NC4 Risk Center
Appendix H – Enterprise Incident Feed
User Guide

Release 7.6 | 10/15/2015

NC4 Disclaimer

NC4 Inc.
100 N. Sepulveda Blvd., Suite 200
El Segundo, CA 90245

Table of Contents

Section 1 – Introduction

The NC4 Enterprise Incident Feed is an optional feature that leverages the Common Alerting Protocol (CAP) v 1.0, a standard proposed by OASIS Emergency Management TC. The CAP spec can be found at http://www.oasis-open.org/committees/download.php/6334/oasis-200402-cap-core-1.0.pdf.

This document is intended for those within the customer organization, who are responsible for setting-up and configuring the NC4 CAP interface within the NC4 Risk Center solution.

The CAP protocol defines a message structure and some metadata level information to exchange alerts between different emergency management systems. However, it doesn't define the delivery mechanisms and leaves this field open for varying implementation interpretation. The National Weather Service http://weather.gov/alerts/ provides its alert data in CAP format as a URL from which the current weather information can be downloaded in CAP format. For example, http://weather.gov/alerts/us.cap provides weather information for the entire continental United States in CAP format. The delivery model is pull-oriented, where each weather incident is enclosed as an info structure within the overall alert structure as defined in the CAP protocol. CAP is a loosely-defined protocol and leaves a lot of semantics of the fields up to the individual user implementations. Other implementations of CAP include DMIS which is available at https://interop.cmiservices.org/. Once again, while the basic data structure being exchanged with DMIS is CAP, the transport delivery mechanism as well as the semantics of some (if not all) fields are up to the individual users to define. EDXML, an upcoming standard, is hopefully going to attempt the issues of standardizing on the data delivery mechanisms and hopefully on tightening the semantics of the underlying data structures. From the information available at this time, it looks like initially, the data structure used to exchange incident level information is still going to be CAP-based.

Given the above scenario, NC4 has written the CAP interface so that we can move quickly to different delivery mechanisms while keeping other pieces of the infrastructure intact. Currently, NC4 has implemented a simple HTTP post-based mechanism, and SOAP-based and email-based delivery mechanisms for delivering NIMC incidents. As the need arises to integrate with other systems, other delivery mechanisms will have to be written to xmit CAP messages.

NC4 has mapped NC4 International Monitoring Center (NIMC) information to the CAP data structure as closely as possible. As time goes by and we recognize other unwritten standards, we may change these mappings, and they will be included as part of our implementations.

NC4 Support Center

The Support Center is available to answer any questions regarding the NC4 Risk Center solution.

The Support Center is available 24 hours a day, 7 days a week.

Phone: 888-624-4411

Email: support@NC4.com

Support Website: https://support.nc4.us

Section 2 – CAP Field Mapping

NC4 International Monitoring Center Real-Time Incidents

The CAP structure, as defined by OASIS, supports multiple info structures enclosed within the alert structures. The idea is to have multiple incidents within the same CAP structures. However, we have found that using only one info structure is a more robust implementation. The following table shows the field mappings utilized by NC4 for its real-time incidents CAP feed:

Field Label	Purpose
alert.identifier	IncidentID.ActivityID (IncidentID is a 32 char string while activityid is a timestamp converted to a long number).
alert.sender	cap@nc4.com or userid of the destination system where applicable.
alert.password	Password for the destination system where applicable.
alert.sent	Time at the time of sending
alert.status	Actual (hardcoded)
alert.scope	Restricted (hardcoded)
alert.restriction	Fully qualified NC4 user name in the format – Organization/SubOrg1/…/SuborgN/UserName Where Organization/SubOrg1/../SubOrgN is the organization hierarchy of the user. For example a user JohnDoe of NC4 Risk Center application, belonging to Accounts department of AcmeCorp will have a fully qualified name of AcmeCorp/Accounts/JohnDoe.
alert.msgType	Alert (hardcoded)
alert.incidents	IncidentID
alert.source	NIMC (hardcoded)

