Open Source CAP Broadcaster

Development Guide

Prepared by:

Vincent Maggard

OpenBroadcaster

https://openbroadcaster.com

Table of Contents

Executive Summary 1
What is CAP? 1
  Pull Vs Push Methods 2
  Profiles 2
Project Breakdown 2
  Alert Processing & Validation 2
  Pre-Air Checklist 3
  Code Examples 3
  Expired / Active Alert Check 3
  CAP Pull Download 4
Appendices 5
  Prerequisite Technical Knowledge 5
  Sample Workplan 5
    Project Tasks 5
    Key Milestones 5

Executive Summary

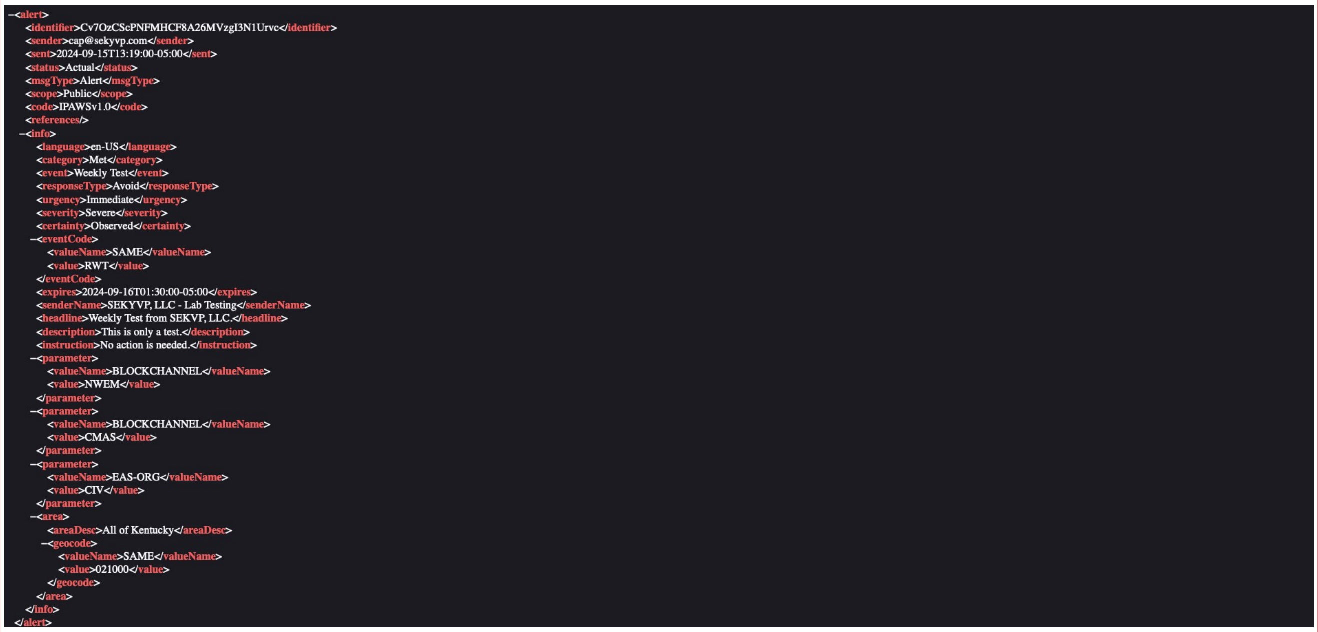
This white paper is aimed at software developers interested in integrating additional alerting system support into OBPlayer and potentially other broadcast automation systems using Common Alerting Protocol (CAP) covering the following elements:

- Understanding the Common Alerting Protocol
- Overview of open source alert systems using CAP
- CAP code examples for implementation

The primary goal of this project is to use bare metal computers including the low cost Raspberry PI and virtual environments to automate the process of receiving and broadcasting a CAP message with open source software. This unattended alerting process for radio and TV is standardized in Canada, the United States and many other countries presently using CAP.

What is CAP?

