Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

T. Pusateri
Seeking affiliation
S. Cheshire
Apple Inc.

March 9, 2015

DNS Push Notifications
draft-ietf-dnssd-push-00

Abstract

The Domain Name System (DNS) was designed to efficiently return matching records for queries for data that is relatively static. When those records change frequently, DNS is still efficient at returning the updated results when polled. But there exists no mechanism for a client to be asynchronously notified when these changes occur. This document defines a mechanism for a client to be notified of such changes to DNS records, called DNS Push Notifications.

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 September 10, 2015.

Table of Contents

1. Introduction

DNS records may be updated using DNS Update [RFC2136]. Other mechanisms such as a Hybrid Proxy [I-D.ietf-dnssd-hybrid] can also generate changes to a DNS zone. This document specifies a protocol for Unicast DNS clients to subscribe to receive asynchronous notifications of changes to RRSets of interest. It is immediately relevant in the case of DNS Service Discovery [RFC6763] but is not limited to that use case and provides a general DNS mechanism for DNS record change notifications. Familiarity with the DNS protocol and DNS packet formats is assumed [RFC1034] [RFC1035] [RFC6195].

1.1. Requirements Language

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 "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

2. Motivation

As the domain name system continues to adapt to new uses and changes in deployment, polling has the potential to burden DNS servers at many levels throughout the network. Other network protocols have successfully deployed a publish/subscribe model to state changes following the Observer design pattern. XMPP Publish-Subscribe [XEP-0060] and Atom [RFC4287] are examples. While DNS servers are generally highly tuned and capable of a high rate of query/response traffic, adding a publish/subscribe model for tracking changes to DNS records can result in more timely notification of changes with reduced CPU usage and lower network traffic.

Multicast DNS [RFC6762] implementations always listen on a well known link-local IP multicast group, and new services and updates are sent for all group members to receive. Therefore, Multicast DNS already has asynchronous change notification capability. However, when DNS Service Discovery [RFC6763] is used across a wide area network using Unicast DNS (possibly facilitated via a Hybrid Proxy [I-D.ietf-dnssd-hybrid]) it would be beneficial to have an equivalent capability for Unicast DNS, to allow clients to learn about DNS record changes in a timely manner without polling.

DNS Long-Lived Queries (LLQ) [I-D.sekar-dns-llq] is an existing deployed solution to provide asynchronous change notifications. Even though it can be used over TCP, LLQ is defined primarily as a UDP-based protocol, and as such it defines its own equivalents of existing TCP features like the three-way handshake. This document builds on experience gained with the LLQ protocol, with an improved design that uses long-lived TCP connections instead of UDP (and therefore doesn't need to duplicate existing TCP functionality), and adopts the syntax and semantics of DNS Update messages [RFC2136] instead of inventing a new vocabulary of messages to communicate DNS zone changes.

3. Overview

The existing DNS Update protocol [RFC2136] provides a mechanism for clients to add or delete individual resource records (RRs) or entire resource record sets (RRSets) on the zone's server. Adopting this existing syntax and semantics for DNS Push Notifications allows for messages going in the other direction, from server to client, to communicate changes to a zone. The client first must subscribe for Push Notifications by connecting to the server and sending DNS message(s) indicating the RRSet(s) of interest. When the client loses interest in updates to these records, it unsubscribes. The DNS Push Notification server for a zone is any server capable of generating the correct change notifications for a name. It may be a master, slave, or stealth name server [RFC1996].

DNS Push Notification clients are NOT required to implement DNS Update Prerequisite processing. Prerequisites are used to perform tentative atomic test-and-set type operations on the server, and that concept has no application when it comes to an authoritative server telling a client of changes to DNS records.

4. Transport

Implementations of DNS Update [RFC2136] MAY use either User Datagram Protocol (UDP) [RFC0768] or Transmission Control Protocol (TCP) [RFC0793] as the transport protocol, in keeping with the historical precedent that DNS queries must first be sent over UDP [RFC1123]. This requirement to use UDP has subsequently been relaxed [RFC5966]. DNS Push Notification is defined only for TCP. DNS Push Notification clients MUST use TCP.

Either end of the TCP connection can terminate all of the subscriptions on that connection by simply closing the connection abruptly with a TCP RST. (An individual subscription is terminated by sending an UNSUBSCRIBE message for that specific subscription.)

If a client closes the connection, it is signaling that it is no longer interested in receiving updates to any of the records it has subscribed. It is informing the server that the server may release all state information it has been keeping with regards to this client. This may occur because the client computer has been disconnected from the network, has gone to sleep, or the application requiring the records has terminated.