Field Label	Purpose
alert.info.category	Since CAP and NIMC Incidents don't have one to one mapping the following NIMC to CAP mappings have been implemented. • Advisory -> Safety • Aviation -> Transport • Fire -> Fire • Geophysical -> Geo • Hazmat -> Env • Health -> Health • Infrastructure -> Infra • Law Enforcement -> Security • Meteorological -> Met • Medical -> Health • Natural Disaster -> Safety • Other -> Other • Structural -> Infra • Terrorism -> Security • Transportation -> Transport • Utility -> Infra
alert.info.event	Incident Type
alert.info.urgency	Following mappings are encoded at this time.. If NIMC status is Open then this field is set to "Immediate" If NIMC status is Closed then this field is set to "Past"
alert.info.severity	Severity
alert.info.senderName	InfoSource
alert.info.certainty	Likely
alert.info.headline	Gist
alert.info.description	Body
Alert.info.parameter	One or more attributes associated with the incident. For example, a Fire incident could have an additional attribute called "% contained" in which case the parameter tag with contain a value "% contained=70" indicating that 70% of fire has been contained.
alert.info.area.areaDesc	Location
alert.info.resource.resourceDesc	Title of attachment/link
alert.info.resource.uri	URL of the link or the filename of the attachment
alert.info.resource.mimeType	Mime type of attachment.
alert.info.resource.size	Attachment size
alert.info.resource.derefURI	Actual attachment content UU encoded.
Alert.info.area.circle	Latitude,longitude 0 – basically a circle with 0 radius.

Field Label	Purpose
alert.info.area.geocode	This field has multiple instances, with each instance of the format Key=Value. Currently we support the following keys.. • street • crossstreet • city • state • county • district • country • region • postal • latitude • longitude

NC4 International Monitoring Center Analysis and Intelligence

The analysis and intelligence (non-real-time intelligence) data developed by the NIMC is also available in the CAP feed. Examples of non-real-time analysis and intelligence data might include food-supply shortages in Haiti, impact of drug cartel in Mexico, country reports for Egypt, etc. The following analysis and intelligence is provided as part of the NC4 CAP feed:

- Special Event Briefings (SEB)
- Situation Reports (SR)
- Global Flashpoints (GFP)
- Analytical Briefs (AB)
- Country Reports (CR)
- Country Risk Rating changes (CRR)

Since this class of data is not specific to short-term and specific incidents, the semantics of this class of data is different from that of the real-time incidents. For example, analysis and intelligence data does not have a concept of an open or a closed incident. The event or situation being monitored might be long-term and developing and hence closure might not be as relevant to this class of intelligence as it is for real-time incidents. We have therefore intentionally left the concept of closure up to the recipients of this data.

Another important aspect of the analysis and intelligence data is that it is not temporally related to each other. In other words, a country report for Libya doesn't have as much temporal relation to a country report on Egypt. Making this assumption allows NC4 to deliver this class of data independent of each other. This simplifies our infrastructure and at the same time provides better fault tolerance for delivery of this data.

In addition, this integration tries to maintain as much consistency with the protocols and formats that we presently have for real-time incidents. The following table shows the field mappings utilized by NC4 for its analysis and intelligence CAP feed:

