

Atoca WG
Internet Draft
Intended Status: Informational
Expires: April 30, 2012

Gabor Bajko
Nokia
October 31, 2011

Emergency Alert Service support in IEEE 802.11 networks
draft-bajko-atoca-wlan-eas-01

Abstract

The IEEE 802.11 specification is evolving by defining new amendments to it, which are then rolled into the base spec. The 802.11u amendment, published in November 2010, contains support for Citizen to Authority type of Emergency Calls and Authority to Citizen type of Alerts.

This document attempts to explain what level of support for Authority to Citizen type of Emergency Alerts has been defined for IEEE 802.11 protocol.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Content

1. Introduction	2
2. Conventions and Terminology	3
3. Emergency Alert Service indication at a wifi AP	3
4. Retrieving an EAS message	5
5.	

1. Introduction

This document assumes the reader is familiar with the basics of the 802.11 protocol.

A STA first listens in a channel to see if there are any APs operating in that channel. If there are, then the STA will receive a beacon. This is the so-called passive scanning procedure. The STA may also choose to send a Probe Request in the channel and wait for a Probe Response. The Probe Request is sent to a broadcast address and contains conditions which an answering AP must satisfy. This is called active scanning.

A STA can only exchange data frames with an AP after the so-called association and authentication (if required) procedure. Management frames however, can be exchanged even before the association and authentication procedure, which the 802.11 specification calls pre-association frame exchange.

2. Conventions and Terminology

2.1 Conventions used in This Document

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 [1].

2.2 Terminology

STA station, a device communicating with an 802.11 access point
GAS Generic Advertisement Service Protocol, defined in 802.11

3. Emergency Alert Service indication at a wifi AP

An AP compliant with the IEEE 802.11-2011 specification will have a MIB variable called dot11EASActivated which has the following definition:

dot11EASActivated OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This is a control variable.

It is written by an external management entity or the SME. Changes take effect as soon as practical in the implementation. This attribute when true, indicates the STA is capable of supporting emergency alert system. The capability is disabled otherwise. "

DEFVAL {false}

::= { dot11StationConfigEntry 128 }

When this MIB variable is set to TRUE, then the AP supports this feature. Else the AP does not support this feature.

When this MIB variable is set to true and there are emergency alert messages in the network, then the beacon will contain one Emergency Alert Information Element for each active alert message, in the beacon. These Emergency Alert Information Elements have to also be included in the Probe Response Frames, when a STA asks for it in a Probe Request Frame.

The Emergency Alert Identifier element provides a hash to identify instances of the active EAS messages that are currently available from the network. The hash allows the STA to assess whether an EAS message advertised by an AP has been previously received and therefore whether it is necessary to download from the network. The format of the Emergency Alert Identifier element is:

0	7	13	79
+-----+-----+-----+-----+			
Element ID	Length	Alert Identifier Hash	
+-----+-----+-----+-----+			

The Length is a 1-octet field whose value is equal to 8. The Alert Identifier Hash (AIH) is an 8-octet field. It is a unique value used to indicate an instance of an EAS message. The value of this field is the hash produced by the HMAC-SHA1-64 hash algorithm operating on the EAS message.

AIH =HMAC-SHA1-64("ES_ALERT", Emergency_Alert_Message)

where AIH is then truncated to the first 64 bits of this function. The Emergency_Alert_Message is the EAS message itself.

NOTE: The same value of hash will be computed by each AP in an ESS and by each AP in different ESSs. Thus a STA, which can download emergency alert messages when in a pre-associated state, can unambiguously determine that it has already downloaded the message, avoiding unnecessary duplicates.

4. Retrieving an EAS message

The STA can find out whether there are any EAS messages in the network by looking for Emergency Alert Information Elements in the beacon or Probe Response messages.

When the STA finds out that there are EAS messages in the network, it can access those messages with the protocol defined in 802.11-2011.

The STA can check if it had previously downloaded that alert message by computing a hash of the message and comparing it with the hash

value present in the Emergency Alert Information Element of the beacon or probe response frame.

The STA does not have to be associated with an AP in order to access and download the EAS message, it can do that in the pre-associated state as well. The STA needs to send a GAS Request public action frame to the AP by setting the Element-ID of the request to the EAS value and including the hash of the EAS message into the Query Request field. Then, in the GAS Response frame, the AP will deliver the requested EAS message to the STA.