If a server closes the connection, it is informing the client that it can no longer provide updates for the subscribed records. This may occur because the server application software or operating system is restarting, the application terminated unexpectedly, the server is

undergoing maintenance procedures, or the server is overloaded and can no longer provide the information to all the clients that wish to receive it. The client can try to re-subscribe at a later time or connect to another server supporting DNS Push Notifications for the zone.

Transport Layer Security (TLS) [RFC5246] is well understood and deployed across many protocols running over TCP. It is designed to prevent eavesdropping, tampering, or message forgery. TLS is REQUIRED for every connection between a client subscriber and server in this protocol specification.

Connection setup over TCP ensures return reachability and alleviates concerns of state overload at the server through anonymous subscriptions. All subscribers are guaranteed to be reachable by the server by virtue of the TCP three-way handshake. Additional security measures such as authentication during TLS negotiation MAY also be employed to increase the trust relationship between client and server. Because TCP SYN flooding attacks are possible with any protocol over TCP, implementers are encouraged to use industry best practices to guard against such attacks [IPJ.9-4-TCPSYN].

5. State Considerations

Each DNS Push Notification server is capable and handling some finite number of Push Notification subscriptions. This number will vary from server to server and is based on physical machine characteristics, network bandwidth, and operating system resource allocation. After a client establishes a connection to a DNS server, each record subscription is individually accepted or rejected. Servers may employ various techniques to limit subscriptions to a manageable level. Correspondingly, the client is free to establish simultaneous connections to alternate DNS servers that support DNS Push Notifications for the zone and distribute record subscriptions at its discretion. In this way, both clients and servers can react to resource constraints. Token bucket rate limiting schemes are also effective in providing fairness by a server across numerous client requests.

6. Protocol Operation

A DNS Push Notification exchange begins with the client discovering the appropriate server, and connecting to it. The client may then add and remove Push Notification subscriptions over this connection. In accordance with the current set of active subscriptions the server sends relevant asynchronous Push Notifications to the client. The exchange terminates when either end closes the TCP connection with a TCP RST.

6.1. Discovery

The first step in DNS Push Notification subscription is to discover an appropriate DNS server that supports DNS Push Notifications for the desired zone. The client MUST also determine which TCP port on the server is listening for connections, which need not be (and often is not) the typical TCP port 53 used for conventional DNS.

1. The client begins the discovery by sending a DNS query to the local resolver with record type SOA [RFC1035] for the name of the record it wishes to subscribe.

2. If the SOA record exists, it MUST be returned in the Answer Section of the reply. If not, the server SHOULD include the SOA record for the zone of the requested name in the Authority Section.

3. If no SOA record is returned, the client then strips off the leading label from the requested name. If the resulting name has at least one label in it, the client sends a new SOA query and processing continues at step 2 above. If the resulting name is empty (the root label) then this is a network configuration error and the client gives up. The client MAY retry the operation at a later time.

4. Once the SOA is known, the client sends a DNS query with type SRV [RFC2782] for the record name "_dns-push._tcp.<zone>", where <zone> is the owner name of the discovered SOA record.

5. If the zone in question does not offer DNS Push Notifications then SRV record MUST NOT exist and the SRV query will return a negative answer.

6. If the zone in question is set up to offer DNS Push Notifications then this SRV record MUST exist. The SRV "target" contains the name of the server providing DNS Push Notifications for the zone. The port number on which to contact the server is in the SRV record "port" field. The address(es) of the target host MAY be

included in the Additional Section, however, the address records SHOULD be authenticated before use as described below in Section 9.2 [I-D.ietf-dane-srv].

7. More than one SRV record may be returned. In this case, the "priority" and "weight" values in the returned SRV records are used to determine the order in which to contact the servers for subscription requests. As described in the SRV specification [RFC2782], the server with the lowest "priority" is first contacted. If more than one server has the same "priority", the "weight" is indicates the weighted probability that the client should contact that server. Higher weights have higher probabilities of being selected. If a server is not reachable or is not willing to accept a subscription request, then a subsequent server is to be contacted.

If a server closes a DNS Push Notification subscription connection, the client SHOULD repeat the discovery process in order to determine the preferred DNS server for subscriptions at that time.

6.2. DNS Push Notification SUBSCRIBE

A DNS Push Notification client indicates its desire to receive DNS Push Notifications for a given domain name by sending a SUBSCRIBE request over the established TCP connection to the server. A SUBSCRIBE request is formatted identically to a conventional DNS QUERY request [RFC1035], except that the opcode is SUBSCRIBE (6) instead of QUERY (0). If neither QTYPE nor QCLASS are ANY (255) then this is a specific subscription to changes for the given name, type and class. If one or both of QTYPE or QCLASS are ANY (255) then this is a wildcard subscription to changes for the given name for any type and/or class, as appropriate.