Field Label	Purpose
alert.identifier	IncidentID.ActivityID (IncidentID is a 32 char string while activityid is a timestamp converted to a long number).
alert.sender	cap@nc4.com or userid of the destination system where applicable.
alert.password	Password for the destination system where applicable.
alert.sent	Time at the time of sending
alert.status	Actual (hardcoded)
alert.scope	Restricted (hardcoded)
alert.restriction	Fully qualified NC4 user name in the format – Organization/SubOrg1/…/SuborgN/UserName Where Organization/SubOrg1/../SubOrgN is the organization hierarchy of the user. For example a user JohnDoe of NC4 Risk Center application, belonging to Accounts department of AcmeCorp will have a fully qualified name of AcmeCorp/Accounts/JohnDoe.
alert.msgType	Alert (hardcoded)
alert.incidents	IncidentID
alert.source	SEB, SR, GFP, AB, CR, CRR (respectively)
alert.info	Only one alert.info object is included per notification
alert.info.category	Other (hardcoded)
alert.info.eventCode	NC4.channelName=Other Where channelName could be either of SEB, SR, GFP, AB, CR or CRR
alert.info.event	SEB = Other SR = Other GFP = Other AB = Other CR = Country Report CRR = Country Report
alert.info.urgency	SEB = Immediate SR = Immediate GFP = Past if GFP has been archived, Immediate otherwise AB = Past if AB has been archived, Immediate otherwise CR = Immediate CRR = Immediate
alert.info.severity	Severity
alert.info.senderName	InfoSource
alert.info.certainty	Likely
alert.info.headline	Gist
alert.info.description	Body

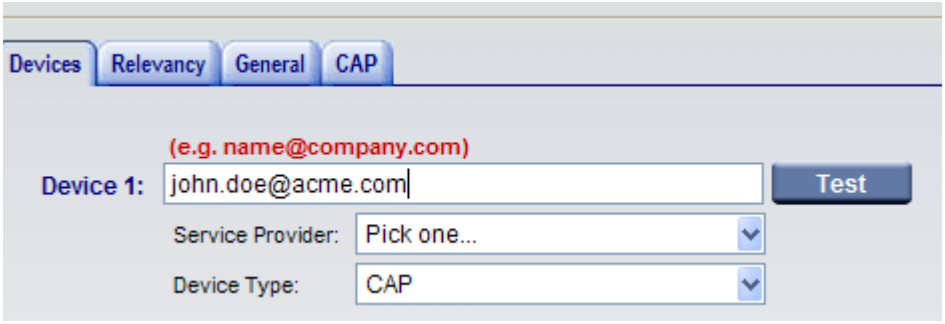
Field Label	Purpose
Alert.info.parameter	This field is where real-time data provides custom, incident specific attributes. For analysis and intelligence data, we store similar attributes. Of significance are the fields Actor and TSCategory. These fields will be provided in the following format. Actor=Al Qaeda Actor=Taliban TSCategory=Crime TSCategory=Terrorism Etc.. It is important to note that there could be multiple categories and multiple actors applicable to a given alert.
alert.info.resource.resourceDesc	Title of attachment/link
alert.info.resource.uri	URL of the link or the filename of the attachment
alert.info.resource.mimeType	Mime type of attachment.
alert.info.resource.size	Attachment size
alert.info.resource.derefURI	Actual attachment content UU encoded.
Alert.info.area	The major difference between real-time and analysis and intelligence data is presence of multiple area objects in a given alert. Real-time data only has one “area” object associated with it. NRT, on the other hand, is often times associated with more than one location. In most scenarios, the primary location of an NRT alert has more details then the others. However, for the purposes of future expandability it is best to assume that there could be multiple “area” objects each with its own geo-location information.
Alert.info.area.areaDesc	Location
Alert.info.area.circle	Latitude,longitude 0 – basically a circle with 0 radius.
alert.info.area.geocode	This field has multiple instances, with each instance of the format Key=Value. Currently we support the following keys. • street • crossstreet • city • state • county • district • country • region • postal • latitude • longitude

Section 3 – Configuring the CAP Interface in NC4 Risk Center

All users on the NC4 Risk Center system do not automatically get CAP access. A special feature needs to be assigned to either the user or the organization in order for the user to exploit the CAP integration.

Please contact NC4 support in order to get access to this feature.

