ATOCA M. Lepinski
Internet-Draft BBN Technologies
Intended status: Informational K. Seo
Expires: January 16, 2013 BBN Technologes
R. Barnes
BBN Technologies
July 15, 2012

# Alert Metadata Protocol (AMP)
## draft-barnes-atoca-meta-03.txt

Abstract

This document specifies a set of mechanisms that devices on an IP network can use to discover an alert metadata server able to provide information about local emergency alert services. Additionally, this document provides a protocol that devices on an IP network can use to retrieve local information from an alert metadata server about sources of emergency alerts and register contact information for receipt of alerts.

Please send feedback to the atoca@ietf.org mailing list.

Status of this Memo



Copyright Notice

Table of Contents

## 1. Introduction

In order for clients to securely receive alerts, both endpoints and servers may need a certain amount of configuration. Clients need to know the identities of trusted alerting authorities so that they can reject false alerts. In some environments, servers need to gather location and contact information for end clients to support alert targeting and delivery, for example client location, language preferences, or device capabilities.

In this context, alert delivery proceeds in three phases. First, a client device connects to a network where alerts are provided and discovers a local alert metadata server. Second, the device discovers an alert metadata server, downloads information about local alert servers, and (optionally) registers some information about itself. Third, an alert server delivers an alert to the client. These roles are illustrated in Figure 1. This document addresses the first two phases (discovery and configuration), and provides one possible channel for alert delivery.

```
                           +-------------------+
                           |   AMP Discovery   |
       +-(1) AMP Srv. URI--| (DHCP, DNS, LoST) |
       |                   +-------------------+
       |
       |
       |
       |
       V
+--------+                 +-------------------+
|  AMP   |--(2) AMP Reg.-->|        AMP        |
| Client |<-(2) AMP Adv.---|       Server      |
+--------+                 +-------------------+
       ^
       |
       |
       |
       |
       |                   +-------------------+
       +-(3) Alert Msg.----|   Alert Server    |
                           +-------------------+
```

Figure 1: AMP alert configuration

This document addresses this problem in two parts. First, we describe the process by which a client discovers an AMP server for a local network or for a location of interest. Second, we define a simple protocol that the client can use to interact with the server to download metadata, register state, and receive alerts.

### 1.1. Open Questions

The current version of this draft specifies transport security (i.e., TLS) as the only mechanism for providing security for AMP messages. However, this document could also specify as an option the use the mechanisms defined by of the JOSE working group to provide object security for the JSON bodies on a per-message basis (independent of the underlying transport).

The current version of this draft specifies only a WebSocket transport for AMP messages. However, as an alternative this document could also specify an option to use HTTP as a transport for AMP messages.

## 2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The following entities are important in the AMP protocol:

Client: An end-user device interested in receiving alerts. The device may be connected to a local network in the area covered by alerts, or may be remote.

Alert Metadata Server: A server that maintains information about clients and information about how alerts are delivered within some scope (e.g., within a jurisdiction).

Alert Server: A server that delivers emergency alerts to clients.

Alerting Authority: An entity that is authorized to originate alerts in a given context (e.g., a jurisdiction)

In a given deployment, the Alert Metadata Server and the Alert Server may be the same server, but this is not necessarily the case.

## 3. Server Discovery

In this section we describe two mechanisms for clients to discover alert metadata servers. The first mechanism enables a client to rely upon its ISP or access network to provide a reference to an appropriate alert metadata server. Many alerting scenarios are local (e.g., natural disasters) and ISPs are often well-positioned to gather information on their local environment. Therefore, it can be

useful for an ISP to provide information about local alerting resources to clients. Likewise, clients should be able to discover information advertised by their local networks. This first mechanism, is based on the discovery procedure described in RFC 5986 [RFC5986]. It relies on a DHCP option specifying the Access Network Domain Name, and a U-NAPTR resolution that uses the Access Network Domain Name to obtain the name of the alert metadata server.