In a SUBSCRIBE request the DNS Header QR bit MUST be zero.
If the QR bit is not zero the message is not a SUBSCRIBE request.

The AA, TC, RD, RA, Z, AD, and CD bits, the ID field, and the RCODE field, MUST be zero on transmission, and MUST be silently ignored on reception.

Like a DNS QUERY request, a SUBSCRIBE request MUST contain exactly one question. Since SUBSCRIBE requests are sent over TCP, multiple SUBSCRIBE requests can be concatenated in a single TCP stream and packed efficiently into TCP segments, so the ability to pack multiple SUBSCRIBE operations into a single DNS message within that TCP stream would add extra complexity for little benefit.

ANCOUNT MUST be zero, and the Answer Section MUST be empty.
Any records in the Answer Section MUST be silently ignored.

NSCOUNT MUST be zero, and the Authority Section MUST be empty.
Any records in the Authority Section MUST be silently ignored.

ARCOUNT MUST be zero, and the Additional Section MUST be empty.
Any records in the Additional Section MUST be silently ignored.

Each SUBSCRIBE request generates exactly one SUBSCRIBE response from the server.

In the SUBSCRIBE response the RCODE indicates whether or not the subscription was accepted. Supported RCODEs are as follows:

Mnemonic	Value	Description
NOERROR	0	SUBSCRIBE successful
FORMERR	1	Server failed to process request due to a malformed request
SERVFAIL	2	Server failed to process request due to resource exhaustion
NOTIMP	4	Server does not implement DNS Push Notifications
REFUSED	5	Server refuses to process request for policy or security reasons

Table 1: Response codes

In a SUBSCRIBE response the DNS Header QR bit MUST be one.
If the QR bit is not one the message is not a SUBSCRIBE response.

The AA, TC, RD, RA, Z, AD, and CD bits, and the ID field, MUST be zero on transmission, and MUST be silently ignored on reception.

The Question Section MUST echo back the values provided by the client in the SUBSCRIBE request that generated this SUBSCRIBE response.

ANCOUNT MUST be zero, and the Answer Section MUST be empty.
Any records in the Answer Section MUST be silently ignored.
If the subscription was accepted and there are positive answers for the requested name, type and class, then these positive answers MUST be communicated to the client in an immediately following Push Notification Update, not in the Answer Section of the SUBSCRIBE response. This simplifying requirement is made so that there is only a single way that information is communicated to a DNS Push Notification client. Since a DNS Push Notification client has to parse information received via Push Notification Updates anyway, it is simpler if it does not also have to parse information received via the Answer Section of a SUBSCRIBE response.

NSCOUNT MUST be zero, and the Authority Section MUST be empty.
Any records in the Authority Section MUST be silently ignored.

ARCOUNT MUST be zero, and the Additional Section MUST be empty.

Any records in the Additional Section MUST be silently ignored.

If accepted, the subscription will stay in effect until the client revokes the subscription or until the connection between the client and the server is closed.

A client MUST not send a SUBSCRIBE message that duplicates the name, type and class of an existing active subscription. For the purpose of this matching, the established DNS case-insensitivity for US-ASCII letters applies (e.g., "foo.com" and "Foo.com" are the same). If a server receives such a duplicate SUBSCRIBE message this is an error and the server MUST immediately close the TCP connection.

Wildcarding is not supported. That is, a wildcard ("*") in a SUBSCRIBE message matches only a wildcard ("*") in the zone, and nothing else.

Aliasing is not supported. That is, a CNAME in a SUBSCRIBE message matches only a CNAME in the zone, and nothing else.

A client may SUBSCRIBE to records that are unknown to the server at the time of the request and this is not an error. The server MUST accept these requests and send Push Notifications if and when matches are found in the future.

6.3. DNS Push Notification UNSUBSCRIBE

To cancel an individual subscription without closing the entire connection, the client sends an UNSUBSCRIBE message over the established TCP connection to the server. The UNSUBSCRIBE message is formatted identically to the SUBSCRIBE message which created the subscription, with the exact same name, type and class, except that the opcode is UNSUBSCRIBE (7) instead of SUBSCRIBE (6).

A client MUST not send an UNSUBSCRIBE message that does not exactly match the name, type and class of an existing active subscription. If a server receives such an UNSUBSCRIBE message this is an error and the server MUST immediately close the TCP connection.

No response message is generated as a result of processing an UNSUBSCRIBE message.

