SIPCORE Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 31, 2019

C. Holmberg
Ericsson
M. Arnold
Metaswitch Networks

September 27, 2018

Push Notification with the Session Initiation Protocol (SIP)
draft-ietf-sipcore-sip-push-16

Abstract

This document describes how a Push Notification Service (PNS) can be used to wake suspended Session Initiation Protocol (SIP) User Agents (UAs), using push notifications, for the UA to be able to send binding refresh REGISTER requests and to receive receive incoming SIP requests. The document defines new SIP URI parameters and new feature-capability indicators that can be used in SIP messages to indicate support of the mechanism defined in this document, to exchange PNS information between the SIP User Agent (UA) and the SIP entity that will request push notifications towards the UA, and to trigger such push notification requests.

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 March 31, 2019.

Copyright Notice

Copyright (c) 2018 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
2. Conventions 6
3. Push Resource ID (PRID) 6
4. SIP User Agent (UA) Behavior 7
4.1. Request Push Notifications from Network 7
4.2. Query Network Push Notification Capabilities 9
5. SIP Proxy Behavior 10
5.1. PNS Provider 10
5.2. Trigger Periodic Binding Refresh 10
5.3. SIP Requests 11
5.3.1. REGISTER 11
5.3.2. Initial Request for Dialog or Stand-Alone Request . . 13
6. Support Of Longlived SIP Dialogs 15
6.1. SIP UA Behavior 16
6.1.1. Initial Request for Dialog 16
6.2. SIP Proxy Behavior 16
6.2.1. REGISTER 16
6.2.2. Initial Request for Dialog 17
6.2.3. Mid-Dialog Request 17
7. Grammar 18
7.1. 555 (Push Notification Service Not Supported) Response Code 18
7.2. 556 (Push Notification Failed) Response Code 18
7.3. sip.pns Feature-Capability Indicator 18
7.4. sip.vapid Feature-Capability Indicator 19
7.5. sip.pnsreg Feature-Capability Indicator 19
7.6. sip.pnsreg Media Feature Tag 19
7.7. SIP URI Parameters 20
8. PNS Registration Requirements 20
9. pn-provider, pn-param and pn-prid URI Parameters for Apple Push Notification service 20
10. pn-provider, pn-param and pn-prid URI Parameters for Google Firebase Cloud Messaging (FCM) push notification service . . 21
11. pn-provider, pn-param and pn-prid URI Parameters for RFC 8030 (Generic Event Delivery Using HTTP Push) 21
12. Security Considerations 22
13. IANA considerations 23
13.1. SIP URI Parameters 23
13.1.1. pn-provider 23

13.1.2. pn-param 23
13.1.3. pn-prid 23
13.2. SIP Response Codes 23
13.2.1. 555 (Push Notification Service Not Supported) . . . 23
13.2.2. 556 (Push Notification Failed) 24
13.3. SIP Global Feature-Capability Indicator 24
13.3.1. sip.pns 24
13.3.2. sip.vapid 25
13.3.3. sip.pnsreg 25
13.4. SIP Media Feature Tag 26
13.4.1. sip.pnsreg 26
13.5. PNS Sub-registry Establishment 27
14. Acknowledgements 28
15. References 28
15.1. Normative References 28
15.2. Informative References 29
Authors' Addresses 30

1. Introduction

In order to save resources (e.g., battery life) some devices (especially mobile devices) and operating systems will suspend applications when not used. In some cases, internal timers cannot be used to wake such applications, nor will incoming network traffic wake the application. Instead, one way to wake the application is by using a Push Notification Service (PNS). Typically each operating system uses a dedicated PNS. For example, Apple iOS devices use the Apple Push Notification service (APNs) while Android devices use the Firebase Cloud Messaging (FCM) service.

Because of the restrictions above, Session Initiation Protocol (SIP) User Agents (UAs) [RFC3261] can not be awoken, in order to send binding refresh SIP REGISTER requests and to receive incoming SIP requests, without using a PNS to wake the UA in order to perform those functions.

Also, without being able to use internal timers in order to wake applications, a UA will not be able to maintain connections e.g., using the SIP Outbound Mechanism [RFC5626], as it requires the UA to send periodic keep-alive messages.

This document describes how PNSs can be used to wake suspended UAs, using push notifications, to be able to send binding refresh REGISTER requests and to receive incoming SIP requests. The document defines new SIP URI parameters and new feature-capability indicators [RFC6809] that can be used in SIP messages to indicate support of the mechanism defined in this document, to exchange PNS information between the UA and the SIP entity (realized as a SIP proxy in this

document) that will request push notifications towards the UA, and to request such push notification requests.

NOTE: Even if a UA is able to be awaken other means than receiving push notifications (e.g., by using internal timers) in order to send periodic binding refresh REGISTER requests, it might still be useful to suspend the application between the sending of binding refresh requests (as it will save battery life) and use push notifications to wake the UA when an incoming SIP request UA arrives.

When a UA registers to a PNS, it will receive a unique Push Resource ID (PRID) associated with the push notification registration. The UA will use a REGISTER request to provide the PRID to the SIP proxy that will request push notifications towards the UA.

