

```
</env-entry>
...
```

For .NET services, add your class to the Web.config for your project. Some services also require an entry for the assembly. For example, for an authentication service, add your assembly to `AuthProviderAssembly` and your class to `AuthProviderImpl` as shown below.

Note: When you create a new project, Visual Studio sets the root namespace to the name of the project.This name is prepended to any namespace you define in the project.The `*Impl` key in the web.config file must reference the full namespace. (To view the root namespace, right-click the project name in Solution Explorer and click Properties.If you clear the root namespace field, the change will not be reflected until the project is rebuilt.)

```
...
<appSettings>
<add key="AuthProviderAssembly" value="dotnetauthsample"/>
<add key="AuthProviderImpl" value="dotnetauthsample.AuthSample"/>
</appSettings>
...
```

The table that follows provides the required `*Impl` parameters and associated service types.

Service Type	***Impl Parameters**
Authentication Service	AuthProviderImpl, SyncProviderImpl
Content Crawler	ContainerProviderImplDocumentProviderImplDocFetchImpl (.NET only) Note: Deploying DocFetch requires additional parameters.
Profile Service	ProfileProviderImpl
Federated Search	SearchImpl
Service Configuration Interface (SCI)	SciImpl

.NET services also require an entry for the associated assembly. The table below provides the required `*Assembly` parameters and associated service types.

Service Type	***Assembly Parameters**
Authentication Service	AuthProviderAssembly SyncProviderAssembly

Service Type	*Assembly Parameters
Content Crawler	ContainerProviderAssemblyDocumentProviderAssemblyDocFetchAssembly
Profile Service	ProfileProviderAssembly
Federated Search	SearchAssembly
Service Configuration Interface (SCI)	AdminEditorAssembly

Deploying a .NET Profile Service

To deploy an authentication service to IIS and .NET, follow these steps.

- The IDK must be installed on the server to which you intend to deploy.
- You must have implemented IProfileProvider and IUser. For details, see *Implementing a Profile Service* on page 412.

This topic describes how to deploy an Authentication Service implemented in Java to a Java application server.

To deploy an authentication service to IIS and .NET:

1. Ensure that you have built your project with the `ProfileProviderSoapBinding.asmx` SOAP endpoint.

   If this service uses SCI, also include SCIProviderBinding.asmx. These files can be found in your IDK installation.

   The IDK installation is typically installed to `C:\Program Files\plumtree\ptedk\5.3\devkit\`

2. Update `Web.config` for profile services.

   Add the following nodes to `<appSettings>`:

   - `<add key="ProfileProviderAssembly" value="`*`assembly name`*`" />`

     Where *assembly name* is the name of the assembly containing your `IProfileProvider` implementation.

   - `<add key="ProfileProviderImpl" value="`*`fully qualified path`*`" />`

Where *fully qualified path* is the fully qualified path to the class implementing `IProfileProvider`.

For example:

```
<appSettings>

    <add key="ProfileProviderAssembly" value="HelloWorldProf_CS"
 />

    <add key="ProfileProviderImpl"
value="HelloWorldProf_CS.Profile" />

...
```

3. Update `Web.config` for services that use SCI.

   If this service uses SCI, add the following nodes to `<appSettings>`:

   - `<add key="AdminEditorAssembly" value="`*`assembly name`*`" />`

     Where *assembly name* is the name of the assembly containing your `IAdminEditor` implementation.

   - `<add key="AdminEditorImpl" value="`*`fully qualified path`*`" />`

     Where *fully qualified path* is the fully qualified path to the class implementing `IAdminEditor`.

Web Service Class Names (*Impl)

To deploy a content service, identity service or SCI page, you must enter the qualified class name in the web.xml (Java) or Web.config (.NET) file for your project.

For Java services, add your class to the web.xml for your project. The `*Impl` key in the web.xml file must reference the fully-qualified name of the class. For content services and identity services, you must enter the name of both provider classes required by the service. If the service uses SCI, enter the fully-qualified name of the appropriate implementation of the `IAdminEditor` interface. For example, for an authentication service, add your class to `AuthProviderImpl` as shown below.

```
...
<env-entry>
<env-entry-name>AuthProviderImpl</env-entry-name>
<env-entry-value>com.plumtree.remote.auth.helloworld.Auth</env-entry-value>
```

```
<env-entry-type>java.lang.String</env-entry-type>
</env-entry>
...
```

For .NET services, add your class to the Web.config for your project. Some services also require an entry for the assembly. For example, for an authentication service, add your assembly to `AuthProviderAssembly` and your class to `AuthProviderImpl` as shown below.

Note: When you create a new project, Visual Studio sets the root namespace to the name of the project.This name is prepended to any namespace you define in the project.The `*Impl` key in the web.config file must reference the full namespace. (To view the root namespace, right-click the project name in Solution Explorer and click Properties.If you clear the root namespace field, the change will not be reflected until the project is rebuilt.)

```
...
<appSettings>
<add key="AuthProviderAssembly" value="dotnetauthsample"/>
<add key="AuthProviderImpl" value="dotnetauthsample.AuthSample"/>
</appSettings>
...
```

The table that follows provides the required `*Impl` parameters and associated service types.

Service Type	***Impl Parameters**
Authentication Service	AuthProviderImpl, SyncProviderImpl
Content Crawler	ContainerProviderImplDocumentProviderImplDocFetchImpl (.NET only) Note: Deploying DocFetch requires additional parameters.
Profile Service	ProfileProviderImpl
Federated Search	SearchImpl
Service Configuration Interface (SCI)	SciImpl

.NET services also require an entry for the associated assembly. The table below provides the required `*Assembly` parameters and associated service types.

Service Type	***Assembly Parameters**
Authentication Service	AuthProviderAssembly SyncProviderAssembly

Service Type	*Assembly Parameters
Content Crawler	ContainerProviderAssemblyDocumentProviderAssemblyDocFetchAssembly
Profile Service	ProfileProviderAssembly
Federated Search	SearchAssembly
Service Configuration Interface (SCI)	AdminEditorAssembly

Configuring an Profile Service

To deploy a profile service in the portal, you must configure a set of portal objects.

This topic describes how to configure portal objects in order to register your authentication service in the portal. The steps are specific to the needs of authentication services and assume that you are familiar with creating and configuring portal objects. For more details on portal objects, see the portal online help.

1. Create and configure a **Remote Service** object.

   This is optional. Configuring a Remote Service object allows multiple services to share a single remote service configuration. Profile Web Services can use either a Remote Service object or hard-coded URLs.

2. Create and configure a **Web Service — Profile**.

   Each remote authentication service must have an associated Profile Web Service object. The Authentication Web Service editor allows you to specify general settings for the back-end system.

   The following settings are necessary for Authentication Services:

   - The **encoding style** must reflect the service implementation (.NET vs Java). The encoding style is set on the **Advanced Settings page**. For .NET, you must set the encoding to `Document/Literal`. Java uses the default, `RPC/Encoded`.
   - All configuration pages must be entered on the **Advanced URLs** page. You can add configuration pages to the **Profile Source** editor. These URLs must be entered on the **Advanced URLs** page.

3. Create and configure a **Profile Source — Remote**

Each Profile Web Service has one or more associated Remote Profile Source objects that define basic settings.

4. Create and configure **Property** objects.

   To create new user information properties, you must first create a **Property** object using the **Property editor.**

5. Configure the **Global Object Property Map**.

   The **Global Object Property Map** displays the types of portal objects with which you can associate properties. Values for a portal object's associated properties are specified on the **Properties and Names** page of the object's editor.

   To import new user information properties into the portal, you must add mappings to the **Global Object Property Map**. Each property must be mapped to the User object.

   To create a property in the portal, choose Create Object... | Property in portal Administration. After you have created a property, you can add it to the Global Object Property Map.

   For more details on the Global Object Property Map, see the portal online help.

6. Associate user information properties with portal user profiles.

   When a profile service imports user information into the portal, the attributes imported must be associated with portal properties. To make these properties available to other services, you must associate them with user information using the **User Profile Manager.**

   On the **User Information Property Map** page of the **User Profile Manager**, add any properties that should be associated with user information settings. (The properties must already exist in the portal and be associated with the User object in the **Global Object Property Map** .)

   For details on the **User Profile Manager**, see the portal online help.

7. Create a configure a **Job**.

   To run the authentication service, you must schedule a job or add the Profile Source to an existing job. The **Remote Profile Source editor** allows you to set a job.

About AquaLogic Interaction 6.5 UI Customization

AquaLogic Interaction 6.5 introduces a selection of new UI customization options. The following topics provide information and detailed instructions.

- *About Adaptive Page Layouts (ALI 6.5)* on page 426: Adaptive page layouts allow you to change the look and feel of the portal user interface using adaptive tags in standard XHTML.
- *About ALI 6.5 Adaptive Styles (CSS Customization)* on page 445: In AquaLogic Interaction 6.5, the available CSS UI customization options have been expanded and restructured. These topics provide information and detailed instructions on implementing UI customization through CSS.
- For an introduction to basic UI customization in all 6.x versions, see *Basic Portal UI Customizations.*

The following legacy customization options are still available. (The options above provide more efficient customization design and implementation.)

- *Advanced Portal UI Customizations*: These advanced customizations require Java or C# coding.
- *Portal Component Replacement*: These advanced customizations require extensive coding. modification of portal source code, and advanced implementation.
- For links to portal UI API documentation for advanced portal customizations that require coding against the internal portal API, see *Portal API Documentation*.

Note: The UI Customization Installer (UICI) is no longer available in ALI 6.5. To download the portal source code, go to the ALI 6.5 product download page on commerce.bea.com and select "Java and .NET Source Code" from the Operating System drop-down list.

About Adaptive Page Layouts (ALI 6.5)

Adaptive page layouts allow you to change the look and feel of the portal user interface using adaptive tags in standard XHTML.

Adaptive page layouts can be used in most areas of the portal. Each page layout type has an associated tag library. The tags in each library only work in the related page layout. The base page library and the legacy adaptive tag libraries (pt:common, pt:core, pt:logic, pt:ptdata, pt:ptui, pt:standard and pt:transformer) work on any page with a portal banner and are used in all page layouts. The following table summarizes the available page layout types:

Page Layout Type	Description	Library	More Information
Base Page	Controls the layout for components that are common to each page (header, footer, navigation).	pt:basepage and legacy libraries	*Creating a Base Page Adaptive Page Layout* on page 427
Profile Page	Controls the layout for components that are common to each user profile page (header, footer, navigation).	pt:basepage and legacy libraries	
Portlet Page	Controls the layout for the portlet area of the portal page.	pt:portletpage and legacy libraries	*Creating a Portlet Adaptive Page Layout* on page 429
Knowledge Directory	Controls the layout for the content area of the Knowledge Directory.	pt:kdpage and legacy libraries	*Creating a Knowledge Directory Adaptive Page Layout* on page 434
Search Results	Controls the layout for the content area of search results.	pt:searchpage and legacy libraries	*Creating a Search Results Adaptive Page Layout* on page 440

Page Layout Type	Description	Library	More Information
Portlet Selection	Controls the layout to use for the pop-up or fly-out editor used to select portlets on a portlet page. **Note:** In most implementations, there is no reason to modify this page layout.	pt:portletpageeditor and legacy libraries	

- For an introduction to adaptive tags, see *About Adaptive Tags* on page 202.
- For a complete list of tags, see the 6.5 tagdocs (*http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/index.html*).
- For details on configuring adaptive page layouts in the portal, see the *AquaLogic Interaction Administrator Guide*.
- For complete examples of each page layout type, see the default page layouts on the ALI image server in the \imageserver\plumtree\portal\private\pagelayouts folder.

Creating a Base Page Adaptive Page Layout

The base page adaptive layout defines components that are common to each page (header, footer, navigation, content area). The profile page layout is almost identical, but uses a user profile search box instead of the standard search box.

The pt:basepage library includes tags to display all common sections of the portal page.

- The `title` tag displays the title of the page.
- The `pagebody` tag displays the html body tag and initializes the JavaScript required by the page.
- The `navarea` tag is used to display legacy portal navigation components in the top bar. The `pt:area` parameter defines which part of the navigation is being defined by each section (ABOVEHEADER, BELOWHEADER, ABOVEBODY, LEFTOFBODY, RIGHTOFBODY, BELOWBODY, ABOVEFOOTER, BELOWFOOTER, and TOPBAR). You can also create a customized display of links to portal components using tags from the pt:ptdata library.
- The `content` tag defines the section where the main content of the page is displayed.

- The `header` and `footer` tags define the sections of the page where the header and footer are displayed. The header and footer are implemented in separate HTML files. For examples, see the header.html and footer.html files in the \imageserver\plumtree\portal\private\pagelayouts folder on your ALI image server.

The example below uses tags from the pt:basepage library to define common page components, and legacy adaptive tags to implement portal links and handle logic.

- For an overview of adaptive tags, see *About Adaptive Tags* on page 202.
- For an introduction to data tags for implementing portal links, see *Data Adaptive Tag Library (pt:ptdata)* on page 238.
- For an introduction to logic tags, see *Logic Adaptive Tag Library (pt:logic)* on page 227.
- For a list of tags in the pt:basepage library, see *http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/basepage/tld-summary.html*. For a complete list of tags, see the 6.5 tagdocs (*http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/index.html*).

Note: This example has been oversimplified for illustration purposes; for a complete implementation of the base page layout, see the basepagelayout.html file in the \imageserver\plumtree\portal\private\pagelayouts folder on your ALI image server.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:pt="http://www.plumtree.com/xmlschemas/ptui/">
<head>
<title><pt:basepage.title/></title>
<pt:standard.stylesheets/>
<link href="pt://images/plumtree/portal/private/css/mainstyle.css"
 rel="stylesheet" type="text/css" />

<!-- Dojo Initialization, create a custom ali namespace -->
<script type="text/javascript">
        var djConfig = {
            isDebug: false,
            scopeMap :  [ ["dojo", "alidojo"] ]
         };
</script>

<!-- Dojo is used by drag and drop and flyout editor -->
<script type="text/javascript"
src="pt://images/plumtree/portal/private/js/dojo.js"></script>

<!-- This tag displays the html body tag and initializes page
JavaScript. -->
```

```
<pt:basepage.pagebody marginwidth="0" marginheight="0"
topmargin="0">
<div id="ali-header-nav">

<!-- This area is used to build links to portal components
displayed in the top bar -->
<pt:basepage.navarea pt:area="TOPBAR"/>

... banner actions ....

<pt:basepage.navarea pt:area="ABOVEHEADER"/>
<pt:basepage.header/>

<!-- This area is used to build links to portal navigation elements
 -->
<pt:basepage.navarea pt:area="BELOWHEADER"/>

... navigation links ...