The second mechanism enables a client to discover an alert metadata server with information about alerts relevant to a particular location. This may be the client's own location, or some other location of interest. This mechanism may be used either in cases where the client's ISP does not provide explicit support for emergency alerting, or in cases where the client is interested in receiving alerts for some region that does not include the client's current location. This mechanism makes use of the LoST protocol [RFC5222], and its corresponding discovery mechanism [RFC5223].

Client implementations SHOULD support both discovery using the Access Network Domain Name (Section 3.1) and discovery based using LoST (Section 3.2). Additionally, client implementations SHOULD support out-of-band discovery by allowing a user to specify a static URI for an appropriate alert metadata server.

### 3.1. Discovery using Access Network Domain Name

The mechanism presented here is based on the discovery procedure described in RFC 5986 [RFC5986]. It relies on the DHCP option for Access Network Domain Name, which is specified in RFC 5986 for both DHCPv4 and DHCPv6. IP networks that support emergency alerting SHOULD provide the Access Network Domain Name option to devices on network that are configured via DHCP. This option provides to the device a domain name that is suitable for service discovery within the access network.. This domain is used as input to the U-NAPTR resolution process for alert server discovery.

In addition to providing the Access Network Domain Name to devices via DHCP, an IP network that supports emergency alerting SHOULD provision DNS records to support a U-NAPTR lookup for AMP Server discovery. U-NAPTR [RFC4848] is a Dynamic Delegation Discovery Service (DDDS) profile that produces a URI (in this case, the URI for the appropriate AMP alert server). Section 3.1.1 specifies the format of the DNS NAPTR record used for this discovery, and Section 3.1,2 provides processing instructions for the client device performing the discovery.

### 3.1.1. NAPTR Record Format

U-NAPTR resolution for an alert server takes a domain name as input and produces a URI that identifies the alert server. This process also requires an Application Service tag and an Application Protocol tag, which differentiate NAPTR records for alert server discovery from other records for that domain. Section 5.1 defines an Application Service tag of "AMP", which is used to identify the AMP alert server that is appropriate for use by devices in a given domain. The Application Protocol tags "ws", and "wss" are used to identify alert servers that support the WebSocket protocol and its secure variant. The NAPTR records in the following example demonstrate the use of the Application Service and Protocol tags. Iterative NAPTR resolution is used to delegate responsibility for the alert server from "zonea.example.net." and "zoneb.example.net." to "outsource.example.com."

```
zonea.example.net.
;;       order pref flags
IN NAPTR 100   10   ""  "AMP:wss" (            ; service
    ""                                          ; regex
    outsource.example.com.                      ; replacement
    )

zoneb.example.net.
;;       order pref flags
IN NAPTR 100   10   ""  "AMP:wss" (            ; service
    ""                                          ; regex
    outsource.example.com.                      ; replacement
    )

outsource.example.com.
;;       order pref flags
IN NAPTR 100   10   "u"  "AMP:wss" (            ; service
    "!.*!wss://alerts.example.org:80/!"          ; regex
    .                                            ; replacement
    )
```

Figure 1: Sample AMP NAPTR Records

U-NAPTR resolution might produce multiple results from each iteration of the algorithm. Order and preference values in the NAPTR record determine which value is chosen. A Device MAY attempt to use alternative choices if the first choice is not successful. An WSS URI for an alert server that is a product of U-NAPTR MUST be authenticated using the domain name method described in Section 3.1 of RFC 6455 [RFC6455]. The domain name that is used in this authentication is the one extracted from the URI, not the one that was input to the U-NAPTR resolution process.

### 3.1.2. Client Processing

In order to discover an appropriate alert server, a client device must first obtain a domain name for the local access network. The client device first attempts to obtain configuration information via DHCP. If the DHCP configuration contains the Access Network Domain Name option, then the client uses the domain name in this option as the domain name for the local access network. Once the client has the domain name of the local access network, it uses this domain name to make a U-NAPTR query [RFC4848] for the Application Service AMP in this domain.

If the DHCP configuration does not contain the Access Network Domain Name option, then the client MUST follow the process described in [I-D.ietf-geopriv-res-gw-lis-discovery] to search the reverse DNS tree for a U-NAPTR record based on the client's IP address.

