Oracle® Universal Records Management Adapter

Software Development Kit
10g Release 3 (10.1.3)

April 2007

This book describes how to develop custom Oracle Universal Records Management Adapters for external repositories

Primary Author: Buddy Robbins

Table of Contents

Table of Figures

Introduction

Purpose

This document provides the information necessary for third parties to construct custom Oracle Universal Records Management adapters for external repositories.

Audience

This document is intended for experienced Microsoft.Net developers. At a minimum, the developers should have experience constructing Windows Services, modifying configuration files, using external assemblies, and should have a firm understanding of object oriented fundamentals, particularly using interfaces and inheritance.

Construction of the File System Adapter Sample Application

Introduction

This section is dedicated to developing a trivial Universal Records Management adapter for providing record and retention management for a computer's file system. The adapter (FileSystemAdapter) is included with the software development kit, as well as the source code for its construction. Detailed instructions are provided in this document to enable a user to follow the construction logic.

Since this is intended as a trivial example, although the adapter will be functional, there are several concept flaws (which will be pointed out). Note that this is not a production-level application.

Adapter Behavior

The Adapter will be able to perform the following functions:

- **Self description.** The adapter supplies several classes to be used by Administration Utility. These classes define what metadata (names, data types, etc.) the adapter can supply to the Oracle Universal Records Management application, define windows controls for use in search creation, and define fields that can be searched.
- **Document declaration.** The Adapter supplies functionality to evaluate searching expressions, and use the results to get a list of documents and their metadata for submittal to the Oracle Universal Records Management application. The metadata supplied include: ItemID (unique identifier), Short Name (8.3 representation of full path), Long Name (full path), Machine Name, File Name, Author, URL, Extension, File Size, Creation Date, Modification Date.
- **Retention management on selected files.** Retention management is performed by collecting metadata belonging to the selected file and sending it to the Oracle Universal Records Management application. When the retention period has expired, the adapter will perform the appropriate disposition action (archive or delete) against the file.
- **Record management on selected files.** Record management ensures that a file cannot be modified or deleted. This is performed by collecting metadata belonging to the selected file and sending it to the Oracle Universal Records Management application, along with the file content itself. To ensure that changes cannot be performed on, or content deleted from the original item, the item itself is removed and an html stub is created containing a hyperlink to the Universal Record Management's content. This stub is placed in the same directory as the original item, and the item is removed.
- **Document freezing.** A litigation freeze issued from the Oracle Universal Records Management application causes the adapter to upload the original content (similar to record management) place an html stub in the original directory, and delete the original item.
- **Document thawing.** Following a litigation freeze, when the Oracle Universal Records Management application issues a thaw (releasing of a freeze), the adapter acknowledges that it receives the command, but does not perform any functions.
- **Document archiving.** Following an archive disposition issued by the Oracle Universal Records Management application, the adapter creates a copy of the original content, and tells the adapter framework the location of the copy. When the framework is finished processing the archiving, it will inform the adapter, and the adapter will delete the copy.
- **Document deletion.** Following a delete disposition issued by the Oracle Universal Records Management application, the adapter deletes the original content.

Design Issues

As stated earlier, this document provides only a sample of how to generate an adapter. The following is a non-exhaustive list of issues relative to the design:

- The URL generated in this application is used to allow Oracle Universal Records Management users to navigate to the original content on the adapter's computer. This is done with the file

protocol and utilizing an administrative share (i.e., FILE://MyComputer/c$/sample/example.txt). This URL will only function if administrative shares are enabled for the computer, and a common security account is used between the Oracle Content Server services and the adapter's computer. A better solution would be to create a control that implements IAdapterControl. This control could create hidden shares dedicated exclusively to the Oracle Universal Records Management adapter.

- The item identifier (ItemID) is required by the Oracle Universal Records Management to uniquely identify an item. To correlate a managed item with the item itself, the combination of machine name + "//" + Short Name (8.3 representation of full path) is used. The Oracle Universal Records Management application limits the item identifier to 100 characters. With a long machine name and a deep path, it is possible to overrun this limitation. A possible better solution would be to create a database table in the framework database that would hold a unique identifier (either a hash representation of the path or a totally opaque unique identifier) and the corresponding full path. When reporting to (or receiving commands from) the Oracle Universal Records Management system the unique identifier would be resolved to the full path.
- Retention-managed documents is can be allowed. When a user deletes a file from the adapter's computer, however, there is no feedback mechanism to inform the Oracle Universal Records Management application that the item has been deleted. A possible solution would be to instantiate an object of type System.IO.FileSystemWatcher to capture delete events. If the deleted object is a directory, the user could call Oracle.UrmAdapter.Core.UrmServices.DeleteSubDirectoryByUrl . If the deleted object is a file, the user could call Oracle.UrmAdapter.Core.UrmServices.DeleteByUrl.
- The control implementing IAdapterSelector iterates through all directories on all drives on the target machine. This currently is a very expensive (time-wise) operation. This could be solved by multithreading the discovery of the directories, or only searching only dedicated hidden shares (see the first design issue).

Construction steps

Create the Project

Using Microsoft Office 2005, create a Windows service application project called "FileSystemAdapter". The assembly name, as well as the default namespace will be FileSystemAdapter.
Add a reference to Oracle.UrmAdapter.dll.

Create the Service Class

Rename Service1.cs (the default service class) to FSAdapterService. This class will enable the adapter to run as a Windows service.

Create the Service Installer Class

Create a new class called FSAdapterServiceInstaller from the installer class template. This class will be used to allow for registering the service.

Create the FSAdapter Class

Create a new class called FSAdapter. It is recommended (for brevity) to add using Oracle.UrmAdapter.Core and Oracle.UrmAdapter clauses. This new class is the main class for the adapter, and should implement the IAdapter and (indirectly) IAdapterConfiguration interfaces. Implementing the interfaces will generate stubs for the following methods:

```
using Oracle.UrmAdapter;
using Oracle.UrmAdapter.Core;

namespace FileSystemAdapter
{
 class FSAdapter: IAdapter
```

```
{
   #region IAdapter Members

   public string GetItemContent(string urmID)
   public void IsMaster()
   public void ProcessDeclarationRequests(Pass searchPass, SearchData search)
   public bool ProcessDisposition(DispositionAction action, string urmID)
   public bool ProcessFreeze(string source, string urmID)
   public bool ProcessThaw(string source, string urmID)

   #endregion

   #region IAdapterConfiguration Members

   public CapabilitiesDefinition AdapterCapabilities
   public Oracle.UrmAdapter.Controls.IArchiveSelector ArchiveSelector
   public IEncryption Encryptor
   public List<Oracle.UrmAdapter.Core.Data.RepositoryField> GetFieldsForRepository(string
            repository)
   public List<string> GetRepositories()
   public List<Oracle.UrmAdapter.Core.Data.SearchData> GetSearches()
   public List<Oracle.UrmAdapter.Controls.IAdapterTab> GetTabs()
   public string Name
   public Oracle.UrmAdapter.Core.Search.ISearchDetails SearchDetails
   public string ServiceName

   #endregion
  }
}
```

When all of the functions have been elaborated, the new adapter will work. The next sections describe the functions (although not in the order listed in the sample method).

Elaborating AdapterCapabilities

The AdapterCapabilities function is used to identify all of the capabilities of the adapter being built. The framework will call this function at various times to retrieve this information. A static function is provided in the class to construct a default set of values. In the example, the default set of values is used as a starting point, and then all deviations to the default values are set. These values are:

- ActivePassive concurrency
- Noncontributing contributions
- Managed declarations
- Managed dispositions
- Archive | DeleteAllRevisions | Destroy | Freeze | Thaw dispositions handled
- Managed logging
- Managed remote access
- Backward | Forward search directions
- Managed searches
- Managed task management
- Allowing updates
- Multithreaded
- With a thread type of STA.

Applicable values are updated as follows:

```
public CapabilitiesDefinition AdapterCapabilities
{
  get
  {
    //Get Default Capabilities
    CapabilitiesDefinition capabilities = CapabilitiesDefinition.DefaultCapabilities;
```

```
      //We won't allow multithreading (just to reduce complexity).
      capabilities.AllowMultiThread = false;

      //Any changes to metadata will be reported back to URM for updates
      capabilities.AllowUpdates = true;

      //Active-Passive does not apply to file system of a single machine.
      capabilities.ConcurrencyModel = ConcurrencyModel.Singleton;

      //For record management, actual content will be sent to URM.
      capabilities.ContributionModel = ContributionModel.ContributesContent;

      //The types of dispositions that the adapter will support.
      capabilities.DispositionTypes = DispositionType.Archive | DispositionType.Destroy |
        DispositionType.Freeze | DispositionType.Thaw | DispositionType.DeleteAllRevisions;

      //The adapter will support both searching new content and
      //searching historical content.
      capabilities.SearchDirections = SearchDirection.Backward | SearchDirection.Forward;

      return capabilities;
    }
  }
```

Elaborating Non-Applicable Functions

Because the adapter is defined as a Singleton concurrency and Managed searches, it is easy to elaborate stubs for functions that will never be called.

```
public void IsMaster()
{
  //Since the adapter is a singleton concurrency, this function will never be called.
  //Typically any functionality that only happens during operation as a master should
  // be kicked off here.
  throw new Exception("The method or operation is not implemented.");
}

public void ProcessSearchRequests(Pass searchPass, SearchData search)
{
  //Since the adapter is search managed, this function will never be called.
  //Typically for adapter managed searches, this function is called during
  //declaration to return all searches defined in the system.
  throw new Exception("The method or operation is not implemented.");
}
```

Elaborating Adapter Description Functions

The interface provides two functions to describe the adapter: ServiceName (name of the service) and Name (user friendly name for display purposes). Typically these would be stored in resource files for internationalization, but for the example, they will be hard-coded.

```
public string ServiceName
{
  //This returns the name of the adapter service (not to be confused
  //with the name of the adapter).
  get { return "URMFileSystemAdapter"; }
}

public string Name
{
  get { return "File System Adapter"; }
}
```

In addition, the adapter needs a string descriptor of all of the repositories it contains. An easy analogy of a repository is a schema of the metadata that will be provided to the Oracle Universal Management

System. Because there is only one set of metadata that can be derived from a file system, just hard-code a string name for this function.

```
public List<string> GetRepositories()
{
  List<string> results = new List<string>();
  results.Add("File System");
  return results;
}
```

Elaborating Repository Metadata Definitions

Because there is only one repository (schema) of metadata supported by the adapter, it is possible to define the metadata in the repository. The items to be added are a unique id, a short name (8.3 style full path name), long name (full path name), the machines NetBIOS name, the filename, the file's author, the URL to the file, file extension, file size, file creation, and last modified dates. With only one repository, it isn't necessary to examine the function parameter. For adapters that contain multiple repositories, return the repository data based upon the repository name.

```
public List<RepositoryField> GetFieldsForRepository(string repository)
{
  List<RepositoryField> results = new List<RepositoryField>();
  RepositoryField theField;

  theField = new RepositoryField("ItemID", "ItemID", RepositoryField.FieldType.Key);
  theField.Size = 200;
  theField.DataType = DataType.String;
  results.Add(theField);

  theField = new RepositoryField("Short Name", "ShortName",
    RepositoryField.FieldType.Field);
  theField.Size = 256;
  theField.DataType = DataType.String;
  results.Add(theField);

  theField = new RepositoryField("Long Name", "LongName",
    RepositoryField.FieldType.Field);
  theField.Size = 1024;
  theField.DataType = DataType.String;
  results.Add(theField);

  theField = new RepositoryField("Machine Name", "MachineName",
    RepositoryField.FieldType.Field);
  theField.Size = 200;
  theField.DataType = DataType.String;
  results.Add(theField);

  theField = new RepositoryField("File Name", "FileName",
    RepositoryField.FieldType.Field);
  theField.Size = 1024;
  theField.DataType = DataType.String;
  results.Add(theField);

  theField = new RepositoryField("Author", "Author", RepositoryField.FieldType.Field);
  theField.Size = 512;
  theField.DataType = DataType.String;
  results.Add(theField);

  theField = new RepositoryField("Url", "Url", RepositoryField.FieldType.URL);
  theField.Size = 512;
  theField.DataType = DataType.String;
  results.Add(theField);

  theField = new RepositoryField("Extension", "Extension",
    RepositoryField.FieldType.Field);
  theField.Size = 512;
  theField.DataType = DataType.String;
  results.Add(theField);
```

```
    theField = new RepositoryField("File Size", "FileSize",
      RepositoryField.FieldType.Field);
    theField.DataType = DataType.Integer;
    results.Add(theField);

    theField = new RepositoryField("Creation Date", "CreationDate",
      RepositoryField.FieldType.Field);
    theField.DataType = DataType.Date;
    results.Add(theField);

    theField = new RepositoryField("Modification Date", "ModificationDate",
      RepositoryField.FieldType.Field);
    theField.DataType = DataType.Date;
    results.Add(theField);

    return results;
  }
```

Creating Search Details

Searches are roughly defined as comparison operations performed on metadata. By elaborating the SearchDetails function, you can define what operations can be performed on what data types. Combining the data type information from the GetFieldsForRepository function, enables you to build a generic search building functionality.

The first step is to create a new class (called SearchDetails) implementing Oracle.UrmAdapter.Core.Search.ISearchDetails. This requires the class to implement the following functions: GetSearchFields (which will define all search fields: their name, display name, if they can be searched upon, if they can be retrieved, and their data types) and GetConditions (which defines the operation that can be performed on each data type).

```
  public List<SearchFieldCollection> GetSearchFields(string repository)
  {  List<SearchFieldCollection> results = new List<SearchFieldCollection>();
    SearchFieldCollection theList = new SearchFieldCollection();

    theList.Add(new FileSystemField("File Name", "FileName", true, true, DataType.String));

    theList.Add(new FileSystemField("Long Name", "LongName", true, true, DataType.String));

    theList.Add(new FileSystemField("Extension", "Extension", true, true,
      DataType.String));

    theList.Add(new FileSystemField("File Size", "FileSize", true, true,
      DataType.Decimal));

    theList.Add(new FileSystemField("Creation Date", "CreationDate", true, true,
      DataType.Date));

    theList.Add(new FileSystemField("Modification Date", "ModificationDate", true, true,
      DataType.Date));

    results.Add(theList);
    return results;
  }
```

The field name, display name, and data type should be identical to the field definitions supplied by the **GetFieldsForRepository** function if they represent the same information. Note, however, that these two sets of fields do not have to be identical. In the example, Author is a valid piece of metadata, however it is not defined as a searchable parameter. If you implement full text searching, that would be a searchable field, but would not be listed as a field from **GetFieldsForRepository**.