  <pt:basepage.navarea pt:area="LEFTOFBODY"/>
</div>
<div style="float:right; width:200px;" >
  <pt:basepage.navarea pt:area="RIGHTOFBODY"/>
</div>
<pt:basepage.navarea pt:area="ABOVEBODY"/>
<pt:common.error/>
<pt:basepage.content/>
<pt:basepage.navarea pt:area="BELOWBODY"/>
<pt:basepage.navarea pt:area="ABOVEFOOTER"/>
<pt:basepage.footer/>
<pt:basepage.navarea pt:area="BELOWFOOTER"/>
</pt:basepage.pagebody>
</html>
```

For details on configuring adaptive page layouts in the portal, see the *AquaLogic Interaction Administrator Guide*.

Creating a Portlet Adaptive Page Layout

Portlet page adaptive layouts allow you to customize the layout of the portlet header as well as how individual portlets are displayed. The outer area (header, footer, navigation areas) is based on the Base Page adaptive layout.

The pt:portletpage library includes tags to define all areas of the portlet section of the portal page.

- All portlets on the page must be within a `portletregiondisplay` tag. This tag defines a container for portlets that specifies where and how portlets inside the region are displayed.
- The `portletdisplay` tag displays an entire portlet (header and content), while the `portletdisplaycontent` tag displays the content of the portlet without the header.
- Portlet data is accessed via the `portletregiondata` tag. The `pt:region` parameter can be set to any value, as long as it corresponds with a `portletregiondisplay` section in the page. The following properties are available for each portlet from the `portletregiondata` tag:

Property	Description
name	The name of the portlet
objid	The ID of the portlet object in the portal.
index	The index of the portlet in the portlet array.
portletidstring	The string identifier of the portlet.
adminprefurl	A link to the associated administrative preference page for the portlet if one exists.
commprefurl	A link to the associated community preference page for the portlet if one exists.
userprefurl	A link to the associated user preference page for the portlet if one exists.
helpurl	A link to the associated help page for the portlet if one exists.
collapseexpandurl	A link for collapsing or expanding the portlet depending on the collapse state of the portlet
iscollapsed	True if the portlet is collapsed, false if it is expanded.
removeurl	A link to remove the portlet from the current page.
hastitlebar	True if portlet title bar is not suppressed, false otherwise.

- The `pagenamebreadcrumbsdata` tag can be used on MyPages and Community Pages (Knowledge Directory adaptive page layout). This tag contains an ordered list representing

the path leading up to the current page. This list contains the name and URL that makes up each breadcrumb.

- On a MyPage, the breadcrumb will display a "Home" link as the parent that leads back to the user's main page, followed by the name of the MyPage that the user is on.
- In Communities, the breadcrumb will display all the Communities/SubCommunities leading up to the page. The breadcrumbs will all be hyperlinks except for the last breadcrumb which represents the current page that the user is on.

Additional tags in the pt:portletpage library can be used to display buttons with access to portlet-specific functionality, including refresh, remove, and collapse/expand.

The example below uses tags from the pt:portletpage library to define portlet page components, and legacy tags to implement portal links and navigation and handle logic.

- For an overview of adaptive tags, see *About Adaptive Tags* on page 202.
- For an introduction to logic tags, see *Logic Adaptive Tag Library (pt:logic)* on page 227.
- For an introduction to data tags for implementing portal links, see *Data Adaptive Tag Library (pt:ptdata)* on page 238.
- This example uses the pt:portletpageeditor.addportletsflyoutdata library to add the portlet flyout editor to the portlet page. The pt:portletpageeditor.sortpropertiesdata tag specifies the portlet sort order. For details on these tags, see the tagdocs.
- For a list of tags in the pt:portletpage library, see *http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/portletpage/tld-summary.html*. For a complete list of tags, see the 6.5 tagdocs (*http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/index.html*).

Note: This example has been oversimplified for illustration purposes; for a complete implementation of the portlet page layout, see the portletdefaultlayout.html file in the \imageserver\plumtree\portal\private\pagelayouts folder on your ALI image server.

```
<!-- Portlet Content Area Begin -->
<span xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>

<pt:portletpage.portletregiondata pt:key="region1" pt:region="1"
 />
<pt:portletpage.portletregiondata pt:key="region2" pt:region="2"
 />
<pt:portletpage.portletregiondata pt:key="region3" pt:region="3"
 />

... breadcrumb display ...
```

```
<!-- Start Page Action Buttons -->
<pt:core.comment><!-- get the Portlet Selection Flyout URL
--></pt:core.comment>
<pt:portletpageeditor.sortpropertiesdata pt:id="sortprops"
pt:scope="session" pt:defaultsort="1"/>

<pt:portletpageeditor.addportletsflyoutdata pt:id="flyoutLink"
pt:sortprops="sortprops" pt:propscope="session" />
<pt:logic.existexpr pt:data="flyoutLink" pt:key="hasFlyout"/>
<pt:logic.if pt:expr="$hasFlyout">
    <pt:logic.iftrue>
     <pt:portletpageeditor.flyoutjs pt:flyoutID="portletSelection"
 pt:onclick="openFlyout" pt:headerId="ali-header-nav"
pt:url="$flyoutLink.url" pt:anchorId="ali-pageEdit-anchor"
pt:flyoutAnchorText="$#130.ptmsgs_portalinfrastructure"
pt:flyinAnchorText="$#301.ptmsgs_portalcommonmsgs"/>
    </pt:logic.iftrue>
</pt:logic.if>

<pt:core.comment><!-- add javascript for collapsing and expanding
 portlets --></pt:core.comment>
<pt:portletpageeditor.collapseexpandjs
pt:onclick="sendCollapseExpandRequest"
pt:flyoutID="portletSelection1" />

... page action implementation ....

<table width="100%" height="100%" style="clear:left;">
    <tr>
        <td class="columnOne" valign="top">
  <!-- Vertical Region One -->
  <pt:portletpage.portletregiondisplay class="portletRegion"
pt:region="1" pt:direction="v">
    <pt:logic.foreach pt:data="region1" pt:var="curr">
      <pt:logic.variable pt:key="titleBarData"
pt:value="$curr.hastitlebar"/>
      <pt:logic.stringexpr pt:expr="($titleBarData) == false"
pt:key="suppressTitleBar" />

      <pt:logic.variable pt:key="portletDivStyle"
pt:value="ali-portlet-regionone "/>
      <pt:logic.if pt:expr="$curr.ismandatory">
      <pt:logic.iffalse>
        <pt:logic.concat pt:key="portletDivStyle"
```

```
pt:value1="$portletDivStyle" pt:value2="dndPortlet"/>
      </pt:logic.iffalse>
      </pt:logic.if>

      <pt:core.html pt:tag="div" class="$portletDivStyle"
id="$curr.portletidstring">


     <pt:logic.if pt:expr="$curr.iscollapsed"><pt:logic.iffalse>

        <pt:logic.variable pt:key="containerclass"
pt:value="ali-portlet-container"/>
      </pt:logic.iffalse><pt:logic.iftrue>
        <pt:logic.variable pt:key="containerclass"
pt:value="ali-portlet-container-collapsed"/>
      </pt:logic.iftrue></pt:logic.if>
        <pt:core.html pt:tag="div" class="$containerclass">
        <pt:logic.if pt:expr="$suppressTitleBar">
        <pt:logic.iftrue>
          <!-- Suppress portlet toolbar -->
          <div class="ali-portlet-toolbar">
            <div class="ali-portlet-cornerleft"></div>
            <div class="ali-portlet-cornerright"></div>
          </div>
        </pt:logic.iftrue>
        <pt:logic.iffalse>
        <!-- Display Portlet Header -->
        <div class="ali-portlet-toolbar">
          <div class="ali-portlet-cornerleft"></div>
          <div class="ali-portlet-cornerright"></div>
          <div
class="ali-portlet-controlone"><pt:portletpage.portletremovebuttondisplay
 pt:datavar="curr" pt:scope="tag"/></div>
          <div
class="ali-portlet-controlone"><pt:portletpage.portletcollapseexpandbuttondisplay
 pt:datavar="curr" pt:onclick="sendCollapseExpandRequest"
pt:scope="tag"/></div>
          <div
class="ali-portlet-controlone"><pt:portletpage.portletpreferencebuttondisplay
 pt:datavar="curr" pt:scope="tag"/></div>
          <div
class="ali-portlet-controlone"><pt:portletpage.portlethelpbuttondisplay
 pt:datavar="curr" pt:scope="tag"/></div>
          <div
class="ali-portlet-controlone"><pt:portletpage.portletrefreshbuttondisplay
```

```
 pt:datavar="curr" pt:scope="tag"/></div>
          <div class="ali-portlet-title"><pt:logic.value
pt:value="$curr.name"/></div>
        </div>
        </pt:logic.iffalse>
        </pt:logic.if>
        <!-- Display Portlet Content Body -->
        <div class="ali-portlet-content">
         <pt:portletpage.portletcontentdisplay pt:datavar="curr"
 pt:colindex="0" pt:scope="tag"/>
        </div>
        <div class="ali-portlet-footer">
          <div class="ali-portlet-botleft"></div>
          <div class="ali-portlet-botright"></div>
        </div>
      </pt:core.html>
      </pt:core.html>
    </pt:logic.foreach>
  </pt:portletpage.portletregiondisplay>
    </td>
    <td class="columnTwo" valign="top">

... portlet regions 2 and 3 ....
      </td>
    </tr>
</table>

</span>
```

For details on configuring adaptive page layouts in the portal, see the *AquaLogic Interaction Administrator Guide*.

Creating a Knowledge Directory Adaptive Page Layout

The Knowledge Directory adaptive page layout defines the content area of the Knowledge Directory. The outer area (header, footer, navigation areas) is based on the Base Page adaptive layout.

All content in the Knowledge Directory can be displayed using tags in the pt:kdpage library. The tags in this library can only be used in the Knowledge Directory page layout.

- Data about the folder in the Directory can be accessed using the `currentfolderdata` and `subfoldersdata` tags.

- Data about each folder's contents can be accessed using the `documentsdata` and `relatedresourcesdata` tags.
- This data can be arranged in a useful way using the `documentcolumnheadersdata`, `paginationdata`, `documentfilterdata`, and `documentsperpagedata` tags.

The example below uses tags from the pt:kdpage library to define Knowledge Directory page components, and logic tags to handle iteration and display.

- For an overview of adaptive tags, see *About Adaptive Tags* on page 202.
- For an introduction to logic tags, see *Logic Adaptive Tag Library (pt:logic)* on page 227.
- For a list of tags in the pt:kdpage library, see . *http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/kdpage/tld-summary.html*For a complete list of tags, see the 6.5 tagdocs (*http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/index.html*).

Note: This example has been oversimplified for illustration purposes; for a complete implementation of the Knowledge Directory page layout, see the knowledgedirectorylayout.html file in the \imageserver\plumtree\portal\private\pagelayouts folder on your ALI image server.

The first section of the page retrieves the folder data.

```
<script type="text/javascript">
<!--
    function toggle_visibility(id) {
       var e = document.getElementById(id);
       if(e.style.display == 'block')
          e.style.display = 'none';
       else
          e.style.display = 'block';
    }
//-->
</script>

<div id="content"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
    <pt:kdpage.currentfolderdata pt:id="currentfolder"/>
    <pt:kdpage.paginationdata pt:id="pagination"
pt:pageslist="pageslist" pt:pagestodisplay="2"/>
    <pt:logic.variable pt:key="currentfolderlevel"
pt:value="$currentfolder.level"/>
```

```
    <pt:logic.stringexpr pt:expr="($currentfolderlevel) == 1"
pt:key="isrootfolder"/>
```

After the breadcrumbs section (not shown here), the next section implements the column display. This section defines an index and divid for each column (3 in this example) The code iterates over the subfolders to display each folder in the correct column, using logic tags to divide the subfolder index by the number of columns to determine which column to display each folder. To change the number of columns, you would add or remove folders from the `<pt:logic.collection pt:key=”foldercolumns”>` section and change the divisor in the `<pt:logic.intops pt:expr="($subfolderindex) % 3" pt:key="col"/>` expression to reflect the correct number of columns.

```
... breadcrumbs section ...

 <!-- Start Root Folder Display -->
 <pt:logic.if pt:expr="$isrootfolder">
 <pt:logic.iftrue>
  <pt:logic.variable pt:key="displayrelatedresource"
pt:value="false"/>
  <div id="ali-kd-main-bar">
  </div>

  <pt:core.comment>This collection contains the meta-data about
the 3 subfolder columns.</pt:core.comment>

  <pt:logic.collection pt:key="foldercolumns">
   <pt:logic.data index="0" divid="ali-kd-main-col1"/>
   <pt:logic.data index="1" divid="ali-kd-main-col2"/>
            <pt:logic.data index="2" divid="ali-kd-main-col3"/>
  </pt:logic.collection>

  <!-- Displaying Columns Start -->
  <pt:logic.foreach pt:data="foldercolumns" pt:var="foldercolumn">


   <pt:kdpage.subfoldersdata pt:id="subfolders"/>
   <pt:logic.collectionlength pt:data="subfolders"
pt:key="flength"/>
   <pt:logic.intexpr pt:expr="($flength) > 0" pt:key="hasfolders"/>


   <pt:logic.if pt:expr="$hasfolders">
    <pt:logic.iftrue>

     <pt:core.html pt:tag="div" id="$foldercolumn.divid">
```

```
      <!-- Display Each Folder Start -->
      <pt:logic.foreach pt:data="subfolders" pt:var="subfolder">
       <pt:logic.intops pt:expr="($subfolderindex) % 3"
pt:key="col"/>
      <pt:logic.intexpr pt:expr="($col) == ($foldercolumn.index)"
 pt:key="incol"/>
       <pt:logic.if pt:expr="$incol">
        <pt:logic.iftrue>
         <div class="ali-kd-main-header">
             <pt:logic.variable pt:key="htmlEncodedName"
pt:value="$subfolder.name" pt:encode="1" />
          <pt:core.html pt:tag="a" href="$subfolder.url"
title="$htmlEncodedName">
           <pt:logic.value pt:value="$subfolder.name"/>
          </pt:core.html>
         </div>
         <!-- Display Subfolders Folder Start -->
         <pt:logic.variable pt:key="subsubfolderskey"
pt:value="$subfolder.subsubfolderskey"/>
         <pt:logic.collectionlength pt:data="$subsubfolderskey"
pt:key="flength"/>
         <pt:logic.intexpr pt:expr="($flength) > 0"
pt:key="hassubsubfolders"/>
         <pt:logic.if pt:expr="$hassubsubfolders">
          <pt:logic.iftrue>
           <div class="ali-kd-main-lists">
            <ul>
             <pt:logic.foreach pt:data="$subsubfolderskey"
pt:var="subsubfolder">
              <pt:core.comment><!-- Only display max 5 subfolder
 --></pt:core.comment>
              <pt:logic.intexpr pt:expr="($subsubfolderindex) <
5" pt:key="undermax"/>
              <pt:logic.if pt:expr="$undermax">
              <pt:logic.iftrue>
               <li>
                <pt:logic.variable pt:key="htmlEncodedName"
pt:value="$subsubfolder.name" pt:encode="1" />
               <pt:core.html pt:tag="a" href="$subsubfolder.url"
 title="$htmlEncodedName">
                 <pt:logic.value pt:value="$subsubfolder.name"/>
                </pt:core.html>
               </li>
              </pt:logic.iftrue>
              </pt:logic.if>
```

```
            </pt:logic.foreach>
           </ul>
          </div>
         </pt:logic.iftrue>
        </pt:logic.if>
       </pt:logic.iftrue>
      </pt:logic.if>
     </pt:logic.foreach>
     <!-- Display Each Folder End -->
    </pt:core.html>

   </pt:logic.iftrue>
  </pt:logic.if>
 </pt:logic.foreach>
 <!-- End Display Columns -->
</pt:logic.iftrue>
<!-- End Root Folder Display -->
```

After the sorting and filtering section (not shown here), the next section displays the document list and content.

```
 <!-- Start Document List and content -->
 <div id="ali-kd-content-container">
  <div id="ali-kd-documents">
      <div id="ali-kd-docs-showing">
       <pt:logic.value
pt:value="$#1932.ptmsgs_portalbrowsingmsgs"/>
       <pt:logic.value pt:value="$pagination.start"/>
       -
       <pt:logic.value pt:value="$pagination.end"/>
       <pt:logic.value
pt:value="$#811.ptmsgs_portalbrowsingmsgs"/>
       <pt:logic.value pt:value="$pagination.total"/>
       <pt:logic.value
pt:value="$#1043.ptmsgs_portalbrowsingmsgs"/>
      </div>

... edit document code ...

