Metreos Communications Environment

Weather Reference Application

8/16/2004
Version 1.0

OVERVIEW

The Weather 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.

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, created to query an XML weather service provided by the NOAA National Weather Service.

The aim of the Weather application is to allow a user with a Cisco IP phone choose a state, and then a station contained within that state to see the current weather.

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 *Weather.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.

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/Weather/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 Weather 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:

- CiscoIPPhoneMenu – lists every state and station for a state.
- CiscoIPPhoneText – used to display error messages and the weather information for a station

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.

The flow of this application is standard for Cisco IP Phone service applications: the application is registered to handle various incoming HTTP requests, based on the action taken by the user.

However, the XML weather feed service provided by the NOAA required the native actions to be constructed in a very particular format, described below.

The weather feed service provides a listing of every single reporting station in a large XML file. Information such as the station ID, station name, state in which the station is located, and the URL which contains the latest XML weather information for that station is all contained in this file.

With the information in this file (located in contrib/valdstations.xml), we could then show a stations-per-state list on the user's phone when they first navigate to the web service. Unfortunately, it is impractical to parse this large file every time the user uses our service. We need a quicker solution.

In plugins, the Weather .NET solution contains a project named Station Parser. This project parses the validstations.xml file and creates a new C# file which essentially contains a massive static method. This method populates a `SortedList` with a listing of all stations by state. Each entry for a station contains the displayable station name, the station ID, and the URL to access the weather information for that station.

With this information now available at runtime, there is no longer a need to parse the validstations.xml file.

Native Actions

The Weather Application uses 3 custom actions, all of which are straightforward, because of the information contained in the `SortedList`

- GetStates – returns all valid state abbreviations extracted from the contrib/validstations.xml file. This action merely iterates through the `SortedList` mentioned above, and returns all state abbreviations found.
- GetAllStations – returns all valid stations for a given state abbreviation. Retrieves the SortedList of stations contained in the `SortedList` of states, and returns the station ID and station display name for each station found.
- GetWeatherStatus – returns a formatted text string for a given state and station. This method looks up the URL for the station in question, parses the XML weather feed, and the formats a text string based on that information.

Native Types

No native types for this application

CLOSING REMARKS

Two additions to the Weather application stand out as needed.

One would be to create paging for states with more than 100 stations (right now, only the first 100 stations are listed for any state). The other would be to allow the administrator to configure the application with a default state and station ID. The application would then need to add an additional menu option in the MainMenu event handler, "My Weather", which would go directly to the DisplayWeather URL based on the configured state and station ID.

