Disaster Management Program
Open Platform for Emergency Networks

Instructions for Using the Common Alerting Protocol (CAP) V1.1 Interface on the Open Platform for Emergency Networks (OPEN)

Draft as of November 6, 2008

Version 0.5

Revision History

Date	Version	Description	Author
April 17, 2006	0.1	Initial draft	Gary Ham
April 20, 2006	0.2	Technical updates	Cathy (Yang) Liu
August 25,2006	0.3	Changes based on new capabilities	Gary Ham
July 20, 2007	0.4	POC Updates	Gary Ham
Nov 6, 2008	0.5	POC updates and updates related to new FEMA Program Management	Gary Ham

Table of Contents

1. Introduction

The Disaster Management Open Platform for Emergency Networks (DM-OPEN) provides interoperability interfaces for sharing of alerts, situation reports, common operational picture snapshots, and other emergency related information. These interfaces provide data structures and rules of operations designed to enable information sharing between diverse systems, both commercial and government. In general, these interfaces conform to open messaging standards as defined through the Emergency Management Technical Committee sponsored by the OASIS standards organization and through the National Information Exchange Model (NIEM).

1.1. Purpose

This document is designed to help programmers and system designers to understand and use OPEN web service interfaces. This is primarily a technical "how to" document which will be updated whenever new releases are made within scope.

1.2. Scope

This document covers the interface that supports cross system alerting using the Common Alerting Protocol (CAP) Version 1.1. Other interfaces are, or will be, covered separately.

1.3. Relationship to Other Documents

This is one of a set of documents that will describe connections to different web services interfaces. There will be a separate document written to cover each OPEN web service that is separately defined using different Web Service Definition Language (WSDL).

1.4. References

The following documents were used to obtain source information or are referenced in this document:

- *OASIS Common Alerting Protocol, v1.1,* OASIS Standard CAPV-1.1, October 2005

2. Before You Begin

[Note: If you have not connected to OPEN in the past, please email the DM-OPEN Interoperability Coordinator at OPEN@eyestreet.com for an FAQ that explains the basics of OPEN. The FAQ will explain how to become approved for OPEN connection and where to find information on other OPEN offerings as well as this one.]

Location of WSDL for generating needed client side classes:

https://interop.cmiservices.org/axis/services/CAP1_1?wsdl

These instructions are Java specific information. If you are a .NET programmer, or use another language capable of consuming WSDL, please use the above WSDL as appropriate, then follow your normal methods for building a web service client. Read what follows below to understand the methods that should be available to you and parameters that you will need to employ.

The following instructions also assume that you understand the structure and usage of a CAP 1.1 message. For further reference please see the OASIS standard available from:

http://www.oasis-open.org/committees/download.php/14759/emergency-CAPv1.1.pdf

3. Generating Service Classes and Beans from the WSDL

Using the Axis libraries, run the WSDL2Java tool on the WSDL to generate the needed client side files. The following example call uses some additional parameters to create a package structure consistent with the OPEN server side. These parameters (NStoPkg) should not be necessary, but may be helpful.

```
java org.apache.axis.wsdl.WSDL2Java
    --NStoPkg http://dmi-services.org=org.cmis.interopserver.services.cap1_1
    --NStoPkg http://dmi-services.org/beans=org.cmis.interopserver.beans
    --NStoPkg urn:oasis:names:tc:emergency:cap:1.1=org.cmis.interopserver.beans.cap1_1
    -v https://interop.cmiservices.org/axis/services/CAP1_1?wsdl
```