           <!-- Start display documents -->
        <pt:logic.foreach pt:data="docs" pt:var="doc">
                          <pt:logic.variable
pt:key="htmlEncodedName" pt:value="$doc.name" pt:encode="1" />
                          <tr>
                              <!-- Document Image Icon -->
                              <td><pt:core.html pt:tag="img"
```

```
src="$doc.imagesrc" alt="$htmlEncodedName"/></td>
                                    <td>
                                        <!-- Document Title -->
                                       <p class="ali-kd-docs-title">

                                         <pt:logic.variable
pt:key="htmlEncodedName" pt:value="$doc.name" pt:encode="1" />
                                       <pt:core.html pt:tag="a"
href="$doc.url" title="$htmlEncodedName"><pt:logic.value
pt:value="$doc.name"/></pt:core.html>
                                    </p>
                                    <!-- Document Description -->
                                    <p>
                                     <pt:logic.variable
pt:key="description" pt:value="$doc.description"/>
                                        <pt:logic.stringexpr
pt:expr="($description) == EMPTY_STRING" pt:key="nodescription"/>

                                        <pt:logic.if
pt:expr="$nodescription">
                                            <pt:logic.iftrue>
                                                <i><pt:logic.value
pt:value="$#832.ptmsgs_portalbrowsingmsgs"/></i>
                                            </pt:logic.iftrue>
                                         <pt:logic.iffalse>
                                             <pt:logic.value
pt:value="$doc.description"/>
                                            </pt:logic.iffalse>
                                        </pt:logic.if>
                                    </p>
                                       <p
class="ali-kd-docs-modified">

... pagination code ...

          <div id="ali-kd-side">
           <ul id="ali-kd-subfolder">
              <pt:kdpage.subfoldersdata pt:id="subfolders"/>
           <pt:logic.collectionlength pt:data="subfolders"
pt:key="flength"/>
           <pt:logic.intexpr pt:expr="($flength) > 0"
pt:key="hasfolders"/>
           <pt:logic.if pt:expr="$hasfolders">
            <pt:logic.iftrue>
             <pt:logic.value
```

```
pt:value="$#1931.ptmsgs_portalbrowsingmsgs"/>
                <pt:logic.foreach pt:data="subfolders"
pt:var="subfolder">
                 <li>
                     <pt:logic.variable pt:key="htmlEncodedName"
pt:value="$subfolder.name" pt:encode="1" />
               <pt:core.html pt:tag="a" href="$subfolder.url"
title="$htmlEncodedName"><pt:logic.value
pt:value="$subfolder.name"/></pt:core.html>
              </li>
             </pt:logic.foreach>
            </pt:logic.iftrue>
           </pt:logic.if>
           </ul>

...related resource display code ...

 </pt:logic.iffalse>
    </pt:logic.if>
</div>
<!-- End documents view -->
```

For details on configuring adaptive page layouts in the portal, see the *AquaLogic Interaction Administrator Guide*.

Creating a Search Results Adaptive Page Layout

The Search Results adaptive page layout defines the search results content area. The outer area (header, footer, navigation areas) is based on the Base Page adaptive layout.

The pt:searchpage library includes a set of tags to customize search results display. The `searchresultsdata` tag provides all the data needed to display search results. Each result is a DataObject that includes the following variables:

Variable	Description
name	The name of the result.
description	The description of the result.
rank	The rank of the result.
resulthref	A gatewayed link to the result.

Variable	Description
resultonclick	An onclick handler for result link if a handler exists.
resulttarget	The target window name for result link if a target exists.
icon	The URL to the icon for the result.
iconalttext	Alternate text for the icon associated with the result.
iconwidth	The width of the icon associated with the result, in pixels.
lastmodified	A string in the current locale representing the last modified date.
propertieshref	A link to the properties for the result if a properties link exists.
propertiesonclick	An onclick handler for result link if a properties link exists.
isbestbet	True if the result is a best bet, false otherwise.
isinmultiplefolders	True if the result occurs in more than one Knowledge Directory folder.
folderpathhref	A link to the Knowledge Directory folder for the result if one exists.
folderpath	The path to the Knowledge Directory folder for the result if one exists.
projectname	The name of the Collaboration project containing the result if the result is a Collaboration item.
projectonclick	An onclick handler for the link to the Collaboration project containing the result if the result is a Collaboration item.
lastpublishedby	The name of the last publishing user if the result is a Publisher item.

Variable	Description
associatedobjectsonclick	An onclick handler for the link to the associated objects for the result if the result is a Publisher item.

The `paginationdata` tag provides variables to handle pagination. The `followupformdata` tag generates the data required to create the follow-up search form shown on the results page. Additional tags define specific form elements.

The example below uses tags from the pt:searchpage library to define Search Results components, and logic tags to iterate over results.

- For an overview of adaptive tags, see *About Adaptive Tags* on page 202.
- For an introduction to logic tags, see *Logic Adaptive Tag Library (pt:logic)* on page 227.
- For a list of tags in the pt:searchpage library, see . *http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/searchpage/tld-summary.html*For a complete list of tags, see the 6.5 tagdocs (*http://edocs.bea.com/en/alui/devdoc/docs6x/api_libraries/tagdocs/ali/6.5/index.html*).

Note: This example has been oversimplified for illustration purposes; for a complete implementation of the search results page layout, see the searchresultslayout.html file in the \imageserver\plumtree\portal\private\pagelayouts folder on your ALI image server.

```
<div id="content"
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'>
 <pt:core.comment>
  <!-- Layout for the search results page.  This layout file
renders the portal search results page.
    It shows a followup search form and results.
   -->
 </pt:core.comment>
    <pt:searchpage.searchresultsdata pt:id="groupedresults"/>
 <pt:searchpage.searchsummarydata pt:id="summary"
          pt:groups="groups"
          pt:spellcorrections="spellcorrections"
          pt:breadcrumbs="breadcrumbs"
          pt:properties="properties"
          pt:collabprojects="collabprojects"
          pt:portlets="portlets"
          pt:folders="folders"
          pt:communities="communities"
          pt:objecttypes="objecttypes"/>
```

```
... search modification and sorting implementation ...

 <div id="ali-search-results">
     <pt:logic.collectionlength pt:data="groupedresults"
pt:key="resultslength"/>
     <pt:logic.intexpr pt:expr="($resultslength) > 0"
pt:key="hasResults"/>
     <pt:logic.if pt:expr="$hasResults">
        <pt:logic.iftrue>
            <table id="ali-search-results-table">
            <pt:logic.foreach pt:data="groupedresults"
pt:var="resultsgroup">
                <pt:logic.variable pt:key="results"
pt:value="$resultsgroup.groupresultskey"/>
                <pt:logic.collectionlength pt:data="$results"
pt:key="resultslength"/>
                <pt:logic.intexpr pt:expr="($resultslength) > 0"
 pt:key="hasResults"/>
                <pt:logic.if pt:expr="$hasResults">
                    <pt:logic.iftrue>
                        <pt:logic.foreach pt:data="$results"
pt:var="result">
                            <pt:logic.variable
pt:key="htmlEncodedName" pt:value="$result.name" pt:encode="1" />

                            <tr>
                                <td><pt:core.html pt:tag="img"
src="$result.icon" alt="$htmlEncodedName"/></td>
                                <td>
                                   <p class="ali-kd-docs-title">

                                       <pt:logic.existexpr
pt:data="result.resultonclick" pt:key="hasResultOnclick"/>
                                     <pt:logic.if
pt:expr="$hasResultOnclick">
                                      <pt:logic.iftrue>
                                          <pt:core.html pt:tag="a"
 href="$result.resulthref" onclick="$result.resultonclick"
title="$htmlEncodedName"><pt:logic.value
pt:value="$result.name"/></pt:core.html>
                                      </pt:logic.iftrue>
                                      <pt:logic.iffalse>
                                     <pt:logic.existexpr
pt:data="result.resulttarget" pt:key="hasResultTarget"/>
                                      <pt:logic.if
```

```
pt:expr="$hasResultTarget">
                                        <pt:logic.iftrue>
                                       <pt:core.html pt:tag="a"
href="$result.resulthref" target="$result.resulttarget"
title="$htmlEncodedName"><pt:logic.value
pt:value="$result.name"/></pt:core.html>
                                        </pt:logic.iftrue>
                                        <pt:logic.iffalse>
                                       <pt:core.html pt:tag="a"
href="$result.resulthref" title="$htmlEncodedName"><pt:logic.value
 pt:value="$result.name"/></pt:core.html>
                                        </pt:logic.iffalse>
                                     </pt:logic.if>
                                      </pt:logic.iffalse>
                                    </pt:logic.if>
                                      </p>
                                      <p>
                                          <pt:logic.existexpr
pt:data="result.description" pt:key="hasDesc"/>
                                  <pt:logic.if pt:expr="$hasDesc">

                                     <pt:logic.iftrue>
                                       <pt:core.comment><!-- The
description contains HTML from search, and is safe, so we shouldn't
 HTML encode it. --></pt:core.comment>
                                        <pt:logic.value
pt:value="$result.description" pt:encode="0"/> 
                                     </pt:logic.iftrue>
                                     <pt:logic.iffalse>
                                       <i><pt:logic.value
pt:value="$#832.ptmsgs_portalbrowsingmsgs"/></i>
                                     </pt:logic.iffalse>
                                    </pt:logic.if>
                                      </p>
                                      <p
class="ali-search-results-modified">
                                          <pt:core.localize
pt:id="1918" pt:file="ptmsgs_portalbrowsingmsgs"
pt:replace0="$result.lastmodified" />
                                        <pt:logic.existexpr
pt:data="result.propertieshref" pt:key="hasPropsLink"/>
                                   <pt:logic.if
pt:expr="$hasPropsLink">
                                     <pt:logic.iftrue>
                                   <pt:core.html pt:tag="a"
```

```
href="$result.propertieshref" onclick="$result.propertiesonclick"
 title="$#1657.ptmsgs_portalbrowsingmsgs"><pt:logic.value
pt:value="$#31.ptmsgs_portalbrowsingmsgs"/></pt:core.html>
                                    </pt:logic.iftrue>
                                  </pt:logic.if>
                                    </p>
                                </td>
                            </tr>
                        </pt:logic.foreach>
                    </pt:logic.iftrue>
                </pt:logic.if>
            </pt:logic.foreach>
            </table>

... page navigation implementation ...

        <pt:logic.iffalse>
            <table id="ali-search-results-table">
                <tr><td>
                    <p><pt:logic.value
pt:value="$#848.ptmsgs_portalbrowsingmsgs"/></p>
                </td></tr>
            </table>
            <br />
        </pt:logic.iffalse>
     </pt:logic.if>
    </div>
</div>
</div>
```

For details on configuring adaptive page layouts in the portal, see the *AquaLogic Interaction Administrator Guide*.

About ALI 6.5 Adaptive Styles (CSS Customization)

In AquaLogic Interaction 6.5, the available CSS UI customization options have been expanded and restructured. These topics provide information and detailed instructions on implementing UI customization through CSS.

ALI 6.5 includes a new UI customization framework based on adaptive page layouts. For details on adaptive page layouts, see *About Adaptive Page Layouts (ALI 6.5)* on page 426 and the

AquaLogic Interaction Administrator Guide. If you are using adaptive page layouts in your portal implementation, additional UI customization options are available through the ALI CSS file.

- To customize default styles for adaptive page layouts, modify the mainstyle.css file. Located in the %PT_HOME%\ptimages\imageserver\plumtree\common\private\css\ folder on the ALI image server, this stylesheet contains all available CSS elements. For examples of CSS customizations, see *Using Adaptive Styles to Customize Portlet Style and Layout* on page 448 and *Using Adaptive Styles to Customize Page Layout* on page 447. For details on the different types of elements available, see the following topics:
  - *ALI 6.5 Adaptive Styles Base Page Elements* on page 450: These elements control style and layout for base page elements, including the action bar and banner.
  - *ALI 6.5 Adaptive Styles Navigation Elements* on page 453: These elements control style and layout for navigation elements, including the menus and breadcrumbs.
  - *ALI 6.5 Adaptive Styles Search Elements* on page 458: These elements control style and layout for search elements, including search forms and the search browse page.
  - *ALI 6.5 Adaptive Styles Editing Elements* on page 460: These elements control style and layout for editing elements, including the page editing elements near the navigation breadcrumb and the flyout page and portlet editor elements.
  - *ALI 6.5 Adaptive Styles Directory Elements* on page 467: These elements control style and layout for Directory elements.
  - *ALI 6.5 Adaptive Styles Portlet Elements* on page 472: These elements control style and layout for portlet elements.
  - *ALI 6.5 Adaptive Styles User Elements* on page 476: These elements control style and layout for user elements, including user profile, user activity stream, user friends and user information components.
- To provide language-specific stylesheets for internationalized portal implementations, create a localized version of the ALI stylesheet and map the language to the each style sheet in the CustomStyles.xml file. For details, see *Implementing Localized Stylesheets for Adaptive Page Layouts* on page 481.

The portal CSS template file follows standard CSS syntax rules. For details on CSS, see *http://www.w3.org/Style/CSS/*.

(The legacy layout is still supported in 6.5. For details on legacy CSS customization, see *Introduction: Using Adaptive Styles (CSS Customization)*.)

Using Adaptive Styles to Customize Page Layout

Adaptive styles allow you to modify page layout and design using the portal CSS template file. You can also use CSS to hide specific functionality exposed in the portal page. This page provides basic syntax rules and customization examples.

Syntax

To apply styles to a specific page, use the page ID. The example below sets the background color for the page with ID 100.

```
#pt-page-100
{
 background-color: red;
}
```

You can change style settings for a specific user or type of user (administrator or guest). The example below displays a special header image on all browse-mode pages for guests. To modify a style for a specific user, replace "guest" with the name of the appropriate portal User object (e.g., `.ptPageUser-mycompany domain ad\Joe Smith`).

```
.ptPageUser-guest #pt-header
{
 background-image:
url(/imageserver/plumtree/portal/private/img/example_guest.gif);
}
```

Style and Branding Customizations

The mainstyle.css file allows you to make a wide range of style changes to the portal page, including adding custom branding and color schemes. For example, you can add a custom image to the portal banner as shown in the example below.

```
#ali-banner  {
 width:100%;
 height:80px;
 background-color: #1D54A6;
 background-image: url(../img/MyCompany_bkg.jpg);
 background-repeat: repeat-x;
 font-family:Verdana, Arial, Helvetica, sans-serif;
 min-width:980px;
}
```

You can also modify the background color for a single page or a specific community. The example below sets the background color for the community with ID 200.

```
.ptCommunity-200
{
 background-color: #AAA;
}
```

Page Element Customizations

The mainstyle.css file allows you to modify the style of form elements in the portal page, including text boxes and buttons. For example, the code below expands the size of the banner search box.

```
#ali-bannerSearch {
 clear:right;
 float:right;
 position:relative;
 width:600px;
 padding: 18px 24px 0px 24px;
}
```

Note: The syntax on this page applies to ALI 6.5 adaptive page layouts. For earlier versions or legacy page layouts, see *Customizing Portal Page Layout and Design*.

Using Adaptive Styles to Customize Portlet Style and Layout

Adaptive styles allow you to customize specific portlets using the unique ID assigned to each portlet, or use CSS classes to modify the design of a group of portlets. You can also set constraints for portlets, including limiting a specific portlet to a three-column layout or preventing users from collapsing portlets.

For an introduction to portlet style elements, see *ALI 6.5 Adaptive Styles Portlet Elements* on page 472.

Syntax

To apply a CSS tag to a specific portlet, use the portlet ID. the example below increases the space around the portlet title for the portlet with ID 43. (You can also define basic styles for a specific portlet within the portlet code.)