When the proxy receives a SIP request for a new dialog, a stand-alone SIP request or a SIP mid-dialog request (in case of longlived dialogs),addressed towards a UA, or when the proxy determines that the UA needs to send a binding refresh REGISTER request, the proxy will request a push notification towards the UA, using the PNS of the UA. Once the UA receives the push notification, it will be able to send a binding refresh REGISTER request and receive the incoming SIP request. The proxy will receive the REGISTER request. If the push notification request was triggered by a SIP request addressed towards the UA (see above), once the REGISTER request has been accepted by the SIP registrar [RFC3261], and the associated SIP 2xx response has been forwarded by the proxy towards the UA, the proxy can forward the SIP request towards the UA using normal SIP routing procedures. In some cases the proxy can forward the SIP request without waiting for the SIP 2xx response to the REGISTER request. Note that this mechanism necessarily adds delay to responding to non-INVITE requests requiring push notification. The consequences of that delay are discussed in Section 5.3.2.

Different PNSs exist today. Some are based on the standardized mechanism defined in [RFC8030], while others are proprietary (e.g., the Apple Push Notification service). Figure 1 shows the generic push notification architecture supported by the mechanism in this document.

Each PNS uses PNS-specific terminology and function names. The terminology in this document is meant to be PNS-independent. If the PNS is based on [RFC8030], the SIP proxy takes the role of the application server.

The proxy MUST be in the signalling path of REGISTER requests sent by the UA towards the registrar, of SIP requests (for a new dialog or a stand-alone) forwarded by the proxy responsible for the UA's domain

(sometimes referred to as home proxy, S-CSCF, etc) towards the UA and of mid-dialog requests that can trigger push notification requests. The proxy can also be co-located with the proxy responsible for the UA's domain. This will also ensure that the Request-URI of SIP requests (for a new dialog or a stand-alone) can be matched against contacts in REGISTER requests.

```
  +--------+           +--------------+       +-----------------+
  | SIP UA |           | Push Service |       |    SIP Proxy    |
  +--------+           +--------------+       +-----------------+
      |                      |                         |
      |      Subscribe       |                         |
      |--------------------->|                         |
      |                      |                         |
      |    Push Resource ID  |                         |
      |<---------------------|                         |
      |                      |                         |
      |          SIP REGISTER (Push Resource ID)       |
      |===============================================>|
      |                      |                         | SIP REGISTER
      |                      |                         |============>
      |                      |                         |
      |                      |                         | SIP 200 OK
      |                      |                         |<============
      |          SIP 200 OK  |                         |
      |<===============================================|
      |                      |                         |
      |                      |                         |
      |                      |                         |
      |                      |                         |  SIP INVITE
      |                      |     Push Message        |<============
      |                      |   (Push Resource ID)    |
      |    Push Message      |<------------------------|
      |  (Push Resource ID)  |                         |
      |<---------------------|                         |
      |                      |                         |
      |          SIP REGISTER (Push Resource ID)       |
      |===============================================>|
      |                      |                         | SIP REGISTER
      |                      |                         |============>
      |                      |                         |
      |                      |                         | SIP 200 OK
```

```
        |                     |                        |<============
        |          SIP 200 OK |                        |
        |<================================================|
        |                     |                        |
        |          SIP INVITE |                        |
        |<================================================|
        |                     |                        |


        ------- Push Notification API

        ======= SIP

    REGISTER sip:alice@example.com SIP/2.0
    Via: SIP/2.0/TCP alicemobile.example.com:5060;branch=z9hG4bKnashds7
    Max-Forwards: 70
    To: Alice <sip:alice@example.com>
    From: Alice <sip:alice@example.com>;tag=456248
    Call-ID: 843817637684230@998sdasdh09
    CSeq: 1826 REGISTER
    Contact: <sip:alice@alicemobile.example.com;
      pn-provider=acme;
      pn-param=acme-param;
      pn-prid=ZTY4ZDJlMzODE1NmUgKi0K>
    Expires: 7200
    Content-Length: 0
```

Figure 1: SIP Push Notification Architecture

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Push Resource ID (PRID)

When a SIP UA registers with a PNS it receives a unique Push Resource ID (PRID), which is a value associated with the registration that can be used to generate push notifications.

The format of the PRID may vary depending on the PNS.

The details regarding discovery of the PNS, and the procedures regarding the push notification registration and maintenance are

outside the scope of this document. The information needed to contact the PNS is typically pre-configured in the operating system of the device.

4. SIP User Agent (UA) Behavior

4.1. Request Push Notifications from Network

Once the SIP UA has registered with the PNS, has received the PRID (using the protocol and procedures associated with the PNS), if the UA wants to receive push notifications (requested by the proxy), the UA MUST include the following SIP URI parameters in the SIP Contact header field URI of the REGISTER request: pn-provider, pn-prid and pn-param (if required for the specific PNS). The pn-provider URI parameter identifies the type of PNS, the pn-prid URI parameter contains the PRID value and the pn-param URI parameter contains additional PNS-specific information.