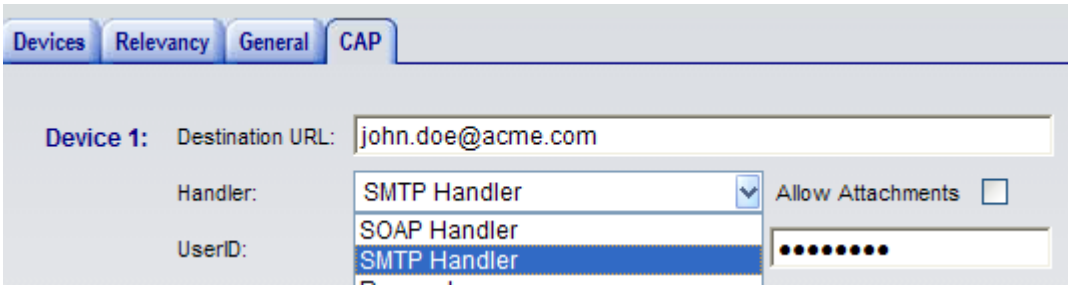
Once assigned, this feature enables the user to see additional screens while configuring the notification devices. Below is a sample screen which shows the additional controls.



Notice the following:

- The additional tab called CAP.
- In the drop down for ***Device Type***, the addition of CAP.

Once a user selects CAP from the ***Device Type*** drop down, and clicks on the CAP tab, the user will see the following:



NC4 supports the following three types of interfaces with CAP:

- Email-based
- SOAP-based.
- HTTP POST-based.

From the CAP tab, the drop down options correspond to the various delivery mechanisms. The field for ***Destination URL*** identifies the email address of the destination or the URL where the SOAP service is listening. An appropriate ***Handler*** is selected while ***Allow Attachments*** is selected depending on whether the destination system is interested in attachments associated with the alert or not.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ns1:alert xmlns:ns1="http://www.incident.com/cap/1.0">
  <identifier>61C672B28048A74338C79EE2A667468B.1238931285019</identifier>
  <sender>cap@nc4.com</sender>
  <sent>2009-04-05T07:40:54.000-04:00</sent>
  <status>Actual</status>
  <msgType>Alert</msgType>
  <source>NIMC</source>
  <scope>Restricted</scope>
  <restriction>NC4/John Doe</restriction>
  <incidents>61C672B28048A74338C79EE2A667468B</incidents>
  <info>
    <category>Geo</category>
    <event>Earthquake</event>
    <urgency>Immediate</urgency>
    <severity>Severe</severity>
    <certainty>Likely</certainty>
    <headline>USGS reports a magnitude 6.0 earthquake struck 3 miles ENE of Miyazaki, in the Hyuga Gulf.</headline>
    <description>The US Geological Survey reports that a magnitude 6.0 earthquake struck approximately 3 miles east-northeast of Miyazaki, in the Hyuga Gulf. There are no immediate reports of injuries or structural damages.</description>
    <parameter>magnitude=6.0</parameter>
    <AREA>
    <areaDesc>Hyuga Gulf</areaDesc>
    <circle>31.954,131.491 0</circle>
    <geocode>street=Hyuga Gulf</geocode>
    <geocode>crossstreet=</geocode>
    <geocode>city=3 Miles ENE of Miyazaki</geocode>
    <geocode>county=</geocode>
    <geocode>district=</geocode>
    <geocode>state=</geocode>
    <geocode>country=Japan</geocode>
    <geocode>region=Asia</geocode>
    <geocode>postal=</geocode>
    <geocode>latitude=31.954</geocode>
    <geocode>longitude=131.491</geocode>
    </AREA>
```