```
#pt-portlet-43 .ali-portlet-title
{
```

```
 padding:8px 0 0 0;
}
```

You can also apply styles to groups of portlets, including those on a specific page or in a specific community. To apply styles to a portlet on a specific page or community, use the page or community ID. The example below makes the same modification as above for all the portlet on the page will ID 100.

```
#pt-page-100 .ali-portlet-title
{
 padding:8px 0 0 0;
}
```

Style Customizations

The mainstyle.css file allows you to make a wide range of style changes to portlets. For example, you can change the color scheme of portlets as shown in the example below.

```
#pt-portlet-43 .ali-portlet-content
{
 clear:left;
 width:100%;
 border-left:solid 1px #6B91C0;
 border-right:solid 1px #6B91C0;
 color:#6B91C0;
}
```

Constraints

The mainstyle.css file allows you to set constraints for portlets. For example, you can set the width of a portlet for a specific page or set of pages. You can define portlet settings by page, layout/column, or community. The example below limits the portlet with ID 43 to a width of 250 pixels on the page with ID 100.

```
#pt-page-100 #pt-portlet-43
{
 width: 250px;
}
```

Note: The syntax on this page applies to ALI 6.5 adaptive page layouts. For earlier versions or legacy page layouts, see *Portlet Style*.

ALI 6.5 Adaptive Styles Base Page Elements

These elements control style and layout for base page elements, including the action bar and banner. This is not a complete list; for all available elements, see the mainstyle.css file.

Note: For search elements, including the banner search form, see *ALI 6.5 Adaptive Styles Search Elements* on page 458.

Element	Page Component	Example
body	The main body of the page (any text-based content not inside another element).	`body { font-family: Verdana, Arial, Helvetica, sans-serif; font-size:100%; color: #FFFFFF; margin: 0px; }`
a:hover	Links in the main body of the page on mouse-over.	`a:hover { text-decoration:underline; }`
#ali-actionbar	The action bar at the top of the page.	`#ali-actionbar { width:100%; height:22px; background-color: #0A2F66; background-image: url(../img/banner_action_bkg.jpg); background-repeat: repeat-x; font-family:Verdana, Arial, Helvetica, sans-serif; min-width:980px; }`
#ali-banner	The main content in the portal banner. This area usually contains company branding, which can include an image referenced in the	`#ali-banner { width:100%; height:80px; background-color: #1D54A6; background-image:`

Element	Page Component	Example

	background-image: url parameter.	url(../img/banner_bkg.jpg);
		
		background-repeat: repeat-x;
		
		font-family:Verdana, Arial, Helvetica, sans-serif;
		min-width:980px;
		}

#ali-bannerWelcome	The welcome text displayed in the portal banner.	#ali-bannerWelcome {
		float:left;
		background:none;
		color:#B2D8FF;
		font-size:.8em;
		
		letter-spacing:1px;
		padding:6px 4px 0px 12px;
		}

#ali-bannerNav	The navigation section of the portal banner.	#ali-bannerNav {
		float:right;
		background:none;
		color:#A6CFF6;
		font-size:.8em;
		letter-spacing:1px;
		padding:6px 12px 4px 4px;
		text-align:right;
		}

#ali-bannerLogo	The logo in the portal banner.	#ali-bannerLogo { clear:left; float:left; padding:10px 10px 0px 14px;

Element	Page Component	Example

		}
		
		#ali-footer {
		clear:both;
		width:100%;
		height:22px;
		
		background-image:url(../images/footer_bg.gif);
		
		background-repeat:repeat-x;
		
		color:#FFFFFF;
		font-family:Arial,
		Helvetica,
		sans-serif;
		font-size:.7em;
		letter-spacing:1px;
		text-align:center;
		margin-top:48px;
		padding:4px 0 0 0;
		min-width:980px;
		}

#ali-footer	The portal page footer.	

ALI 6.5 Adaptive Styles Navigation Elements

These elements control style and layout for navigation elements, including the menus and breadcrumbs. This is not a complete list; for all available elements, see the mainstyle.css file.

Note: Parameters marked "!important;" are related to drop-down lists within navigation elements. These paramaters can be customized, but must not be removed; eliminating them will cause the drop-downs either to not work or to distort.

Element	Navigation Component	Example
#ali-mainNav	The main navigation section of the portal page.	`#ali-mainNav { clear:left; float:left; width:100%; height:30px; background-color:#3068CF; background-image:url(../images/main_nav_tab.gif); background-repeat:repeat-x; border-bottom:solid 1px #5083CB; letter-spacing: 1px; min-width:980px; }`
#ali-nav	All lists within navigation sections in the portal page.	`#ali-nav, #ali-nav ul { padding: 0; margin: 0; list-style: none; line-height: 1; }`
a.ali-navmenu	The portal navigation menu.	`a.ali-navmenu { color:#385ABD !important; width:170px !important; background-color:#F1F5F9 !important; border-bottom:solid 1px #146BC5; font-size:1.15em; }`

Element	Navigation Component	Example
a.ali-nav-actions	A separate style for items in the portal navigation menu that are actions, as opposed to links to pages/sections on the portal. Sets a different background color for these items in the menu to offer a clear distinction between menu navigation links and action links.	`a.ali-nav-actions { color:#2B49AC !important; width:170px !important; background-color:#C9D4E9 !important; border-bottom:solid 1px #146BC5; font-size:1.15em; }`
#ali-secondNavBar	The second-level navigation bar is used on the User Profile page and Community Pages. A second bar appears below the main navigation bar in a different color and is mainly	`#ali-secondNavBar { clear:both; float:left; width:100%; padding:0; margin:0 0 -21px 0;`

Element	Navigation Component	Example

	used to list pages within a community.	background-image:url(../images/nav2_pages.gif);
		background-repeat:repeat-x;
		letter-spacing:0;
		font:bold .725em Helvetica;
		min-width:980px;
		}

#ali-secondPages	The list of secondary pages in a community or other sections of the portal that use a second level of navigation.	#ali-secondPages {
		float:left;
		color:#51617a;
		background-color:none;
		width:80%;
		}

#ali-secondSub	The drop-down menu for pages within a community ("sub-communities").	#ali-secondSub {
		float:right;
		padding:0;
		margin:0;
		background-image:url(../images/nav2_subg.gif);
		background-repeat:repeat-x;
		border-bottom:solid 1px #82A8F3;
		border-left:solid 1px #82A8F3;
		}

#ali-secondNav	The list of links in the sub-community drop-down	#ali-secondNav,

Element	Navigation Component	Example
	menu.	`#ali-secondNav ul { padding: 0; margin: 0; list-style: none; }`
a.ali-secondMenu	The link color and background color for the sub-community drop-down menu.	`a.ali-secondMenu { color:#4467CB; font:bold 8pt Arial, Helvetica, sans-serif; width:161px; background-color:#F1F6FF; border-bottom:solid 1px #83A1D8; }`
#ali-community-name	The look and placement of the community (or home page) name for the second navigation bar.	`#ali-community-name { position:relative; left:-40px; background-image:url(../images/arrow2_down.gif); background-repeat:repeat-x; border-right:solid 1px #82A8F3; letter-spacing:1px; color:#5374A1; padding:8px 12px 6px 12px; margin-right:-3px; }`

Element	Navigation Component	Example
#ali-breadcrumb	The breadcrumb list displayed at the top of the portal page content section.	`#ali-breadcrumb { float:left; margin:4px 0 0 12px; padding:0; color:#888888; padding:0; font-family:Helvetica, Arial, sans-serif; font-size:.7em; letter-spacing:1px; }`

ALI 6.5 Adaptive Styles Search Elements

These elements control style and layout for search elements, including search forms and the search browse page. This is not a complete list; for all available elements, see the mainstyle.css file.

Element	Search Component	Example
#ali-bannerSearch	The basic (banner) search form in the portal banner.	`#ali-bannerSearch { clear:right; float:right; position:relative; width:400px; padding: 18px 24px 0px 24px; }`
#ali-searchAdvanced	The div below the banner search box that contains the advanced search text link.	`#ali-searchAdvanced { clear:right; float:right; width:320px; margin:0 0 0 0; padding:0 90px 0 0; }`
input.ali-searchBox	The search input box in the search form. Applies to the input box itself only when it appears in the banner.	`input.ali-searchBox { color:#999999; border:outset 1px; padding: 1px; }`
input.ali-searchButton	The search button in the search form. Applies to the search button in the banner only.	`input.ali-searchButton { background-image:url(../images/button_search_gadget.gif); background-repeat:repeat-x; border:outset 0px; padding:2px 6px; margin-left:4px; color:#1A48A4; font-size:.8em;`

Element	Search Component	Example
		}

input[type="button"]:hover	The search button on mouse-over. Applies to the search button in the banner.	input[type="button"]:hover
		{
		color:#FF6000;
		}

#ali-search-modifier-container	The search form on the search results/browse page.	#ali-search-modifier-container
		{
		clear:both;
		float:left;
		width:99%;
		margin:6px 0 0 0;
		
		padding:0;
		min-width:980px;
		}

#ali-search-results	The search results section on the search browse page.	#ali-search-results
		{
		clear:both;
		float:left;
		width:78%;
		min-width:625px;
		margin:12px 0 0 32px;
		padding:0;
		}

ALI 6.5 Adaptive Styles Editing Elements

These elements control style and layout for editing elements, including the page editing elements near the navigation breadcrumb and the flyout page and portlet editor elements. This is not a complete list; for all available elements, see the mainstyle.css file.

Element	Editing Component	Example

#ali-pageEdit	The portal actions such as Edit Page, Create Page, etc. Appears opposite the breadcrumb at the top or the portal page.	`#ali-pageEdit { float:right; padding: 0px 0px 12px 0px;`
		
		`font-family:Helvetica, Arial, sans-serif; color:#96b7ED; text-align:right; }`

#ali-edit-container	The div that contains the flyout page editor.	`#ali-edit-container { clear:left; float:left; width:97%; margin:0px 12px 12px 12px; min-width:950px; }`
#ali-edit-toolbar	The bar at the top of the flyout page editor and contains the "Edit Page" text and the "X"	`#ali-edit-toolbar { float:left; width:100%; height:21px;`

Element	Editing Component	Example

	(close editor) button for the editor.	margin-top:6px;
		background-color:#6B91C0;
		background-image:url(../images/edit_title_bg.gif);
		background-repeat:repeat-x;
		color:#FFFFFF;
		font-family:Verdana, Arial, Helvetica, sans-serif;
		font-size:11px;
		font-weight:bold;
		letter-spacing:1px;
		}

#ali-edit-cornerleft and #ali-edit-cornerright	The top right and top left rounded corners for the flyout page editor.	#ali-edit-cornerleft {
		clear:left;
		float:left;
		width:8px;
		height:21px;
		background-image:url(../images/edit_corner_left.gif);
		background-repeat:no-repeat;
		}
		#ali-edit-cornerright {
		float:right;
		position:relative;
		right:-2px;
		width:8px;
		height:21px;
		background-image:url(../images/edit_corner_right.gif);

Element	Editing Component	Example
		background-repeat:no-repeat; margin:0; padding:0; }

#ali-edit-content	The div containing the main section of the flyout editor below the toolbar and above the rounded corners at the bottom.	#ali-edit-content {
		width:100%;
		border-left:solid 1px #6B91C0;
		border-right:solid 1px #6B91C0;
		
		background-color:#ECEFF4;
		
		color:#6B91C0;
		}

#ali-edit-tabs-container	Can be used to set tabbed navigation within the flyout page editor. (Not used in the	#ali-edit-tabs-container { clear:left; float:left;

Element	Editing Component	Example

	default implementation of the flyout page editor.)	width:100%;
		margin:0;
		padding:0;
		height:30px;
		
		background-image:url(../images/edit_tab_gradient.gif);
		
		background-repeat:repeat-x;
		
		background-color:#C8DCFF;
		
		border-bottom:solid 1px #7497C4;
		border-right:solid 1px #7497C4;
		border-left:solid 1px #7497C4;
		}

.ali-edit-tabs	Can be used to set the style of tabbed navigation within the flyout page editor. (Not used in the default implementation of the flyout page editor.)	.ali-edit-tabs {
		float:left;
		margin:6px 0 0 0;
		padding:0;
		
		background-color:#CFd3E7;
		
		border-left:solid 1px #7497C4;
		border-top:solid 1px #7497C4;
		border-right:solid 1px #7497C4;
		}

#ali-edit-footer	Contains the elements for the bottom of the flyout page editor, including the bottom	#ali-edit-footer { width:100%; clear:left; float:left;

Element	Editing Component	Example

	outline and the left and right rounded corners.	background-image:url(../img/edit_bot.gif);
		
		background-repeat:repeat-x;
		
		height:8px;
		}

#ali-edit-botleft and #ali-edit-botright	The divs that contain the rounded corners for the bottom right and bottom left corners of the flyout page editor. The images for these corners are	#ali-edit-botleft { clear:left; float:left; width:8px; height:8px;

Element	Editing Component	Example

	specified in the style as the background image.	position:relative;
		left:-1px;
		
		background-image:url(../images/edit_portlet_tab_left.gif);
		
		background-repeat:no-repeat;
		}

#ali-edit-table	Flyout portlet editor table.	#ali-edit-table {
		font-size:1em;
		color:#000000;
		margin:0px;
		}

#ali-edit-portlets	Flyout portlet editor tab.	#ali-edit-portlets {
		float:left;
		width:100%;
		border-left:solid 1px #6B91C0;
		border-right:solid 1px #6B91C0;
		
		background-color:#ECEFF4;
		
		color:#000000;
		font-size:11px;
		}

#ali-edit-portlets-search	Flyout portlet editor search form.	#ali-edit-portlets-search {
		float:left;
		padding:0 18px 0 14px;
		
		border-right:solid 1px #D5D6DA;
		line-height:31px;
		}

Element	Editing Component	Example
#ali-edit-breadcrumb	Flyout portlet editor breadcrumbs.	`#ali-edit-breadcrumb { float:left; color:#2B4A7B; padding: 2px 24px 12px 2px; font-family:Helvetica, Arial, sans-serif; font-size:11px; letter-spacing:1px; }`
#ali-edit-main-col1	Flyout portlet editor folder display.	`#ali-edit-main-col1 { float:left; width:212px; margin:0; padding:0; }`

ALI 6.5 Adaptive Styles Directory Elements

These elements control style and layout for Directory elements. This is not a complete list; for all available elements, see the mainstyle.css file.

Element	Directory Component	Example
#ali-kd-title	The title displayed on the Directory page.	`#ali-kd-title { float:left; height:22px; padding:4px 12px 0 12px; border-right:solid 1px #9BBEEE; color:#7197c6; font-size:.8em; font-weight:bold; letter-spacing:2px; }`
#ali-kd-main*	These elements control the main page of the Directory	`#ali-kd-main-bar { clear:both; float:left; width:100%; min-width:980px; margin:0; padding:0; border-top:solid 1px #DBD9D9; border-bottom:solid 1px #C9CED9; background-color:#CFDFFF; background-repeat:repeat-x; height:7px; } #ali-kd-main-col1 { float:left; width:212px; margin:0 60px 48px 36px; } .ali-kd-main-header {`

Element	Directory Component	Example
		width:100%; margin-top:24px; padding:2px 4px; background-color:#E5E9F6; border-bottom:solid 1px #C6CAD4; }

#ali-kd-breadcrumb	The breadcrumb displayed on the Directory page.	#ali-kd-breadcrumb {
		float:left;
		height:22px;
		padding:5px 0 0 0;
		margin-left:-28px;
		font-size:.75em;
		font-weight:bold;
		color:#2B4A7B;
		}

#ali-kd-sorting-bar	The div in the Directory and search results pages that contains the pull-down menu for sorting results or listings by	#ali-kd-sorting-bar { clear:both; float:left; width:100%;

Element	Directory Component	Example

	"items per page", "item type", etc.	min-width:1000px;
		margin:0;
		padding:0;
		height:31px;
		border-top:solid 1px #D5D4D4;
		border-bottom:solid 1px #A8B8D9;
		
		background-image:url(../img/kd_sort_bg.gif);
		
		background-repeat:repeat-x;
		
		font-family:Verdana, Arial, Helvetica, sans-serif;
		font-size:.8em;
		color:#000000;
		font-weight:normal;
		}

#ali-kd-documents	The div for listings of items in Directory folders.	#ali-kd-documents {
		clear:both;
		float:left;
		width:64%;
		min-width:625px;
		min-height:500px;
		margin:0 0 0 32px;
		padding:6px 64px 48px 0;
		
		background-image:url(../img/kd_folder_bg.gif);
		
		background-repeat:repeat-y;
		
		background-position:right;
		}

Element	Directory Component	Example

.ali-kd-doc-office, .ali-kd-doc-web, .ali-kd-doc-text	The icon and text for specific document types.	.ali-kd-doc-office {
		clear:both;
		color:#000000;
		font-size:.8em;
		padding:0 24px 24px 34px;
		
		background-image:url(../images/icon_office_24x24.gif);
		
		background-repeat:no-repeat;
		}

#ali-kd-pagenav	Directory navigation section.	#ali-kd-pagenav {
		clear:both;
		float:right;
		line-height:2em;
		color:#A1B2C4;
		font-size:.7em;
		
		padding-right:24px;
		
		margin-bottom:12px;
		}

#ali-kd-side	Directory subfolders and related links sections.	#ali-kd-side {
		float:left;
		right:18px;
		width:25%;
		margin:0 0 0 -12px;
		padding:20px 0 48px 0;
		min-width:250px;
		letter-spacing:1px;
		
		font-family:Helvetica, sans-serif;
		font-size:.8em;
		font-weight:bold;
		color:#7197C6;
		}

Element	Directory Component	Example
#ali-kd-subfolder li	Directory subfolder links.	#ali-kd-subfolder li { padding:2px 0 2px 24px; list-style:none; background-image:url(../images/folder_16x16.gif); background-repeat:no-repeat; background-position:0 50%; }
.ali-kd-related a	Directory related links.	.ali-kd-related a { font-size:90%; font-family:Arial, Helvetica, sans-serif; font-weight:normal; color:#3761B7; text-decoration:none; }

ALI 6.5 Adaptive Styles Portlet Elements

These elements control style and layout for portlet elements. This is not a complete list; for all available elements, see the mainstyle.css file.

For portlet flyout editor elements, see *ALI 6.5 Adaptive Styles Editing Elements* on page 460.

Element	Portlet Component	Example
.ali-portlet-container	Contains the nested elements of the portlet toolbar, controls rounded corners and content.	`.ali-portlet-container { min-width:250px; margin:4px 0px 6px 0px; }`
.ali-portlet-cornerleft and .ali-portlet-cornerright	Divs that contain the rounded corners for the top right and top left corners of the portlet. The images for these corners are	`.ali-portlet-cornerright { float:right; width:8px; height:21px;`

Element	Portlet Component	Example

	specified in the style as the background image.	position:relative;
		right:-2px;
		background-image:url(../images/portlet_corner_right.gif);
		background-repeat:no-repeat;
		margin:0;
		padding:0;
		}

.ali-portlet-toolbar	The top section of the portlet, where the title of the portlet sits along with action buttons such as minimize, edit and refresh.	.ali-portlet-toolbar
		{
		width:100%;
		height:21px;
		background-color:#5C91D8;
		background-image:url(../images/portlet_title_bar.gif);
		background-repeat:repeat-x;
		color:#FFFFFF;
		font-family:Verdana,
		Arial, Helvetica,
		sans-serif;
		font-size:1.1em;
		font-weight:bold;
		letter-spacing:1px;
		}

.ali-portlet-controlone and .ali-portlet-controltwo	Used to position the action buttons in the portlet toolbar for actions such as minimize, edit and refresh.	.ali-portlet-controltwo { float:right; width:13px; margin-bottom:-13px; padding:0px 0px 0px

Element	Portlet Component	Example
		6px; border: solid 1px #FF0000; }

.ali-portlet-content	The div that contains the main content of the portlet. Sets the left and right outlines of the portlet.	.ali-portlet-content {
		clear:left;
		width:100%;
		border-left:solid 1px #6B91C0;
		border-right:solid 1px #6B91C0;
		color:#6B91C0;
		}

.ali-portlet-footer	Contains the elements for the bottom of the portlet, including the bottom outline and the left and right rounded corners.	.ali-portlet-footer {
		width:100%;
		background-image:url(../my/portlet_bot.gif);
		background-repeat:repeat-x;
		height:8px;
		}

.ali-portlet-botleft and.ali-portlet-botright	Divs that contain the rounded corners for the bottom right and bottom left corners of the portlet. The images for these	.ali-portlet-botright { float:right; width:8px; height:8px;

Element	Portlet Component	Example

	corners are specified in the style as the background image.	position:relative;
		right:-2px;
		background-image:url(../images/portlet_cornerright.gif);
		background-repeat:no-repeat;
		}

ALI 6.5 Adaptive Styles User Elements

These elements control style and layout for user elements, including user profile, user activity stream, user friends and user information components. This is not a complete list; for all available elements, see the mainstyle.css file.

Element	User Component	Example
#ali-user-navbar	The user profile navigation menu in portal navigation.	`#ali-user-navbar { clear:both; width:100%; padding:0 0 2px 0; margin:0; height:27px; background-image:url(../images/nav_2_bg.gif); background-repeat:repeat-x; letter-spacing:1px; font:bold .725em Helvetica; line-height:24px; min-width:980px; }`
.ali-user-activity*	These elements control the display of user activity stream components. The User Activities portlet usually	`.ali-user-activity-pulldown { clear:both; float:right; width:99%;`

Element	User Component	Example

	appears on the user profile page.	padding:3px 16px 0 0;
		color:#000000;
		text-align:right;
		font-size:.75em;
		}
		
		.ali-user-activity-stream {
		margin: 8px 0 0 0;
		padding:0 0 0 4px;
		
		background-color:#EFF2FA;
		
		border-bottom:solid 1px #DBDEE4;
		font-size:.75em;
		font-weight:bold;
		letter-spacing:1px;
		}

.ali-user-friends*	These elements control the display of the user friends components. The User's Friends list portlet usually	.ali-user-pulldown { clear:both; float:right; width:99%; padding:3px 16px 0

Element	User Component	Example

	appears on the User Profile page. .	0;
		color:#000000;
		font-size:.75em;
		text-align:right;
		}
		
		.ali-friends-info-title
		{
		padding-right:6px;
		text-align:right;
		color:#6E7686;
		font-size:.75em;
		font-weight:bold;
		letter-spacing:1px;
		}

#ali-user-geninfo*	These elements control the display of the user general information components, the	#ali-user-geninfo-edit { float:right; width:100px;

Element	User Component	Example

	main information on the User Profile page.	margin:-4px;
		padding:6px 12px 6px 12px;
		
		background-color:#EFF3FF;
		
		border-left:solid 1px #C4C8DB;
		text-align:center;
		}
		
		.ali-user-geninfo-title {
		
		padding-right:6px;
		text-align:right;
		
		color:#6E7686;
		font-size:.65em;
		font-weight:bold;
		
		letter-spacing:0;
		}

#ali-user-search	The search section in the user general info component.	#ali-user-search {
		float:right;
		padding:3px 24px;
		
		margin:0;
		height:22px;
		width:310px;
		
		background-image:url(../images/a_2_sbg.gif);
		
		background-repeat:repeat-x;
		
		border-left:solid 1px #6f90cf;
		}

Implementing Localized Stylesheets for Adaptive Page Layouts

To provide language-specific stylesheets for internationalized portal implementations, create a localized version of the ALI stylesheet and map the language to the each style sheet in the CustomStyles.xml file.

The CustomStyles.xml file is located in the %PT_HOME%\settings\portal\ folder on the portal server. This file also contains default mappings to legacy stylesheets to support any products that do not use adaptive page layouts.

Note: The language-specific stylesheet mappings in CustomStyles.xml only apply to pages that use adaptive page layouts.

1. Create a localized version of the ALI stylesheet for each language. For example, mystyle-ar.css for Arabic.
2. Modify CustomStyles.xml to specify stylesheets for each supported language. For example, to use mystyle-ar.css for Arabic, add the following mapping to CustomStyles.xml:

```
<StyleSettings>
  <cssMapping>
    <language>ar</language>
    <styles>mystyle-ar.css</styles>
  </cssMapping>
</StyleSettings>
```

3. Include the `pt://styles` adaptive tag in the head element of any page that should use a localized stylesheet. The head element must also include the `pt.standard.stylesheets` tag to reference the legacy stylesheet, which contains the legacy portlet styles required by any preexisting portlets and by the admin UI.

```
<head>
<pt.standard.stylesheets/>
<link href="pt://styles" type="text/css"
rel="styleSheet"></link>
... </head>
```

Note: The pt://styles tag can only be used to implement localized stylesheets in pages that use adaptive page layouts.

About the ALI Activity Stream API

The ALI Activity Stream API is a REST-based API that allows remote web services to post stories to portal user's activity stream through simple HTTP requests.

User status and activities are displayed in the Status Portlet and User Activity Portlet included with the ALI installation. New activities (stories) can be posted to these portlets from remote applications using the ALI Activity Stream API. For details on using the ALI Activity Stream API, see the following topics:

- *Using the ALI Activity Stream API* on page 483
- *Configuring Web Services that Use the ALI Activity Stream API* on page 491

Note: To use the Activity Stream API from a remote web service, the Remote Portlet Service must be installed and enabled. The Remote Portlet Service and Activity Stream sample portlets are included in the activityservice.pte file provided with the ALI installation package. For details on installation and configuration, see the *AquaLogic Interaction Installation Guide* and the *AquaLogic Interaction Administrator Guide*.

Using the ALI Activity Stream API

To add a new story to a users Activity Stream, send a POST request containing the necessary data to the story server URL.

The post must use the following syntax:

```
POST
http://${STORY_SERVER_URL}/api/story/v1/activities/${USER_ID_TYPE}/${USER_ID_PREFIX}/${USER_ID_SUFFIX}

<soc:activity xmlns:soc="http://social.bea.com/activity">
  <body><![CDATA[${EVENT}]]></body>
</soc:activity>
```

Variable	Description
STORY_SERVER_URL	The URL to the story server, typically, %HOSTNAME%:21030/activityservice.
USER_ID_TYPE	The type of user identifier that follows; acceptable values are: username (login name), UUID, & ID (portal object integer ID). **Note:** These strings must be lowercase.
USER_ID_PREFIX	The identifier of the user that the story is about; in the case of domain qualified names, this is the domain name. (For example, bea\jking will be represented as .../username/bea/jking in the post URL.)
USER_ID_SUFFIX	(Optional.) In the case of domain qualified names, this is the username; omitted in all other cases.
EVENT	The story itself (the CDATA envelope is optional if the story does not include any markup).

Note: The Activity Stream API is protected by ALI security. Access is restricted to portal users by login token verification or basic authentication. For details, see *Configuring Web Services that Use the ALI Activity Stream API* on page 491.

The response will include one of the following HTTP codes:

- 201 // story successfully created
- 401 // authorization failed
- 404 // story user not found
- 400 // other error, including XML syntax errors

The response body uses the following syntax:

```
<soc:activity xmlns:soc="http://social.bea.com/activity">
    <body>post content</body>
    <userId>4258</userId>
    <senderFullName>Joe King</senderName>
    <senderIP>10.60.28.195</senderIP>
    <userUUID>E523D1A7-475E-0B4D-76CA-6F9001480000</userUUID>
    <userFullName>Joe King</userFullName>
    <version>v1</version>
</soc:activity>
```

The post can be implemented in many ways, as shown in the examples that follow.

HTML/JavaScript Example

The simple HTML/JavaScript example below posts a story entered by the user.

```
<script type="text/javascript">
 // Get the gatewayed URL - This will give us authentication and
 prevent cross domain POST errors.
 var activitystreamBaseURL = "<pt:common.url
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'
pt:href='http://bfraser03.bea.com:21030/activityservice/api/story/v1/activities/'/>";

 // Get the sender's full name.
 var senderName = "<pt:common.userinfo pt:info='FullName'
xmlns:pt='http://www.plumtree.com/xmlschemas/ptui/'/>";

 var portalURL = "http://bfraser03.bea.com:8080/portal/server.pt";


 function sendStory() {
  var to = document.getElementById('to').value;
  var message = linkedName() + ' wrote: '
    + document.getElementById('message').value;

  doRestCall(
    'POST',
    activitystreamBaseURL + 'username/' + to,
    '<?xml version="1.0" encoding="UTF-8"?>'
     + '<soc:activity xmlns:soc="http://social.bea.com/activity">'

      + '<body><![CDATA[' + message + ']]></body>'
      + '</soc:activity>');
 }
```

```
 function doRestCall(requestType, postURL, xml) {
  var xmlhttp;
  // Get the XMLHttpRequest object -- needs to work for IE and
non-IE browsers
  if (window.XMLHttpRequest) {
   xmlhttp = new XMLHttpRequest();
  } else if (window.ActiveXObject) {
   xmlhttp = new ActiveXObject("Microsoft.XMLHttp");
  }
  // Set the callback as an anonymous funcion
  xmlhttp.onreadystatechange = function() {
   if (xmlhttp.readyState == 4) {
    var str = '<div>';
    // We should get 201 CREATED on successful POST
    if (xmlhttp.status == 201) {
     str = str + 'Message sent!';
    } else {
     str = str + 'ERROR: ' + xmlhttp.status + '</div><div>'
       + xmlhttp.responseText + '</div>';
    }
    document.getElementById('output').innerHTML = str;
   }
  }
  xmlhttp.open(requestType, postURL, true);
  xmlhttp.setRequestHeader("Content-Type", "text/xml");
  xmlhttp.send(xml);
 }

 function linkedName() {
  // This function returns viewing userÕs name hyperlinked to
their homepage.
  var nameurl = senderName.replace(" ", "_");
  nameurl = nameurl.toLowerCase();
  var link = '<a href="' + portalURL + '/user/' + nameurl + '">'
    + senderName + '</a>';
  return link;
 }
</script>
<div>Send a message to: <input id="to" type="text"
size="30"/></div>
<div>Message: <input id="message" type="text" size="40" /></div>
<div><a href="#" onclick="sendStory();"> send it! </a></div>
<div id="output"></div>
```

Java Example

The Java example below sends the message “Check out the BEA home page!”.

ActivityStreamUtil.java

```
/* Copyright (c) 2008, BEA Systems, Inc. All rights reserved. */

package bea.sample.activitystream;

import java.io.UnsupportedEncodingException;

import org.apache.commons.httpclient.*;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.*;

public class ActivityStreamUtil {

 public static HttpClient CreateAuthenticatedClient(String
username,
   String password) {
  HttpClient client = new HttpClient();

  // This is promiscuous be careful
  client.getState().setCredentials(AuthScope.ANY,
    new UsernamePasswordCredentials(username, password));
  // Send credentials with request without waiting for challenge
  client.getParams().setAuthenticationPreemptive(true);

  return client;
 }

 public static PostMethod CreateActivityStreamPost(String url,
String body) {
  PostMethod postMethod = new PostMethod(url);
  RequestEntity requestEntity = null;
  try {
   requestEntity = new
StringRequestEntity("<activity><body><![CDATA["
     + body + "]]></body></activity>", "text/xml", "UTF-8");
  } catch (UnsupportedEncodingException e) {
   e.printStackTrace();
  }
  postMethod.setRequestEntity(requestEntity);
  return postMethod;
```

```
 }
}
```

ActivityStreamPost.java

```
/* Copyright (c) 2008, BEA Systems, Inc. All rights reserved. */

package bea.sample.activitystream;

import java.io.IOException;

import org.apache.commons.httpclient.*;
import org.apache.commons.httpclient.methods.PostMethod;

public class ActivityStreamPost {
 public static final String ACT_USERNAME = "username";
 public static final String ACT_ID = "id";
 public static final String ACT_UUID = "uuid";

 public static void main(String[] args) {
  // Post to the guest user by name from the administrator account.