The implementation of the GetConditions function is as follows:

```
  public List<Condition> GetConditions(DataType type)
  {
```

```
    List<Condition> theConditions = new List<Condition>();

    if (type == DataType.String)
    {
      theConditions.Add(new Condition("equals", "="));
      theConditions.Add(new Condition("not equals", "!="));
      theConditions.Add(new Condition("contains", "contains"));
    }
    if (type == DataType.Decimal || type == DataType.Integer || type == DataType.Float)
    {
      theConditions.Add(new Condition("greater than", ">"));
      theConditions.Add(new Condition("less than", "<"));
      theConditions.Add(new Condition("equals", "="));
    }
    if (type == DataType.Date)
    {
      theConditions.Add(new Condition("after", ">"));
      theConditions.Add(new Condition("before", "<"));
      theConditions.Add(new Condition("equals", "="));
    }
      return theConditions;
  }
```

When the class has been built, the remaining task is to elaborate the SearchDetails property of the FSAdapter class, returning a new instance of the SearchDetails class.

```
  public Oracle.UrmAdapter.Core.Search.ISearchDetails SearchDetails
    {get {return new SearchDetails();}}
```

Creating Encryptor Functionalty

Usernames and passwords are stored in the framework database, as well as configuration files. It is the responsibility of the adapter developer to implement an encryption scheme to encrypt and decrypt the information. A sample encryptor is provided (encrypting the information as pig-latin).

The first step of creating the encryptor is to create a new class (called PigLatinEncryptor) and have it implement the Oracle.UrmAdapter.IEncryptor interface. This will require it to implement two functions: Encrypt and Decrypt. The code for implementing the functions is as follows:

```
  class PigLatinEncryptor: IEncryption
  {
    public PigLatinEncryptor() { }

    #region IEncryption Members

    public string Decrypt(string inString)
    {
      string firstLetter = inString.ToCharArray()[inString.Length-2].ToString();
      return firstLetter + inString.Substring(0,inString.Length-2);
    }

    public string Encrypt(string inString)
    {
      return inString.Substring(1) + inString.Substring(0, 1) + "a";
    }

    #endregion
  }
```

When the class has been defined, the remaining task is to elaborate the Encryptor property of the FSAdapter class, returning a new instance of the PigLatinEncryptor class.

```
  public IEncryption Encryptor
  { get {return new PigLatinEncryptor();}}
```

Creating the Archive Selector Component

The Archive Selector component is used to identify where in the repository to perform searches. As it applies to the file system adapter, it identifies which directories to search. The sample Archive Selector component will be a dot net TreeView component that will reflect all drives and directories on the drive.

The first step in the process is to create a new Windows control, adding it to the project. After it is created, make the control implement the Oracle.UrmAdapter.Controls.IArchiveSelector class. This will require the elaboration of the following methods: Initialize, Search, DeleteSearch, IsReady, ReadyChanged, DirtyChanged, IsDirty, Save, LoadSearch.

The elaboration of these functions are quite long, so only a quick narrative is provided. Please see the supplied sample code for the exact algorithms and syntax.

Initialize method is called when the control is first loaded. It should interrogate the system (using the System.IO.DriveInfo) to get all the drives, and then recursively get all the directories. As drives and directories are discovered, they should be added to the TreeView.

Search is the property that is called when a user selects a search via the administration tool. It should be held in a class level variable.

DeleteSearch is called when a user deletes a search via the administration tool. A SQL table (FSO_ARCHIVES) is used to map all searches to the directories that should be searched. When DeleteSearch is called, an open DbConnection instance is passed to it. A SQL command should be run against FSO_ARCHIVES deleting all records where the SEARCH_NAME = the Search property's Name property.

Because a search has to be mapped to at least one directory, IsReady is easily constructed by returning true if any object is selected, otherwise false.

Track the IsDirty property by creating a private varaibale to hold the IsDirty value. The property will set and retrieve that value. By creating an event handler on the TreeView's AfterCheck event, you can set the value to true and also throw the DirtyChanged event. You can also throw the ReadyChanged event (with the parameter set to true if any boxes are set).

To implement the Save function, take the open DbConnection passed to the function. Delete all records in the FSO_ARCHIVES table that exist for the current search (using the Search property's Name property). Then walk the TreeView; for every checked box, insert a record into the FSO_ARCHIVES table with the Search property's Name property and the directory name.

The LoadSearch function is the opposite of the Save function. Take the open DbConnection passed to the function. Get the all records in the FSO_ARCHIVES table and for each record, walk the tree, identify the correct node, and set its checked state to true.

At this point, the Archive Selector Component is complete. The ArchiveSelector property of FSAdapter can now be easily implemented with the following code segment:

```
public Oracle.UrmAdapter.Controls.IArchiveSelector ArchiveSelector
{get { return new FileSystemAdapter.FSArchiveAdapter(); }}
```

Creating Additional Tabs

The administration utility has the ability to have additional tabs displayed for custom configuration items

required by adapters. These tabs are windows controls that will inherit from AdapterControl and implement the IAdapterTab interface.

In this example, one tab that contains a configurable setting to set a files archive bit if it is declared.

The first step is to define a database table that will hold the configuration data (FSA_CONFIGURATION). Then create a windows control (FileSystemAdvanced) and add it to the project. After setting the control to inherit Oracle.UrmAdapter.Controls.IAdapterTab, elaborate the following functions: HelpBookmark, Index, IsRequired. Also, you must override the Save and Initialize functions.

The HelpBookmark function is a read-only property that provides a string to call up a file in the help directory. This is required if the user creates an online help htm file.

The index function is a read-only property that is used to order the placement of the tab in the administration utility. The lower the number, the higher its position is in the administration utility's function list.

The IsRequired function is a read-only property that indicates that the value must be set and saved prior to the user being able to navigate to tabs with higer index numbers using the administration utility.

The Initialize function retrieves the current configuration setting (either from the configuration file or a database table). In the example, a database connect is established, the FSA_CONFIGURATION file is queried, and the control is set to reflect the persisted information.

The Save function does the opposite functionality—taking the control's values, establishing a database connection, and persisting the values.

When the control is created, the GetTabs property of the FSAdapter can be implemented:

```
public List<Oracle.UrmAdapter.Controls.IAdapterTab> GetTabs()
{
  List<Oracle.UrmAdapter.Controls.IAdapterTab> results =
    new List<Oracle.UrmAdapter.Controls.IAdapterTab>();
  results.Add(new FileSystemAdvanced());
  return results;
}
```

Elaborating the ProcessDeclarationRequests

The ProcessDelcarationReqests is the main entry point to the adapter to declare data to the Oracle Universal Records Management system. It is called and passes parameters to define the search. The adapter programmer is expected to take those parameters and transform them into a search native to the repository. The adapter is expected to run the search, place the results into a dataset, and send the dataset back to the framework.

The basic algorithm for the ProcessDeclarationRequests is as follows:

```
Get a table to put metadata found into.
Get List of Directories to search
  Iterate through directories
    Get the list of files in the direction
    Load the Search
    Iterate through files in directories
      Get file's shortname
      Get the file's fileInfo

      Evaluate search
      If values fall in the searchPass search window and satisfy the search
        Create new row from result table
        Get Id for the file (machine name + shortname) and store in row
        Set ShortName property if in query and store in row
```

Set LongName property if in query and store in row
Set MachineName property if in query and store in row
Set FileName property if in query and store in row
Set Author property if in query and store in row
Set Url property if in query and store in row
Set File extension if included (trimming of leading ".") and store in row
Set File Size id in query and store in row
Set Modification Date if in query and store in row
Set Creation Date if in query
Add Row to output table

If rowcount > 100 submit results and reset table
submit incomplete batches

Compute retrieval and execution times and assign to searchPass values
For this agent, retrieval and execution times will be identical.
Assign total number found to searchPass.Total.

The complete code can be viewed in the sample code.

A class SearchProcessor is included with the sample code as an example of an expression parser. It uses postfix tree traversal (RPN) to evaluate the expression.

Elaborating the GetItemContent Method

The GetItemContent method is used to inform the adapter that the original content is required by the Oracle Universal Records Management system. This can be either for content contribution for performing records management, or for archive dispositions.

The value passed to the function is the unique ID, defined as the machine name + "\\" + short name of the file (8.3 representation). This value should be parsed so the original short name is revealed.

The basic process is to copy the content to a work area (defined in the Oracle.UrmAdapter.Core.Persistence.Database's GetKeyValue function as "Working Directory"), and return the full path of the temporary file as the result of the function call.

Later, an event can be called indicating the content is no longer needed, so the adapter will know it can delete the temporary content.

Elaborating the ProcessDisposition Method

The ProcessDisposition method is used for instructing the adapter to perform disposition functions. Typically, disposition functions are divided into two areas: archive and deletion. The ProcessDisposition function will only handle dispositions related to deletions. The action and urmID parameters are the disposition action to perform and the unique identifier, respectively.

```
public bool ProcessDisposition(DispositionAction action, string urmID)
{
  bool result = false;
  try
  {
     //Convert urmID to Machine Name and full path
     int start = urmID.IndexOf(@"\\", 0);
     string machineName = urmID.Substring(0, start);
     string fullPath = urmID.Substring(start + 2);

    //Since file systems do not support revisions, we can only
    //handle DeleteAll or Destroy disposition types.
    if (action.Value == DispositionAction.ActionType.da_DeleteAll ||
         action.Value == DispositionAction.ActionType.da_Destroy)
    {
      if (System.IO.File.Exists(fullPath))
```

```
            {
              System.IO.File.Delete(fullPath);
            }
            result = true;
          }
        }
        catch
        {
          result = false;
        }
        return result;
      }
```

Elaborating the ProcessFreeze Method

The ProcessFreeze method is called by the framework to instruct the adapter to perform a litigation freeze on a file. This means that the file cannot be further edited or deleted. Because you cannot place an effective lock on the file system, the strategy is to copy the content to the Oracle Universal Records Manager, create an HTML stub pointing to the file's new location, then delete the original content.

```
      public bool ProcessFreeze(string source, string urmID)
      {
        bool result = false;
        try
        {
          UrmServices theService = new UrmServices();
          System.IO.FileInfo fi = new System.IO.FileInfo(urmID);

          string url = theService.TransferItemToInternal(source, urmID,
            fi.Name,fi.DirectoryName, GetAuthor(fi.FullName));
          CreateStub(urmID,url);
          fi.Delete();
          result = true;
        }
        catch (Exception ex)
        {
          //Report exception
          Oracle.UrmAdapter.Diagnostics.UrmAdapterTrace.Exception( ex);
        }
        return result;
      }
```

The CreateStub function is a simple function that creates an HTML file containing one hyperlink pointing to the URL supplied by the TransferItemToInternal function.

```
private string CreateStub(string fileName, string url)
{
  string stubFileName = null;
  System.IO.FileStream theStream = null;

  try
  {
    //Convert urmID to Machine Name and full path
    int start = fileName.IndexOf(@"\\", 0);
    string machineName = fileName.Substring(0, start);
    string fullPath = fileName.Substring(start + 2);

    if (System.IO.File.Exists(fullPath))
    {
      stubFileName = fileName + ".htm";
      theStream = System.IO.File.Create(stubFileName);
      StringBuilder sb = new StringBuilder();
      sb.AppendLine("<html xmlns='http://www.w3.org/1999/xhtml'><head>");
      sb.AppendLine("<head><title>Content Replacement Notice</title></head><body>");
      sb.AppendLine("This data has been replaced.<br>");
      sb.AppendLine("Click on the following link to access the document<br />");
      sb.AppendLine("<a href=" + url + ">" + fullPath + "</a>");
      sb.AppendLine("</body></html>");
```

```
      System.Text.UTF8Encoding encoder = new UTF8Encoding();
      byte[] buffer = encoder.GetBytes(sb.ToString());
      theStream.Write(buffer, 0, buffer.Length);
      theStream.Close();
      System.IO.File.Delete(fileName);
    }
  }
  catch
  {
    stubFileName = "";
  }
  finally
  {
    if (theStream != null)
    {
      theStream.Close();
    }
  }
  return stubFileName;
}
```

Elaborating the ProcessThaw Method

The ProcessFreeze method is called by the framework to instruct the adapter to release a litigation freeze on a file. You cannot extract the content from the Oracle Universal Records Management system, or change the status from being record managed back to retention managed. Threrefore, report that the process thaw process succeeded, even though no processing was performed.

```
public bool ProcessThaw(string source, string urmID)
{
  //Since we can't reconstitute the content, we'll just say that the thaw was successful
  return true;
}
```

Finishing the Adapter

When all of the interface methods have been elaborated, the final methods to complete are the service's control methods.

First, you must create a class level variable in the FSAdapterService class for the adapter.

```
private FSAdapter _adapter;
```

Then, in the FSAdapterService class, implement start and stop routines.

```
protected override void OnStart(string[] args)
{

   _adapter = new FSAdapter();
  _adapter.OnStart(args);

}

protected override void OnStop()
{
  _adapter.OnStop();
  ExitCode = 0;
}
```

In the FSAdapter class, the OnStart and OnStop functions must be constructed.

```
public void OnStart(string[] args)
{
```

```
    _AdapterEngine = new AdapterEngine(this);
    _AdapterEngine.CheckinComplete += new
      CheckinCompleteEventHandler(_AdapterEngine_CheckinComplete);
    _AdapterEngine.ReleaseContent += new
      ReleaseContentEventHandler(_AdapterEngine_ReleaseContent);
    _AdapterEngine.Start();
  }

  public void OnStop()
  {
    if (_AdapterEngine != null)
    {
      _AdapterEngine.Stop();
    }
  }
```

A class level variable _AdapterEngine is added to the FSAdapter class to bind the adapter to the framework.

```
  private AdapterEngine _AdapterEngine;
```

Finally, two event handlers are constructed for the _AdapterEngine's CheckinComplete and ReleaseContent events.

When checkin complete is thrown, the adapter should check to see if the item checked in is record managed. If so, since it was already checked in, a stub should be created, and the original content should be deleted, just as it was for the ProcessFreeze workflow.

```
  void _AdapterEngine_CheckinComplete(object sender, CheckinCompleteEventArgs e)
  {
    if (!string.IsNullOrEmpty(e.URL) && e.Status == CheckinStatus.Declared)
    {
      try
      {
        string theStub = CreateStub(e.ItemID, e.URL);
        if (!string.IsNullOrEmpty(theStub))
        {
          if (System.IO.File.Exists(e.ItemID))
          {
            System.IO.File.Delete(e.ItemID);
          }
        }
      }
      catch
      {//do nothing}
    }
  }
```

As described in GetItemContent, when a temporary file is copied, the framework tells the adapter when processing is complete and the temporary file can be deleted. This is handled through the _AdapterEngine's ReleaseContent event handler:

```
  void _AdapterEngine_ReleaseContent(object sender, ReleaseContentEventArgs e)
  {
    string theFile = e.FileName;
    if (System.IO.File.Exists(theFile))
    {
      System.IO.File.Delete(theFile);
    }
  }
```

Setting up Configuration Files

In order to use the logging functionality intrinsic to the adapter technology, a configuration file should be added to the project.