Having being successfully revoked with a correctly-formatted UNSUBSCRIBE message, the previously referenced subscription is no longer active and the server MAY discard the state associated with it immediately, or later, at the server's discretion.

6.4. DNS Push Notification Update Messages

Once a subscription has been successfully established, the server generates Push Notification Updates to send to the client as appropriate. An initial Push Notification Update will be sent immediately in the case that the answer set was non-empty at the moment the subscription was established. Subsequent changes to the answer set are then communicated to the client in subsequent Push Notification Updates.

The format of Push Notification Updates borrows from the existing DNS Update [RFC2136] protocol, with some simplifications.

The following figure shows the existing DNS Update header format:

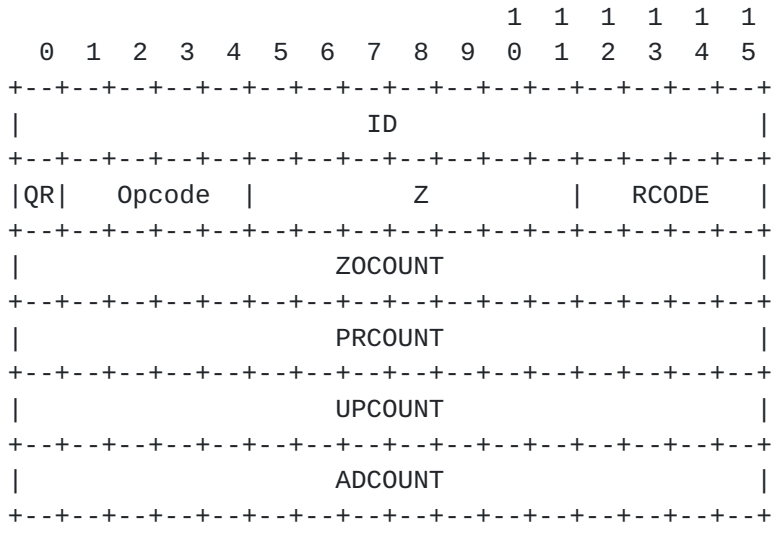

Figure 1

For DNS Push Notifications the following rules apply:

The QR bit MUST be zero, and the Opcode MUST be UPDATE (5). Messages received where this is not true are not Push Notification Update Messages and should be silently ignored for the purposes of Push Notification Update Message handling.

ID, the Z bits, and RCODE MUST be zero on transmission, and MUST be silently ignored on reception.

ZOCOUNT MUST be zero, and the Zone Section MUST be empty. Any records in the Zone Section MUST be silently ignored.

PRCOUNT MUST be zero, and the Prerequisite Section MUST be empty. Any records in the Prerequisite Section MUST be silently ignored.

ADCOUNT MUST be zero, and the Additional Data Section MUST be empty. Any records in the Additional Data Section MUST be silently ignored.

The Update Section contains the relevant change information for the client, formatted identically to a DNS Update [RFC2136]. To recap:

Delete all RRsets from a name:
TTL=0, CLASS=ANY, RDLENGTH=0, TYPE=ANY.

Delete an RRset from a name:
TTL=0, CLASS=ANY, RDLENGTH=0;
TYPE specifies the RRset being deleted.

Delete an individual RR from a name:
TTL=0, CLASS=NONE;
TYPE, RDLENGTH and RDATA specifies the RR being deleted.

Add an individual RR to a name:
TTL, CLASS, TYPE, RDLENGTH and RDATA specifies the RR being added.

Upon reception of a Push Notification Update Message, the client receiving the message MUST validate that the records being added or deleted correspond with at least one currently active subscription on that connection. Specifically, the record name MUST match the name given in the SUBSCRIBE request, subject to the usual established DNS case-insensitivity for US-ASCII letters. If the QTYPE was not ANY (255) then the TYPE of the record must match the QTYPE given in the SUBSCRIBE request. If the QCLASS was not ANY (255) then the CLASS of the record must match the QCLASS given in the SUBSCRIBE request. If a matching active subscription on that connection is not found, then that individual record addition/deletion is silently ignored. Processing of other additions and deletions in this message is not affected. The TCP connection is not closed. This is to allow for the race condition where a client sends an outbound UNSUBSCRIBE while inbound Push Notification Updates for that subscription from the server are still in flight.

In the case where a single change affects more than one active subscription, only one update is sent. For example, an update adding a given record may match both a SUBSCRIBE request with the same QTYPE and a different SUBSCRIBE request with QTYPE=ANY. It is not the case that two updates are sent because the new record matches two active subscriptions.