  String host = "localhost:21030";
  String userIDType = ACT_USERNAME;
  String userID = "guest";
  String username = "administrator";
  String password = "admin";
  String url = "http://" + host
    + "/activityservice/api/story/v1/activities/" + userIDType
    + "/" + userID;
  String message = "Check out the <a
href=\"http://www.bea.com\">BEA</a> home page!";
  HttpClient client = ActivityStreamUtil.CreateAuthenticatedClient(

    username, password);
  PostMethod post =
ActivityStreamUtil.CreateActivityStreamPost(url,
    message);
  try {
   int status = client.executeMethod(post);
   if (status == HttpStatus.SC_CREATED) {
    System.out.println("Post successful");
    System.out.println(post.getResponseBodyAsString());
   } else {
    System.err.println("Method failed: " + post.getStatusLine());
```

```
    }
   } catch (HttpException e) {
    System.err.println("Fatal protocol violation: " +
e.getMessage());
    e.printStackTrace();
   } catch (IOException e) {
    System.err.println("Fatal transport error: " + e.getMessage());

    e.printStackTrace();
   } finally {
    post.releaseConnection();
   }
  }
}
```

.NET Example

The .NET (C#) example below sends the message 'Greetings from .NET!".

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.IO;

namespace ActivityServiceTest
{
    class Program
    {
        static void Main(string[] args)
        {
            string url =
"http://localhost:8090/activityservice/api/story/v1/activities/username/bea/nsuravar";

            string username = "test";
            string password = "plumtree";
            string data = "<soc:activity
xmlns:soc=\"http://social.bea.com/activity\"><body>Greetings from
 .NET!</body></soc:activity>";

            HttpWebRequest request =
(HttpWebRequest)WebRequest.Create(url);
            request.Method = "POST";
           request.Credentials = new NetworkCredential(username,
 password);
            request.ContentLength = data.Length;
```

```
            request.KeepAlive = false;
            System.Net.ServicePointManager.Expect100Continue =
false;
            Uri uri = new Uri(url);

            NetworkCredential Credentials =
request.Credentials.GetCredential(uri, "Basic");

            request.ContentType = "text/xml;charset=UTF-8";

            if (Credentials != null)
            {
                byte[] credentialBuffer = new
UTF8Encoding().GetBytes(
                    Credentials.UserName + ":" +
                    Credentials.Password);
                request.Headers["Authorization"] =
                    "Basic " +
Convert.ToBase64String(credentialBuffer);
            }

            StreamWriter writer = new
StreamWriter(request.GetRequestStream());
            writer.Write(data);
            writer.Close();

            try
            {
                WebResponse response = request.GetResponse();

              byte[] ByteArray = response.Headers.ToByteArray();

                using (StreamReader reader = new
StreamReader(response.GetResponseStream()))
                {
                    while (reader.Peek() != -1)
                    {
                        Console.WriteLine(reader.ReadLine());
                     }
                }
            }
            catch (Exception ex)
            {
                Console.Write(ex.ToString());
            }
```

```
            }
        }
}
```

Configuring Web Services that Use the ALI Activity Stream API

To send the portal login token or basic authentication information to a remote application that uses the ALI Activity Stream API, register the remote application as a Web Service in the portal.

To use the ALI Activity Stream API, a remote application must have access to the portal login token or a portal user's basic authentication information. To send this information to the remote application, create a Web Service in the portal for the remote application and configure the following settings. (For detailed information on Web Service configuration settings, see the portal online help.)

- On the **Main Settings** page, enter the location of the remote application.
- On the **HTTP Configuration** page of the Web Service editor, add the root folder(s) for the remote application to the list of **Gateway URL Prefixes**. Gateway URL prefixes must include a trailing slash ("http://MyServer/") and are case sensitive. For details on the gateway, see *About Server Communication and the Gateway* on page 40.

  Note: All folders or pages that communicate with the Activity Stream API must be gatewayed. This includes the path to the service and any pages that are accessed by the service (for example, cross-domain scripting with XMLHttpRequest).

- To send the portal login token, on the **Advanced Settings** page, select the **Send Login Token** option. Configure the Login Token duration or leave it at the default. (Alternatively, you can choose to send the user's basic authentication information as described next.)
- To send the user's basic authentication information, on the **Authentication Settings** page, select the **User's Basic Authentication Information** option. (Alternatively, you can choose to send the portal login token as described in the previous bullet.)

Note: To use the Activity Stream API from a remote web service, the Remote Portlet Service must be installed and enabled. The Remote Portlet Service and Activity Stream sample portlets are included in the activityservice.pte file provided with the ALI installation package. For details

on installation and configuration, see the *AquaLogic Interaction Installation Guide* and the *AquaLogic Interaction Administrator Guide*.

About the Pathways API

The Pathways API allows a client application to query and manage Pathways objects through simple HTTP requests.

The Pathways API is a REST-based API. All requests are executed through HTTP using specific URL patterns and standard parameters.

The Pathways API supports the representation of request and response payloads in two formats: JSON (JavaScript Object Notation) or XML. HTTP requests can designate this preference in the Accept: header (Accept: application/xml or Accept: application/json). Additionally, GET type requests allow this preference to be passed as a query string argument (format=XML or format=JSON). The JSON format might be easier to work with in the context of a JavaScript call, while XML is usually preferable from application server code. For example, the responses below are returned from GET /pathways/api/privileges?format=XML and GET /pathways/api/privileges?format=JSON respectively.

XML response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PPrivileges xmlns:soc="http://social.bea.com/pathways">
 <access>HAS_PORTAL_ACTIVITY_ACCESS_PATHWAYS</access
 ><access>CAN_VIEW_TAGS</access><
 access>HAS_APPLICATION_WRITE</access>
</soc:PPrivileges>
```

JSON response:

```
{"soc$PPrivileges":{"@xmlns":{"soc":"http:\/\/social.bea.com\/pathways"},
"access":[{"access":"HAS_PORTAL_ACTIVITY_ACCESS_PATHWAYS"},
```

```
{"access":"CAN_VIEW_TAGS"},
{access":"HAS_APPLICATION_WRITE"}]}}
```

The base URL for all requests is the URL of the Remote Server object defined for the Web Service in the portal, plus "/pathways/api/". (For details on configuring a Remote Server, see the portal online help.)

- For a complete list of URL patterns for specific requests, see *Pathways API Interfaces* on page 494.
- For examples of common requests, see the following topics:
  - *Retrieving Data Using the Pathways API* on page 506
  - *Executing Searches Using the Pathways API* on page 503
  - *Creating and Managing Saved Searches Using the Pathways API* on page 514
  - *Creating and Managing Tags Using the Pathways API* on page 510
  - *Creating and Managing Views Using the Pathways API* on page 516

Note: All HTTP requests to the Pathways API must include security credentials. There are two ways of providing these credentials:

- To use the standard REST API URL in the form of http://server/pathways/api/items, you must provide a portal login token. The login token can be borrowed from a portal-to-portlet request or retrieved using the IDK. For details on configuring the portal to send a login token, see *Configuring Web Services that Use the Pathways API* on page 517. For details on using the IDK to access a login token, see *Initiating a PRC Session to Use IDK Remote APIs* on page 55.
- To use the /direct/ URL pattern (http://server/pathways/api/direct/items), you must provide username/password credentials in a traditional basic authentication format. For details on configuring the portal to send basic authentication information, see *Configuring Web Services that Use the Pathways API* on page 517.

For details on Pathways objects, see the Pathways documentation.

Pathways API Interfaces

The Pathways API includes a collection of interfaces for accessing and managing items, users, views, tags, and saved searches.

Function	URL (Interface)	HTTP Method	Method Parameters
Retrieve the version of the Pathways API.	/pathways/api/version	GET	
Retrieve suggested tags.	/pathways/api/suggest/[tag-prefix]	GET	
Retrieve tags created after a specific date.	/pathways/api/tags/created-after	GET	`date=[encoded date]`
Retrieve tags applied to a specific item.	/pathways/api/item/[id]/tags	GET	

Remove a tag from a specific item.	/pathways/api/item/[id]/tag	DELETE	`<soc:PTagList xmlns="http://schemas.com/pathways">`
			`<tag>`
			`<value>TagToRemove</value>`
			`</tag>`
			`</soc:PTagList>`

Add a tag to a specific item.	/pathways/api/item/[id]/tag	POST	`<soc:PTagList xmlns="http://schemas.com/pathways">`
			`<tag>`
			`<value>TagToAdd</value>`
			`</tag>`
			`</soc:PTagList>`

Retrieve a tag cloud for a search	/pathways/api/tags/[viewid] [viewid] must be one of the following: • "my" for the current user's tags • "ev" for everyone's tags • a custom view ID	GET	Search query. For a list of query parameters, see *Pathways API Search Query Parameters* on page 500.

Function	URL (Interface)	HTTP Method	Method Parameters
Retrieve the list of saved searches.	/pathways/api/savedsearches	GET	
Retrieve the details of an existing saved search.	/pathways/api/savedsearch/[id]	GET	
Remove an existing saved search	/pathways/api/savedsearch/[id]	DELETE	

Function	URL (Interface)	HTTP Method	Method Parameters
Edit an existing saved search.	/pathways/api/savedsearch/[id]	PUT	<soc:PSavedSearchesList xmlns:soc="http://socialcomputing/pathways"> <savedSearch> <creationDate>2007-09-06T16:52:00.0</creationDate> <dailyEmail>false</dailyEmail> <ownerId>PROFILE:1:1</ownerId> <queryString>Tabloid News</queryString> <rssFormat>false</rssFormat> </savedSearch> </soc:PSavedSearchesList>
Create a new saved search.	/pathways/api/savedsearch	POST	<soc:PSavedSearchesList xmlns:soc="http://socialcomputing/pathways"> <savedSearch> <creationDate>2007-09-06T16:52:00.0</creationDate> <dailyEmail>false</dailyEmail> <ownerId>PROFILE:1:1</ownerId> <queryString>Tabloid

Function	URL (Interface)	HTTP Method	Method Parameters
			News</queryString> <rssFormat>false</rssFormat> </savedSearch> </soc:PSavedSearchesList>
Get the list of existing views.	/pathways/api/views	GET	
Get the details of a specific view.	/pathways/api/view/[id]	GET	
Remove an existing view.	/pathways/api/view/[id]	DELETE	
Edit an existing view.	/pathways/api/view/[id]	PUT	<soc:PViewDetail <name>Marketing view</name> </soc:PViewDetail>
Create a new view.	/pathways/api/view	POST	Expressed in XML payload.name=[view name]members=[comma separated list of user/group ids]subscribers=[comma separated list

Function	URL (Interface)	HTTP Method	Method Parameters
			`of user/group ids]`
Get a list of items.	/pathways/api/items	GET	Search query. For a list of query parameters, see *Pathways API Search Query Parameters* on page 500.
Get details of a specific item.	/pathways/api/item/[ptsource]/[id]	GET	
Retrieve the favorite state for an item or contact state for a user.	/pathways/api/isfavorite/[id]	GET	
Modify the favorite state for an item or contact state for a user.	/pathways/api/isfavorite/[id]	PUT	`<soc:PIsItemFavorite xmlns:soc="http://schemas.com/pathways"> <result>true</result> </soc:PIsItemFavorite>`
Get a list of experts.	/pathways/api/people	GET	Search query. For a list of query parameters, see *Pathways API Search Query Parameters* on page 500.
Get details of a specific expert.	/pathways/api/people/[id]	GET	
Get access privileges for a specific user.	/pathways/api/privileges	GET	

PUT and DELETE methods can be overloaded by using a POST method with specific headers to resolve issues with browser support for asynchronous PUT and DELETES.

- `X-HTTP-Method-Override: PUT` can be used in a POST request header to signal that a PUT operation is intended.
- `X-HTTP-Method-Override: DELETE` can be used in a POST request header to signal that a DELETE operation is intended.

All requests should include the parameter `format=XML`.

Pathways API Search Query Parameters

The Pathways API provides a set of standard query parameters to execute search queries.

Query Parameter	Purpose	Valid Values	Default Value
order	Defines the order of the search results.	• "ASC" — Ascending • "DES" — Descending	“DES” (Descending)
orderBy	Designates result field for sorting.	• "RANK" • "CONTENTTYPE" • "CONTENTURL" • "DESCRIPTION" • "DETAILSURL" • "LAST_MODIFIED • "NAME" • "PATH" • "PROJECT" • "PROJECTURL" • "TAGS"	“RANK” (The search score for each item.)
tagCount	Defines the maximum number of tags to return. (Applies only to tag queries.)	int	50

Query Parameter	Purpose	Valid Values	Default Value
startAt	Offsets starting result within complete result set.	0 > int < total result count	1
resultCount	Defines the maximum number of results to return.	int > 0	10
query	Establishes the search query itself.	Any terms with the existing search syntax	(everything)
viewId	Limits result set to an existing View.	Existing View ID	(everyone)
resultType	Limits result set to items of a particular type.	• "ALL" • "DOCS_ONLY" • "USERS_ONLY"	"ALL" (both documents and users)
onlyFavorites	Limits result set to Favorites/Contacts.	"true" or "false"	"false"
kdFolders	Limits result set to items in specific Knowledge Directory folders.	Comma delimited list of folder IDs	(all — no constraint)
adminFolders	Limits result set to items in specific Administrative folders.	Comma delimited list of folder IDs	(all — no constraint)
collabProjects	Limits result set to items in specific Collaboration projects.	Comma delimited list of project IDs	(all — no constraint)
newerThan	Limits search result to items created since a specific date.	URL-encoded short form date ("11/7/2007")	(all — no constraint)

Query Parameter	Purpose	Valid Values	Default Value
objectTypes	Restricts search to specific object types.	Comma delimited list of any of the following strings: • AdminFolders • AuthenticationSources • CollabDiscussion • CollabDocument • CollabEvent • CollabMessage • CollabProject • CollabTask • Communities • CommunityPages • CommunityTemplates • ContentSources • ContentTypes • Crawlers • DocumentFolders • ExperienceDefinitions • ExternalOperations • FederatedSearches • Filters • Groups • Invitations • Jobs • PageTemplates • PortalDocuments • Portlets • PortletTemplates • ProfileSources • Properties • PublisherItems • RemoteServers	(all — no constraint)

Query Parameter	Purpose	Valid Values	Default Value
		• SnapshotQueries • Users • WebServices • FromUserPrefs (object types from Pathways user preferences) • FromAdminPrefs (object types pulled from Pathways admin preferences)	
propertyIDs	Designates indexed properties by ID to include with each search result.	Comma delimited property IDs (for example, “210,215” or “PT210,PT215”)	ID, Name, Description, Folder, URL, Last Modified, Activity Rank
propertyNames	Designates indexed properties by name to include with each search result.	Comma delimited property names (for example, “Office,Sales+Region”)	ID, Name, Description, Folder, URL, Last Modified, Activity Rank

Executing Searches Using the Pathways API

To search for items, users, and tags, use a search interface with standard query parameters.

The Pathways API allows you to search for items in the portal, Collaboration, or Publisher, as well as for users (experts). All search queries use a standard set of query parameters. For a detailed list of parameters, see *Pathways API Search Query Parameters* on page 500. You can also retrieve specific items or lists of items without using a search query; for details, see *Retrieving Data Using the Pathways API* on page 506. To create and manage saved searches, see *Creating and Managing Saved Searches Using the Pathways API* on page 514.

- To search for items, use GET /pathways/api/items. You can search for items in the portal Knowledge Directory and Administration, Collaboration, and Publisher. To restrict the search to specific types of items, use the objectTypes query parameter.
- To search for experts, use GET /pathways/api/people.
- To retrieve a tag cloud for search, use GET /pathways/api/tags.

For example, the request below submits a search for items containing the text “Planetary Science” with the result set constrained to two items:

```
GET
/pathways/api/items?format=XML&query=planetary+science&resultCount=2
```

The response contains the search results, as shown in the example below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PResultList xmlns:soc="http://social.bea.com/pathways">
    <item>
        <activityRank>0.0</activityRank>
        <classId>18</classId>
       <description>...featured a hit piece on Dr. S. Fred Singer,
 the esteemed Professor Emeritus of environmental &lt;span
class=&quot;ptHighlight&quot;&gt;science&lt;/span&gt; at the
University of Virginia...featured a hit piece on Dr. S. Fred
Singer, the esteemed Professor Emeritus of environmental &lt;span
 class=&quot;ptHighlight&quot;&gt;science&lt;/span&gt; at the
University of Virginia. In a segment disgracefully entitled
&amp;quot;Welcome to 'The Denial Machine...scientists that disagree
 with what your saying. The IPCC, NASA, NOAA, the National Academy
 of &lt;span
class=&quot;ptHighlight&quot;&gt;Sciences&lt;/span&gt;, the
American Association for the Advancement of Science, the American
 Geophysical Union, the American Meteorological...cyclicity in
varves and turbidites from the Arabian Sea: hypothesis of tidal
origin (Global and &lt;span
class=&quot;ptHighlight&quot;&gt;Planetary&lt;/span&gt; Change,
Volume 34, Issues 3-4, Pages 313-325, November 2002) - W. H.
Bergera, U...M. Kauffman, Ph.D. Organic Chemistry, MIT, USA Joel
 Schwartz, B.S. Chemistry, M.S. &lt;span
class=&quot;ptHighlight&quot;&gt;Planetary&lt;/span&gt; &lt;span
 class=&quot;ptHighlight&quot;&gt;Science&lt;/span&gt;, California
 Institute of Technology, USA John Brignell, Ph.D. Professor
Emeritus, Department of Electronics &amp;amp;
Computer...</description>
        <folderClassId>17</folderClassId>
        <folderId>200</folderId>
```

```
        <folderName>\Knowledge Directory\Test</folderName>

<iconLink>plumtree/portal/public/img/sml{52a5a5a0-84c9-11d2-a0c5-0060979c42d8}.gif</iconLink>

        <isFavorite>false</isFavorite>
        <lastModified>2008-03-24T05:38:00-07:00</lastModified>
       <name>ABC's Global Warming Hit Piece &quot;Welcome to 'The
 Denial Machine'&quot; | NewsBusters.org</name>
        <objectId>8267</objectId>
        <path>\Knowledge Directory\Test</path>
        <rank>1</rank>
        <resultScore>2.84252</resultScore>
        <source>PTPORTAL</source>
        <type>Document</type>

<URL>http://snahendr03.amer.bea.com:8080/portal/server.pt?open=18&amp;objID=8267&amp;mode=2&amp;in_hi_userid=1</URL>

<viewURL>http://snahendr03.amer.bea.com:8080/portal/server.pt?objID=8267&amp;open=18&amp;mode=3&amp;space=Space&amp;space=Space</viewURL>

    </item>
    <item>
        <activityRank>0.0</activityRank>
        <classId>18</classId>
        <description>...Coverage of &lt;span
class=&quot;ptHighlight&quot;&gt;science&lt;/span&gt; by
SignOnSanDiego.com and The San Diego Union-Tribune...Pension Crisis
 Special Reports Video Multimedia Photo Galleries Topics Education
 Features Health | Fitness Military Politics &lt;span
class=&quot;ptHighlight&quot;&gt;Science&lt;/span&gt; Solutions
Opinion Columnists Steve Breen Forums Weblogs Communities U-T
South County U-T East...asteroid to determine whether it is on a
 potentially disastrous collision course with Earth, The &lt;span
 class=&quot;ptHighlight&quot;&gt;Planetary&lt;/span&gt; Society
 announced Tuesday. WASHINGTON, 2:01 p.m. Feb. 25 (AP) Limiting
fish catch to...Museum of Man San Diego Natural History Museum
SDSU College of Engineering SDSU College of &lt;span
class=&quot;ptHighlight&quot;&gt;Sciences&lt;/span&gt; San Diego
 Supercomputer Center San Diego Zoological Society Scripps
Institution of Oceanography Scripps Research
Institute...</description>
        <folderClassId>17</folderClassId>
        <folderId>200</folderId>
        <folderName>\Knowledge Directory\Test</folderName>
```

```
<iconLink>pluntree/portal/public/img/sml{52a5a5a0-84c9-11d2-a0c5-0060979c42d8}.gif</iconLink>

        <isFavorite>false</isFavorite>
        <lastModified>2008-03-24T05:41:00-07:00</lastModified>
        <name>SignOnSanDiego.com &gt; News &gt; Science</name>
        <objectId>9238</objectId>
        <path>\Knowledge Directory\Test</path>
        <rank>2</rank>
        <resultScore>2.68074</resultScore>
        <source>PTPORTAL</source>
        <type>Document</type>

<URL>http://snahendr03.amer.bea.com:8080/portal/server.pt?open=18&amp;objID=9238&amp;mode=2&amp;in_hi_userid=1</URL>

<viewDetailURL>http://snahendr03.amer.bea.com:8080/portal/server.pt?objID=9238&amp;open=18&amp;mode=3&amp;space=Space&amp;parentspace=Space</viewDetailURL>

    </item>
    <totalCount>102</totalCount>
</soc:PResultList>
```

Retrieving Data Using the Pathways API

To retrieve data about items, views, saved searches and users (experts), use one of the interfaces provided by the Pathways API.

The Pathways API provides a collection of interfaces to retrieve data.

- To retrieve details about a specific item in the portal, Collaboration, or Publisher, use GET /pathways/api/item/[ptsource]/[id]. For example, the following request retrieves details on a portal item with an object ID of 201: `GET /pathways/api/item/PTPORTAL/18_201?format=XML`.

  The response contains detailed information about the item:

  ```
  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <soc:PItemDetail xmlns:soc="http://social.bea.com/pathways">
      <type>PTPORTAL</type>
      <classId>18</classId>
      <objectId>201</objectId>
      <name>APOD: 2007 October 12 - The Whale and the Hockey
  Stick</name>
  ```

```
    <description>A different astronomy and space science related
 image is featured each day, along with a brief
explanation.</description>
    <lastModified>2008-02-25T02:33:00-08:00</lastModified>

<URL>http://sahendf3.aver.bea.com:8080/portal/server.pt?open=18&amp;objID=201&amp;mode=2&amp;in_hi_userid=1</URL>


<viewDetailURL>http://sahendf3.aver.bea.com:8080/portal/server.pt?objID=201&amp;open=18&amp;mode=3&amp;space=Space&amp;cached=true&amp;spaceID=2</viewDetailURL>

    <path>\Knowledge Directory\Test</path>
    <folderId>200</folderId>
    <folderName>\Knowledge Directory\Test</folderName>
    <property>
        <id>PT8</id>
        <name>Content Language</name>
        <value>en</value>
    </property>
    <property>
        <id>PT6</id>
        <name>Content Type ID</name>
        <value>106</value>
    </property>
    <property>
        <id>PT2</id>
        <name>Description</name>
        <value>A different astronomy and space science related
 image is featured each day, along with a brief
explanation.</value>
    </property>
    <property>
        <id>PT1</id>
        <name>Name</name>
        <value>APOD: 2007 October 12 - The Whale and the Hockey
 Stick</value>
    </property>
    <property>
        <id>PT3</id>
        <name>Object Created</name>
        <value>4232743.0</value>
    </property>
    <property>
        <id>PT4</id>
        <name>Object Last Modified</name>
        <value>4287993.0</value>
```

```
    </property>
    <property>
        <id>PT5</id>
        <name>Open Document URL</name>
        <value>http://apod.nasa.gov/apod/ap071012.html</value>

    </property>
</soc:PItemDetail>
```

You can also find out whether a specific item has been marked as a favorite using GET /pathways/api/isfavorite/[id]. For details on searching for items, see *Executing Searches Using the Pathways API* on page 503.

- To retrieve details about an existing view by ID, use GET /pathways/api/view/[id]. For example, `GET /pathways/api/view/8aa289731303557b0113086b1ece0003?format=XML`. (To retrieve a list of existing saved searches, use GET /pathways/api/views?format=XML.)

  The response contains detailed information about the view:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PViewDetail xmlns:soc="http://social.bea.com/pathways">
    <created>2007-06-07T16:00:44.367-07:00</created>
    <id>8aa289731303557b0113086b1ece0003</id>
    <lastModified>2008-04-03T08:30:35.807-07:00</lastModified>

    <members>PTPORTAL:1:203</members>
    <name>Marketing view</name>
    <ownerId>PTPORTAL:1:1</ownerId>
    <published>true</published>

<subscribers>PTPORTAL:1:203,PTPORTAL:1:204,PTPORTAL:1:1</subscribers>
</soc:PViewDetail>
```

  For details on creating new views, see *Creating and Managing Views Using the Pathways API* on page 516.

- To retrieve details about an existing saved search by ID, use GET /pathways/api/savedsearch/[id]. For example, `GET /pathways/api/savedsearch/8aa289731587336f011587345c110001?format=XML`. (To retrieve a list of existing saved searches, use GET /pathways/api/savedsearches?format=XML.)

  The response contains detailed information about the saved search:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PSavedSearchesList
```

```
xmlns:soc="http://social.bea.com/pathways">
    <savedSearch>
        <created>2007-10-09T16:58:16.593-07:00</created>
        <dailyEmail>true</dailyEmail>
        <id>8aa289731587336f011587345c110001</id>
        <language>en</language>

<lastModified>2008-02-22T16:07:32.583-08:00</lastModified>
        <ownerId>PTPORTAL:1:1</ownerId>
        <queryString>news</queryString>
        <rssFormat>false</rssFormat>
        <timeZoneId>America/Los_Angeles</timeZoneId>
    </savedSearch>
</soc:PSavedSearchesList>
```

For details on creating new saved searches, see *Creating and Managing Saved Searches Using the Pathways API* on page 514.

- To retrieve details about an existing user (expert) by ID, use GET /pathways/api/people/[id]. For example, `GET /pathways/api/people/PTPORTAL/1_201?format=XML`.

  The response contains detailed information about the user:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PItemDetail xmlns:soc="http://social.bea.com/pathways">
    <type>PTPORTAL</type>
    <name>user1</name>
    <description></description>
    <lastModified>2008-01-14T07:34:00-08:00</lastModified>

<URL>http://smahendr03.amer.bea.com:8080/portal/server.pt?open=1&amp;objID=201&amp;mode=2&amp;in_hi_userid=1</URL>

    <path>\Admin Objects Directory\Test Resources</path>
    <folderId>306</folderId>
    <folderName>\Admin Objects Directory\Test
Resources</folderName>
    <property>
        <id>PT201</id>
        <name>Secret Property</name>
        <value></value>
    </property>
    <property>
        <id>PT156</id>
        <name>Company</name>
        <value></value>
    </property>
```

```
    <property>
        <id>PT152</id>
        <name>Phone Number</name>
        <value></value>
    </property>
    <property>
        <id>PT26</id>
        <name>Email Address</name>
        <value></value>
    </property>
</soc:PItemDetail>
```

You can retrieve the current user's access privileges using GET /pathways/api/privileges. For example, `GET /pathways/api/privileges?format=XML`. The response lists all available privileges:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PPrivileges xmlns:soc="http://social.bea.com/pathways">
    <access>HAS_PORTAL_ACTIVITY_ACCESS_PATHWAYS</access>
    <access>CAN_VIEW_TAGS</access>
    <access>CAN_VIEW_PEOPLE</access>
    <access>CAN_ADD_TAGS</access>
    <access>CAN_CREATE_VIEWS</access>
    <access>CAN_PRESCRIBE_VIEWS</access>
    <access>HAS_MY_VIEW</access>
    <access>HAS_EVERYONE_VIEW</access>
    <access>CAN_ADD_FAVORITES_CONTACTS</access>
    <access>CAN_SET_USER_PREFS</access>
    <access>CAN_ACCESS_ENHANCED_UI</access>
    <access>CAN_SAVE_SEARCHES</access>
    <access>HAS_APPLICATION_WRITE</access>
</soc:PPrivileges>
```

You can also find out whether a specific expert has been marked as a contact of the current user, use GET /pathways/api/isfavorite/[id].

Creating and Managing Tags Using the Pathways API

To create new tags and assign tags to specific items, use the /pathways/api/item/[id]/tag interface.

The Pathways API provides a collection of interfaces to access and manage tags.

- To retrieve tags that contain a specific string, use GET /pathways/api/suggest/[tag-prefix]. For example, the following request retrieves tags that start with “port”: `GET /pathways/api/suggest/port?format=xml`.
- To retrieve a weighted tag cloud, use GET /pathways/api/tags/[viewid] where [viewid] is “my” for the current user's tags, “ev” for everyone's tags, or a custom view ID.
- To retrieve tags applied to a specific item, use GET /pathways/api/item/[id]/tags. For example, the following request retrieves tags associated with the portal item with ID 1985: `GET /pathways/api/item/PTPORTAL/18_1985/tags?format=XML`.

