Metreos Communications Environment

LDAP Directory Reference Application

8/16/2004
Version 1.0

OVERVIEW

The LDAP Directory application bundled with the Metreos Communications Environment (MCE) is first and foremost intended for reference and educational purposes only. Additional features and error handling must be implemented before this application is suitable for use in a production environment.

However, basic functionality is present and the application should run without issues on a properly configured Application Server. See the installation chapter for a list of required components.

This application demonstrates the considerations, flow, and behavior necessary to create a Cisco IP phone-based service. This sample will make use of HTTP provider-managed sessions, to allow an approach which should resonate with a web developer when using sessions found in ASP.NET or PHP, for example. A more thorough explanation of HTTP sessions can be found in the WebDialer documentation.

Along with sessions and the use of Cisco IP phone XML objects, this sample will also demonstrate the reasoning behind the decision to make a custom native action and a custom native type, created to query and manipulate the data obtained from the LDAP server.

The aim of the LDAP Directory application is to allow a user with a Cisco IP phone to browse the company directory, either by browsing a list of users defined by first and last name, or by navigating through all people with a last name beginning with a particular letter.

While the user is browsing the results, he may choose to dial any directory item he has highlighted on the screen, by pressing the SoftKey 'Dial'. Navigation is achieved with the use of the following SoftKeys:

- Next: Moves one page forward.
- Far Next: Moves 3 pages.
- Prev: Moves one page back.
- Far Prev: Moves 3 pages back.

A 'page' is at most 32 directory entries, a limit determined by the maximum number of entries allowable in a directory listing on the Cisco IP phone.

Also important to understand is that Cisco IP phones come in different versions and capabilities; for this application, we distinguish between 7970 and non-7970 phones. The marked difference of 7970 phones is that they support the use of PNG images in two 7970-only commands. In this example, we utilize one of these commands in displaying the main menu on the 7970. If a non-7970 phone is detected, a non-graphical menu is displayed to the user.

INSTALLATION

There are several components which must be configured in order for this application to function properly.

Media Server

The Media Server is not needed for this application.

Application Server

This guide assumes that the Application Server has been setup and configured according to the procedure outlined in the User's Guide. In particular, this application requires that the HTTP protocol provider be installed.

Start the Application Server.

The application must now be installed. This is done by clicking on the *Applications* link on the MCE Admin main page. Clicking *Add Application* will bring up a file chooser dialog for files with a *.mca* extension. Locate the bin/mca directory and double-click on *LdapDirectory.mca*. This will initiate the application installation procedure. The Application server will log a series of Info-level messages ending with an "installation successful" message.

The MCE Admin will then return to the application listing page. The LDAP Directory application will be listed with a *configure* link next to it. Following is a list of the values and how they should be configured:

- Ldap_Address: The IP address of an LDAP server to use for this application.
- Ldap_Port: The port that the LDAP server uses. Typically 389.
- Ldap_Username: The username which has access to LDAP when making queries. This value is of the form of an LDAP 'dn' entry, such as 'cn=Directory Manager dc=metreos dc=com'. This is used particularly for binding to the LDAP server. If you do not want a binding to occur, leave the username blank. Not all LDAP setups require that a binding take place. *The LDAP library which this application uses can not bind on Windows.*
- Ldap_Password: This is the password to use when binding to the LDAP server.
- Search_Base: The base element of all entries you would like returned by the LDAP query. This is also in the form of an LDAP entry.
- First_Name_Attribute: The LDAP attribute used to specify the first name of a person in the entries configured on the LDAP server.
- Last_Name_Attribute: The LDAP attribute used to specify the last name of a person in the entries configured on the LDAP server.
- Telephone_Number: The LDAP attribute used to specify the telephone number of a person in the entries configured on the LDAP server.

- Unknown_Number: This number will be used in a directory entry presented to the phone if the person in the LDAP query has no telephone number. For instance, providing the human resources phone number for the company would be a reasonable choice for this field.
- Main_Menu_URL: An external URL pointing to the PNG menu image used for the 7970 Cisco IP phone for the main menu of this application. This image can be found in the 'contrib' directory found in this sample, and should be hosted at a URL for which the test phone can access.

Cisco CallManager

In CallManager 3.x or 4.x, a Cisco IP phone service must be set up to the URL which initiates an instance of this application, i.e., http://<ApplicationServerIP>:8000/LdapDirectory/MainMenu.

Once this is successfully completed, register a Cisco IP phone with this service for testing.