According to the 802.11 specifications, the EAS message in the GAS Response frame is expected to be formatted in accordance with OASIS EDXL.

A STA which is already associated and authenticated with an AP, can also use GAS ANQP protocol to download the URI of a local Emergency Alert Server. The STA can then retrieve the EAS message using a URI formed by concatenating the downloaded local Emergency Alert Server URI with the hexadecimal numerals of the Alert Identifier Hash converted to UTF-8 encoded characters and the ".xml" file extension. For example, if the Emergency Alert Server URI is `http://eas.server.org` and the Alert Identifier Hash is `"0x1234567890abcdef"`, then the URI would be `http://eas.server.org/1234567890abcdef.xml`

The XML file is expected to be formatted in accordance with OASIS EDXL.

The mechanism by which the EAS message is retrieved from the formed URI is not specified in the 802.11 specification.

When an EAS Message has expired, an AP with `dot11EASActivated` set to TRUE shall remove the corresponding instance of an Emergency Alert Identifier element from its Beacon and Probe Response frames.

5. Protocol considerations

An Emergency Alert distribution protocol intended to work with WiFi Access Points will need to deliver the Alert message to the AP, which in turn will need to save it in a MIB variable. Currently there are no MIB variables defined in the 802.11 standard to store Alert messages.

Once the AP, which has EAS implemented and enabled, receives an Alert message, it needs to include an Emergency Alert Information Element into the beacon and probe responses, and remove it once the Alert message expires.

The AP would probably need to register itself to an Alert Distribution Server in order to receive the Alerts.

6. IANA considerations

None.

7. Security considerations

This document is provided for information to the IETF community. The protocol described here relies on the existence of a protocol which can distribute emergency alert messages to the wifi access points. As described here, the STAs can access an alert message in pre-associated state as well, when the STA did not authenticate the AP or the network behind the AP. Careful consideration should be taken on when such an alert message can be trusted and be displayed on the screen of a wifi device.

8. Normative References

9. Informative References

<http://standards.ieee.org/getieee802/download/802.11-2007.pdf>
<http://standards.ieee.org/getieee802/download/802.11-2011.pdf> (to be available in 2012)

10. Author's Addresses

Gabor Bajko
gabor.bajko@nokia.com

ATOCA
Internet-Draft
Intended status: Informational
Expires: May 3, 2012

R. Barnes
BBN Technologies
October 31, 2011

Lightweight Emergency Alerting Protocol (LEAP)
draft-barnes-atoca-delivery-01.txt

Abstract

Emergency alerts need to be delivered reliably from one source to many recipients at once. TCP is unsuitable for this style of delivery, because the large number of acknowledgements would likely cause network congestion. This document defines a UDP-based protocol for delivering alerts that supports fragmentation and retransmission for reliability, and allows the sender of a datagram to control whether acknowledgements are sent.

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

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Open Questions	3
2. Definitions	3
3. Packet Format	3
4. URI Format	4
5. Server Processing	5
6. Client Processing	6
7. IANA Considerations	7
8. Security Considerations	8
9. Acknowledgements	8
10. References	9
10.1. Normative References	9
10.2. Informative References	9
Author's Address	9

1. Introduction

Servers that provide emergency alerts to end hosts have two conflicting requirements. They need to deliver alerts reliably to a large number of hosts, but in a scalable fashion that does not cause undue network congestion. In particular, TCP is unsuitable for delivering alerts because of the overhead imposed by connection establishment and acknowledgement messages [RFC0793]. Sending alerts directly in a UDP datagram is not appropriate either, because of the size limits imposed by link maximum transmission units (MTUs) [RFC0768].

This document defines the Light-weight Emergency Alerting Protocol (LEAP) as a simple, UDP-based way to deliver emergency alerts. This protocol defines a simple fragmentation layer over UDP, and retransmission and reassembly algorithms that allow for reliable transmission of alerts without a need for acknowledgements. We also define a URI format for specifying alert sources, so that alert servers can inform alert recipients about what sorts of alerts they should accept over this protocol.

1.1. Open Questions

Should we randomize the order in which fragments are transmitted in order to deal with correlated loss?