Common Alerting Protocol (CAP) is a worldwide open standard for alert creation and transmission via IP networks. CAP is used by many systems to ingest and distribute alerts for broadcast. Some of the common systems included IPAWS in the US managing Wireless Emergency Alert (WEA) on cell phones, sirens, and more.[I]



```
–<alert>
    <identifier>Cv7OzCScPNFMHCF8A26MVzgI3N1Urvc</identifier>
    <sender>cap@sekyvp.com</sender>
    <sent>2024-09-15T13:19:00-05:00</sent>
    <status>Actual</status>
    <msgType>Alert</msgType>
    <scope>Public</scope>
    <code>IPAWSv1.0</code>
    <references/>
  –<info>
      <language>en-US</language>
      <category>Met</category>
      <event>Weekly Test</event>
      <responseType>Avoid</responseType>
      <urgency>Immediate</urgency>
      <severity>Severe</severity>
      <certainty>Observed</certainty>
    –<eventCode>
        <valueName>SAME</valueName>
        <value>RWT</value>
      </eventCode>
      <expires>2024-09-16T01:30:00-05:00</expires>
      <senderName>SEKYVP, LLC - Lab Testing</senderName>
      <headline>Weekly Test from SEKVP, LLC.</headline>
      <description>This is only a test.</description>
      <instruction>No action is needed.</instruction>
    –<parameter>
        <valueName>BLOCKCHANNEL</valueName>
        <value>NWEM</value>
      </parameter>
    –<parameter>
        <valueName>BLOCKCHANNEL</valueName>
        <value>CMAS</value>
      </parameter>
    –<parameter>
        <valueName>EAS-ORG</valueName>
        <value>CIV</value>
      </parameter>
    –<area>
        <areaDesc>All of Kentucky</areaDesc>
      –<geocode>
          <valueName>SAME</valueName>
          <value>021000</value>
        </geocode>
      </area>
    </info>
  </alert>
```

Figure 1: Example of CAP formatted message:

I IPAWS CAP Technology Guide

Pull Vs Push Methods

Alerts can be pulled or pushed from the CAP alert server(s) to the clients. CAP alerts can be life critical, favouring to publish/subscribe (RSS and ATOM). Otherwise, a CAP alert might be missed

Pull uses HTTP(s) for alerts fetching. This fetching is normally done via an ATOM or RSS feed. Each alert is given a feed entry with a link to the full cap XML. The client polls the server every x number of seconds, with common poll rates of every 30-40 seconds. The client must keep track of alerts it's already used in the feed, so as to not keep repeating it over and over. When a new message is found the clients logs and downloads it. At that point it can treat it much the same as Push.

Push on the other hand sends the alert or pushes it to the client. This allows for near-real time activation. With push alerts they are still logged but since messages are normally sent once so the receiving system can be sure it's new. If an internet outage or system reboot happens when an alert is sent, the client would miss it. Some cap servers handle this with alert repeats via heartbeat messages, however if used it adds more processing and code to do so to the system.

Profiles

Profiles are used with CAP to add needed requirements for special systems like the US's EAS, or Canada's Alert Ready system. CA-CP handles unique French language accent characters [II]

Common profile for the United States is IPAWS. [III]

All alerts have the CAP v.1.2 elements.

There are profiles for other countries as well such as CAP-AU.

Project Breakdown

Alert Processing & Validation

Alerts received require parsing of the XML and alert validation (simple pre-air checks).

The order of handling an alert is as follows:

1. Parse the XML into an object.
2. Run through checks listed in the example table using the object.
3. If the alert passes all checks, download the attached audio, or create text to speech. For video create a crawl via concatenating the description & instructions elements values.
4. Place audio & crawl live on air.