  The response provides a list of tags:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PTagList xmlns:soc="http://social.bea.com/pathways">
    <tag>
        <count>1</count>
        <isAppliedByCurrentUser>true</isAppliedByCurrentUser>
        <normalizedCount>9</normalizedCount>
        <value>phenomenon</value>
    </tag>
    <tag>
        <count>1</count>
        <isAppliedByCurrentUser>true</isAppliedByCurrentUser>
        <normalizedCount>9</normalizedCount>
        <value>sudden</value>
    </tag>
</soc:PTagList>
```

- To add a tag from a specific item, use POST /pathways/api/item/[id]/tag as shown in the example below.

  The following request creates tags ‘foo’ and ‘bar’ and assigns them to portal document 210:

```
POST /pathways/api/item/PTPORTAL/1_210/tags
CSP-Session-Token: 1|1207243836|0d6PFr/Z+mRrarbn9uhZJyYL2uI=
Host: smahendr03.amer.bea.com:11990
Accept-Language: en-us
ACCEPT: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PTagList xmlns:soc="http://social.bea.com/pathways">
    <tag>
        <value>foo</value>
    </tag>
    <tag>
        <value>bar</value>
```

```
    </tag>
</soc:PTagList>
```

The response confirms that the request was successful:

```
HTTP/1.1 201 Created

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PTagList xmlns:soc="http://social.bea.com/pathways">
    <tag>
        <value>foo</value>
    </tag>
    <tag>
        <value>bar</value>
    </tag>
</soc:PTagList>
```

This same request could be used with the HTTP DELETE method to remove these tags from the item. The successful response code would be 202 Accepted instead of 201 Created.

- To delete a tag from multiple items within a specific view, use DELETE /pathways/api/tag/[viewid]/[tag] where [viewid] is “my” for the current user's tags, “ev” for everyone's tags, or a custom view ID; and [tag] is the tag value.

  Users with WRITE_APPLICATION capability can delete tags that they applied or tags applied by other users. For example, an administrator could remove all tags corresponding to "acquisition" for everyone (the "ev" view) or, more narrowly, for another group view, or for a specifically identified user (“custom/[ptsource]/[userid]”).

  For example, the following request deletes tag ‘foo’ from all items in view ‘bar’: `DELETE /pathways/api/tag/bar/foo`.

  The response confirms that the request was successful:

```
HTTP/1.1 202 Accepted

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PTagViewStatus xmlns:soc="http://social.bea.com/pathways">

    <value>foo</value>
    <appliedItem>PTPORTAL:18:201</appliedItem>
    <appliedItem>PTPORTAL:18:203</appliedItem>
</soc:PTagViewStatus>
```

  In the example above, ‘foo’ was removed from Knowledge Directory items with object IDs of 201 and 203.

- To rename a tag applied to multiple items within a specific view, use PUT /pathways/api/tag/[viewid]/[tag] where [viewid] is “my” for the current user's tags, “ev” for everyone's tags, or a custom view ID; and [tag] is the tag value.

  Important: The "appliedItem" elements are returned in the response to the 'rename' request, but they should not be passed when making the request. If you add the “appliedItem” element to the request, an error is produced indicating how the call should be invoked.

  Users with WRITE_APPLICATION capability can rename tags that they applied or tags applied by other users. For example, an administrator could rename all “HR” tags to “human resources” for everyone (the "ev" view) or, more narrowly, for another group view, or for a specifically identified user (“custom/[ptsource]/[userid]”).

  For example, the following request renames tag ‘foo’ to ‘bar’ from all items in view ‘ev’:

```
PUT /pathways/api/tag/ev/foo
CSP-Session-Token: 1|1207243836|0d6PFr/Z+mRrarbn9uhZJyYL2uI=
Host: smahendr03.amer.bea.com:11990
Accept-Language: en-us
ACCEPT: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PTagViewStatus xmlns:soc="http://social.bea.com/pathways">

    <value>bar</value>
</soc:PTagViewStatus>
```

  The response confirms that the request was successful:

```
HTTP/1.1 202 Accepted

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PTagViewStatus xmlns:soc="http://social.bea.com/pathways">

    <value>bar</value>
    <appliedItem>PTPORTAL:18:201</appliedItem>
    <appliedItem>PTPORTAL:18:203</appliedItem>
</soc:PTagViewStatus>
```

  In the example above, the tag ‘bar’ replaced the tag ‘foo’ for Knowledge Directory items with object IDs of 201 and 203.

Creating and Managing Saved Searches Using the Pathways API

To create new saved searches and modify existing saved searches, use the /pathways/api/savedsearch interfaces.

The Pathways API provides a collection of interfaces to access and manage saved searches.

- To retrieve the list of existing saved searches, use GET /pathways/api/savedsearches.
- To retrieve details about a specific saved search, use GET /pathways/api/savedsearch/[id]. For details, see *Retrieving Data Using the Pathways API* on page 506.
- To create a new saved search, use POST /pathways/api/savedsearch/new. For example, the request below creates a new saved search for “gravity”.

```
POST /pathways/api/savedsearch/new
CSP-Session-Token: 1|1207329770|lKh5QpON3TQykMZ3a2x4ygh6Drs=
Accept-Language: en-us
ACCEPT: text/xml,application/xml

<?xml version="1.0" encoding="UTF-8"?>
<soc:PSavedSearchesList
xmlns:soc="http://social.bea.com/pathways">
  <savedSearch>
<created>2007-10-09T16:58:16.593-07:00</created>
<dailyEmail>true</dailyEmail>
<ownerId>PTPORTAL:1:1</ownerId>
<queryString>Gravity</queryString>
<rssFormat>false</rssFormat>
  </savedSearch>
</soc:PSavedSearchesList>
```

The response confirms that the saved search was created and includes the details passed in the request.

```
HTTP/1.1 201 Created
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PSavedSearchesList
xmlns:soc="http://social.bea.com/pathways">
    <savedSearch>
        <created>2007-10-09T16:58:16.593-07:00</created>
        <dailyEmail>true</dailyEmail>
```

```
        <ownerId>PTPORTAL:1:1</ownerId>
        <queryString>Gravity</queryString>
        <rssFormat>false</rssFormat>
    </savedSearch>
</soc:PSavedSearchesList>
```

- To modify an existing saved search, use PUT /pathways/api/savedsearch/[id] as shown in the example request below.

```
PUT /pathways/api/savedsearch/8aa28973157e6c3901157e74bab70001
CSP-Session-Token: 1|1207331374|2/XMqYsZj6DwW5b220VWnB2o4S8=
Accept-Language: en-us
ACCEPT: text/xml,application/xml

<?xml version="1.0" encoding="UTF-8"?>
<soc:PSavedSearchesList
xmlns:soc="http://social.bea.com/pathways">
  <savedSearch>
<created>2007-10-09T16:58:16.593-07:00</created>
<dailyEmail>true</dailyEmail>
<ownerId>PTPORTAL:1:1</ownerId>
<queryString>New Search Target</queryString>
<rssFormat>false</rssFormat>
  </savedSearch>
</soc:PSavedSearchesList>
```

  If the update was successful, the response code will be HTTP 202 Accepted. As with creating a new saved search, the response also includes the details passed in the request.

- To delete an existing saved search, use DELETE /pathways/api/savedsearch/[id]. There is no need to include details about the saved search, as shown in the example request below.

```
DELETE
/pathways/api/savedsearch/8aa28973191a6c1701191a6d903f0001
CSP-Session-Token: 1|1207330372|Ybc8ZmBc8vd7svvaa39CzZ5lp18=
Accept-Language: en-us
ACCEPT: text/xml,application/xml
```

  If the delete was successful, the response code will be HTTP 202 Accepted. The response also includes details about the saved search that was deleted, including the creation date.

Creating and Managing Views Using the Pathways API

To create new views and update existing views, use the /pathways/api/view interfaces.

The Pathways API provides a collection of interfaces to access and manage views.

- To retrieve the list of existing views, use GET /pathways/api/views.
- To retrieve details about a specific views, use GET /pathways/api/view/[id]. For details, see *Retrieving Data Using the Pathways API* on page 506.
- To create a new view, use POST /pathways/api/view/new. For example, the request below adds a new view called “Customer View”.

```
POST /pathways/api/view
CSP-Session-Token: 1|1207244890|EBNGlMeU4zFYKJIFbhIavCVk4/U=
Accept-Language: en-us
ACCEPT: text/xml,application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PViewDetail xmlns:soc="http://social.bea.com/pathways">
    <members>PTPORTAL:1:200,PTPORTAL:1:1</members>
    <name>Customer View</name>
    <ownerId>PTPORTAL:1:1</ownerId>
    <published>true</published>

<subscribers>PTPORTAL:1:201,PTPORTAL:1:1,PTPORTAL:2:201</subscribers>
</soc:PViewDetail>
```

The response confirms that the new view was created and includes the details passed in the request.

```
HTTP/1.1 201 Created
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PViewDetail xmlns:soc="http://social.bea.com/pathways">
    <created>2008-04-03T10:50:55.477-07:00</created>
    <id>8aa289731910767e01191569a5f30003</id>
    <lastModified>2008-04-03T10:50:55.507-07:00</lastModified>

    <members>PTPORTAL:1:200,PTPORTAL:1:1</members>
    <name>Customer View</name>
    <ownerId>PTPORTAL:1:1</ownerId>
    <published>true</published>
```

```
<subscribers>PTPORTAL:2:201,PTPORTAL:1:201,PTPORTAL:1:1</subscribers>
</soc:PViewDetail>
```

- To modify an existing view, use PUT /pathways/api/view/[id] as shown in the example request below.

```
PUT /pathways/api/view/8aa289731303557b0113086b1ece0003
CSP-Session-Token: 1|1207332050|KHt3F4KlgmlBuntcrEKQpFcQGq8=
Accept-Language: en-us
ACCEPT: text/xml,application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soc:PViewDetail xmlns:soc="http://social.bea.com/pathways">
<members>PTPORTAL:1:203</members>
 <name>Marketing view</name>
 <published>false</published>
<subscribers>PTPORTAL:1:203,PTPORTAL:1:204</subscribers>
</soc:PViewDetail>
```

  If the update was successful, the response code will be HTTP 202 Accepted. As with creating a new view, the response also includes the details passed in the request.

- To delete an existing view, use DELETE /pathways/api/view/[id]. There is no need to include details about the view, as shown in the example request below.

```
DELETE /pathways/api/view/8aa289731910767e01191569a5f30003
CSP-Session-Token: 1|1207330830|loGCMTzUMaJMBW4w22ACHoLUH0Y=
Accept-Language: en-us
ACCEPT: text/xml,application/xml
```

  If the delete was successful, the response code will be HTTP 202 Accepted. The response also includes details about the view that was deleted, including the creation date.

Configuring Web Services that Use the Pathways API

To send the portal login token or basic authentication information to a remote application that uses the Pathways API, register the remote application as a Web Service in the portal.

To use the Pathways API, a remote application must have access to the portal login token or a portal user's basic authentication information. To send this information to the remote application, create a Remote Server and a Web Service in the portal for the remote application and configure

the following settings. (For detailed information on Remote Server and Web Service configuration settings, see the portal online help.)

- On the **Main Settings** page, enter the location of the remote application.
- On the **HTTP Configuration** page of the Web Service editor, add the root folder(s) for the remote application to the list of **Gateway URL Prefixes**. Gateway URL prefixes must include a trailing slash ("http://MyServer/") and are case sensitive. For details on the gateway, see *About Server Communication and the Gateway* on page 40.

  Note: All folders or pages that communicate with the Pathways API must be gatewayed. This includes the path to the service and any pages that are accessed by the service.

- To send the portal login token, on the **Advanced Settings** page, select the **Send Login Token** option. Configure the Login Token duration or leave it at the default. (Alternatively, you can choose to send the user's basic authentication information as described next.)
- To send the user's basic authentication information, on the **Authentication Settings** page, select the **User's Basic Authentication Information** option. (Alternatively, you can choose to send the portal login token as described in the previous bullet.)

API Libraries

This page provides links to API libraries for use in ALUI development.

AquaLogic Interaction Development Kit

These API libraries provide detailed documentation on IDK objects and methods. To access documentation for previous versions or download the entire documentation package, download the appropriate version of the IDK. For details on using these APIs, see *About the IDK ALI Portlet API* on page 284 and *About the IDK Proxy API* on page 328.

IDK Version	Java Doc	NDocs
6.0	http://edocs.bea.com/aluidk/docs60/javadocs/index.html	http://edocs.bea.com/aluidk/docs60/ndocs/index.html
5.4	http://edocs.bea.com/aluidk/docs54/javadocs/index.html	http://edocs.bea.com/aluidk/docs54/ndocs/index.html

ALI Scripting Framework

The ALI Scripting Framework is a collection of client-side JavaScript libraries that provide services to portlets and hosted gatewayed pages. Most services are provided by the JSPortlet API. Opener functionality is provided by the Common Opener API. For lower-level functionality, see the JSXML package. For details on using these APIs, see *About the ALI Scripting Framework* on page 260.

Library	Browse	Download
JSPortlet jsdocs	http://edocs.bea.com/aluidk/docs60/jsapi/jsportlet/index.html	http://edocs.bea.com/aluidk/docs60/jsapi/JSPortletALIScriptingFramework.zip
Common Opener jsdocs	http://edocs.bea.com/aluidk/docs60/jsapi/jscommonopener/index.html	

Library	Browse	Download
JSXML jsdocs	[illegible]	[illegible]

Adaptive Tags

Adaptive Tags are XML tags that can be included in the markup returned by any gatewayed page, including portlets. These tags provide access to key portal components and support advanced attribute replacement. For details on using tags, see *About Adaptive Tags* on page 202.

Product/Version	Browse	Download
ALI 6.5	[illegible]	[illegible]
ALI 6.1 MP1	[illegible]	[illegible]
ALI 6.1	[illegible]	[illegible]
ALI 6.0	[illegible]	[illegible]
Ensemble 1.0	[illegible]	[illegible]

The following API documentation includes the classes used to create custom Adaptive Tags for use in portlets and gatewayed pages. For details on creating custom tags, see *Creating Custom Adaptive Tags* on page 252.

Language	Browse	Download
JavaDocs	[illegible]	[illegible]
NDocs	[illegible]	[illegible]

AL Analytics OpenUsage API

ALI Analytics 2.0 delivers detailed information about portal usage. The OpenUsage API allows you to raise Analytics events from custom portlets and applications and store them in the Analytics database. You can raise events using the OpenUsage Java API or an OpenUsage event tag. For details, see the dev2dev article *Using the ALI Analytics 2.0 OpenUsage API*.

Package	Browse	Download
JavaDocs	[illegible]	[illegible]
TagDocs	[illegible]	[illegible]

AquaLogic .NET Application Accelerator / .NET Portlet Toolkit

The AquaLogic .NET Application Accelerator is a collection of libraries and Visual Studio 2005 integration features that support easy authoring of ASP.NET 2.0 and WSRP portlets. Portlets can be authored for both BEA portal environments: AquaLogic Interaction (ALI) and WebLogic Portal (WLP). The .NET Application Accelerator includes the .NET Portlet Toolkit. A .chm file for the .NET Portlet API can be found here: *http://edocs.bea.com/en/alui/dotnetportlettoolkit/docs11/ndocs/PortletAPI.chm*.

Portal UI API

For links to portal UI API documentation for advanced portal customizations that require coding against the internal portal API, see *Portal API Documentation*. For details on portal UI customization, see *About AquaLogic Interaction 6.5 UI Customization* on page 425.

Additional Development References

The following references provide additional information for use in AquaLogic development.

CSP

CSP is a platform-independent protocol based on the open standard of HTTP 1.1. The syntax of communication between the portal and remote servers is defined by CSP. CSP defines custom headers and outlines how ALI services use HTTP to communicate and modify settings. The AquaLogic Interaction Development Kit (IDK) provides simplified, stable interfaces that allow you to write code that communicates using CSP.

- *CSP 1.3*
- *CSP 1.4*

AquaLogic .NET Application Accelerator / .NET Portlet Toolkit

The AquaLogic .NET Application Accelerator is a collection of libraries and Visual Studio 2005 integration features that support easy authoring of ASP.NET 2.0 and WSRP portlets. The AquaLogic .NET Application Accelerator includes the .NET Portlet Toolkit. Portlets can be authored for both BEA portal environments: AquaLogic Interaction (ALI) and WebLogic Portal (WLP). Development Guide are available for both environments:

- *AquaLogic Interaction Development Guide*
- *WebLogic Portal WSRP Development Guide*
- *.NET Portlet API documentation (.chm file)*.

Analytics APIs

AquaLogic Analytics delivers comprehensive reporting on activity and content usage within portals and composite applications, allowing you to know and meet user information needs. The OpenUsage and Query APIs provide access the Analytics functionality from custom applications.

- The Analytics OpenUsage API allows you to to raise Analytics events from custom portlets and applications and store them in the Analytics database. For details, see *Using the AquaLogic Interaction Analytics 2.0 OpenUsage API*.
- The Analytics Query API allows you to query data in the Analytics database. For details, see the *AquaLogic Analytics Query API Usage Guide* .

JSR-168 Container

The AquaLogic Interaction JSR-168 Container is an implementation of the JSR-168 JCP standard for portlet authoring. For details on developing portlets with the ALI JSR-168 Container 1.2, see the *JSR-168 Container Developer Guide*. For other information on the JSR-168 Container, see the *JSR-168 Container documentation index page*.