The server SHOULD encode change notifications in the most efficient manner possible. For example, when three AAAA records are deleted from a given name, and no other AAAA records exist for that name, the server SHOULD send a "delete an RRset from a name" update, not three

separate "delete an individual RR from a name" updates. Similarly, when both an SRV and a TXT record are deleted from a given name, and no other records of any kind exist for that name, the server SHOULD send a "delete all RRsets from a name" update, not two separate "delete an RRset from a name" updates.

Reception of a Push Notification Update Message results in no response back to the server.

The TTL of an added record is stored by the client and decremented as time passes, with the caveat that for as long as a relevant subscription is active, the TTL does not decrement below 1 second. For as long as a relevant subscription remains active, the client SHOULD assume that when a record goes away the server will notify it of that fact. Consequently, a client does not have to poll to verify that the record is still there. Once a subscription is cancelled (individually, or as a result of the TCP connection being closed) record aging resumes and records are removed from the local cache when their TTL reaches zero.

7. Acknowledgements

The authors would like to thank Kiren Sekar and Marc Krochmal for previous work completed in this field. This draft has been improved due to comments from Ran Atkinson.

8. IANA Considerations

This document defines the service name: "_dns-push._tcp".
It is only applicable for the TCP protocol.
This name is to be published in the IANA Service Name Registry.

This document defines two DNS OpCodes: SUBSCRIBE with (tentative) value 6 and UNSUBSCRIBE with (tentative) value 7.

9. Security Considerations

Strict TLS support is mandatory in DNS Push Notifications. There is no provision for opportunistic encryption using a mechanism like "STARTTLS".

9.1. Security Services

It is the goal of using TLS to provide the following security services:

Confidentiality All application-layer communication is encrypted with the goal that no party should be able to decrypt it except the intended receiver.

Data integrity protection Any changes made to the communication in transit are detectable by the receiver.

Authentication An end-point of the TLS communication is authenticated as the intended entity to communicate with.

Deployment recommendations on the appropriate key lengths and cypher suites are beyond the scope of this document. Please refer to TLS Recommendations [I-D.ietf-uta-tls-bcp] for the best current practices. Keep in mind that best practices only exist for a snapshot in time and recommendations will continue to change. Updated versions or errata may exist for these recommendations.

9.2. TLS Name Authentication

As described in Section 6.1, the client discovers the DNS Push Notification server using an SRV lookup for the record name "_dns-push._tcp.<zone>". The server connection endpoint SHOULD then be authenticated using DANE TLSA records for the associated SRV record. This associates the target's name and port number with a trusted TLS certificate [I-D.ietf-dane-srv]. This procedure uses the TLS Sever Name Indication (SNI) extension [RFC6066] to inform the server of the name the client has authenticated through the use of TLSA records.

9.3. TLS Compression

In order to reduce the chances of compression related attacks, TLS-level compression SHOULD be disabled when using TLS versions 1.2 and earlier. In the draft version of TLS 1.3 [I-D.ietf-tls-tls13], TLS-level compression has been removed completely.

9.4. TLS Session Resumption

TLS Session Resumption MUST be disabled on DNS Push Notification servers. It is not useful to have subscription state cached for long periods of time. It is also not desirable for subscription state to be maintained while the client is not connected.

10. References

10.1. Normative References

[I-D.ietf-dane-srv]
Finch, T., Miller, M., and P. Saint-Andre, "Using DNS-Based Authentication of Named Entities (DANE) TLSA Records with SRV Records", draft-ietf-dane-srv-11 (work in progress), February 2015.

[I-D.ietf-tls-tls13]
Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-04 (work in progress), January 2015.

[I-D.ietf-uta-tls-bcp]
Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of TLS and DTLS", draft-ietf-uta-tls-bcp-11 (work in progress), February 2015.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

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

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, August 2010.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6195] Eastlake, D., "Domain Name System (DNS) IANA Considerations", RFC 6195, March 2011.

10.2. Informative References

[I-D.ietf-dnssd-hybrid]
Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-00 (work in progress), November 2014.

[I-D.sekar-dns-llq]
Sekar, K., "DNS Long-Lived Queries", draft-sekar-dns-llq-01 (work in progress), August 2006.

[IPJ.9-4-TCPSYN]
Eddy, W., "Defenses Against TCP SYN Flooding Attacks", The Internet Protocol Journal, Cisco Systems, Volume 9, Number 4, December 2006.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 2005.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[XEP-0060]
Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe", XSF XEP 0060, July 2010.

Authors' Addresses

Tom Pusateri
Seeking affiliation
Hilton Head Island, SC
USA

Phone: +1 843 473 7394
Email: pusateri@bangj.com

Stuart Cheshire
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
USA

Phone: +1 408 974 3207
Email: cheshire@apple.com