Assuming your classpath was correctly set, running the above WSD2Java call will generate the following package and class structure:

```
org.cmis.interopserver.beans.SimpleCOG.java
org.cmis.interopserver.beans.cap1_1.Alert.java
org.cmis.interopserver.beans.cap1_1.Area.java
org.cmis.interopserver.beans.cap1_1.Category.java
org.cmis.interopserver.beans.cap1_1.Certainty.java
org.cmis.interopserver.beans.cap1_1.EventCode.java
org.cmis.interopserver.beans.cap1_1.Geocode.java

org.cmis.interopserver.beans.cap1_1.Info.java
org.cmis.interopserver.beans.cap1_1.MsgType.java
org.cmis.interopserver.beans.cap1_1.Parameter.java
```

org.cmis.interopserver.beans.cap1_1.Resource.java
org.cmis.interopserver.beans.cap1_1.ResponseType.java
org.cmis.interopserver.beans.cap1_1.Scope.java
org.cmis.interopserver.beans.cap1_1.Severity.java
org.cmis.interopserver.beans.cap1_1.Status.java
org.cmis.interopserver.beans.cap1_1.Urgency.java
org.cmis.interopserver.services.cap1_1.CAP1_1.java
org.cmis.interopserver.services.cap1_1.CAP1_1Service.java
org.cmis.interopserver.services.cap1_1.CAP1_1ServiceLocator.java
org.cmis.interopserver.services.cap1_1.CAP1_1SoapBindingStub.java

With these classes you can retrieve and post CAP 1.1 messages by executing following instructions:

4. Connecting to the Web Service

Run the following class (call it from a main function and pass in your user id, password and COG id) to prove that you can establish a connection. A return string of "pong" proves that you are using appropriate credentials and that the service is alive. It is always good to do a ping to check your connection before attempting other functionality. It isolates any connection errors from other possible problems.

```
import java.net.*;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import org.cmis.interopserver.services.cap1_1.CAP1_1Service;
import org.cmis.interopserver.services.cap1_1.CAP1_1ServiceLocator;
import org.cmis.interopserver.services.cap1_1.CAP1_1SoapBindingStub;

public class SimplePingExample {

public SimplePingExample (String userId, String pwd, String cogId ){
    URL svcURL;
    CAP1_1Service service = new CAP1_1ServiceLocator();
    CAP1_1SoapBindingStub cap;

    try {
       svcURL = new URL("https://interop.cmiservices.org/axis/services/CAP1_1");
       System.out.println("Invoking service at " + svcURL.toString());
       cap = (CAP1_1SoapBindingStub) service.getCAP1_1(svcURL);
       cap.setUsername(cogId+"/"+userId);
       cap.setPassword(pwd);
       try {
          System.out.println(cap.ping());
       } catch (RemoteException e1) {
          e1.printStackTrace();
       }
      } catch (MalformedURLException e) {
       e.printStackTrace();
      } catch (ServiceException e) {
       e.printStackTrace();
    }
 }
```

```
}
```

5. Brief Explanations of Service Methods

Including the ping() method, you now have access to fifteen proven CAP 1.1 service methods that allow access to the DM-OPEN interoperability server:

a. To ping the service and ensure that it is up and running:

```
String ping() throws java.rmi.RemoteException;
```

The ping method returns the string "pong" if you are able to connect to the web service. It is used simply to verify that the service is up and running and that the has been successfully authenticated.

b. To get your COG in SimpleCOG format (id as long, name as String):

```
SimpleCOG getMyCog() throws java.rmi.RemoteException;
```

The getMyCog method returns an object of type SimpleCOG which contains the connection cogId as a long and the name of the COG as a String. (A SimpleCOG object is an instance of the SimpleCOG class generated from WSDL2Java). This is not a particularly useful function on operational basis, but its simplicity can be used to determine whether database connectivity is alive before attempting to diagnose more complicated system connection issues.

c. To get all of the COGs that you are allowed to post to in an array of type SimpleCOG:

```
SimpleCOG[] getCogs() throws java.rmi.RemoteException;
```

From the returned array, you may choose the target for your Alert posts. The getCogs method returns an array of SimpleCOG objects where each object includes a cogid as a long and the name of the COG as a String. Each SimpleCOG instance represents an organization that can retrieve an OPEN posted alert. For testing and initial development, OPEN interoperability partners are given passwords and ids in COG 1737, the OPEN Interoperability COG. For members of COG 1737, this method will return only two SimpleCOGs that have been set up expressly for test purposes (COG 1000 – the DMI Services Project Team and COG 2 – OPEN Interoperability COG 2). When testing is sufficient and a vendor or project has been sponsored by a vetted emergency organization, fully operational COGs can be set up for each sponsor. At that point, this method will retrieve an array of more than 2000 available COGs sponsored by organizations throughout the United States (and a couple of location in Canada). As a developer, you could make this a dynamically built pick list for your users. Since a list of this size may not be practical for your users, you can run this query and manually pick from those of interest to your users to build a static pick list for your users. (Note: See coming attractions below. There are plans to remedy this shortcoming.)