How should we manage UDP ports? Require that destination==source? Require that destination==default? If there is any flexibility in port selection, should the URI format allow these to be indicated?

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].

3. Packet Format

LEAP transmits ESCAPE-encoded CAP alerts as a collection of fragments [I-D.barnes-atoca-escape]. Alert servers divide alerts into fragments that are small enough to fit into an MTU, and clients reassemble these fragments to obtain the complete alert. (See Section 5 and Section 6 for details on the fragmentation and reassembly processes.

LEAP payloads are encapsulated in UDP datagrams with source and

destination ports equal to XXX. Each datagram comprises a 4-octet LEAP header, followed by alert data:

[[Note to RFC Editor: Please replace the XXX above with the port number assigned by IANA]]

[illegible]

The LEAP header has the following fields:

- o alert-id: A 16-bit unsigned integer uniquely identifying this alert among alerts sent from the server IP address and port for this packet
- o frag-count: An 8-bit unsigned integer describing the total number of fragments in an alert
- o frag-no: An 8-bit unsigned integer describing the position of this payload in the sequence of alert fragments

The remainder of the UDP payload contains the body of the alert fragment itself. The reassembled fragments of a LEAP-transmitted alert MUST comprise a valid ESCAPE-formatted alert. Note that because each alert can be split into at most 256 fragments, the total size of the alert is still limited to a multiple of the MTU. If the available payload size after IP, UDP, and LEAP headers is 1KB, then the maximum alert that can be transmitted is 256KB.

4. URI Format

A LEAP URI describes an alert server that will transmit alerts using LEAP. Clients can use these URIs to determine which LEAP messages they should accept based on a list of authorized LEAP URIs.

```
[[ TODO: ABNF for URI format leap:[host/IP] ]]
```

5. Server Processing

An alert server transmits an ESCAPE-encoded alert according to the following steps:

1. Choose a 16-bit pseudo-random alert ID.
2. Divide the alert into fragments that are sufficiently small that they are likely to be less than the MTU on all links between the server and end clients. A 512-octet maximum fragment size is RECOMMENDED.
3. Attach to each fragment a LEAP header with the following values:
 - * alert-id: The 16-bit value chosen in step 1
 - * frag-count: The number of fragments generated in step 2
 - * frag-no: The index of this fragment in the sequence of fragments, starting at zero
4. Transmit each fragment (with its header) in a UDP datagram to the client(s)
5. Re-transmit the fragment sequence as necessary to achieve the desired level of reliability

Servers increase the reliability of alert delivery by retransmitting the sequence of alert fragments. Servers SHOULD compute the number of retransmissions R based on three factors:

- o p: The estimated probability of a packet successfully reaching the client from the server (one minus the loss rate)
- o q: The probability that a client receives all fragments successfully
- o F: The number of fragments in the alert

When clients apply the reassembly algorithm described below, the probability of receiving an entire alert after R retransmissions is given by the following formula:

$$q = (1 - (1-p)^R)^F$$

Solving this equation for R, the number of retransmissions required to achieve a resiliency q is as follows:

$$R = \log(1-q^{(1/F)}) / \log(1-p)$$

For example, if the server estimates that there is a 10% loss rate to clients ($p=.9$) and wishes to transmit a 10-fragment alert ($F=10$) with 99% reliability ($q=.99$), then it should transmit the entire sequence of alert fragments at least 3 times ($R=2.998$).

6. Client Processing

LEAP clients reassemble alert fragments from alert servers in order to obtain a complete alert. A LEAP client maintains a set of alert buffers (possibly empty) to hold fragments of incomplete alerts. Each buffer is identified by the IP address of the alert server and the 16-bit alert ID of the alert being reassembled. Each alert buffer contains the following data elements:

- o IP address of the alert server
- o Alert ID for this alert
- o Number of fragments in this alert
- o List of fragment numbers that have been received
- o List of fragment bodies that have been received

A LEAP client processes an incoming LEAP datagram according to the following steps:

1. Search for an existing alert buffer that matches this datagram's IP address and alert ID
2. If there is no current alert buffer, initialize one with the following values:
 - * IP address: The source IP address of the incoming datagram
 - * Alert ID: The alert ID from the LEAP header in the incoming datagram
 - * Number of fragments: The fragment count from the LEAP header in the incoming datagram
 - * Received fragment number list: A one-element list containing the fragment number from the LEAP header in the incoming datagram