Add the following sections to the file:

```
<configSections>
  <section name="microsoft.web.services3"
    type="Microsoft.Web.Services3.Configuration.WebServicesConfiguration,
      Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral,
      PublicKeyToken=31bf3856ad364e35" />
</configSections>
<microsoft.web.services3>
  <messaging>
    <mtom clientMode="On" />
  </messaging>
</microsoft.web.services3>

<system.diagnostics>
  <switches>
    <add name="DebugOutput" value="0"/>
    <add name="ErrorOutput" value="1"/>
    <add name="InfoOutput" value="1"/>
    <add name="WarningOutput" value="1"/>
  </switches>
  <sources>
    <source name="UrmAdapterTrace" switchValue="Verbose">
      <listeners>
        <add name="AdapterTracer"
          type="Oracle.UrmAdapter.Diagnostics.UrmAdapterTraceListener, Oracle.UrmAdapter"
              initializeData="FileSystemAdapter" />
        <remove name="Default" />
      </listeners>
    </source>
  </sources>
</system.diagnostics>
```

Branding the Adapter

The generic adapter provides a mechanism for the developer to brand for the Adapter with their own company logo. A default icon is compiled into the project and appears in the top left corner of the administration and database wizard title bars. This icon can be overridden at run time by providing a file named “product.ico” in adapter install directory. The file is loaded at run time. In order to support the full range of Windows operating system environments generate multiple rendentions of the icon in the full standard set of dimensions.

Deploying the Application

When deploying the application, the following must be in the target directory:

- FileSystemAdapter.exe – File system adapter service
- FileSystemAdapter.exe.config – Adapter configuration file
- Oracle.UrmAdapter.dll – Oracle Universal Records Management framework component
- Oracle.UrmAdapter.UrmWebServices.dll – Oracle Universal Records Management Web service definitions
- Administration.exe – Administration utility
- Administration.exe.config – Administration configuration file
- DatabaseWizard.exe – Utility for creating framework database
- Framework_MS.SQL – Microsoft SQL Server script for creating framework database
- Framework_Oracle.SQL – Oracle script for creating framework database
- FileSystemSDK_MS.SQL – Microsoft SQL Server script for creating file system specific tables
- FileSystemSDK_Oracle.SQL – Oracle script for creating file system specific tables.

The Administration.exe.config file must be edited prior to deployment to identify the type of adapter that will be administered. The following is the configuration for the example adapter:

```
<appSettings>
  <add key="AdapterConfiguration" value="FileSystemAdapter.FSAdapter" />
  <add key="AdapterAssembly" value="FileSystemAdapter.exe" />
</appSettings>
```

The AdapterConfiguration key value identifies the class (namespace.class) to instantiate to perform configuration functions. The AdapterAssembly key value identifies the executable that contains the adapter functions.

Any additional SQL scripts required should be put in the same deployment directory and should end with _MS.SQL and _ORACLE.SQL. The DatabaseWizard utility will always run the appropriate Framework_*.SQL script followed by every other script for that database type. Note that the order of the scripts running after the framework script is not guaranteed.

Prior to running the application, the executable must be installed as a service via the InstallUtil included with Microsoft 2.0 Framework.

Sample Install

A sample InstallShield basic MSI project has been provided that includes examples of the custom actions, dialogs and redistributable packages that are required for successful installation, configuration and setup of the Oracle Universal Records Management .NET SDK Adapter Framework. The sample install was written and compiled using FlexNet InstallShield Premier Edition Version 12 for Windows.

Prerequisites

1. Install FlexNet InstallShield Premier Edition Version 12 for Windows
2. Open the InstallShield IDE
3. GoTo the Tools drop down menu and select the Redistributable Downloader option
4. The InstallShield Redistributable Downloader Welcome dialog will be displayed. Click Next.
5. A new dialog will open displaying all of the InstallShield Redistributable packages that are available for download. Select the following packages by checking the checkbox next to the package's name:

   .NET Framework Version 2.0 Redistributable Package
   Visual J# .NET Version 2.0 Redistributable Package

6. Click Next
7. A download progress dialog is displayed
8. Once the download has completed, click the enabled Finish button
9. Run the Oracle Universal Records Management Adapter SDK .NET install

Working with the Sample Install

1. Browse to the following directory Program Files\InstallShield 12 Projects\SampleCustomAdapter
2. Open the SampleCustomAdapter.ism project in the InstallShield IDE
3. Expand the Application Data tree control and select the Redistributables (Objects) View
4. Scroll to the bottom of the redistributable packages list and select the Visual JSharp .NET Redistributable Package by checking the checkbox.
5. Verify the Visual JSharp .NET Redistributable Package displays the following settings:

   Type: Prerequisite Setup
   Location: Installed Locally

Note: If the package is not installed locally, then select the Download Selected Item option from the right-click menu.

6. Open the File drop down menu and select the Save As option
7. The Save As dialog is displayed
8. Enter a new project name and location for your custom install package and check the following checkboxes:

   Create 'Project Name' subfolder and save the project in the created folder
   Create and assign a new project GUID to the saved project
   Update the project settings appropriately based on the new project name

9. Click the Save button
10. Open the re-named project that was created above in the InstallShield IDE.
11. Open the Build drop down menu and select the Build Release option
12. Verify the new project compiles successfully
13. Make desired changes to the sample install project template to support installation of custom adapters.

URM Adapter Framework Object Model

Introduction

The rest of the document is dedicated to an exhaustive description of the Oracle Universal Records Management Adapter Framework. All of the following classes are contained in the Oracle.UrmAdapter.dll component.

Namespace: Oracle.UrmAdapter

Assembly: Oracle.UrmAdapter.dll

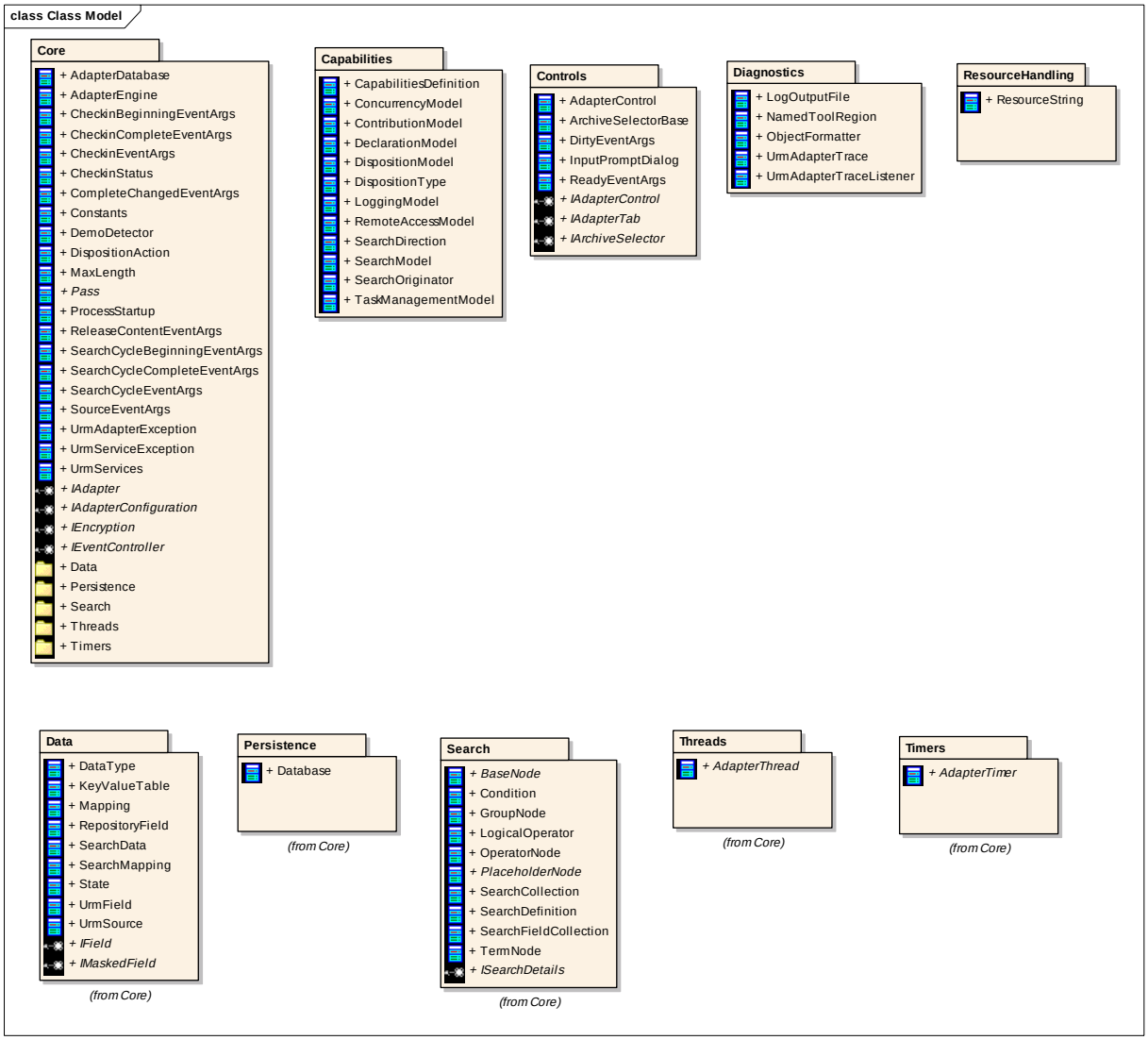

Figure 1: UrmAdapter Class Model

Namespace: Oracle.UrmAdapter.Capabilities

Assembly: Oracle.UrmAdapter.dll

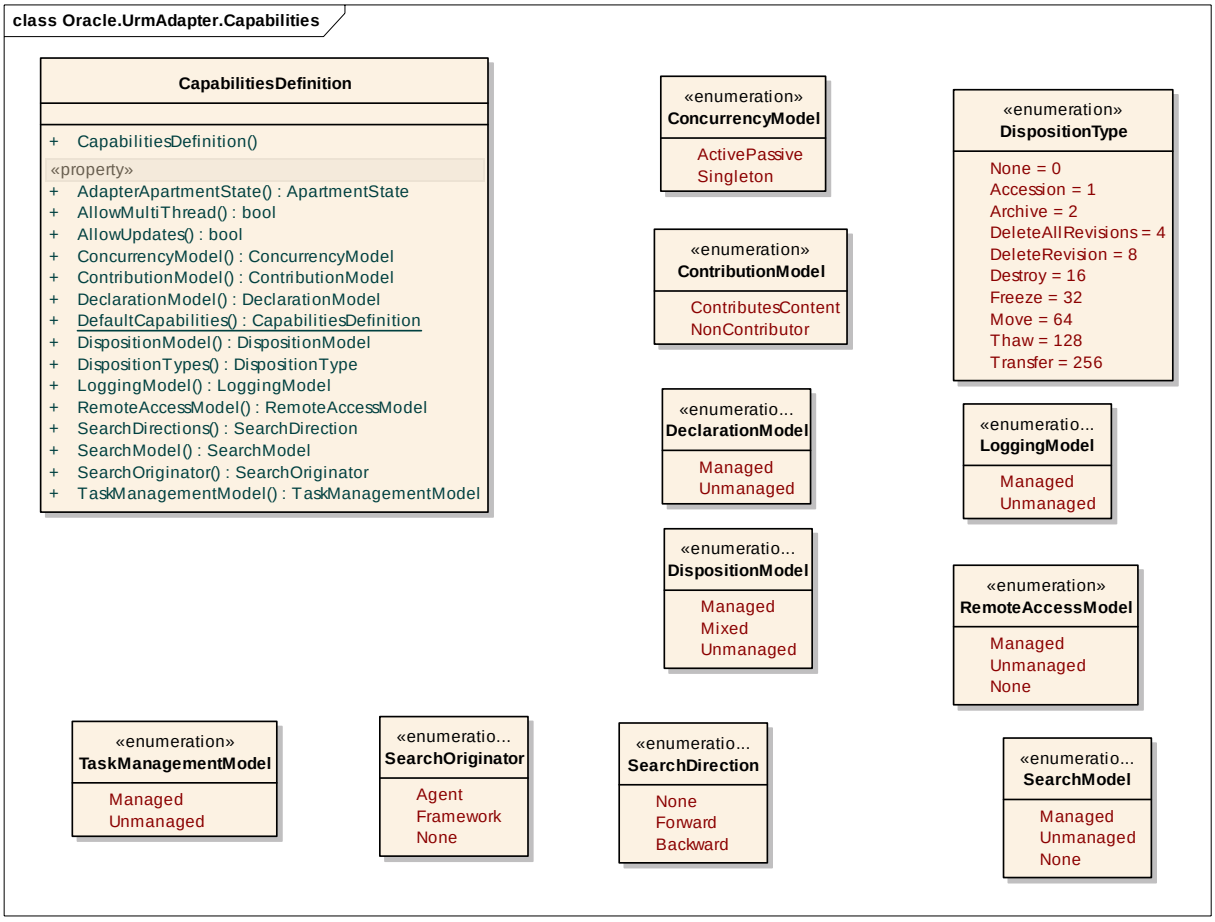

Figure 2: Oracle.UrmAdapter.Capabilities

CapabilitiesDefinition

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Container class used to describe the capabilities of the adapter. This enables the framework (Oracle.UrmAdapter) to only provide appropriate messages and calls to the adapter. Typically an adapter implementing the IAdapterConfiguration interface will create a local instance of the class from the DefaultCapabilities function and change relevant properties. The AdapterCapabilities property will be implemented to return this local instance.

Operations

Method	Notes	Parameters
AdapterApartmentState() ApartmentState	Used to identify the Apartment state of threads passed to the Adapter. Default value is System.Threading.ApartmentState.STA (Apartment threading) and is recommended.	
AllowMultiThread() bool	The adapter framework is capable of multithreading, issuing multiple calls to the adapter. If the adapter is thread-safe and able to handle multiple calls, set the value to true,	

Method	Notes	Parameters
	otherwise, false.	
Allow Updates() bool	Indicates that the adapter intends on updating metadata in Content Server. Set true if updates are allowed, otherwise false.	
CapabilitiesDefinition()	Constructor	
ConcurrencyModel() ConcurrencyModel	Describes how multiple agents behave.	
ContributionModel() ContributionModel	Indicates if the adapter will be sending content (not just metadata) to the Content Server.	
DeclarationModel() DeclarationModel	Describes how declarations will be managed by the framework.	
Static **DefaultCapabilities()** CapabilitiesDefinition	Static property to retrieve the default capabilities for an adapter. These are: ConcurrencyModel: ActivePassive ContributionModel: NonContributor DeclarationModel: Managed DispositionTypes: Archive & DeleteAllRevisions & Destroy & Freeze & Thaw LoggingModel: Managed RemoteAccessModel: Managed SearchDirections: Backward & Forward SearchModel: Managed TaskManagementModel: Managed AllowUpdates: true AllowMultiThread: true AdapterApartmentState: ApartementState.STA	
DispositionModel() DispositionModel	If dispositions will be managed by the framework, use Managed, otherwise Unmanged.	
DispositionTypes() DispositionType	Type of dispositions that the adapter can perform. Consists of the following flags (or'd together) Archive: copies the content to URM DeleteAllRevisions: deletes the content DeleteRevision: deletes the content Destroy: deletes the content Freeze: prevents the content from being altered or deleted Thaw: allows previously frozen content to be altered or deleted	
LoggingModel() LoggingModel	If logging will be managed by the framework, use Managed, otherwise Unmanaged.	
RemoteAccessModel() RemoteAccessModel	If remote access will be managed by the framework, use Managed, otherwise Unmanaged.	
SearchDirections() SearchDirection	Indicates directions that declaration searchs can be performed. OR Forward and Backward values together if necessary.	
SearchModel() SearchModel	If searches will be managed by the framework, use Managed, otherwise Unmanged.	
SearchOriginator() SearchOriginator		
TaskManagementModel() TaskManagementModel	If tasks will be managed by the framework, use Managed, otherwise Unmanged.	