II Canada Alert Ready Specifications
III IPAWS Profile

Pre-Air Checklist

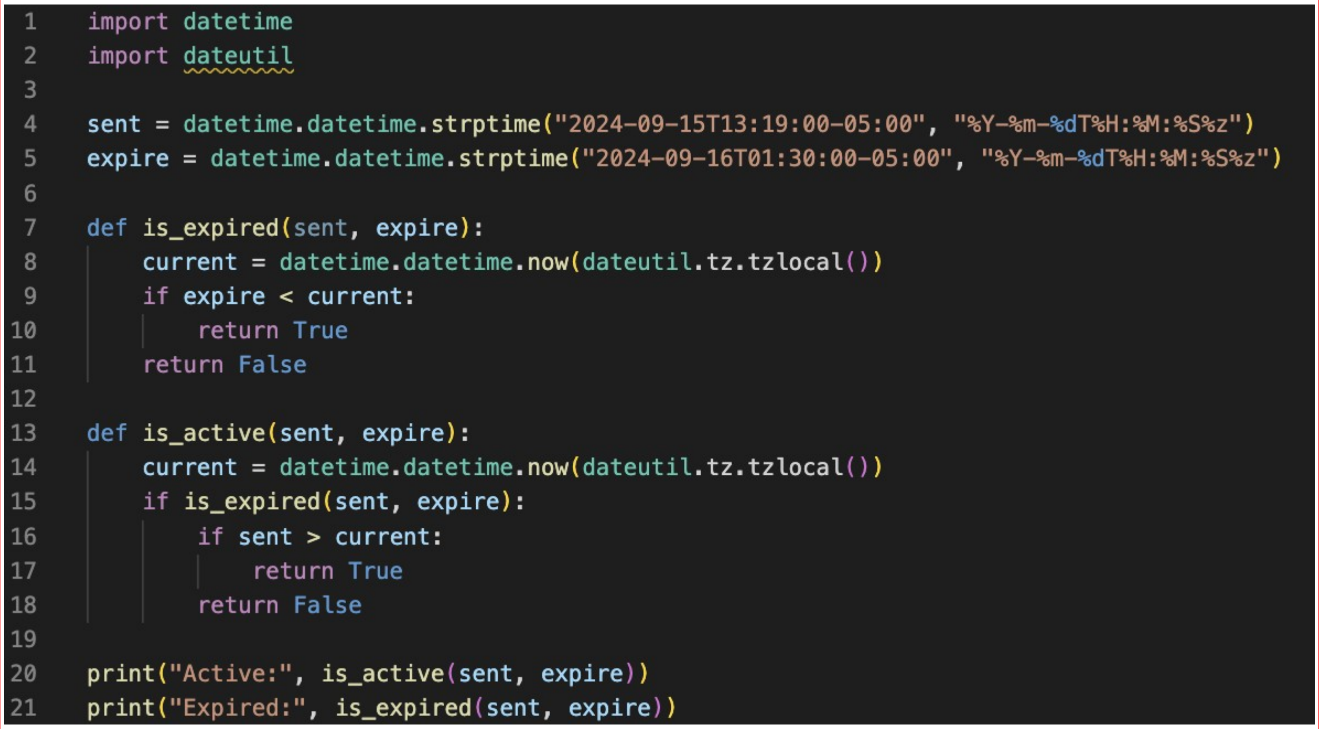
Check Name	Needed Actions
Is the alert active?	Current time > Issue time but < expire time.
Is the alert for the broadcasting area?	Check for one or more locations is included
Check if event code is set for broadcast?	Run though list of allowed event codes.
Check for the language code for the broadcaster viewer/listener?	Use the info block's language tag to find the right version of the message.

Code Examples

Below is group of code examples using Python 3.x This code is open source and can be found on GitHub [IV].

Expired / Active Alert Check

To check for a given alerts status expired, or not, is quite simple. To get the difference from the sent time and expire time examine if the difference is > 0 then the alert is active.