## 3.2. Discovery using LoST

The mechanism presented here is based on the Location to Service Translation protocol (LoST) [RFC5222]. This protocol enables a client to query with an arbitrary location (either its own location or an alternative location of interest) and obtain the URI for an alert metadata server that is able to provide information for alerts relevant to the given location.

### 3.2.1. LoST Server Discovery

In order to utilize LoST to discovery an alert metadata server, the client must first obtain the address or URI of a LoST server. Implementations supporting LoST-based discovery of alert metadata servers MUST also support DHCP-based LoST discovery as specified in RFC 5223 [RFC5223]. Implementations MAY provide an interface whereby a user can directly configure a static LoST server URI or IP address, but MUST prefer a discovered LoST server to a configured one.

### 3.2.2. Client Processing

To discover an alert metadata server for a given geography, a client makes a LoST <findservice> request. The client populates the <service> element of this request with the URN "urn:service:alert-info", the URN specifying the alert metadata service. The client populates the <location> element of the request with a location for which the client is interested in receiving emergency alerts. (This may be the client's own location, or may be an alternate location of interest to the client.)

## 4. AMP Protocol

The Alert Metadata Protocol (AMP) consists of a set of messages encoded as JSON objects [RFC4627] exchanged over the WebSocket protocol [[WebSocket]]. In this section we describe the format of each AMP message type, and the overall flow of an AMP session.

### 4.1. Message Format

Each AMP message is a JSON object containing a "type" and other fields that depend on the message type. An AMP object MUST contain the following field:

"type": REQUIRED Token. The type of AMP message encoded in this object.

This document defines four values of the "type" field, corresponding to the four different alert types:

"advertisement": A message describing local alert servers and authorities

"registration": A message registering client data with the server

"refer": A message referring the client to another AMP server

"alert": An emergency alert

Future documents may define additional message types. Implementations MUST ignore any AMP message with an unknown type, or any unknown field in an AMP message.

#### 4.1.1. Advertisement

Advertisement messages are sent from servers to clients. These messages allow servers to notify clients about local alert authorities and local alert servers. This information enables the client to determine whether future alerts are valid, regardless of the protocol mechanism used to transport the alert. An advertisement message can contain the following fields:

"token": OPTIONAL String. This field is an opaque string that the server uses to identify the client on subsequent requests.

"contacts": REQUIRED Array of String. This field is an array of strings, where each string contains a URI from which local alerts may be sourced. This array MUST NOT have length zero.

"certs": OPTIONAL Array of String. This field is an array of strings, where each string contains an X.509 certificate for a local authority. Each certificate is encoded with DER and base64url encoded [[BASE64]]. These certificates are used to validate local alerts signed by the given alert authority.

"public_keys": OPTIONAL Array of String. This field is an array of strings, where each string contains Subject Public Key Information (SPKI) for a local authority, encoded in DER and base64url encoded. These are the public keys used to validate alerts signed by the given alert authority.

"hash_values": OPTIONAL Array of String. This field is an array of hash values that are used in ESCAPE verification, base64url encoded.

"ttl": REQUIRED Number. This field is a positive integer that indicates the length of time (in seconds) for which this advertisement is valid. If the client does not receive a new advertisement message from the server before the ttl indicates that the advertisement is stale, then the client should attempt to obtain a new advertisement message by sending a registration message to the server.

An advertisement message MUST contain either a "certs" field or a "public_keys" field.

The "token" field MUST be present except when the server does not maintain state for clients. If the server sets the "token" field, then the values it uses MUST be chosen to minimize the possibility that one client will be able to guess another's token, since that would allow one client to change or delete another client's registered state. One algorithm for generating these tokens would be to compute the HMAC of another client identifier (e.g., an IP address and timestamp) using a secret key known only to the AMP server.