d. To post an Alert to one or more OPEN COGs:

```
void postCAPAlert(Alert alertObj, SimpleCOG[] simpleCOGs)
      throws java.rmi.RemoteException;
```

The postCapAlert method lets you post an alert to one or more different organizations where each organization is represented by its COG. The first parameter is an instance of the Alert object generated from the WSDL. This instance must be populated according to CAP 1.1 schema. The second parameter is an array of SimpleCOG (see 3.c above) where each SimpleCOG represents an organization that you are allowing to retrieve the alert. Do NOT post to your own COG (do NOT include your own COG in the second parameter). It will raise errors. (Note: this differs from the EDXL DE implementation where posting to your own cog is required for retrieval of your own posted messages). An error will also be raised if the combination of <sender> and <identifier> coincides with an already posted Alert. Under normal circumstances, this method does not return anything; if no error is returned then the posting has succeeded.

e. To post an Alert globally across the entire OPEN network:

```
void postCAPAlert(Alert alertObj) throws
      java.rmi.RemoteException;
```

The postCAPAlert method is used to post an Alert globally across the system. Once posted, the Alert will be accessible by users of all COGs (including the sender's) via the "getCAPAlertGlobal" methods. This method does not return anything; if no error is returned then the posting has succeeded. Posting with this method will allow retrieval by any COG in the DM environment.

f. To retrieve a particular Alert with known identifier and sender from the OPEN interface:

```
Alert getCAPAlert(String senderIdentifier) throws
      java.rmi.RemoteException;
```

The getCAPAlert method returns a specific CAP1.1 Alert object based upon the combined value of the <sender> and <identifier> tags. The String argument is colon delimited string consisting of the identifier and the sender (e.g., `"hamg@battelle.org" + ":" + "DM001567"` or "`sender:identifier`)". (Note: This particular method fails when a colon (":") is an integral part of the identifier or sender. It is likely that the delimiter will be changed to a character that cannot be used as part of the identifier or sender fields in the next release). You can only retrieve an alert using this method if the alert was posted by your COG, the alert was expressly posted to your COG by a user member of a different COG, or the Alert was posted globally across the system. If the "senderIdentifier" does not correspond to a CAP Alert in the DMIS system, an empty value (NULL), will be returned. (done-gh)

g. To retrieve all Alerts posted to or from your COG later than time specified in the <sent> field:

```
Alert[] getCAPAlerts(Calendar dateTime) throws
      java.rmi.RemoteException;
```

The getCAPAlerts method returns an array of CAP1.1 Alert objects where time specified in each Alert object's <sent> tag is later than the Java Calendar argument. This includes Alerts that have been posted directly to the caller's COG, Alerts sent by the caller, as well as those that have been posted globally. The Alerts are returned in chronological order, from newest to oldest. If the date is invalid, an error is returned. If there are no alerts since that date-time, an empty list is returned.

This method allows retrieving systems to choose not to retrieve older alerts, making the retrieval process somewhat less onerous. Users of the getCAPAlerts(Calendar) method should still allow some overlap in time from most recent retrieval when choosing the Calendar value for subsequent retrievals. This overlap is needed because relevant messages can be posted to OPEN at some time after the original message creation time and OPEN does not alter the originating <sent>tag value, regardless of the actual post time. Of course, this also means duplicate Alert retrievals should be checked for and discarded as appropriate.

h. To retrieve all Alerts posted to or from your COG later than the date and time specified by the Calendar value:

```
Alert[]getCAPAlertsByPostedDate(Calendar dateTime) throws
      java.rmi.RemoteException;
```

The getCAPAlertsByPostedDate method returns an array of CAP1.1 Alert objects posted to the DM OPEN service after the time specified in the Java Calendar argument. This includes Alerts that have been posted directly to the caller's COG, Alerts sent by the caller, as well as those that have been posted globally. The Alerts are returned in chronological order, from newest to oldest. If the date is invalid, an error is returned. If there are no alerts since that date-time, an empty list is returned.