```
  </info>
</ns1:alert>
```

The fields are mapped according to the mapping table shown on page 6. The type of alerts are determined by the alert profiles that a user creates on the NC4 Risk Center system. The NC4 CAP interface just hooks into the NC4 alert profile mechanism.

The SMTP Interface

The NC4 CAP interface supports the SMTP interface for sending CAP alerts using regular email. Several NC4 customers use this interface to receive NC4 Risk Center alerts in XML format. This format is the easiest to integrate as it requires minimal tie-up between NC4 and the destination systems. As its SMTP based, it's pretty loosely coupled in the sense that all a destination system needs to have is an email account setup on their systems to receive the alert.

Another advantage of integrating using the SMTP transport is that this doesn't involve dealing with firewall issues which one would encounter if using the SOAP mechanism.

While sending CAP alerts to an SMTP CAP device, the NC4 CAP interface simply sends an email to the email address provided in the ***Destination URL*** field. In the email, a file called cap.xml is attached which has an alert of the format as shown above.

The SOAP Interface

The NC4 CAP Interface also supports sending CAP alerts using a SOAP interface. The following items need to be kept in mind regarding this interface:

- Like the SMTP interface, the SOAP interface also uses the push model – the NC4 Risk Center application sends an alert to the destination system whenever an incident create/update meets the criteria specified by the user in their alert profiles.
- NC4 has defined a WSDL for a web service definition. The NC4 Risk Center application acts as a web service client. When sending an alert to a customer destination system, the NC4 SOAP client connects to the SOAP service developed by the customer and makes a SOAP call to deliver the alert to the customer system.
- The web service developed by the customer has to implement the SOAP service as defined in the WSDL defined by NC4. NC4 does provide a Java reference implementation but customers are free to develop their own independent implementation as long as it conforms to the WSDL provided by NC4.
- The webservice has to be running on a webserver that is accessible from the Internet. If a customer has a firewall in place, then the port on which the webservice is running has to be open for receiving the SOAP calls made by the NC4 CAP SOAP client.
- Starting with version NC4 Risk Center 7.0.1, NC4 has added client side SSL capability. What this means is that if a server that is hosting the service requests a client side SSL, then NC4 will send a client side certificate.
- The WSDL of the webservice is as follows.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:s1="http://www.incident.com/cap/1.0"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="http://www.nc4.us/cap"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://www.nc4.us/cap"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.incident.com/cap/1.0">
      <s:element name="alert" type="s1:Alert" />
      <s:complexType name="Alert">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="identifier" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="sender" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" name="sent" type="s:dateTime" />
          <s:element minOccurs="1" maxOccurs="1" name="status" type="s1:AlertStatus" />
          <s:element minOccurs="1" maxOccurs="1" name="msgType" type="s1:AlertMessageType" />
          <s:element minOccurs="0" maxOccurs="1" name="password" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="source" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="scope" type="s1:AlertScope" />
          <s:element minOccurs="0" maxOccurs="1" name="restriction" type="s:string" />
```

```
      <s:element minOccurs="0" maxOccurs="1" name="addresses" type="s:string" />
      <s:element minOccurs="0" maxOccurs="unbounded" name="code" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="note" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="references" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="incidents" type="s:string" />
      <s:element minOccurs="0" maxOccurs="unbounded" name="info" type="s1:Info" />
    </s:sequence>
  </s:complexType>
  <s:simpleType name="AlertStatus">
    <s:restriction base="s:string">
      <s:enumeration value="Actual" />
      <s:enumeration value="Exercise" />
      <s:enumeration value="System" />
      <s:enumeration value="Test" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="AlertMessageType">
    <s:restriction base="s:string">
      <s:enumeration value="Alert" />
      <s:enumeration value="Update" />
      <s:enumeration value="Cancel" />
      <s:enumeration value="Ack" />
      <s:enumeration value="Error" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="AlertScope">
    <s:restriction base="s:string">
      <s:enumeration value="Public" />
      <s:enumeration value="Restricted" />
      <s:enumeration value="Private" />
    </s:restriction>
  </s:simpleType>
  <s:complexType name="Info">
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" default="en-US" name="language" type="s:language" />
      <s:element minOccurs="0" maxOccurs="unbounded" name="category" type="s1:InfoCategory" />
      <s:element minOccurs="0" maxOccurs="1" name="event" type="s:string" />
      <s:element minOccurs="1" maxOccurs="1" name="urgency" type="s1:InfoUrgency" />
      <s:element minOccurs="1" maxOccurs="1" name="severity" type="s1:InfoSeverity" />
      <s:element minOccurs="1" maxOccurs="1" name="certainty" type="s1:InfoCertainty" />
      <s:element minOccurs="0" maxOccurs="1" name="audience" type="s:string" />
      <s:element minOccurs="0" maxOccurs="unbounded" name="eventCode" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="effective" type="s:dateTime" />
      <s:element minOccurs="0" maxOccurs="1" name="onset" type="s:dateTime" />