- * Received fragment body list: A one-element list containing the fragment body in the incoming datagram
3. If there is a current alert buffer, add this datagram to the buffer:
 - A. If the fragment count field in the datagram differs from the fragment count field in the buffer, discard the datagram
 - B. Add the fragment number from the incoming datagram to the list of fragment numbers
 - C. Add the fragment body from the incoming datagram to the list of fragment bodies
 - D. If all fragments have been received, re-assemble the fragment bodies in order by fragment number and return the reassembled alert

In order to limit the amount of state that needs to be stored, clients SHOULD apply access controls before accepting incoming datagrams and limit the time that an individual buffer is stored. When a client has been configured with local alert servers (e.g., using the Alert Metadata Protocol [I-D.barnes-atoca-meta]), then it SHOULD only accept LEAP datagrams from configured servers.

Clients MUST apply a buffer timeout T1 to incoming alerts. If all fragments for a buffer do not arrive within T1 milliseconds, then the buffer is discarded. The RECOMMENDED default value for T1 is 5000 milliseconds.

Clients MAY also impose an absolute limit on the number of buffers they will store at one time, although this may cause them to miss a legitimate alert if an attacker sends many false alerts. If a client wishes to limit the number of buffers stored, it SHOULD place limits on a per-IP-address basis, rather than on a global basis. This will prevent attackers from creating many buffers, but still allow a legitimate alert server to transmit the few alerts that it needs to get through.

7. IANA Considerations

[TODO: Request a default port number]

[TODO: Register URI scheme]

8. Security Considerations

The primary risk for alerting systems is the introduction of false alert information, either by injecting false alerts or by modifying valid alerts. This protocol addresses these risks by using the authentication and integrity features of the ESCAPE alert format [I-D.barnes-atoca-escape].

The main security concern for this protocol is denial of service on the client, both in the sense of resource exhaustion and in the sense of preventing legitimate alerts from arriving. Clients are required to maintain state, so there is a risk that this state will be exhausted. Rejecting LEAP datagrams based on resource limits, however, can lead to legitimate alert datagrams being dropped.

Several DOS mitigations are described in Section 6 above. The LEAP protocol itself also imposes an absolute upper bound on the amount of data stored per source IP address. Due to the limited set of alert IDs and fragment numbers available, the worst-case amount of buffer is 2^{24} times the link MTU, for example 4GB for a 1KB MTU. An attacker can only force a client to accept more data than this by spoofing IP addresses or sending alerts from multiple hosts.

As discussed above, clients SHOULD apply resource constraints to limit the amount of state that an attacker can require a client to store. These resource constraints must be constructed so that legitimate alerts are still likely to get through. Since there is no authentication in LEAP, it is not possible to apply access controls based on cryptographic credentials. But if alert server IP addresses can be pre-provisioned, then the client can choose to accept datagrams only from those IP addresses. Limiting resources on a per-IP-address basis also increases the likelihood that legitimate alerts will be received. While attackers may try to send many alerts simultaneously in order to exhaust resources, real alert servers are much more likely to only send a few alerts at any given time.

9. Acknowledgements

Thanks to Martin Thomson, Brian Rosen, Hannes Tshofenig for help in developing and refining the ideas in this document.

10. References

10.1. 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.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

- [I-D.barnes-atoca-meta]
Barnes, R., "Alert Metadata Protocol (AMP)", draft-barnes-atoca-meta-00 (work in progress), October 2011.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

Author's Address

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

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

ATOCA
Internet-Draft
Intended status: Informational
Expires: May 2, 2012

M. Lepinski
K.Seo
R. Barnes
BBN Technologies
October 30, 2011

Alert Metadata Protocol (AMP)
draft-barnes-atoca-meta-01.txt

Abstract

Recipients of emergency alerts need to discover information about local alert distribution servers, and to register contact points where they can receive alerts. This document defines a mechanism for IP networks to advertise a local alert server, and a protocol that devices on the network can use to retrieve local information and register information about themselves.