ConcurrencyModel

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Describes how multiple adapters behave when running concurrently

Attributes

Attribute	Notes
ActivePassive	Indicates that with multiple agents running, only one instance will be active at a time.
Singleton	Indicates that only one instance will be run (no backup activation).

ContributionModel

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Indicates that the adapter will be sending content (not just metadata) to the Content Server.

Attributes

Attribute	Notes
ContributesContent	Adapter contributes original content to URM for record management.
NonContributor	Adapter does not contribute content for record management.

DeclarationModel

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Indicates whether the framework or the adapter will be responsible for managing declarations.

Attributes

Attribute	Notes
Managed	Indicates that declarations will be managed by the framework
Unmanaged	Indicates that declarations will be managed by the adapter

DispositionModel

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Indicates whether the framework or the adapter will be responsible for managing dispositions.

Attributes

Attribute	Notes
Managed	Dispositions will be managed exclusively by the framework
Mixed	Disposition requests will be supplied by the framework, the agent will decide what to handle, and what to allow the framework to handle.
Unmanaged	Dispositions will be handled by the adapter.

DispositionType

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Identifies the types of dispositions the framework will try to process. To handle multiple types, OR ("|") the values together.

Attributes

Attribute	Notes
None	
Accession	Not currently Supported
Archive	Content is copied to URM
DeleteAllRevisions	All revisions (for retention managed content) should be deleted by the adapter
DeleteRevision	A particular revision is deleted (not currently supported)

Attribute	Notes
Destroy	Record (for record managed content) should be deleted by the adapter
Freeze	Content and metadata should be frozen by adapter. Edits and deletions denied.
Move	Not currently supported
Thaw	Content and metadata should be thawed -- Edits and deletions are re-enabled.
Transfer	Not currently supported

LoggingModel

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Indicates whether the framework or the adapter will be responsible for moving log files to the URM.

Attributes

Attribute	Notes
Managed	Movement of logs to URM are managed by framework.
Unmanaged	Movement of logs to URM are managed by Adapter.

RemoteAccessModel

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Indicates if the adapter is capable of performing remoting functions to the administration utility.

Attributes

Attribute	Notes
Managed	Communication via remoting is enabled
Unmanaged	Communication via remoting is not supported
None	Communication via remoting state is unknown

SearchDirection

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Identifies the directions (new content and/or historical content) that the framework will try to declare. To handle multiple types, OR (|) the values.

Attributes

Attribute	Notes
None	Search Directions is unknown
Forward	New content will be detected for declarations
Backward	Historical content will be detected for declarations

SearchModel

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Indicates if the searching will be initiated by the framework or the adapter

Attributes

Attribute	Notes

Attribute	Notes
Managed	Search initiation is handled in framework.
Unmanaged	Search initiation is handled in the adapter.
None	Search Model state is unknown.

SearchOriginator

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Indicates if the declaration searches will be constructed by the administration utility (handled in framework) or done through the adapter.

Attributes

Attribute	Notes
Public **Agent**	Indicates that the Adapter is responsible for creating searches
Public **Framework**	Indicates that the Framework is responsible for creating searches
Public **None**	Indicates that Search Originator is unknown.

TaskManagementModel

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Indicates if the declaration batches will be checked and verified through the framework or by the adapter.

Attributes

Attribute	Notes
Managed	Batch declaration management is handled by framework.
Unmanaged	Batch declaration management is handled by adapter.

Namespace: Oracle.UrmAdapter.Controls

Assembly: Oracle.UrmAdapter.dll

The Oracle.UrmAdapter.Controls namespace contains classes needed to generate controls for the Administrator UI.

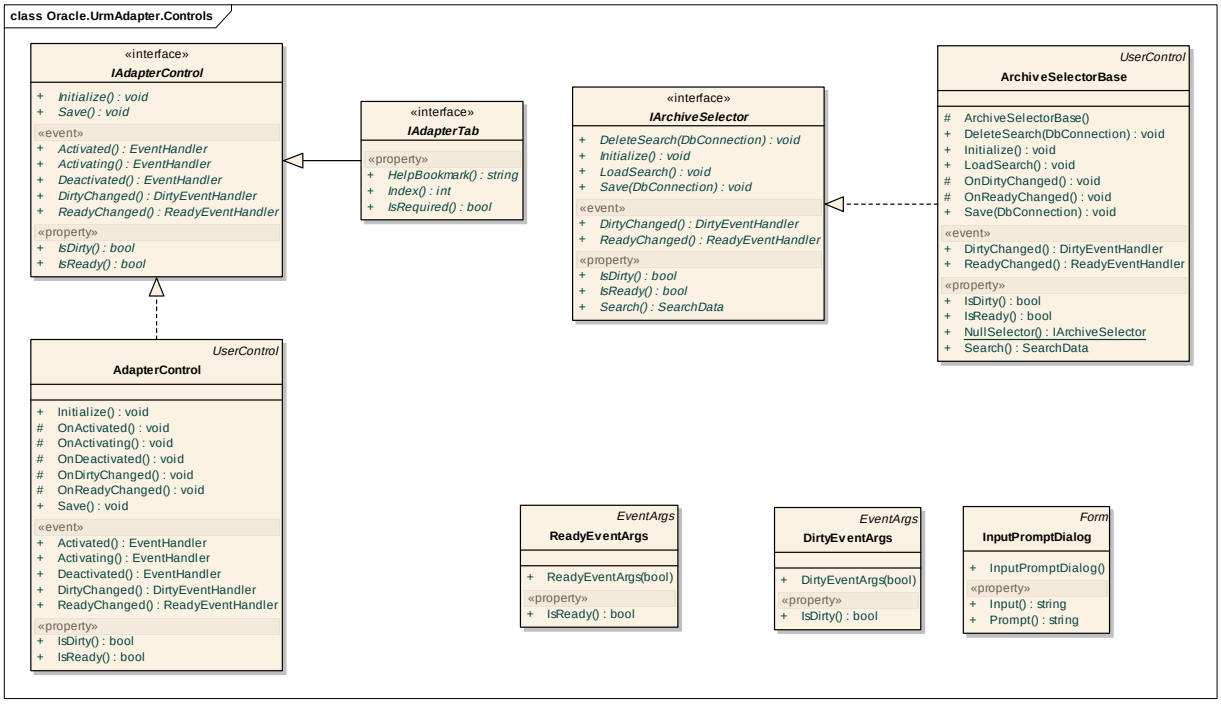

Figure 3: Oracle.UrmAdapter.Controls

AdapterControl

Type: **Class** **UserControl**
Assembly: Oracle.UrmAdapter.dll

The AdapterControl is the base class for creating controls that will be used by the Administrator UI

Operations

Method	Notes	Parameters
Activated() EventHandler	Event thrown when the control has been activated	
Activating() EventHandler	Event thrown when the control is in the process of activating	
Deactivated() EventHandler	Event thrown when the control is being deactivated	
DirtyChanged() DirtyEventHandler	Event thrown when the control differs from the persisted values (returns true) or matches persisted values (returns false)	
Initialize() void	Called to inform control to initialize itself.	
IsDirty() bool	Called to retrieve "dirty" status of control values.	
IsReady() bool	Called to retrieve validation status of control values.	
Protected **OnActivated()** void	Event handler for Activated event	
Protected **OnActivating()** void	Event handler for Activating event	

Method	Notes	Parameters
Protected **OnDeactivated()** void	Event handler for Deactivated event	
Protected **OnDirtyChanged()** void	Event handler for "dirty" value changed event	
Protected **OnReadyChanged()** void	Event handler for value validation changed event	
ReadyChanged() ReadyEventHandler	Event thrown when the control values validations pass (returns true) or fail (returns false)	
Save() void	Called to inform control so persist its values.	

ArchiveSelectorBase

Type: **Class** **UserControl**
Assembly: Oracle.UrmAdapter.dll

Base class used to implement control to identify areas of repository that the framework will search.

Operations

Method	Notes	Parameters
Protected **ArchiveSelectorBase()**	Constructor	
DeleteSearch() void	Called by administration UI, indicating that a search has been deleted, and all mappings to repository areas corresponding to that search should be removed.	**DbConnection** [in] connection Open database connection to Framework database
DirtyChanged() DirtyEventHandler		
Initialize() void		
IsDirty() bool		
IsReady() bool		
LoadSearch() void	Called by administration UI, indicating that a search has been deleted, and all mappings to repository areas corresponding to that search should be removed.	
Static **NullSelector()** IArchiveSelector	Read-Only static property used to get null selector (no archive selectors are used by the adapter).	
Protected **OnDirtyChanged()** void		
Protected **OnReadyChanged()** void		
ReadyChanged() ReadyEventHandler		
Save() void	Called by administration utility to tell control to save values.	**DbConnection** [in] connection An open DbConnection to connect to framework database.
Search() SearchData	Property used for providing SearchData information to the control.	

DirtyEventArgs

Type: **Class** **EventArgs**
Assembly: Oracle.UrmAdapter.dll

Event Args class used by the DirtyChanged event handlers.

Operations

Method	Notes	Parameters
DirtyEventArgs()	`Constructor that sets the IsDirty values`	**bool** [in] isDirty Indicates the control's values match persisted values (false) or do not match (true)).
IsDirty() bool	`Read-only property indicating if the control's values match persisted values (false) or not (true).`	

InputPromptDialog

Type: **Class** **Form**
Assembly: Oracle.UrmAdapter.dll

Simple dialog box that prompts the user for a string value.

Operations

Method	Notes	Parameters
Input() string	`Value provided to /retrieved from the user.`	
InputPromptDialog()	`Constructor`	
Prompt() string	`Prompt text for the window.`	

ReadyEventArgs

Type: **Class** **EventArgs**
Assembly: Oracle.UrmAdapter.dll

Event Args class used by the ReadyChanged event handler.

Operations

Method	Notes	Parameters
IsReady() bool	`Read-only property indicating the control is ready (passes validation and can be persisted).`	
ReadyEventArgs()	`Constructor used to create an instance and set the IsReady value.`	**bool** [in] ready Indicates that the control values pass validation

IAdapterControl

Type: **Interface**
Assembly: Oracle.UrmAdapter.dll

The IAdapterControl is the primary interface implemented by controls used by the Administrator UI.

Operations

Method	Notes	Parameters
Activated() EventHandler	`Event thrown when the control is activating.`	

Method	Notes	Parameters
Activating() EventHandler	Event thrown when the control is activating.	
Deactivated() EventHandler	Event thrown when the control is deactivated.	
DirtyChanged() DirtyEventHandler	Event thrown when the dirty state (persisted values matching current values) has changed.	
Initialize() void	Function called by administration UI to tell control to initialize itself.	
IsDirty() bool	Constructor used to create an instance and set the IsReady value.	
IsReady() bool	Function called by the administration UI to see if the control has passed its business rules (in ready state).	
ReadyChanged() ReadyEventHandler	Event thrown when the ready state (passing business rules) has changed.	
Save() void	Function called when the administration UI wants the control to persist its values.	

IAdapterTab

Type: **Interface** **IAdapterControl**
Assembly: Oracle.UrmAdapter.dll

The IAdapterTab is the primary interface implemented by tabs used by the Administrator UI. Typically, the AdapterControl will implement both the IAdapterControl and IAdapterTab interfaces. The IAdapterTab will describe behavior of the tab itself.

Operations

Method	Notes	Parameters
HelpBookmark() string	The filename of the html help file in the help directory.	
Index() int	The desired index of the tab in the Administration UI.	
IsRequired() bool	Indicates if the tab must be completed and saved before proceeding to the next tab.	

IArchiveSelector

Type: **Interface**
Assembly: Oracle.UrmAdapter.dll

The IArchiveSelector interface is used for creating selector controls for selecting areas for the agent to record/retention manage. These controls will be useable in the administration UI. Typically they will contain some kind of TreeView control and a mechanism to identify what directories, mailboxes, sites, etc. should be checked for content declarations.