### 4.1.2. Registration

Registration messages are sent from clients to servers. They are used by the clients to register with a server in order to receive future alerts of the proper type and format (e.g., language). The same message is also used to update existing registration information or to request deletion of existing registration information. Note that for location information, the Registration makes use of the PIDF-LO format, which is defined in [RFC4119]. Registration messages contain the following fields:

"token": REQUIRED String. An opaque a string that identifies the client. Once a client has received an advertisement message from a server, it SHOULD copy the token from that message into all future registration messages to that server, so that the server can distinguish between new registrations and updates to existing registrations.

"contacts": OPTIONAL Array of String. This field is an array of strings, where each string contains a URI that can be used contact the client. If this field is included, but the array is empty, then the the server MUST delete any existing registration information for this client.

"location": OPTIONAL String. This field is a string containing a "geopriv" element from a PIDF-LO, base64url encoded.

"language": OPTIONAL String. This field is a string containing the language in which the client wishes to receive alerts, in the format defined by RFC 5646 [RFC5646].

If a server receives a new registration message from a previously registered client (i.e., a registration message containing a token that the server has previously sent in an advertisement message), then the server should replace the existing registration information for that client with the information contained in the new registration message. If the server receives a registration message containing only the token field, then the server should delete any existing registration information associated with this client.

#### 4.1.3. Refer

Refer messages are sent from servers to clients. These messages allow servers to notify clients of a different AMP server that the client should contact. For example, if an AMP server receives a registration message indicating a location outside its jurisdiction, it might send a refer message that refers the client to an appropriate server for the client's current location. A refer message must contain the following fields:

"to": REQUIRED String. The URI of the AMP server to which the client is being referred.

Upon receiving a Refer message, a client SHOULD send establish a new AMP session with the AMP server indicated in the "to" field of the refer message.

#### 4.1.4. Alert

Alert messages are sent from servers to client. These messages are one mechanism for distributing local alerts. (Other mechanisms for transporting local alerts include LEAP [I-D.barnes-atoca-delivery].) Alerts sent using an AMP alert message are encoded using ESCAPE [I-D.barnes-atoca-escape], then base64url encoded. An Alert message contains the following fields:

"alert_data": REQUIRED String. An ESCAPE-encoded, base64url-encoded alert message.

The procedure for validating ESCAPE-encoded alert messages can be found in [I-D.barnes-atoca-escape]

### 4.2. AMP Sessions

An AMP session is a WebSocket connection over which AMP messages are conveyed. The first goal of an AMP session is to inform a client about local alerting resources (alerting configuration information), but the client may maintain a long-lived AMP session in order to provide updated status (e.g., location or contact changes) as well as to get updated configuration information over time.

The client initiates an AMP session by establishing a WebSocket connection to the AMP server. The client sends the first message, providing a Registration message with relevant information.

The AMP Server MUST respond with an Advertisement message containing local alert information immediately upon the establishment of a session. If the initial Registration contained a "token" value, then the "token" field in the Advertisement MUST be either empty or equal to the registered token value.

Once the initial handshake is complete, either side may send a message at any time. When a message is received, the action taken depends on the type of message:

- Client receives Refer: The client SHOULD close the current AMP session and initiate a new AMP session with the server indicated in the "to" field of the message. If the client received a token in the first session, then it SHOULD include that token in the initial Registration for the new session.

- Client receives Advertisement: The client MUST replace its local alert configuration information with the contents of the Advertisement. If the "token" field is present, then the client MUST update its token.

- o Client receives Alert: The client MUST decode the encoded "alert_data" element and process the resulting ESCAPE message according to the ESCAPE validation rules. If the alert is valid, the client renders it to the user.

- o Server receives Registration: The server MUST replace its current state for the client with the state in the message.

  * If the "token" field is present, the server MUST verify that it matches a token that it has assigned in an Advertisement message in this session; if not, then this message MUST be ignored.

  * If the Registration message contains only the "type" field, then the server MUST delete any state associated with this session.

  * If the location of the client has moved out of the server's coverage area, then the server MUST close the connection. If the responsible AMP server for the client's new location is known, then the server SHOULD send a Refer message before closing the connection.

If either side receives a message sent in the incorrect direction, it MUST ignore it. For the server, this includes Advertisement, Refer, and Alert messages. For the client, Registration messages.