      <s:element minOccurs="0" maxOccurs="1" name="expires" type="s:dateTime" />
      <s:element minOccurs="0" maxOccurs="1" name="senderName" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="headline" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="description" type="s:string" />
      <s:element minOccurs="0" maxOccurs="1" name="instruction" type="s:string" />
```

```
      <s:element minOccurs="0" maxOccurs="1" name="web" type="s:anyURI" />
      <s:element minOccurs="0" maxOccurs="1" name="contact" type="s:string" />
      <s:element minOccurs="0" maxOccurs="unbounded" name="parameter" type="s:string" />
      <s:element minOccurs="0" maxOccurs="unbounded" name="resource" type="s1:Resource" />
      <s:element minOccurs="0" maxOccurs="unbounded" name="area" type="s1:Area" />
    </s:sequence>
  </s:complexType>
  <s:simpleType name="InfoCategory">
    <s:restriction base="s:string">
      <s:enumeration value="Geo" />
      <s:enumeration value="Met" />
      <s:enumeration value="Safety" />
      <s:enumeration value="Security" />
      <s:enumeration value="Rescue" />
      <s:enumeration value="Fire" />
      <s:enumeration value="Health" />
      <s:enumeration value="Env" />
      <s:enumeration value="Transport" />
      <s:enumeration value="Infra" />
      <s:enumeration value="Other" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="InfoUrgency">
    <s:restriction base="s:string">
      <s:enumeration value="Past" />
      <s:enumeration value="Immediate" />
      <s:enumeration value="Expected" />
      <s:enumeration value="Future" />
      <s:enumeration value="Unknown" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="InfoSeverity">
    <s:restriction base="s:string">
      <s:enumeration value="Minor" />
      <s:enumeration value="Moderate" />
      <s:enumeration value="Severe" />
      <s:enumeration value="Extreme" />
      <s:enumeration value="Unknown" />
    </s:restriction>
  </s:simpleType>
  <s:simpleType name="InfoCertainty">
    <s:restriction base="s:string">
      <s:enumeration value="Unlikely" />
      <s:enumeration value="Possible" />
      <s:enumeration value="Likely" />
      <s:enumeration value="Very Likely" />
      <s:enumeration value="Unknown" />
    </s:restriction>
  </s:simpleType>
```

```
    <s:complexType name="Resource">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="resourceDesc" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="mimeType" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="size" type="s:integer" />
        <s:element minOccurs="0" maxOccurs="1" name="uri" type="s:anyURI" />
        <!-- s:element minOccurs="0" maxOccurs="1" name="derefUri" type="s:string" / -->
        <s:element minOccurs="0" maxOccurs="1" name="derefUri" type="s:base64Binary" />
        <s:element minOccurs="0" maxOccurs="1" name="digest" type="s:string" />
      </s:sequence>
    </s:complexType>
    <s:complexType name="Area">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="areaDesc" type="s:string" />
        <s:element minOccurs="0" maxOccurs="unbounded" name="polygon" type="s:string" />
        <s:element minOccurs="0" maxOccurs="unbounded" name="circle" type="s:string" />
        <s:element minOccurs="0" maxOccurs="unbounded" name="geocode" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="altitude" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="ceiling" type="s:string" />
      </s:sequence>
    </s:complexType>
  </s:schema>
</wsdl:types>
<wsdl:message name="CAPAlertRequest">
  <wsdl:part name="parameters" element="s1:alert" />
</wsdl:message>
<wsdl:message name="CAPAlertResponse">
  <wsdl:part name="parameters" element="s1:alert" />
</wsdl:message>
<wsdl:portType name="CAPAlertPortType">
  <wsdl:operation name="postCAPAlert">
    <wsdl:input message="tns:CAPAlertRequest" />
    <wsdl:output message="tns:CAPAlertResponse" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CAPAlertBinding" type="tns:CAPAlertPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  <wsdl:operation name="postCAPAlert">
    <!-- soap:operation soapAction="http://www.nc4.us/postCAPAlert" style="document" / -->
    <soap:operation soapAction="" style="document" />
    <wsdl:input>
      <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="CAP">
```

```
    <wsdl:port name="CAPAlertSOAPPort" binding="tns:CAPAlertBinding">
      <soap:address location="http://localhost:8080/services/CAPAlertSOAPPort" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

The HTTP Interface

The NC4 CAP interface also supports sending CAP alerts using a simple HTTP POST. NC4 will simply use HTTP POST to send the CAP XML. It will also send the user ID and password provided at the time of device configuration as a HTTP Basic Authentication header. No response is expected from the URL that is servicing the requests. All that is required is that the code returns an HTTP 200 OK response.

The SOAP Reference Implementation

NC4's reference implementation for the WSDL included above comes in the form of a Java implementation that runs on Apache Tomcat. It has been tested to run on Tomcat version 6.0. It uses the following components.

- JDK 1.6
- Tomcat 6.0
- Apache Axis version 1.4
- Apache Commons Logging
- Log4j version 1.2.8
- Apache Commons I/O
- Apache Commons CLI (command line interface)
- Java Activation Framework
- Apache Commons Discovery

NC4's reference implementation comes in the form of a Web Application Archive (WAR file). For the most part, dropping this file in the Tomcat webapps directory should be all that is required for getting the reference implementation up and running.

The default reference implementation does nothing but print the alert received by the webservice. Customers can override this default behavior by extending **us.nc4.cap.service.binding.CAPAlertBindingImpl** class. For example, a customer called Acme Corp can create a Java class as shown following:

```
package com.acme.cap;
```

```
import us.nc4.cap.util.AlertConversionHelper;

public class AcmeCAPImpl extends us.nc4.cap.service.CAPAlertBindingImpl {

    public void postCAPAlert(us.nc4.cap.beans.holders.AlertHolder parameters) throws
java.rmi.RemoteException {
            String tempstr = "";
            try {
             AlertConversionHelper ah = new AlertConversionHelper();
                  tempstr = ah.getString(parameters.value);
                  System.out.println("received a CAP alert");
            } catch (Exception e) {
                     e.printStackTrace();
            }
    }
}
```

Once this class is put in a place where it can be loaded by Tomcat, edit the file **nc4ws.properties** that can be found in the webapp that gets created when the WAR file provided by NC4 is dropped in the Tomcat webapps directory.

```
us.nc4.cap.service.CAPAlertBindingImpl=com.acme.cap.AcmeCAPImpl
```

Once Tomcat is recycled, this class will now be used to provide the CAP implementation when a SOAP client is used to contact the webservice. The reference implementation uses Apache Commons Discovery to resolve the class that should provide the implementation for the webservice.

NC4's reference implementation also comes with a simple test client that can be used to test the CAP service.

```
set classpath=.;c:dist\nc4ws.jar;lib\commons-cli-1.0.jar;lib\log4j-
1.2.8.jar;lib\axis.jar;lib\jaxrpc.jar;lib\commons-logging-1.0.4.jar;lib\commons-discovery-0.2.jar;lib\wsdl4j-
1.5.1.jar;lib\activation.jar;lib\mail.jar;lib\commons-io-1.3.1.jar

java -Dlog4j.configuration=my.properties us.nc4.cap.util.Test  --url
"http://localhost:8080/nc4ws/services/CAPAlertSOAPPort" --file file://c:/cvshome/nc4ws/cap.xml
```

The first line simply sets the **classpath** to indicate where the libraries used by the client can be found. The second line then invokes the test client to make the SOAP call. The –url parm provides the URL at which the webservice is running while the –file parm provides an XML in the CAP format. A sample CAP file has been provided earlier.