Operations

Method	Notes	Parameters
DeleteSearch() void	Called to delete search's archive mappings. The adapter is responsible for managing a search/archive table.	**DbConnection** [in] connection Open dbConnection providing access to the framework database
DirtyChanged()	Event indicating that the control's values	

Method	Notes	Parameters
DirtyEventHandler	differ from persisted values (true) or matches persisted values (false)	
Initialize() void	Called to tell control to perform initialization functions	
IsDirty() bool	Called to see if the control's values differ from persisted values (true) or matches persisted values (false)	
IsReady() bool	Called to see if the control's values pass validations (true) or not (false)	
LoadSearch() void	Called to load search's archive mappings into the control.	
ReadyChanged() ReadyEventHandler	Event indicating that the control's values pass validations (true) or not (false)	
Save() void	Called to save search's archive mappings	**DbConnection** [in] connection Open dbConnection providing access to the framework database
Search() SearchData	Uses to set/get SearchData	

Namespace: Oracle.UrmAdapter.Core

Assembly: Oracle.UrmAdapter.dll

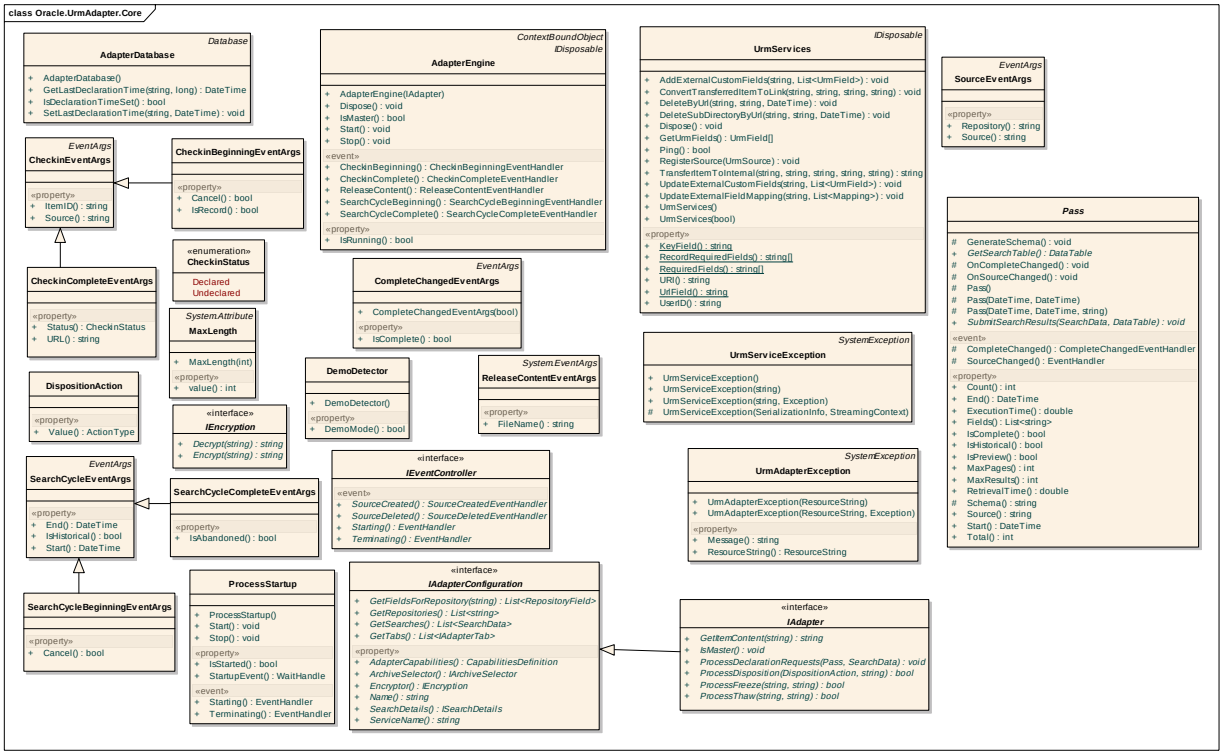

Figure 4: Oracle.UrmAdapter.Core

AdapterDatabase

Type: **Class** **Database**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
AdapterDatabase()		
GetLastDeclarationTime() DateTime		**string** [in] task **long** [in] interval
IsDeclarationTimeSet() bool		
SetLastDeclarationTime() void		**string** [in] task **DateTime** [in] value

AdapterEngine

Type: **Class** **ContextBoundObject, IDisposable**
Assembly: Oracle.UrmAdapter.dll

Framework engine used to interface the adapter with the URM. Typically, an adapter will create an instance of this class, and call the Start method to get the framework running

Operations

Method	Notes	Parameters
AdapterEngine()	Constructor	**IAdapter** [in] adapter Instance of the service class implementing IAdapter
AssumeMastery() void		
CheckinBeginning() CheckinBeginningEventHandler	Event that indicates that a declared item is about to be checked into URM.	
CheckinComplete() CheckinCompleteEventHandler	Event that indicates that a declared item has been checked into URM.	
Dispose() void		
IsMaster() bool	For adapters with the ActivePassive ConcurrencyModel, this function will indicate if the adapter has the role as the Active master. @returns boolean - true if master	
IsRunning() bool	Read-only property indicating the framework is in a running state	
ReleaseContent() ReleaseContentEventHandler	Event that indicates downloaded content has been processed and can be released.	
SearchCycleBeginning() SearchCycleBeginningEventHandler	Event that indicates that the declaration search cycle is beginning.	
SearchCycleComplete() SearchCycleCompleteEventHandler	Event that indicates that the declaration search cycle has completed.	
Start() void	Starts Service threads	
Stop() void	Stop Service threads	

CheckinBeginningEventArgs

Type: **Class** **CheckinEventArgs**
Assembly: Oracle.UrmAdapter.dll

Event handler argument for the AdapterEngine.CheckinBeginning event.

Operations

Method	Notes	Parameters
Cancel() bool	Set to true if the checkin should be cancelled (for errors detected in adapter, for example).	
IsRecord() bool	Read-only property indicating that the search is record managed (vs. retention managed).	

CheckinCompleteEventArgs

Type: **Class** **CheckinEventArgs**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
Status() CheckinStatus	Read-only property. Describes what type of checkin has been done.	
URL() string	Read-only property. Describes the url location of the content in URL. Only applicable for content contributed to URM.	

CheckinEventArgs

Type: **Class** **EventArgs**
Assembly: Oracle.UrmAdapter.dll

Base class event handler argument for the AdapterEngine.xComplete events. CheckinBeginningEventArgs and CheckinCompleteEventArgs inherit from this class.

Operations

Method	Notes	Parameters
ItemID() string	`Read-only property. The unique key that identifies the item`	
Source() string	`Read-only property identifying the URM source that the item is being declared to`	

CheckinStatus

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Enumeration of checkin types (Declared successfully, or Undeclared).

Attributes

Attribute	Notes
Declared	
Undeclared	

Constants

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
Static **BacklogCheckInterval()** string		
Static **BatchCheckInterval()** string		
Static **BatchSize()** string		
Static **DemoMode()** string		
Static **DispositionInterval()** string		
Static **ForwardEnabledKey()** string		
Static **ForwardEndKey()** string		
Static **ForwardIntervalKey()** string		
Static **ForwardStartKey()** string		
Static **ForwardTask()** string		
Static **HeartbeatInterval()** string		
Static **HistoryComplete()** string		
Static **HTMLEncoding()** string		
Static **InvalidNameChars()** char		
Static **MasterCheckInterval()** string		
Static **MaxCaptionLength()** int		
Static **MaxColumnNameLength()** int		
Static **MaxDefaultLength()** int		
Static **MaxDescriptionLength()** int		

Method	Notes	Parameters
Static **MaxFieldPrecision()** int		
Static **MaxFieldScale()** int		
Static **MaxFieldSize()** int		
Static **MaxKeyValueLength()** int		
Static **MaxNameLength()** int		
Static **MaxNumberOfFields()** int		
Static **MaxSize()** int		
Static **MaxTableNameLength()** int		
Static **NoSize()** int		
Static **OutstandingBatchThreshold()** string		
Static **RemotingPort()** string		
Static **ReverseEnabledKey()** string		
Static **ReverseEndKey()** string		
Static **ReverseIntervalKey()** string		
Static **ReverseStartKey()** string		
Static **ReverseTask()** string		
Static **StartOfTime()** string		
Static **UrmAlternateKeyColumn()** string		
Static **UrmKeyColumn()** string		
Static **WorkingDirectory()** string		

DemoDetector

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

The DemoDetector class is a utility class used for system demonstrations only. It is used to detect of the demo file is present, and detecting if the framework is in demo mode. In demo mode, all timing intervals should be reduced for demonstration and testing purposes.

Operations

Method	Notes	Parameters
DemoDetector()	`Constructor`	
DemoMode() bool	`Read-only property indicating if the framework is in demo mode.`	

DispositionAction

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Class identifying a disposition action

Operations

Method	Notes	Parameters
Value() ActionType	`Type of action the disposition expects.`	

MaxLength

Type: **Class** **System.Attribute**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
MaxLength()		**int** [in] length
value() int		

Pass

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

The Pass class is used by the framework to provide adapter with information that is required for a declaration cycle .

Operations

Method	Notes	Parameters
Protected **CompleteChanged()** CompleteChangedEventHandler		
Count() int	The number of items returned by the declaration cycle's pass.	
End() DateTime	Read-Only property indicating the datetime the declaration cycle's pass end window time.	
ExecutionTime() double	The length of time (in milliseconds) of the declaration cycle's pass execution	
Fields() List<string>	Read-Only list of fields required for the declaration's pass	
GetSearchTable() DataTable	Generates an ADO.Net datatable for population of items found in declaration cycle.	
IsComplete() bool	Indicates that the declaration cycle's pass is complete.	
IsHistorical() bool	Indicates that the declaration cycle's pass is for a historical search.	
IsPreview() bool	Read-Only property indicating that the declaration is for search preview.	
MaxPages() int	Read-Only property indicating the maximum number of result pages the framework is expecting. 0 indicates unlimited.	
MaxResults() int	Read-Only property indicating the maximum number of results the framework is expecting. 0 indicates unlimited.	
RetrievalTime() double	The length of time (in milliseconds) required to retrieve all of the data for the declaration cycle's pass.	
Source() string	Read-Only property indicating the Source name for the declaration cycle's pass	
Start() DateTime	Read-Only property indicating the datetime the declaration cycle's pass start window time.	
SubmitSearchResults() void	Used to submit declaration results to the framework.	**SearchData** [in] search **DataTable** [in] results
Total() int	The total number of items returned by the declaration cycle's pass. This is the number found vs. the number returned (Count property).	

ProcessStartup

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Defining a constructor on your class which implements IAdapterConfiguration which accepts an IEventController will result in the Administration UI passing you an object which implements that interface when it constructs your configuration class. You can then listen for the events IEventController exposes.

Operations

Method	Notes	Parameters
IsStarted() bool	`Indicates that the framework processing has started.`	
ProcessStartup()		
Start() void	`Starts the framework processing.`	
Starting() EventHandler	`Event handler indicating that framework processing has begun.`	
StartupEvent() WaitHandle	`Used to get a handle to framework processor.`	
Stop() void	`Stops the framework processing.`	
Terminating() EventHandler	`Event handler indicating that framework processing is terminating.`	

ReleaseContentEventArgs

Type: **Class** **System.EventArgs**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
FileName() string	`Absolute path of staged file.`	

SearchCycleBeginningEventArgs

Type: **Class** **SearchCycleEventArgs**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
Cancel() bool	`Set cancel to true to cancel the search`	

SearchCycleCompleteEventArgs

Type: **Class** **SearchCycleEventArgs**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
IsAbandoned() bool		

SearchCycleEventArgs

Type: **Class** **EventArgs**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters

Method	Notes	Parameters
End() DateTime	The ending time of the search	
IsHistorical() bool	Flag indicating this is a historical search	
Start() DateTime	The beginning time of the search	

SourceEventArgs

Type: **Class** **EventArgs**
Assembly: Oracle.UrmAdapter.dll

Event Args class used for IEventController's SourceCreated and SourceDeleted events.

Operations

Method	Notes	Parameters
Repository() string	Read-only property for reading repository name.	
Source() string	Read-only property for reading URM source name.	

UrmAdapterException

Type: **Class** **SystemException**
Assembly: Oracle.UrmAdapter.dll

Generalized exception generated by the adapter framework.

Operations

Method	Notes	Parameters
Message() string	Read-only property containing error message	
ResourceString() ResourceString	Read-only property containing error message in resource string form.	
UrmAdapterException()	Constructor	**ResourceString** [in] message Resource String containing exception message
UrmAdapterException()	Constructor that contains inner exception (good for bubbling exceptions).	**ResourceString** [in] message Resource String containing exception message **Exception** [in] innerException inner exception to be wrapped

UrmServices

Type: **Class** **IDisposable**
Assembly: Oracle.UrmAdapter.dll

Provides direct access to URM functions.

Operations

Method	Notes	Parameters
AddExternalCustomFields() void	Adds custom fields for a source in URM.	**string** [in] source Urm Source Name **List<UrmField>** [in] fields List of Urm Fields to add
ConvertTransferredItemToLink()	Used to correlate a replacement	**string** [in] source

Method	Notes	Parameters
void	`stub to the original content.`	Urm Source **string** [in] id Unique ID of original document **string** [in] newId Unique ID of stub document **string** [in] newUrl Url of stub document
DeleteByUrl() void	`Deletes an item based on its url that exist on or before the expiration date.`	**string** [in] source Urm Source Name **string** [in] url Url of item to be deleted **DateTime** [in] expirationDate date/time of deletion
DeleteSubDirectoryByUrl() void	`Deletes all items in a url directory that exist on or before the expiration date`	**string** [in] source Urm Source Name **string** [in] url Url of directory to delete **DateTime** [in] expirationDate date/time of deletion
GetUrmFields() UrmField	`Gets a list of default URM fields @returns Array of UrmFields representing default URM fields`	
Static **KeyField()** string		
Ping() bool	`Ping's URM Server @returns boolean indicating if URM services are available`	
Static **RecordRequiredFields()** string	`Returns an array of fields in addition to RequuiredFields necessary to submit record managed items.`	
RegisterSource() void	`Used to register a new source in URM.`	**UrmSource** [in] SourceData Source Definition
Static **RequiredFields()** string	`Returns an array of fields required to submit retention managed items.`	
SetupExtraMetadataField() void		**string** [in] source **List<Mapping>** [in] map
TransferItemToInternal() string	`For an adapter that contributes content, used to record manage a previously retention managed item @returns URL of internal document`	**string** [in] source URM Source Name **string** [in] key Unique Key of item **string** [in] fileName Content File Name **string** [in] directory Location of content file **string** [in] author Content Author
UpdateExternalCustomFields() void	`Updates the definitions of external custom fields for a source in URM.`	**string** [in] source URM source name **List<UrmField>** [in] fields

Method	Notes	Parameters
		List of edited fields
UpdateExternalFieldMapping() void	`Used to update adapter to URM mappings.`	**string** [in] source URM source name **List<Mapping>** [in] maps List of urm field to repository mappings
URI() string		
Static **UrlField()** string		
Static **UrmServices()**		
UrmServices()	`Constructor`	
UrmServices()	`Constructor with retry`	**bool** [in] enableRetry Indicates if the framework should repeatedly call the function until successful (good for unreliable or heavy-trafficed networks).
UserID() string		

IAdapter

Type: **Interface** **IAdapterConfiguration**
Assembly: Oracle.UrmAdapter.dll

The IAdapter interface it the primary interface to implement to create a URM adapter. By implementing the interface, the framework will make calls to the adapter to perform URM functions.

Operations

Method	Notes	Parameters
GetItemContent() string	`Request from framework to retrieve and stage content, returning the filename where the staged content is located. @returns full path to content`	**string** [in] urmID Unique ID of document
IsMaster() void	`Used to inform Adapter it is now the master, not backup, adapter`	
ProcessDeclarationRequests() void	`Processes declaration requests.`	**Pass** [in] searchPass Search parameters **SearchData** [in] search Search Definition
ProcessDisposition() bool	`Processes disposition operations, returning true if successful. @returns boolean denoting success status`	**DispositionAction** [in] action Type of disposition to process **string** [in] urmID Unique key identifying record
ProcessFreeze() bool	`Processes freeze (hold) operations, returning true if successful. @returns Boolean indicating success status`	**string** [in] source URM source name **string** [in] urmID Unique key identifying record
ProcessThaw() bool	`Processes thaw (unhold) operations, returning true is successful. @returns Boolean indicating success status`	**string** [in] source URM source name **string** [in] urmID Unique key identifying record

IAdapterConfiguration

Type: **Interface**
Assembly: Oracle.UrmAdapter.dll

The IAdapterConfiguration should be implemented by the adapter to enable management by the administration utility.