HOW IT WORKS

Overview

A Cisco IP phone sends HTTP requests based on user action. Given that we have constructed our application correctly, if the user presses a SoftKey, an HTTP request is sent to the script instance governing the current session. Multiple instances are created as multiple users concurrently use the LDAP Directory service, so a requirement for correct behavior of this application is that the phone, when making requests, will signify that it is making a session-bound request. This behavior is made possible by instructing the phone to include a 'metreosSessionId' query parameter in every request to the Application Server. The Application Server will test for the presence for this query parameter, and route it to the script instance which matches its value.

The Cisco IP Phone has a number of predefined modes which this application makes use of. These modes are triggered by sending the phone certain XML-formatted data containing the appropriate information to active the desired mode.

The modes which the application uses are:

- CiscoIPPhoneDirectory – lists people found in LDAP server query, with phone numbers.
- CiscoIPPhoneMenu – lists every letter in the alphabet, to allow 'Search by Letter' listing of people. Also used to display the main menu on all non-7970 phones.
- CiscoIPPhoneInput – used to accept alphabetic input from the user, which is then used to create the query based on first name / last name text strings.
- CiscoIPPhoneGraphicalFileMenu – a 7970-only command, which will display a PNG image, and also uses the touch screen capabilities of the 7970.
- CiscoIPPhoneText – used to display error messages.

The Cisco IP phone is expecting properly formatted XML for each of these modes; here we make use of native actions found in the MCE framework which make creating these XML objects a simple matter for the developer, in that understanding the XML schema or even XML itself, is not necessary.

Native Actions

The one custom native action needed by this application was one which would perform a query against the LDAP server. It makes use of a Novell-created, freely available library which handles all of the LDAP protocol and connection control to communicate with the LDAP server. So, we created an action which performs the LDAP search query by making use of this library.

- *Search* uses the Novell LDAP library to connect to the LDAP server, format a LDAP search query, make the query, and then create a sorted list of all the results

returned by the server. The sorted list used to hold the resultant values also corresponds to the custom native type created for this application. It follows then that the type of the one result data field of this action should be 'Metreos.Types.LdapDirectory.ResultSet'. Given that the application assigns this result data field to a variable of this same type, the application can then access the data contained in the sorted list after this native action finishes executing via this variable.

Native Types

ResultSet contains as an inner value a custom collection derived from a System.Collection.SortedList. The derivation was created as the amount of information needed for storage exceeds a simple value type, such as name and telephone number of a single user. *ResultSet* provides a number of methods to easily obtain the information found within this inner collection in a manner geared towards making C# snippets in the MVD environment as succinct and error-free as possible. For example, by only using an integer argument, the GetNameAtIndex and GetNumberAtIndex methods of this type allow the application developer to quickly obtain a formatted name and number at any index of the collection.

```
string username = resultSet.GetNameAtIndex(5);
string userTelephoneNumber = resultSet.GetNumberAtIndex(5);
```

With these utilities, we can then easily construct a CiscoIPPhoneDirectory object containing the name and number of a series of users: in this case, the users returned by an LDAP query.

Triggering Event Handler: OnGotRequest

The script is triggered on an HTTP request from the phone. This particular request is generated when the user navigates to the 'services' listing on his Cisco IP phone, and chooses the LDAP Directory application. The exact name of the application in this services listing is determined when the service was set up in CallManager.

In creating the XML response to send to the Cisco IP phone for this request, the event handler, 'OnGotRequest', will create either a CiscoIPPhoneMenu or CiscoIPPhoneGraphicalFileMenu based on simple determination logic as to whether the phone is a 7970 or non-7970 model. A small aside about determining the model: the event handler parses the 'Accept' header which comes in with every Cisco IP Phone request. If the header contains mention of PNG, then it is assumed to be a 7970. In CallManager 4.0, there is also a specific header sent by Cisco IP Phones which allows better determination of the model of the phone. For our purposes, and for backwards compatibility with versions of CallManager other than 4.0, the method we are using in this application appears to be most suitable.

One should take a moment and review the process in creating the CiscoIPPhoneMenu and CiscoIPPhoneGraphicFileMenu object. There are a number of modular components found within these and all other Cisco IP Phone XML objects, such as menu items, softkeys, touch-areas, etc. In order to create a fully functional menu, one will typically use not only the 'CreateMenu' action, which returns a Metreos.Types.CiscoIPPhone.Menu object, but would then also proceed to append these additional components. In this particular event handler, we add menu items in the case of either the 7970-specific GraphicFileMenu object or the Menu object. For those initiated with the different versions of Cisco IP Phone phones and the capabilities for each type, it is not surprising that it is meaningless and not advised to fill in the touch screen coordinates when using the AddMenuItem action in creating a Menu object, as this non-graphical menu does not support custom-defined touch areas.