Servers SHOULD maintain information about AMP servers covering neighboring jurisdictions and their respective coverage areas. That way, the server can issue a Refer message to the client as soon as the client reports that it has left the coverage area. This will help ensure that the client always has up-to-date alerting configuration information, without the client having to repeatedly perform AMP discovery.

## 5. IANA Considerations

This document requires several registrations by IANA into existing registries, and creates a new registry of AMP message codes.

[[ TODO: Register the URN: "urn:service:alert-info" ]]

[[ TODO: Register NAPTR service tag "AMP" and application protocols "http", "https" ]]

[[ TODO: Register media type application/amp+json ]]

### 5.1. AMP Message Type Registry

IANA is requested to create a new registry of AMP Message Types. This registry contains two fields, the name of the name of the registered message type, and a specification pointer containing a reference to the document that defines the registered message type.

IANA is requested to populate this new registry with the following four entries:

| Message Type Name | Specification Pointer |
|---|---|
| Registration | draft-barnes-atoca-meta |
| Advertisement | draft-barnes-atoca-meta |
| Refer | draft-barnes-atoca-meta |
| Alert | draft-barnes-atoca-meta |

## 6. Security Considerations

[Author's Note: The Security Considerations will be fleshed out in more detail in the next version of this document.]

The AMP protocol contains contact and location information for a device which for many devices will consist of private information regarding the user of the device. Therefore, confidentiality protection should be used when the registration request contains private information.

The modification of AMP messages can cause client devices to accept false alerts (in the case where the advertisement is modified) or to receive alerts for the improper location (if the registration is modified). Therefore, integrity protection should be applied to AMP messages.

The AMP protocol runs over HTTP. Therefore, the use of HTTP over TLS can provide confidentiality and integrity protection for AMP messages.

Alert server discovery makes use of NAPTR. Standard security considerations involving the use of NAPTR apply. DNSSEC SHOULD be used to protect the DNS responses provided during the discovery procedure.

## 7. Acknowledgements

The authors would like to thank Derrick Kong for help in creating the JSON schema for the AMP protocol.

## 8. References

### 8.1. Normative References

[I-D.barnes-atoca-escape]
Barnes, R., "Encoding of Secure Common Alert Protocol Entities (ESCAPE)", draft-barnes-atoca-escape-00 (work in progress), October 2011.

[I-D.ietf-geopriv-res-gw-lis-discovery]
Thomson, M. and R. Bellis, "Location Information Server (LIS) Discovery using IP address and Reverse DNS", draft-ietf-geopriv-res-gw-lis-discovery-03 (work in progress), March 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4119] Peterson, J., "A Presence-based GEOPRIV Location Object Format", RFC 4119, December 2005.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.

[RFC5222] Hardie, T., Newton, A., Schulzrinne, H., and H. Tschofenig, "LoST: A Location-to-Service Translation Protocol", RFC 5222, August 2008.

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

[RFC5986] Thomson, M. and J. Winterbottom, "Discovering the Local Location Information Server (LIS)", RFC 5986, September 2010.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

## 8.2. Informative References

[I-D.barnes-atoca-delivery]
Barnes, R., "Lightweight Emergency Alerting Protocol (LEAP)", draft-barnes-atoca-delivery-01 (work in progress), October 2011.

[RFC5223] Schulzrinne, H., Polk, J., and H. Tschofenig, "Discovering Location-to-Service Translation (LoST) Servers Using the Dynamic Host Configuration Protocol (DHCP)", RFC 5223, August 2008.

Authors' Addresses

Matt Lepinski
BBN Technologies
10 Moulton St
Cambridge, MA 02138
US

Phone: +1 617 873 5939
Email: mlepinski@bbn.com

Karen Seo
BBN Technologes
10 Moulton St
Cambridge, MA 02138
US

Phone: +1 617 873 3152
Email: kseo@bbn.com

Richard Barnes
BBN Technologies
9861 Broken Land Parkway
Columbia, MD 21046
US

Phone: +1 410 290 6169
Email: rbarnes@bbn.com