Operations

Method	Notes	Parameters
AdapterCapabilities() CapabilitiesDefinition	Read-only property describing the functionalities the adapter can perform.	
ArchiveSelector() IArchiveSelector	Read-only property supplying the control to the administration utility that identifies archives that the URM will manage.	
Encryptor() IEncryption	Read-only property providing an class that can perform encryption/decryption services	
GetFieldsForRepository() List<RepositoryField>	Supplies the fields that the adapter can supply to the URM based upon the repository name. @returns List of repository fields	**string** [in] repository Name of repository
GetRepositories() List<string>	Provides a list of repositories (schemas) that the adapter supports. @returns List of repository names	
GetSearches() List<SearchData>	Provides a list of searches that are defined for the adapter. This is only applicable for adapters that support the Unmanaged search model. @returns List of search data	
GetTabs() List<IAdapterTab>	Provides a list of custom tabs that will be displayed by the administration tool @returns List of instantiated object that implement the IAdapterTab interface	
Name() string	Read-only property providing the name of the adapter	
SearchDetails() ISearchDetails	Read-only property providing a list of search details (search item names, data type, and operations).	
ServiceName() string	Read-only property providing the name of the service	

IEncryption

Type: **Interface**
Assembly: Oracle.UrmAdapter.dll

The class implementing IEncryption interface should be constructed to perform encryption/decryption services for userid's and passwords.

Operations

Method	Notes	Parameters
Decrypt() string	Decrypts the supplied string. @returns decrypted string.	**string** [in] inString
Encrypt() string	Encrypts the supplied string. @returns encrypted string.	**string** [in] inString

IEventController

Type: **Interface**
Assembly: Oracle.UrmAdapter.dll

Used to monitor source creation and deletion via the administration tool.

Operations

Method	Notes	Parameters
SourceCreated() SourceCreatedEventHandler	Event fired whan a source is created through the administration tool	
SourceDeleted() SourceDeletedEventHandler	Event fired whan a source is deleted through the administration tool	
Starting() EventHandler	Adapter processes are starting	
Terminating() EventHandler	Adapter processes are terminating	

Namespace: Oracle.UrmAdapter.Core.Data

Assembly: Oracle.UrmAdapter.dll

The Oracle.UrmAdapter.Core.Data namespace contains classes used for configuring Agent Framework, Agent Sources, Searches, and Search Mappings.

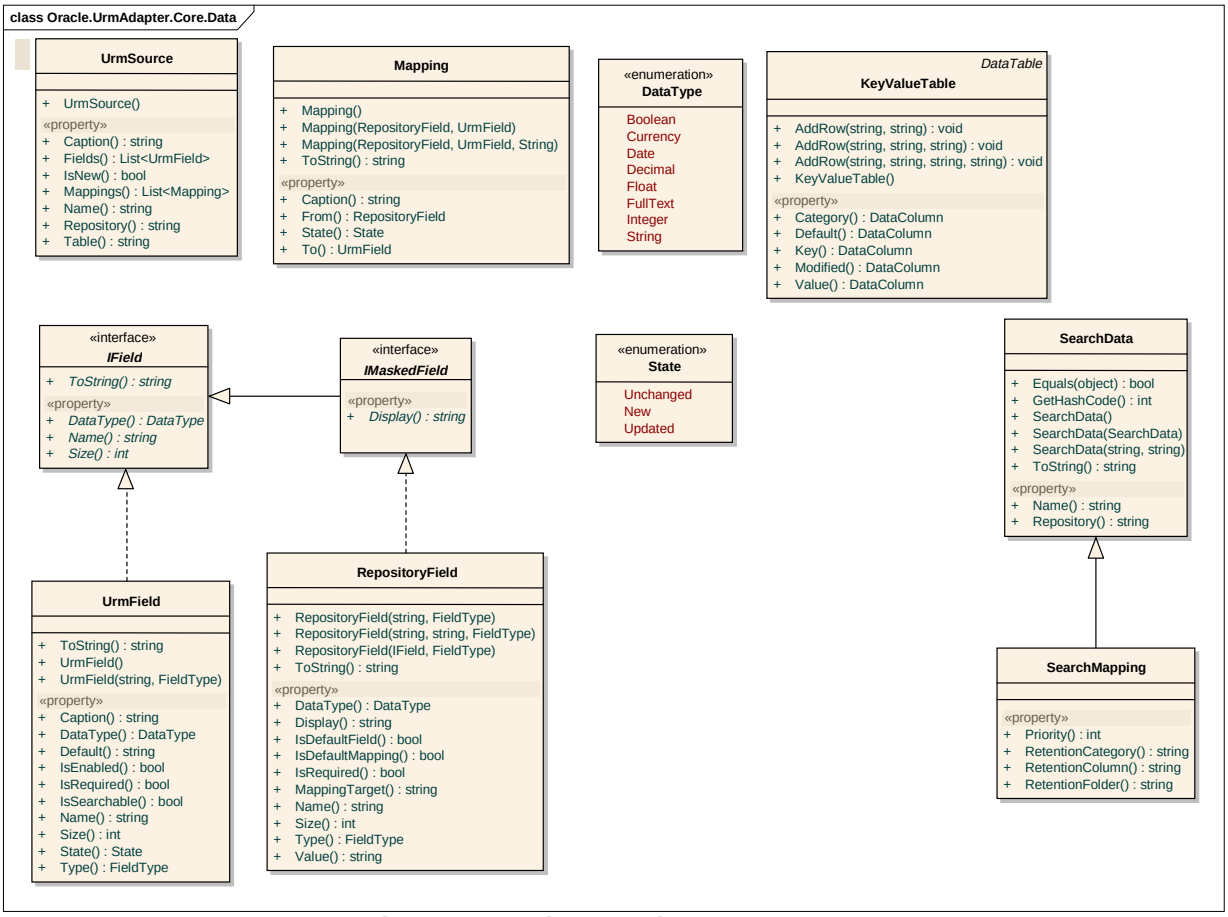

Figure 5: Oracle.UrmAdapter.Core.Data

DataType

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Enumeration to describe datatypes in adapter

Attributes

Attribute	Notes
Boolean	
Currency	
Date	
Decimal	Exact numerical value
Float	Approximate numerical value

Attribute	Notes
FullText	Full Text indicates a full text search can be performed.
Integer	Int 32 value
String	

KeyValueTable

Type: **Class** **DataTable**
Assembly: Oracle.UrmAdapter.dll

DataTable derived class used to hold configuration items and values.

Operations

Method	Notes	Parameters
AddRow() void		**string** [in] key **string** [in] value
AddRow() void		**string** [in] category **string** [in] key **string** [in] value
AddRow() void		**string** [in] category **string** [in] key **string** [in] value **string** [in] def
Category() DataColumn		
Default() DataColumn		
Key() DataColumn		
KeyValueTable()		
Modified() DataColumn		
Value() DataColumn		

Mapping

Type: **Class**
Assembly: Oracle.UrmAdapter.Core.Data

Class used to map Adapter metadata to URM metadata.

Operations

Method	Notes	Parameters
Caption() string	`Mapping caption (for display in URM)`	
From() RepositoryField	`Repository field side of mapping.`	
Mapping()	`Constructor`	
Mapping()	`Constructor setting intial mapping values`	**RepositoryField** [in] from Repository field **UrmField** [in] to Urm field
Mapping()	`Constructor setting initial mapping values`	**RepositoryField** [in] from Repository field **UrmField** [in] to Urm field **String** [in] caption Field caption (for display in URM)
State() State	`Property indicating the state of the mapping (Updated, New, Unchanged)`	
To() UrmField	`Urm field side of mapping.`	
ToString() string	`Overridden function used to display mapping as string @returns string`	

RepositoryField

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Definition for a repository's metadata field to be managed by URM.

Operations

Method	Notes	Parameters
DataType() DataType	Property representing the URM datatype for the field	
Display() string	Display (caption) for the field in URM	
IsDefaultField() bool	Property denoting that a default custom field will be created for the field	
IsDefaultMapping() bool	Property denoting that a default mapping will be created for the field	
IsRequired() bool	Property denoting that the field is required to be checked into URM	
MappingTarget() string	For default mappings, denotes the URM field for the map	
Name() string	Name of the field	
RepositoryField()	Constructor	**string** [in] name Name of repository field **FieldType** [in] type Data type of repository field
RepositoryField()	Constructor	**string** [in] display Display string **string** [in] name Field name **FieldType** [in] type Field type
RepositoryField()	Constructor	**IField** [in] field Field **FieldType** [in] type Field type
Size() int	Size of the field	
ToString() string		
Type() FieldType	Field type	
Value() string	Value of the field	

FieldType

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Enumeration of repository field types

Attributes

Attribute	Notes
Date	Current Date/time
Field	Repository field
Key	Key uniquely identifying item
Literal	Literal value (string or date)
URL	URL location of item

SearchData

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Top level description of searches.

Operations

Method	Notes	Parameters
Equals() bool	`Equality function` `@returns boolean`	**object** [in] obj object to compare to
GetHashCode() int		
Name() string	`Search name`	
Repository() string	`Repository name`	
SearchData()	`Constructor`	
SearchData()	`Constructor`	**SearchData** [in] search SearchData
SearchData()	`Constructor`	**string** [in] name Search name **string** [in] repository Repository name
ToString() string	`overridden function returning search name` `@returns Search name`	

SearchMapping

Type: **Class** **SearchData**
Assembly: Oracle.UrmAdapter.dll

Used to map searches to URM retention categories.

Operations

Method	Notes	Parameters
Priority() int	`Priority of the search.  The lower the number, the higher the priority.`	
RetentionCategory() string	`Retention category name`	
RetentionColumn() string	`Used to merge RetentionCategory and RetentionFolder properties. If no folder is defined, it will contain the retention category name. If a folder is defined, it will contain the folder name`	
RetentionFolder() string	`Retention folder name`	

State

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.Core.Data

Enumeration used to define the state of a record/data row

Attributes

Attribute	Notes
Unchanged	Record has not changed since last persisted state.
New	Record is new and has not been persisted
Updated	Record has changed since last persistence

UrmField

Type: **Class**

Assembly: Oracle.UrmAdapter.dll

The definition of a field from URM.

Operations

Method	Notes	Parameters
Caption() string		
DataType() DataType	`Field data type property`	
Default() string		
IsEnabled() bool		
IsRequired() bool		
IsSearchable() bool		
Name() string	`Field name property`	
Size() int	`Read-only property. Maximum field size. Only relevant to string type fields.`	
State() State		
ToString() string		
Type() FieldType		
UrmField()	`Constructor`	
UrmField()	`Constructor`	**string** [in] name Name of the field **FieldType** [in] type Field data type

FieldType

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Enumerated types of fields supported by URM

Attributes

Attribute	Notes
Text	Character string less than 101 characters
BigText	Character string less than 201 characters
Checkbox	Boolean (represented by a checkbox in URM User Interface)
Date	Date Time.
Int	Int 32 value
Memo	String > 200 characters

UrmSource

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Defines a URM Source

Operations

Method	Notes	Parameters
Caption() string	`Source caption (viewable in URM user interface)`	
Fields() List<UrmField>	`List of fields supported by URM`	
IsNew() bool	`Property that indicates if the URM Source has been persisted (false) or not (true)`	
Mappings() List<Mapping>	`Property for assigning a list of mappings between repository fields and URM fields`	
Name() string	`Source name`	

Method	Notes	Parameters
Repository() string	Repository (schema) name for the source Once the source has been created in URM, this property becomes read-only	
Table() string	Data table in URM database used to persist metadata information This value must be unique among all sources. Once the source has been created in URM, this property becomes read-only.	
UrmSource()	Constructor	

IField

Type: **Interface**
Assembly: Oracle.UrmAdapter.dll

Interface used to describe fields provided by a repository

Operations

Method	Notes	Parameters
DataType() DataType	Read-only property describing the data type of the field	
Name() string	Read-only property describing the name of the field	
Size() int	Read-only property describing the size of the field	
ToString() string		

IMaskedField

Type: **Interface** **IField**
Assembly: Oracle.UrmAdapter.dll

Interface to class used to perform field formatting

Operations

Method	Notes	Parameters
Display() string	Read-only property called to format the field.	

Namespace: Oracle.UrmAdapter.Core.Persistence

Assembly: Oracle.UrmAdapter.dll

The Oracle.UrmAdapter.Core.Persistence namespace contains a class used for configuring Adapter Framework / URM connectivity

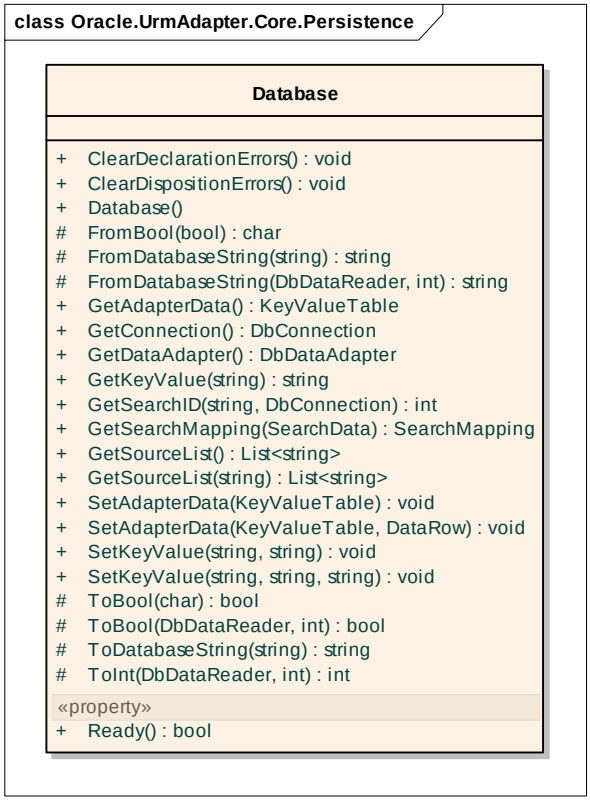

Figure 6: Oracle.UrmAdapter.Core.Persistence

Database

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Class used to access the framework database

Operations

Method	Notes	Parameters
ClearDeclarationErrors() void		
ClearDispositionErrors() void		
Public **Database()**	`Constructor`	
Protected **FromBool()** char	`Utility function to convert a boolean value to string @returns string representation`	**bool** [in] value boolean value
Protected **FromDatabaseString()** string	`Utility function to retrieve a string value from supplied string defaulting a string.Empty for null values`	**string** [in] value string to convert