Before this event handler exits, it responds to the phone with the XML object as the content of the body, and sets the session query parameter, 'metreosSessionId' with the value of 'Routing GUID'. Specifying this query parameter will allow requests to navigate to the new session, created when the initial HTTP request came in.

From either the non-7970 or 7970 menu, the user will be able to navigate to the 'NameSearch' or the 'LetterListing' event handler.

Session-Bound Event Handler: NameSearch

This event handler creates a CiscoIPPhoneInput object which allows the user to specify a first name and/or last name to search on. The first name and last name do not have to be complete; partial names are allowed.

The 'NameSearchResults' event handler is triggered when the user selects the 'submit' key. Query parameters in the URL specify the first and/or last name that the user wishes to search on.

Session-Bound Event Handler: NameSearchResults

The presence of the 'firstname' and 'lastname' query parameters is essential to this event handler executing properly. The HTTP Provider and QueryParamCollection native type provides a rather simple way of validating and accessing this information.

In every HTTP request which comes up in a Metreos.Providers.Http.GotRequest event, the incoming event parameter 'query' defines all the information after and including the '?' of URL used to reach the Application Server. Unfortunately, a text string is not very useful when dealing with key-value pairs found in the query parameters. However, the Metreos.Types.Http.QueryParamCollection is already available for the developer to access the key-values of the query parameters, using the name as the index into the collection. Also be sure to initialize the QueryParamCollection with the incoming query string event parameter, 'query'.

```
string queryParameterValue = queryParams["someQueryParameter"];
```

The LDAP search is then executed, using these incoming search parameters as well as the LDAP server-specific values configured once the application was installed.

Once the search completes, the ResultSet is ready to use for populating the first CiscoIPPhoneDirectory directory to send to the user. After the object is created and the appropriate SoftKeys are added, a response is sent to the phone containing the object. The SoftKeys contained in this response which relate to navigation specify an 'index', 'forward', and 'far' as query parameters. 'Index' is used to specify the last entry index sent to the phone, 'forward' indicates the direction of the incoming navigation request, and 'far' indicates if the paging logic should move 3 pages of entries, instead of 1. An example using the actual navigational SoftKeys: for the 'Next' SoftKey, the underlying URL will contain an index of 32, far will be false, and forward will be true. For 'Far Prev', the underlying URL will contain an index of 32, far will be true, and forward will be false. When the user selects one of these navigation SoftKeys, the NameSearchPagedResults handler will be responsible for handling the HTTP request generated by this user action.

Session-Bound Event Handler: NameSearchPagedResults

With the three incoming query parameters defined; index, far and forward, this event handler will be able to compute the new index in order to begin creating a new CiscoIPPhoneDirectory object for a new page. In other words, a different set of 32 entries will be shown depending on which of the four navigational SoftKeys, 'Next', 'Prev','Far Next', and 'Far Prev', was selected.

The SoftKeys for this new response are each individually computed. Depending on the location in the ResultSet, some SoftKeys are not shown. A good example is when the user has navigated to the end of the ResultSet. At this point, the 'Next' and 'Far Next' SoftKeys are not shown, as they have no purpose other than to confuse the user. Moreover, the absence of these keys also serves to indicate to the user that the end of the search results has been reached.

The CiscoIPPhoneDirectory object sent to the user has navigational SoftKeys which point back to this same event handler. At this point, NameSearchPagedResults will handle all navigation for this search.

Session-Bound Event Handlers: LetterListing, LetterListingResults, and LetterListingPagedResults

The design for the 'Search by Letter' menu option is the same as the 'Name Search' option. In fact, these three event handlers proceed almost identically to their counterparts: NameSearch, NameSearchResults, and NameSearchPagedResults. The differences are so small as to make expounding on them in this example meaningless.

CLOSING REMARKS

A significant limitation of this application is that it has no means to solve the common problem of paging found in large LDAP result sets, which occurs if the number of entries in the query reaches the maximum number of results that the server will allow to be sent back down the connection created for querying. This value is usually 500 or 1000 by default in most installations. There are a number of workarounds, some LDAP-server version and vendor specific. A discussion of the techniques which could be used to solve this problem is beyond the scope of this document.

One practical enhancement would be to allow connecting to v2 LDAP server based on a configuration option, which is very simple in terms of working with the Novell LDAP stack. Also, the allowing the administrator to configure the format of the results would be a sensible addition. For example, [Last Name], [First Name] versus [First Name] [Last Name], for each row in the directory listing.