This method is almost identical to `getCAPAlerts(Calendar)` (paragraph 5g above) except that the date-time value is being compared to the actual time that Alert is posted to the DM OPEN server instead of the sent field in the Alert.

i. To retrieve all Alerts posted to the global COG later than the date and time specified by the Calendar value:

```
Alert[]getCAPAlertsGlobalByPostedDate(Calendar dateTime)
      throws java.rmi.RemoteException;
```

The getCAPAlertsGlobalByPostedDate method returns an array of CAP1.1 Alert objects globally posted to the DM OPEN service after the time specified in the Java Calendar argument. This includes Alerts that have been sent by any COG for distribution to the global COG in the DM OPEN environment. The Alerts are returned in chronological order, from newest to oldest. If the date is invalid, an error is returned. If there are no alerts since that date-time, an empty list is returned.

This method can be used to get alerts that have been designated as global in scope separately from those posted individually to the retriever's COG.

j. To retrieve all Alerts posted to the global COG with a <sent> field value later than the date and time specified by the Calendar value:

```
Alert[]getCAPAlertsGlobalBySentDate(Calendar dateTime) throws
      java.rmi.RemoteException;
```

The getCAPAlertsGlobalBySentDate method returns an array of CAP1.1 Alert objects globally posted to the DM OPEN service where time specified in each Alert object's <sent> tag is later than the Java Calendar argument. This includes Alerts that have been posted by any COG for distribution to all COGs in the DM OPEN environment. The Alerts are returned in chronological order, from newest to oldest. If the date is invalid, an error is returned. If there are no alerts since that date-time, an empty list is returned.

This method works the same way as getCAPAlertsGlobalByPostedDate except that the comparison is made to the <sent> value in the CAP message instead of the actual time that the message is posted to DM OPEN.

k. To retrieve all Alerts posted to the caller on DM OPEN later than the date and time specified by the Calendar value:

```
Alert[]getMyCogsAlertsByPostedDate(Calendar dateTime) throws
      java.rmi.RemoteException;
```

The getMyCogsAlertsByPostedDate method returns an array of CAP1.1 Alert objects posted to the caller after the time specified in the Java Calendar argument. The time argument is being compared to the time each Alert is posted to the DM OPEN service**.** This method returns only those Alerts that have been expressly posted to the polling COG by another DM OPEN COG. The Alerts are returned in chronological order, from newest to oldest. If the date is invalid, an error is returned. If there are no alerts since that date-time, an empty list is returned. Global Alerts are not returned by this method. Neither are Alerts that were sent by the caller.

This method can be used for finer grained retrieval, since Alerts posted by the polling COG and global Alerts are not retrieved. It works exactly the same as the getCAPAlertsByPostedDate function except that only Alerts posted to the polling COG are returned.

l. To retrieve all Alerts posted to the caller with a <sent> field value later than the date and time specified by the Calendar value:

```
Alert[]getMyCogsAlertsBySentDate(Calendar dateTime) throws
      java.rmi.RemoteException;
```

The getMyCogsAlertsBySentDate method returns an array of CAP1.1 Alert objects posted to the caller where time specified in each Alert object's <sent> tag is later than the Java Calendar argument. This only includes Alerts posted to the polling COG by another DM OPEN COG. The Alerts are returned in chronological order, from newest to oldest. If the date is invalid, an error is returned. If there are no alerts since that date-time, an empty list is returned. Neither global Alerts nor Alerts sent by the polling COG are returned.