Method	Notes	Parameters
	`@returns string value`	
Protected **FromDatabaseString()** string	`Utility function to retrieve a boolean value from a data reader column defaulting a string.Empty for null values @returns string value`	**DbDataReader** [in] reader open datareader **int** [in] column column index to check
GetAdapterData() KeyValueTable	`Retrieves a KeyValue table of all configuration parameters for the adapter. @returns KeyValueTable containing all defined Key/value pairs`	
GetConnection() DbConnection	`Retrieves an open DbConnection object to the Adapter database. @returns open DbConnection`	
GetDataAdapter() DbDataAdapter	`Retrieves an appropriate adapter of type DbDataAdapter. @returns DbDataAdapter`	
GetKeyValue() string	`Retrieves a configuration value based upon the key. @returns Configuration value`	**string** [in] key Configuration key
GetSearchID() int	`Retrieves the search ID based on the search name. @returns Search Id`	**string** [in] searchName Search name **DbConnection** [in] connection requires an Open DbConnection
GetSearchMapping() SearchMapping	`Retrieves a search mapping based on a search definition. @returns Search mapping for search`	**SearchData** [in] search Search definition
GetSourceList() List<string>	`Retrieves a list of defined URM sources @returns List of URM source names`	
GetSourceList() List<string>	`Retrieves a list of defined URM sources based on repository name @returns List of URM source names`	**string** [in] repository Repository name
Ready() bool		
SetAdapterData() void	`Updates the adapter configuration KeyValue table.`	**KeyValueTable** [in] keyValueTable Data table containing all key/value pairs
SetAdapterData() void	`Updates a particular row of an adapter configuration KeyValue table`	**KeyValueTable** [in] table Data table containing all key/value pairs **DataRow** [in] row Specific row to update
SetKeyValue() void	`Sets a configuration value based upon the key.`	**string** [in] key Configuration key **string** [in] value Configuration value
SetKeyValue() void	`Sets a configuration value based upon the key.`	**string** [in] key Configuration key **string** [in] value Configuration value **string** [in] category Key category
Protected **ToBool()** bool	`Utility function to convert a character (Y`	**char** [in] value

Method	Notes	Parameters
	or N) to a boolean @returns boolean	string value
Protected **ToBool()** bool	Utility function to retrieve a boolean value from a data reader column defaulting a false for null values @returns boolean value	**DbDataReader** [in] reader open datareader **int** [in] column column index to check
Protected **ToDatabaseString()** string	Utility function to substitute single quotes to double single quotes for insertion into databases @returns string value	**string** [in] value source value
Protected **ToInt()** int	Utility function to retrieve a integer value from a data reader column defaulting a -1 for null values @returns integer value	**DbDataReader** [in] reader open datareader **int** [in] column column index to check

Namespace: Oracle.UrmAdapter.Core.Search

Assembly: Oracle.UrmAdapter.dll

The Oracle.UrmAdapter.Core.Search namespace contains classes for creating declaration search definitions.

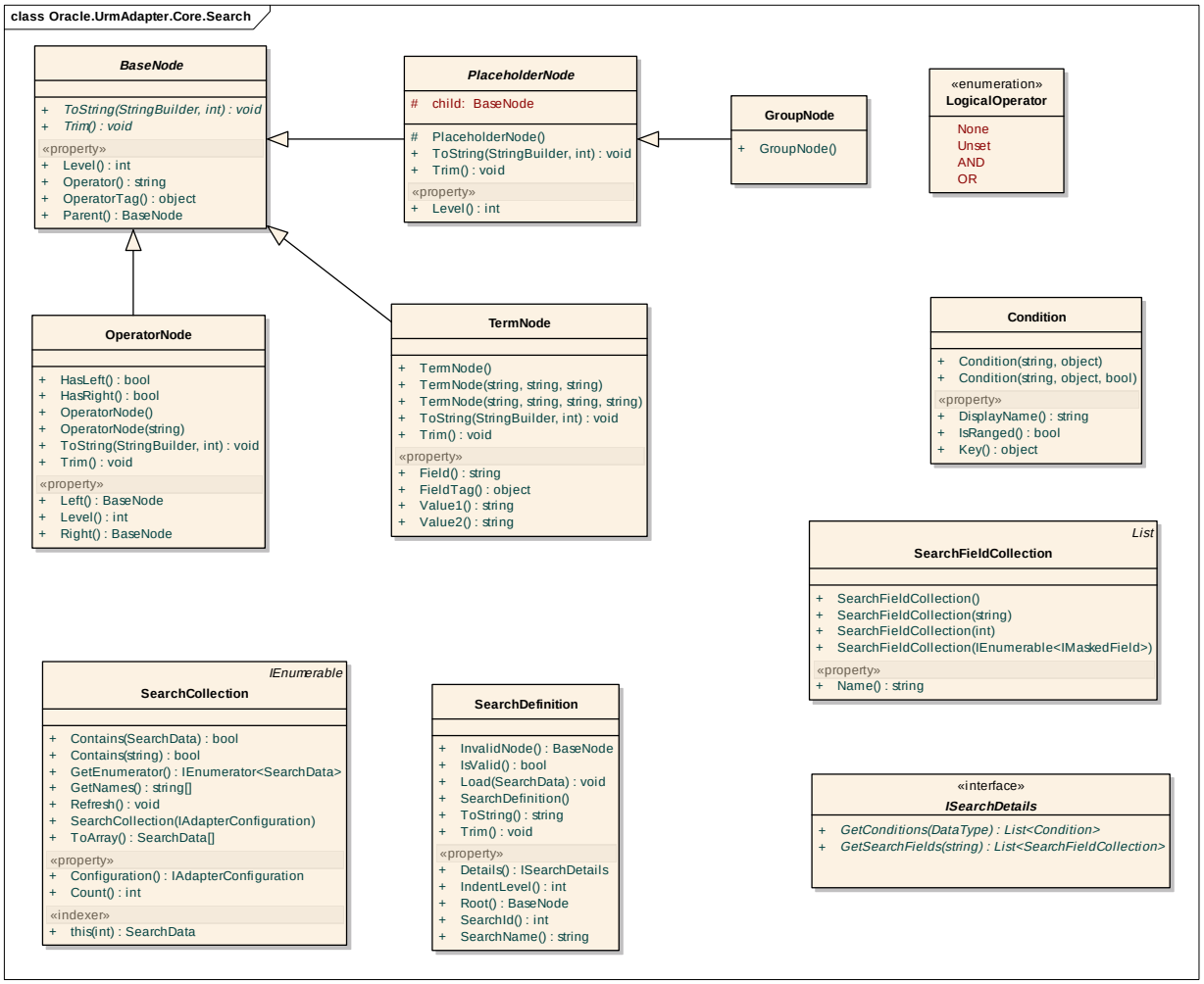

Figure 7: Oracle.UrmAdapter.Core.Search

BaseNode

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Abstract class used as base class for all search nodes in expression tree

Operations

Method	Notes	Parameters
Level() int	`Nodes level in the expression tree`	
Operator() string	`Expression operator`	
OperatorTag() object	`Ancillary operator information`	
Parent() BaseNode	`Node's parent node`	
ToString() void		**StringBuilder** [in] sb **int** [in] indentLevel

Method	Notes	Parameters
Trim() void		

Condition

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Defines search operations that can be performed on metadata data types.

Operations

Method	Notes	Parameters
Condition()	`Constructor`	**string** [in] displayName Displayable operator name **object** [in] key key used to identify operator for a type
Condition()	`Constructor`	**string** [in] displayName Displayable operator name **object** [in] key Key used to identify operator for a type **bool** [in] isRanged Indicates a tertiary condition (example "between")
DisplayName() string	`Display name`	
IsRanged() bool	`Indicates a tertiary condition (example "between")`	
Key() object	`Key used to identify operator for a type`	

GroupNode

Type: **Class** **PlaceholderNode**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
Public **GroupNode()**		

LogicalOperator

Type: **Enumeration**
Assembly: Oracle.UrmAdapter.dll

Enumerated set of logical operators for connecting terms in a search

Attributes

Attribute	Notes
None	`Operator is unknown`
Unset	`Operator is unset (generates an error when validating the expression tree)`
AND	`Logical AND`
OR	`Logical OR`

OperatorNode

Type: **Class** **BaseNode**
Assembly: Oracle.UrmAdapter.dll

Node in expression tree that defines an operation

Operations

Method	Notes	Parameters
HasLeft() bool	`Indicates if the left side of the expression is set @returns true if left side is present`	
HasRight() bool	`Indicates if the right side of the expression is set @returns true if right side is present`	
Left() BaseNode	`The left side (node) of the expression`	
Level() int	`Returns the node level in the expression tree`	
OperatorNode()	`Constructor`	
OperatorNode()	`Constructor`	**string** [in] theOperator Operator to store in the structure
Right() BaseNode	`The right side (node) of the expression`	
ToString() void	`Overriding function used to get a map of the expression tree`	**StringBuilder** [in] sb string builder instance **int** [in] indentLevel number of spaces to indent per level
Trim() void	`Utility used to remove grouping placeholders from the expression tree. Typically grouping placeholders are only needed for display purposes, so Trim should be called prior to parsing the expression tree.`	

PlaceholderNode

Type: **Class** **BaseNode**
Assembly: Oracle.UrmAdapter.dll

Attributes

Attribute	Notes
Protected **child** BaseNode	

Operations

Method	Notes	Parameters
Public **Level()** int		
Protected **PlaceholderNode()**		
Public **ToString()** void		**StringBuilder** [in] sb **int** [in] indentLevel
Public **Trim()** void		

SearchCollection

Type: **Class** **IEnumerable**
Assembly: Oracle.UrmAdapter.dll

Collection if Search Data items

Operations

Method	Notes	Parameters
Configuration() IAdapterConfiguration	Property for identifying configuration object (IAdapterConfiguration)	
Contains() bool	Used to check to see if a search data item is currently in the list @returns boolean, true if present	**SearchData** [in] otherData Item to check for
Contains() bool	Used to check to see if a search, by name is currently in the list @returns boolean, true if present	**string** [in] searchName Search name to check for
Count() int	Read-only property showing number of items in collection	
GetEnumerator() IEnumerator<SearchData>		
GetNames() string	Retrieves a string array of all search names in the list @returns string array of search names	
Refresh() void	Updates the list with all searches currently in the framework database or defined by the adapter	
SearchCollection()		**IAdapterConfiguration** [in] configuration
this() SearchData	Read-only property for retrieving the search data based on index @returns SearchData item	**int** [in] index
ToArray() SearchData	Returns a SearchData array of all items currently in the collection @returns SearchData array	

SearchDefinition

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Complete definition of a search. Search definitions are loaded in as an expression tree.

Operations

Method	Notes	Parameters
Details() ISearchDetails		
IndentLevel() int		
InvalidNode() BaseNode	Identifies the first invalid node of the expression tree. If the tree is valid, this will return null. @returns First invalid node (null if valid tree)	
IsValid() bool	Identifies if the expression tree is valid @returns boolean true if valid tree	
Load() void	Loads the SearchDefinition's expression tree based on a SearchData object	**SearchData** [in] search SearchData object
Root() BaseNode		
SearchDefinition()	Constructor	
SearchId() int		
SearchName() string		
ToString() string	Retrieves a string representing the expression tree @returns String representation of the expression	
Trim() void	Used to remove grouping placeholders from the expression tree. Typically the	

Method	Notes	Parameters
	placeholders are only useful for display of the tree, and expression parsing will be done after the Trim function is called.	

SearchFieldCollection

Type: **Class** **List**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
Name() string		
SearchFieldCollection()		
SearchFieldCollection()		**string** [in] name
SearchFieldCollection()		**int** [in] capacity
SearchFieldCollection()		**IEnumerable<IMaskedField>** [in] collection

TermNode

Type: **Class** **BaseNode**
Assembly: Oracle.UrmAdapter.dll

TermNode is a node in the expression tree that contains actual values to be operated on.

Operations

Method	Notes	Parameters
Field() string	Metadata field to perform the operation on	
FieldTag() object	Ancillary field information	
TermNode()	Constructor	
TermNode()	Constructor	**string** [in] field Metadata field to perform the operation on **string** [in] theOperator Comparison operator **string** [in] value1 Value to compare to metadata field
TermNode()	Constructor	**string** [in] field Metadata field to perform the operation on **string** [in] theOperator Comparison operator **string** [in] value1 Value to compare to metadata field **string** [in] value2 Second value to compare (for ranged operators)
ToString() void	Overridden function used to display node's contents	**StringBuilder** [in] sb String Builder instance **int** [in] indentLevel number of spaces to

Method	Notes	Parameters
		indent per level
Trim() void	Used to remove grouping placeholders from the expression tree. Typically the placeholders are only useful for display of the tree, and expression parsing will be done after the Trim function is called.	
Value1() string	Value to compare to metadata field	
Value2() string	Second value to compare (for ranged operators)	

ISearchDetails

Type: **Interface**
Assembly: Oracle.UrmAdapter.dll

Interface to describe search fields and search conditions supported by the adapter

Operations

Method	Notes	Parameters
GetConditions() List<Condition>	Provides a list of conditions for a datatype @returns List of conditions	**DataType** [in] type Data type requested
GetSearchFields() List<SearchFieldCollection>	Provides a list of SearchFieldCollections for a particular repository(schema) @returns List of SearchFieldCollections	**string** [in] repository Name of repository

Namespace: Oracle.UrmAdapter.Core.Threads

Assembly: Oracle.UrmAdapter.dll

The Oracle.UrmAdapter.Core.Threads namespace contains a class for creating threads that can be monitored via the Administration UI.

Figure 8: Oracle.UrmAdapter.Core.Threads

AdapterThread

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Abstract class used for creating threads that can be monitored by the Administration utility. This class extends Microsoft's System.Threading.Thread class. Information on all operations (except the Run virtual method) can be found in Microsoft documentation for the thread class. Typically, the user will create a Run method that does the work of the class.

Operations

Method	Notes	Parameters
Abort() void		
Abort() void		**object** [in] stateInfo
Protected **AdapterThread()**		
Static **AllocateDataSlot()** LocalDataStoreSlot		
Static **AllocatedNamedDataSlot()** LocalDataStoreSlot		**string** [in] name
Static **BeginCriticalRegion()** void		
Static **BeginThreadAffinity()** void		
Static **CurrentContext()** Context		
CurrentCulture() CultureInfo		
Static **CurrentPrincipal()** IPrincipal		
Static **CurrentThread()** Thread		
CurrentUICulture() CultureInfo		
Static **EndCriticalRegion()** void		
Static **EndThreadAffinity()** void		
Equals() bool		**object** [in] obj
Equals() bool		**AdapterThread** [in] otherThread
Equals() bool		**Thread** [in] otherThread
ExecutionContext() ExecutionContext		
Static **FreeNamedDataSlot()** void		**string** [in] name
GetApartmentState() ApartmentState		
Static **GetData()** object		**LocalDataStoreSlot** [in] slot
Static **GetDomain()** AppDomain		
Static **GetDomainID()** int		
GetHashCode() int		
Static **GetNamedDataSlot()** LocalDataStoreSlot		**string** [in] name
Interrupt() void		
IsAlive() bool		
IsBackground() bool		
IsThreadPoolThread() bool		
Join() void		
Join() void		**int** [in] millisecondsTimeout
Join() void		**TimeSpan** [in] timeout
Static **JoinAll()** void		**AdapterThread[]** [in] agentThreads
ManagedThreadId() int		
Static **MemoryBarrier()** void		

Method	Notes	Parameters
Name() string		
Protected **OnStarted()** void		
Protected **OnStatusChanged()** void		
Protected **OnStopped()** void		
Protected **OnUnhandledException()** void		**UnhandledExceptionEvent Args** [in] e
Priority() ThreadPriority		
Static **ResetAbort()** void		
Protected **Run()** void	`This function must be elaborated, and is the starting point for the thread. Additional parameters needed for the function should be placed in class level variables and set prior to calling the Run function.`	
SetApartmentState() void		**ApartmentState** [in] state
Static **SetData()** void		**LocalDataStoreSlot** [in] slot **object** [in] data
Static **Sleep()** void		**int** [in] millisecondsTimeout
Static **Sleep()** void		**TimeSpan** [in] timeout
Static **SpinWait()** void		**int** [in] iterations
Start() void		
Start() void		**object** [in] parameter
Started() EventHandler		
Status() string		
StatusChanged() EventHandler		
Stopped() EventHandler		
Thread() Thread		
Static **Threads()** List<AdapterThread>		
ThreadState() ThreadState		
ToString() string		
Static **ToThreadArray()** Thread		**AdapterThread[]** [in] agentThreads
TrySetApartmentState() bool		**ApartmentState** [in] state
UnhandledException() UnhandledExceptionEventHandler		

Namespace: Oracle.UrmAdapter.Core.Timers

Assembly: Oracle.UrmAdapter.dll

The Oracle.UrmAdapter.Core.Timer namespace contains a class for creating timed threads that can be monitored via the Administration UI.