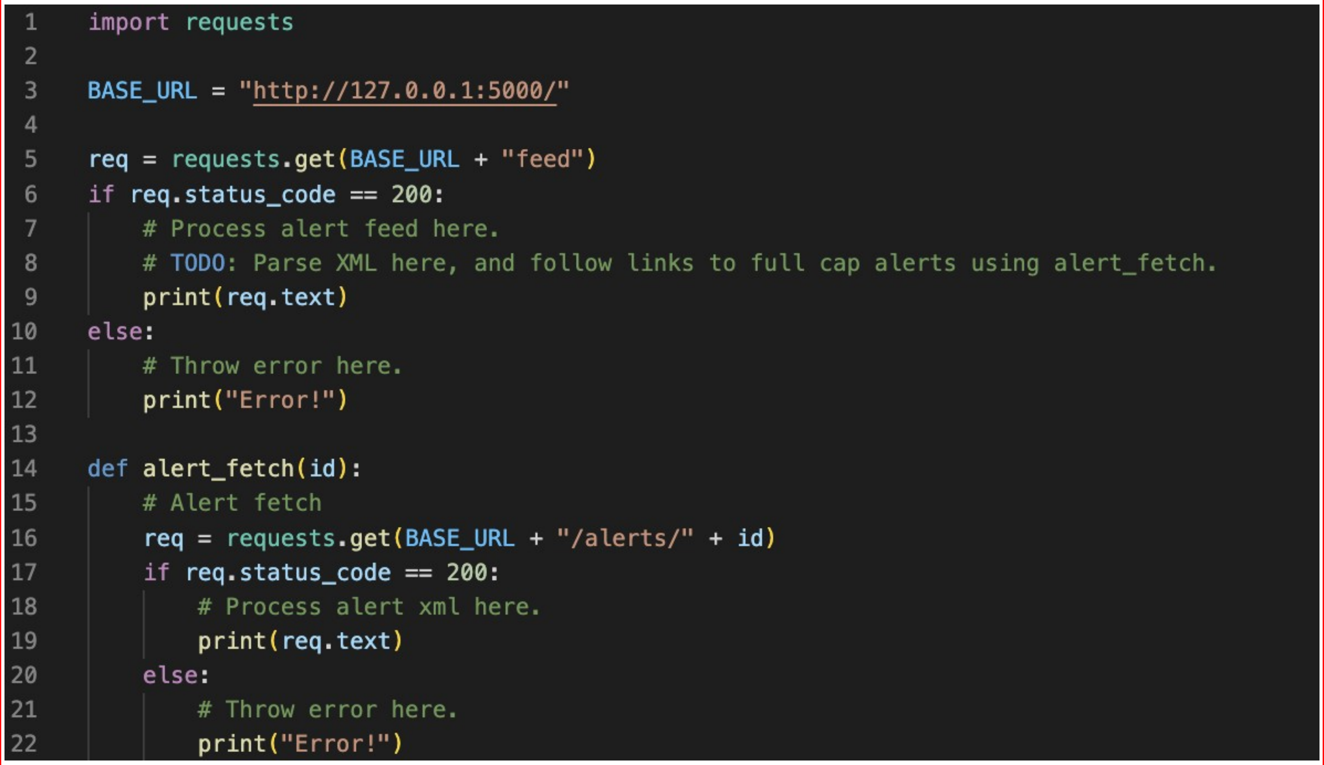
```python
import datetime
import dateutil

sent = datetime.datetime.strptime("2024-09-15T13:19:00-05:00", "%Y-%m-%dT%H:%M:%S%z")
expire = datetime.datetime.strptime("2024-09-16T01:30:00-05:00", "%Y-%m-%dT%H:%M:%S%z")

def is_expired(sent, expire):
    current = datetime.datetime.now(dateutil.tz.tzlocal())
    if expire < current:
        return True
    return False

def is_active(sent, expire):
    current = datetime.datetime.now(dateutil.tz.tzlocal())
    if is_expired(sent, expire):
        if sent > current:
            return True
        return False

print("Active:", is_active(sent, expire))
print("Expired:", is_expired(sent, expire))
```

Figure 2: Example using Python date objects handling it with two steps.

IV CAP Examples

CAP Pull Download

Example using an Open source CAP server.



```python
import requests

BASE_URL = "http://127.0.0.1:5000/"

req = requests.get(BASE_URL + "feed")
if req.status_code == 200:
    # Process alert feed here.
    # TODO: Parse XML here, and follow links to full cap alerts using alert_fetch.
    print(req.text)
else:
    # Throw error here.
    print("Error!")

def alert_fetch(id):
    # Alert fetch
    req = requests.get(BASE_URL + "/alerts/" + id)
    if req.status_code == 200:
        # Process alert xml here.
        print(req.text)
    else:
        # Throw error here.
        print("Error!")
```

Figure 3: Example downloading the xml file

Step by Step Guide

1. HTTP get request (download xml) CAP feed of alerts.
2. Parse xml
3. Compare with alerts already processed while handling multiple alerts.
4. Validate CAP alert and pass along to the broadcast chain.*

** OBPlayer presently logs the CAP alerts including off air audio logs as required by regulators.*

Appendices

Term	Definition
Crawl	Video overlay of scrolling alert text.
PR	GitHub **Pull Request** used for submitting code changes for review.
HTTP	Hyper Text Transport
IPAWS	Integrated Public Alert & Warning System

Prerequisite Technical Knowledge

Coding Skills:

- Python3 Basic knowledge
- Can parse and handle XML

Hands On Experience:

- Text to Speech
- Common A/V codecs
- Web stack
- Git & GitHub

Sample Workplan

Project Tasks

CAP Pull client for player.

Task	Predicted Timeline
Coding CAP pull client for OBPlayer.	50 hours
Testing alerting playback sequence	8 hours
Quality Assurance (QA)	4 hours
Issue Pull Request	**10 minutes**

Key Milestones

1. CAP Pull client for player.
2. Complete Alerting playback for the broadcast location. (~~US, CA,~~ IN, BD etc)
3. Issue Pull Request (PR) with changes into Git repo.
4. Wait for PR review.