This method can be used for finer grained retrieval, since Alerts sent by the polling COG and global Alerts are not retrieved. It works exactly the same as the getCAPAlerts function except that only Alerts specifically identified for retrieval by the polling COG are returned.

m. To retrieve all Alerts sent by the caller to DM OPEN later than the date and time specified by the Calendar value:

```
Alert[]getAlertsSentByPostedDate(Calendar dateTime) throws
      java.rmi.RemoteException;
```

The getAlertsSentByPostedDate method returns an array of CAP1.1 Alert objects sent by the caller after the time specified in the Java Calendar argument. The time argument is being compared to the time each Alert is posted to the DM OPEN service**.** This includes only those Alerts that have been expressly posted by the polling COG to other COGs and/or the global COG. The Alerts are returned in chronological order, from newest to oldest. If the date is invalid, an error is returned. If there are no alerts since that date-time, an empty list is returned. Alerts that were posted to the polling COG by other COGS are not retrieved by this method.

This method allows a COG to retrieve messages it sent to others. It works exactly the same as the getCAPAlertsByPostedDate function except that only Alerts sent by the polling COG are returned.

n. To retrieve all Alerts sent by the caller with a <sent> field value later than the date and time specified by the Calendar value:

```
Alert[]getAlertsSentBySentDate(Calendar dateTime) throws
      java.rmi.RemoteException;
```

The getAlertsSentBySentDate method returns an array of CAP1.1 Alert objects sent by the caller where time specified in each Alert object's <sent> tag is later than the Java Calendar argument. This includes only those Alerts that have been sent by the polling COG to other COGs and/or the global COG. The Alerts are returned in chronological order, from newest to oldest. If the date is invalid, an error is returned. If there are no alerts since that date-time, an empty list is returned. Alerts that were posted to the polling COG from other COGs are not retrieved by this method.

This method allows a COG to retrieve those messages that it has sent to others. It works exactly the same as the getCAPAlerts function except that only Alerts sent from the polling COG are returned.

o. To retrieve the current OPEN server time in Calendar format:

```
Calendar getServerTime()throws java.rmi.RemoteException
```

Retrieves the current time on the server; useful for time-synching to ensure that getCAPAlertsByPostedDate() is accurate. Note that current server time is also returned as a SOAP header with each request; this method is here only for convenience.

6. More Examples and Contact Info

Sample code is available. E-mail the OPEN Interoperability Coordinator at OPEN@eyestreet.com for some WSDL generated classes, appropriate Axis libraries, and some test drive code to get you started. The Interoperability Coordinator is also your first point-of-contact for any problems related to programming issues related to OPEN.

7. Coming Attractions

a. getCogs(geography) – There will soon be a version of getCogs using a circle structure (point and radius) or similar geographic reference that will limit the number of COGs retrieved by this method. This will make the creation of dynamic posting pick lists more effective. Date of release TBD (near future).

b. COG Profile – More complete COG information is needed if user systems wish any form of dynamic creation of COG posting pick lists. This requirement is still in the analysis phase.

8. Known Issues

a. drefUri – The drefUri tag is designed to contain an embedded base-64 encoded mime type structure. This capability is not implemented at this time. Users may substitute a URL where the appropriate content can be referenced.

b. Circular post bug – A posted alert with identical <sender> and <identifier> is rejected as duplicate and an exception. In most case, this is appropriate. However, if the Alert is already in the system for COG A where COG A represents a rule-based distribution activity, COG A is prevented from using those rule to post the same alert back to COG B because the alert is already in the DM OPEN system. A temporary work around would be for COG B to slightly alter the identifier and repost. The proper solution will be to add COG B to the list of COGs posted to without adding a new alert to the system. This would allow COG B to retrieve the alert in an unaltered state, without duplicating the data in the database.

9. Suggestions, Recommendations and Assistance

a. Please send your suggestions, recommendations for improvement, etc. by e-mail to help@cmiservices.org. Be sure to specify that you are referencing the OPEN Interop CAP1.1 interface. Be specific. Remember that resources are limited and that the DM Program emphasizes the use of open standards for data and uses a directed distribution (user chooses recipients) sharing paradigm.

b. Initial programming and connection assistance can be provided through our OPEN Systems Interoperability Coordinator, Yohannes Tilahun (tilahuny@battelle.org).