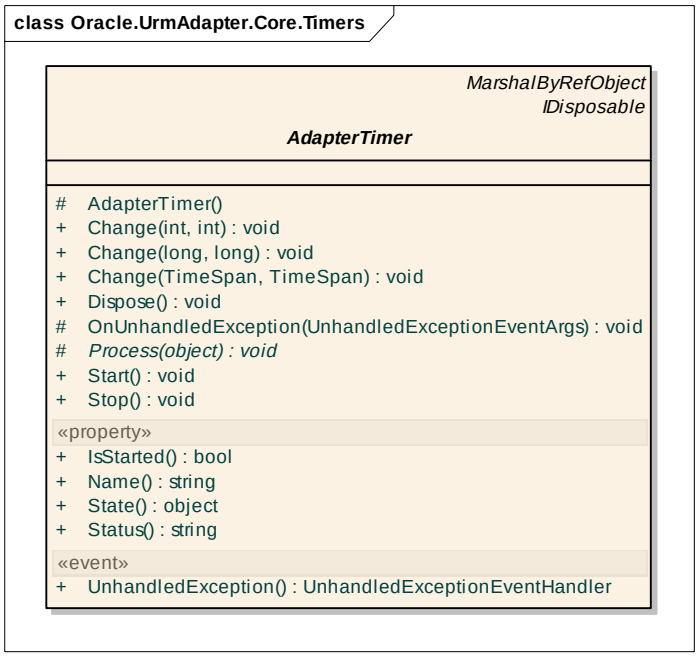

Figure 9: Oracle.UrmAdapter.Core.Timers

AdapterTimer

Type: **Class** **MarshalByRefObject, IDisposable**
Assembly: Oracle.UrmAdapter.dll

The AdapterTimer abstract class provides the ability to create timers that are monitored by the administration utility. Typically the user will elaborate the Process function that will be called with the timer fires.

Operations

Method	Notes	Parameters
Protected **AdapterTimer()**	`Constructor`	
Change() void	`Sets the timer intervals`	**int** [in] dueTime Time (in milliseconds) the first interval fires **int** [in] period Time (in milliseconds) successive intervals fire
Change() void	`Sets the timer intervals`	**long** [in] dueTime Time (in milliseconds) the first interval fires **long** [in] period Time (in milliseconds) successive intervals fire
Change() void	`Sets the timer intervals`	**TimeSpan** [in] dueTime Timespan to wait for the first interval to fire **TimeSpan** [in] period Timespan to wait for successive intervals to fire
IsStarted() bool	`Indicates if the timer has started`	

Method	Notes	Parameters
Name() string	`Display name to give the timer for display on administration utility`	
Protected **OnUnhandledException()** void		**UnhandledExceptionEventArgs** [in] e
Protected **Process()** void	`Abstract function that must be elaborated. This function will be called with the timer fires.`	**object** [in] state
Start() void	`Starts the timer thread`	
State() object	`Current state of the timer`	
Status() string	`Status name to give the timer for display on administration utility`	
Stop() void	`Stops the timer thread`	
UnhandledException() UnhandledExceptionEvent Handler		

Namespace: Oracle.UrmAdapter.Diagnostics

Assembly: Oracle.UrmAdapter.dll

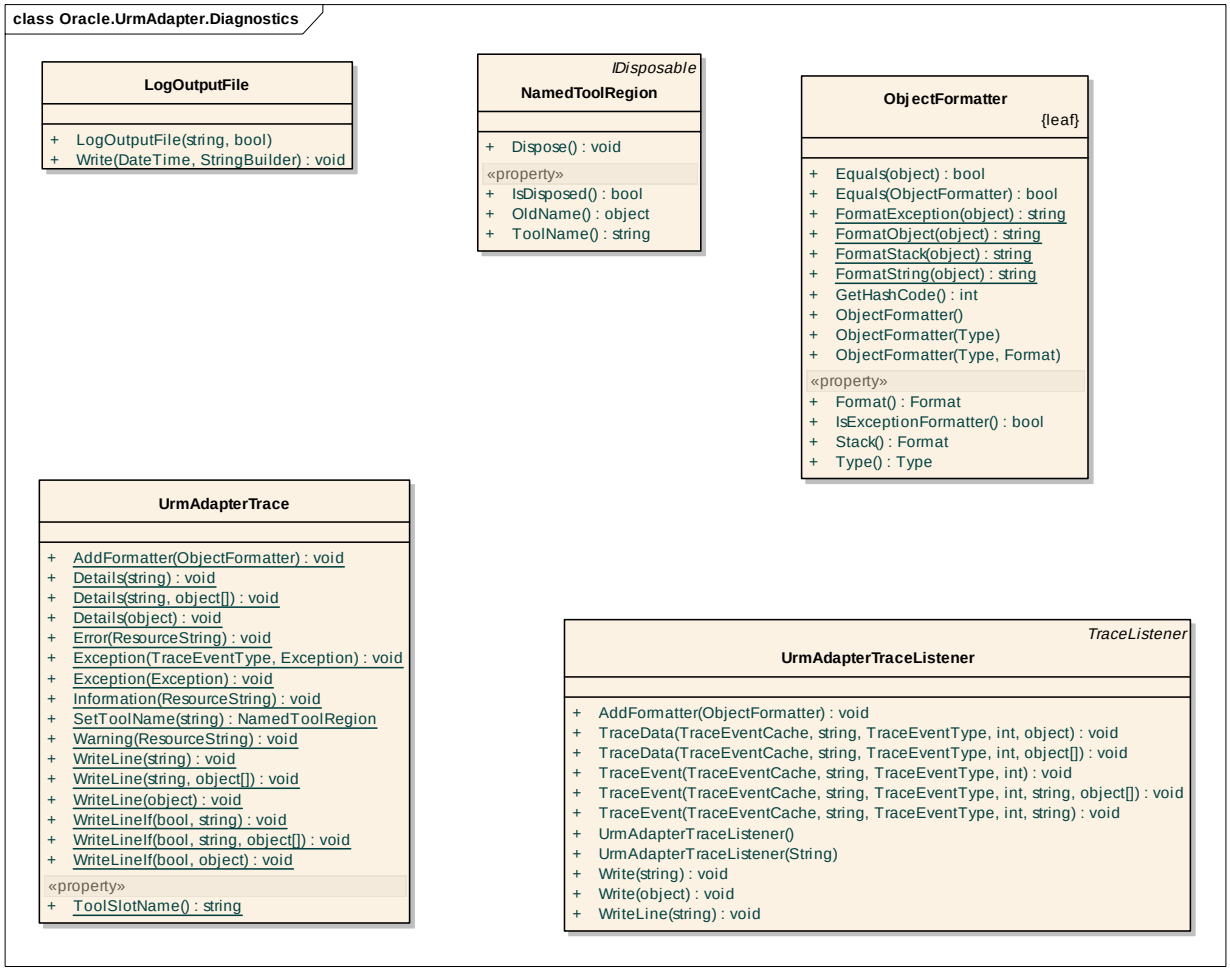

Figure 10: Oracle.UrmAdapter.Diagnostics

LogOutputFile

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
LogOutputFile()		**string** [in] path **bool** [in] destinationIsUnicode
Write() void		**DateTime** [in] fileNameTime **StringBuilder** [in] line

NamedToolRegion

Type: **Class** **IDisposable**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
Dispose() void		
IsDisposed() bool		
OldName() object		
ToolName() string		

ObjectFormatter

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Sealed class used for formatting objects

Operations

Method	Notes	Parameters
Equals() bool	`Equality function` `@returns True if equal`	**object** [in] obj Object to compare to
Equals() bool	`Equality function` `@returns True if equal`	**ObjectFormatter** [in] otherFormatter ObjectFormatter to compare to
Format() Format	`Object formatter property`	
Static **FormatException()** string	`Returns a message exception` `@returns Exception message`	**object** [in] data The exception to format
Static **FormatObject()** string	`Returns the ToString() value of the object` `@returns ToString() value`	**object** [in] data Object to be formatted
Static **FormatStack()** string	`Returns the stack trace of an exception` `@returns StackTrace value`	**object** [in] data Exception to be formatted
Static **FormatString()** string	`Returns of data casted as string` `@returns String representation of object`	**object** [in] data Object to be formatted
GetHashCode() int		
IsExceptionFormatter() bool	`Read-only property identifying if the formatter is an Exception formatter`	
ObjectFormatter()		
ObjectFormatter()		**Type** [in] type
ObjectFormatter()		**Type** [in] type **Format** [in] format
Stack() Format	`Stack formatter property`	
Type() Type	`Type formatter property`	

UrmAdapterTrace

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Custom trace functionality for use with the administration UI and URM logging.

Operations

Method	Notes	Parameters
Static	`Used to add a type formatter to the`	**ObjectFormatter** [in] formatter

Method	Notes	Parameters
AddFormatter() void	`trace`	Formatter object to add
Static **Details()** void	`Sends a "details" level message to the trace listener`	**string** [in] message Message to add
Static **Details()** void	`Sends a formatted "details" level message to the trace listener`	**string** [in] format Format encoded string **object[]** [in] parameters Parameters to add to message
Static **Details()** void	`Sends a "details" level message to the trace listener`	**object** [in] data Object (converted to string) to add
Static **Error()** void	`Sends an "error" level message to the trace listener`	**ResourceString** [in] resource Resource String containing message
Static **Exception()** void	`Sends a formatted exception to the trace listener`	**TraceEventType** [in] type Identifies type of trace **Exception** [in] exception Exception to be traced
Static **Exception()** void	`Sends a formatted exception to the trace listener`	**Exception** [in] exception Exception to be traced
Static **Information()** void	`Sends an "information" level message to the trace listener`	**ResourceString** [in] resource Resource String containing message
Static **SetToolName()** NamedToolRegion	`Sets the tool name for viewing through the administration utility` `@returns NamedToolRegion object`	**string** [in] toolName Name of tool
Static **ToolSlotName()** string		
Static **Warning()** void	`Sends a "warning" level message to the trace listener`	**ResourceString** [in] resource Resource String containing message
Static **WriteLine()** void	`Writes a string to the trace listener`	**string** [in] message message to write
Static **WriteLine()** void	`Writes a formatted string to the trace listener`	**string** [in] format The format encoded string to write **object[]** [in] parameters list of parameters to populate the format string
Static **WriteLine()** void	`Writes an object (converted to string) to the trace listener`	**object** [in] data Object (converted to string) to add
Static **WriteLineIf()** void	`Writes a string to the trace listener if a condition is true`	**bool** [in] shouldWrite Boolean indicating if message should be written **string** [in] message Message to write
Static **WriteLineIf()** void	`Writes a string to the trace listener if a condition is true`	**bool** [in] shouldWrite Boolean indicating if message should be written **string** [in] format Format encoded string **object[]** [in] parameters Parameter list to add to format string
Static **WriteLineIf()** void	`Writes a string to the trace listener if a condition is true`	**bool** [in] shouldWrite Boolean indicating if message should be written **object** [in] data Object converted to string

UrmAdapterTraceListener

Type: **Class** **TraceListener**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
AddFormatter() void		**ObjectFormatter** [in] formatter
TraceData() void		**TraceEventCache** [in] eventCache **string** [in] source **TraceEventType** [in] eventType **int** [in] id **object** [in] data
TraceData() void		**TraceEventCache** [in] eventCache **string** [in] source **TraceEventType** [in] eventType **int** [in] id **object[]** [in] data
TraceEvent() void		**TraceEventCache** [in] eventCache **string** [in] source **TraceEventType** [in] eventType **int** [in] id
TraceEvent() void		**TraceEventCache** [in] eventCache **string** [in] source **TraceEventType** [in] eventType **int** [in] id **string** [in] format **object[]** [in] args
TraceEvent() void		**TraceEventCache** [in] eventCache **string** [in] source **TraceEventType** [in] eventType **int** [in] id **string** [in] message
UrmAdapterTraceListener()		
UrmAdapterTraceListener()	`This constructor is used when the initializeData attribute is set for this trace listener's configuration node. The name of the server (or tool) which owns this logging may be specified via app.config like this: <listeners> <add name="AdapterTracer" type="Oracle.UrmAdapter.Diagnostics.UrmAdapterTraceListener, Oracle.UrmAdapter" initializeData="VaultAdapter" /> </listeners></source>`	**String** [in] server The name of the server, or tool, generating this logging
Write() void		**string** [in] message
Write() void		**object** [in] o
WriteLine() void		**string** [in] message

Namespace: Oracle.UrmAdapter.ResourceHandling

Assembly: Oracle.UrmAdapter.dll

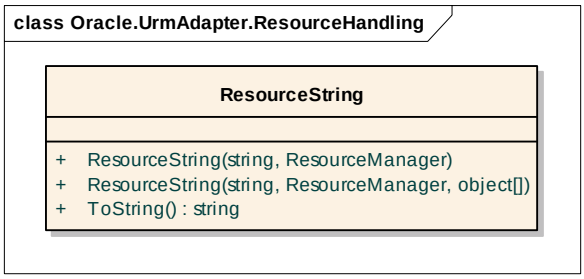

Figure 11: Oracle.UrmAdapter.ResourceHandling

ResourceString

Type: **Class**
Assembly: Oracle.UrmAdapter.dll

Operations

Method	Notes	Parameters
ResourceString()		**string** [in] stringName **ResourceManager** [in] resourceManager
ResourceString()		**string** [in] stringName **ResourceManager** [in] resourceManager **object[]** [in] args
ToString() string		