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

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 2, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Open Questions	3
2. Definitions	3
3. Server Discovery	4
3.1. NAPTR Record Format	4
3.2. Client Processing	5
4. Protocol	6
4.1. Message Format	6
4.1.1. Registration	6
4.1.2. Advertisement	7
4.1.3. Refer	8
4.1.4. Alert	8
4.2. HTTP Transport of AMP Messages	9
5. IANA Considerations	10
5.1. AMP Message Type Registry	10
6. Security Considerations	11
7. Acknowledgements	11
8. References	12
8.1. Normative References	12
8.2. Informative References	12
Appendix A. JSON Schema for AMP Messages	13
Authors' Addresses	15

1. Introduction

In order for clients to securely receive alerts from alert servers, both endpoints and servers need a certain amount of configuration. For example, 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.

This document defines a protocol that addresses this problem in two parts. First, a client discovers a local alert server using information provided by its local network. Second, the client connects to the server and conducts an exchange of alerting-related metadata.

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 that Local Alert Distribution Servers will be discovered by via a U-NAPTR query using the domain name of the local network (in a fashion analogous to LIS discovery in [RFC5986]). An alternative approach would be to use standard LOST discovery [RFC5223] and then find the appropriate Local Alert Distribution Server by making a LOST query for some newly defined alert URN.

The current version of this draft specifies only an HTTP transport for AMP messages. However, as an alternative this document could also specify an option to use WebSockets 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 entities involved in this protocol are referred to as the "client" and "server". A client is any entity that is interested in receiving emergency alerts. A server in the sense of this document is an entity that maintains information about clients and information

about how alerts are delivered within some scope (e.g., within a jurisdiction); it may or may not be the server that actually delivers emergency alerts.

3. Server Discovery

Since many alerting scenarios are local (e.g., natural disasters) and ISPs are well-positioned to gather information on their local environment, 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.

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 LADS 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 specifies the format of the DNS NAPTR record used for this discovery, and Section 3.2 provides processing instructions for the client device performing the discovery.

3.1. 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 "http", "https", "ws", and "wss" are used to identify alert servers that support these protocols. 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:http" (           ; service
    ""                                     ; regex
    outsource.example.com.                 ; replacement
)

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

outsource.example.com.
;;      order pref flags
IN NAPTR 100 10 "u" "AMP:http" (         ; 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 HTTPS or 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 2818 [RFC2818]. 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.

3.2. 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 should look up its own IP address in the reverse DNS to obtain a domain name. The client should then attempt

to use this domain name as the domain name for the local access network. Note that if the U-NAPTR query using this domain name fails, then the client device should iteratively repeat the U-NAPTR query using as the domain name of the local access network the domain name obtained by removing the left-most portion of the domain name used in the previous attempt.

4. Protocol

The Alert Metadata Protocol (AMP) consists of a set of messages encoded as JSON objects [RFC4627]. Section 4.1 describes the four message types for the AMP protocol, Registration, Advertisement, Refer, and Alert. The complete JSON schema for these four message types appears in Appendix A. Section 4.2 specifies the use of HTTP as a transport for AMP messages. A MIME Type, `application/amp+json`, for use with AMP messages is registered in Section 5.

[Author's Note: Future versions of this document may define other transports AMP messages, e.g., WebSockets]

4.1. Message Format

Each AMP message is a JSON object consisting of a "type" and an array of "fields" that depend on the message type. Each of the four message types, Registration, Advertisement, Refer, and Alert, are described along with their corresponding properties in the following subsections. (A complete JSON scheme for these four message types appears in Appendix A.) Future documents may define additional message types. Therefore, implementations MUST ignore any AMP message containing a type field that it does not recognize.

4.1.1. 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 This field is a string that identifies the client. This field is optional and does not appear in the first registration message sent by a client to a particular server. However, once a client has received an advertisement message from a server, it copies the token from that message into all future registration messages so

that the server can distinguish between new registrations and updates to existing registrations.

contacts This field is an array of strings, where each string contains a URI that can be used to contact the client. This field is optional. If this field is not included then the registration is interpreted by the server as a request to delete existing registration information for this client.

location This field is a string containing a "location-info" element from a PIDF-LO. This field is optional, but it must be present if the **contacts** field is present.

language This field is a string containing the language in which the client wishes to receive alerts. This field uses the same values as the HTTP language tag. This field is optional, but it must be present if the **contacts** field is present.

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.

4.1.2. Advertisement

Advertisement messages are sent from servers to clients. These messages allow servers to notify clients about local alert authorities that sign authoritative alerts. This enables the client to validate future alerts regardless of the protocol mechanism used to transport the alert. An advertisement message contains the following fields:

token This field is a string that identifies the client. This field is mandatory. The server selects a token for the client when it first receives a registration from that client. The server then includes the token in all advertisement messages sent to that client.

contacts This field is an array of strings, where each string contains a URI from which local alerts may be sourced. This field is mandatory and the array **MUST NOT** be empty.

certs This field is an array of strings, where each string contains an X.509 certificate for a local authority. These certificates are used to validate local alerts signed by the given authority. This field is optional, but either this field or the **public_keys** field **MUST** be present and not empty.

public_keys This field is an array of strings, where each string contains Subject Public Key Information (SPKI) for a local authority. These are the public keys used to validate alerts signed by the given authority. This field is optional, but either this field or the **public_keys** field **MUST** be present and not empty.

hash_values This field is an array of hash values that are used in ESCAPE verification. This field is mandatory and the array **MUST NOT** be empty.

ttl: 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.

4.1.3. Refer

Advertisement 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 This field is a string that contains the URI of the AMP server to which the client is being referred.

Upon receiving a Refer message, a client **SHOULD** send a new registration message to 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 ESCAPE-encoded [I-D.barnes-atoca-escape]. An alert message contains the following fields:

alert_data This field is a string that contains an ESCAPE-encoded alert message.

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

4.2. HTTP Transport of AMP Messages

This section describes the use of HTTP [RFC2616] and HTTP over TLS [RFC2818] as transport mechanisms for the AMP protocol, which a conforming alert server and client MUST support.

Although AMP uses HTTP as a transport, it uses a strict subset of HTTP features, and due to the restrictions of some features, an alert server is not a fully compliant HTTP server. It is intended that an alert server can easily be built using an HTTP server with extensibility mechanisms and that an AMP client can trivially use existing HTTP libraries. This subset of requirements helps implementors avoid ambiguity with the many options that the full HTTP protocol offers.

A client that conforms to this specification MAY choose not to support HTTP authentication [RFC2617] or cookies [RFC2965]. Because the client and the alert server may not necessarily have a prior relationship, the alert server SHOULD NOT require a client to authenticate, either using the above HTTP authentication methods or TLS client authentication. Unless all clients that access a LIS can be expected to be able to authenticate in a certain fashion, denying access to alerts could prevent a client from receiving critical emergency information.

An AMP Registration message MUST be carried in the body of an HTTP POST request. The client MUST include a Host header in the request. The alert server response to a registration request MUST be 200 unless there is an HTTP-layer error. The Response SHOULD contain an AMP Advertisement message. The POST method is the only method REQUIRED for AMP.

The MIME type of an AMP registration request and response bodies is "application/amp+json". The alert server and the client MUST provide this value in the HTTP Content-Type and Accept header fields. If the alert server does not receive the appropriate Content-Type and Accept header fields, the alert server SHOULD fail the request, returning a 406 (not acceptable) response. AMP responses SHOULD include a Content-Length header.

Clients MUST NOT use the "Expect" header or the "Range" header in AMP requests. The alert server MAY return 501 (not implemented) errors

if either of these HTTP features are used. In the case that the alert server receives a registration request from the client containing an If-* (conditional) header, the alert server SHOULD return a 412 (precondition failed) response.

The alert server populates the HTTP headers of responses so that they are consistent with the contents of the message. In particular, the "Cache-Control" header SHOULD be set to disable caching of AMP Advertisement messages by HTTP intermediaries. Instead, the alert server controls caching of AMP Advertisement messages by setting the TTL field in the Advertisement message.

The alert server SHOULD NOT use an HTTP 3xx response to redirect an AMP Registration request. Instead, the alert server SHOULD redirect AMP Registration requests by providing an HTTP 200 response containing the AMP Refer message.

Implementations of AMP that implement HTTP transport MUST implement transport over TLS [RFC2818]. TLS provides message integrity and confidentiality between the client and alert server. The client MUST implement the server authentication method described in Section 3.1 of [RFC2818], with an exception in how wild cards are handled. The leftmost label MAY contain the wild card string "*", which matches any single domain name label. Additional characters in this leftmost label are invalid (that is, "f*.example.com" is not a valid name and does not match any domain name). The client uses the URI obtained during alert server discovery to authenticate the server. The details of this authentication method are provided in Section 3.1 of HTTPS [RFC2818]. When TLS is used, the client SHOULD fail a request if server authentication fails, except in the event of an emergency.

5. IANA Considerations

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

```
[[ 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 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.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [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.
- [RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.
- [RFC5986] Thomson, M. and J. Winterbottom, "Discovering the Local Location Information Server (LIS)", RFC 5986, September 2010.

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.
- [I-D.ietf-atoca-requirements]
Schulzrinne, H., Norreys, S., Rosen, B., and H. Tschofenig, "Requirements, Terminology and Framework for Exigent Communications", draft-ietf-atoca-requirements-02 (work in progress), October 2011.
- [RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP

Authentication: Basic and Digest Access Authentication",
RFC 2617, June 1999.

[RFC2965] Kristol, D. and L. Montulli, "HTTP State Management
Mechanism", RFC 2965, October 2000.

[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.

Appendix A. JSON Schema for AMP Messages

```
# Registration
{
  "type": "Registration",
  "fields": {
    "token": {
      "type": "string",
      "description": "Identifier for client",
      "required": false
    },
    "contacts": {
      "description": "Array of URIs",
      "type": "array",
      "uri": {
        "type": "string"
      },
      "required": false
    },
    "location": {
      "type": "string",
      "description": "Location-info element from PIDF-LO",
      "required": false
    },
    "language": {
      "type": "string",
      "description": "Language tag",
      "required": false
    }
  }
}
```

```
}

# Advertisement
{
  "type": "Advertisement",
  "fields":
  {
    "contacts":
    {
      "description": "Array of source URIs for alerts sourced",
      "type": "array",
      "uri":
      {
        "type": "string"
      }
      "required": true
    },
    "token":
    {
      "type": "string",
      "description": "Identifier that client should use in future",
      "required": true
    },
    "certs":
    {
      "description": "Array of certificates for local authorities",
      "type": "array",
      "cert":
      {
        "type": "string"
      }
      "required": false
    },
    "public_keys":
    {
      "description": "Array of SPKIs for local authorities",
      "type": "array",
      "spki":
      {
        "type": "string"
      }
      "required": false
    },
    "hash_values":
    {
      "description": "Array of hash values for ESCAPE verification",
      "type": "array",
      "hash":

```

```
        {
            "type":"string"
        }
        "required":true
    },
    "ttl":
    {
        "type":"number",
        "description":"Number of seconds for advertisement validity",
        "required":true
    }
}

# Refer
{
    "type":"Refer",
    "fields":
    {
        "to":
        {
            "type":"string",
            "description":"URI of new AMP server client should contact",
            "required":true
        }
    }
}

# Alert
{
    "type":"Alert",
    "fields":
    {
        "alert_data":
        {
            "type":"string",
            "description":"ESCAPE-encoded alert data",
            "required":true
        }
    }
}
```

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 Technologies
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

This Internet-Draft, draft-ietf-atoca-requirements-01.txt, has expired, and has been deleted from the Internet-Drafts directory. An Internet-Draft expires 185 days from the date that it is posted unless it is replaced by an updated version, or the Secretariat has been notified that the document is under official review by the IESG or has been passed to the RFC Editor for review and/or publication as an RFC. This Internet-Draft was not published as an RFC.

Internet-Drafts are not archival documents, and copies of Internet-Drafts that have been deleted from the directory are not available. The Secretariat does not have any information regarding the future plans of the authors or working group, if applicable, with respect to this deleted Internet-Draft. For more information, or to request a copy of the document, please contact the authors directly.

Draft Authors:

Henning Schulzrinne<hgs+ecrit@cs.columbia.edu>

Steve Norreys<steve.norreys@bt.com>

Brian Rosen<br@brianrosen.net>

Hannes Tschofenig<Hannes.Tschofenig@gmx.net>