When the UA receives a 2xx response to the REGISTER request, if the response contains a Feature-Caps header field [RFC6809] with a 'sip.pns' feature-capability indicator with a parameter value identifying the same type of PNS that was identified by the pn-provider URI parameter in the REGISTER request, the UA can assume that a SIP proxy will request push notifications towards the UA. In other cases, the UA MUST NOT assume that push notifications will be requested, and the actions taken by the UA might be dependent on implementation or deployment architecture, and are outside the scope of this document.

In addition, if the response contains a Feature-Caps header field with a 'sip.vapid' feature-capability indicator, the proxy supports use of the Voluntary Application Server Identification (VAPID) mechanism [RFC8292] to restrict push notifications to the proxy.

NOTE: The VAPID specific procedures of the SIP UA are outside the scope of this document.

When the UA receives a push notification, it MUST send a binding refresh REGISTER request, using normal SIP procedures. If there are Network Address Translators (NATs) between the UA and the proxy, the REGISTER request will create NAT bindings that will allow incoming SIP requests to reach the UA. Once the UA has received a 2xx response to the REGISTER request, the UA might receive a SIP request for a new dialog (e.g., a SIP INVITE), or a stand-alone SIP request (e.g., a SIP MESSAGE), if such SIP request triggered the push notification request. Note that, depending on which transport protocol is used, the SIP request might reach the UA before the REGISTER response.

If the SIP UA has created multiple bindings (e.g., one for IPv4 and one for IPv6), the UA MUST send a binding refresh REGISTER request for each of those bindings when it receives a push notification.

If the UA is able to send binding refresh REGISTER requests using a non-push mechanism (e.g., using an internal timer that periodically wakes the UA), the UA MUST insert a 'sip.pnsreg' media feature tag [RFC3840] in the Contact header field URI of each REGISTER request. Then, if the response to the REGISTER request contains a 'sip.pnsreg' feature-capability indicator with an indicator value, the UA MUST send REGISTER requests prior to the registration expires. The indicator value indicates a minimum time (given in seconds), prior to the registration expires when the UA MUST send the REGISTER request. Even if the UA is able to to send REGISTER requests using a non-push mechanism, the UA MUST still send a REGISTER request when it receives a push notification, following the procedures in this section. If the REGISTER response does not contain a a 'sip.pnsreg' feature-capability indicator, the UA SHOULD only send a re-registration REGISTER request when it receives a push notification (even if the UA is able to use a non-push mechanism for sending re-registration REGISTER requests), or when there are circumstances (e.g., if the UA is assigned new contact parameters due to a network configuration change) that require an immediate REGISTER request to be sent.

NOTE: In some cases the UA might be able to use a non-push mechanism to wake and send binding refresh REGISTER requests. Such REGISTER request will update the registration expiration timer, and the proxy does not need to request a push notification towards the UA in order to wake the UA. The proxy will still request a push notification towards the UA when the proxy receives a SIP request addressed towards the UA (Section 5.3.2). This allows the UA to e.g., use timers for sending binding refresh REGISTER requests, but to be suspended (in order to save battery resources etc) between sending the REGISTER requests and use push notification to wake the UA to process incoming calls.

NOTE: This specification does not define any usage of a push notification payload. As defined in Section 5.3.2, a proxy must not include any payload in the push notification request. If a SIP UA receives a push notification that contains a payload the UA can discard the payload, but the UA will still send a binding refresh REGISTER request.

NOTE: If the SIP UA application wants to use push notifications for other purposes than to trigger binding refresh requests, it needs to be able to distinguish between the different purposes when receiving push notifications. Mechanisms for doing that are outside the scope of this specification.

As long as the UA wants to receive push notifications (requested by the proxy), the UA MUST include a pn-provider, pn-prid and a pn-param (if required for the specific PNS provider) SIP URI parameter in each binding refresh REGISTER request. Note that, in some cases, the PNS might update the PRID value, in which case the UA will include the new value in the pn-prid SIP URI parameter in the binding refresh REGISTER request.

If the UA no longer wants to receive push notifications (requested by the proxy), the UA MUST send a binding refresh REGISTER request without including the SIP URI parameters described above, or the UA MUST remove the registration.

If a UA's subscription to a PNS (and the associated PRID) is only valid for a limited time then the UA is responsible for retrieving new subscription details from the PNS and sending a REGISTER request with the updated pn parameters to the proxy prior to the expiry of the existing subscription. If a UA's subscription to a PNS expires then the UA MUST send a REGISTER without the pn parameters (to tell the proxy that it no longer wants push notifications) or terminate the registration. The exact mechanisms for expiry and renewal of PRIDs will be PNS-specific and are outside the scope of this document.

For privacy and security reasons, the UA MUST NOT include the SIP URI parameters defined in this document in non-REGISTER request, to prevent the PNS information associated with the UA from reaching the remote peer. For example, the UA MUST NOT include the SIP URI parameters in the Contact header field of an INVITE request.

4.2. Query Network Push Notification Capabilities

A SIP UA might need to query the network for push notification capabilities (e.g., related to VAPID) before it registers with the PNS, and before it indicates that it wants to receive push notifications. The UA can do so by only including the pn-provider SIP URI parameter in the SIP Contact header field URI of the REGISTER request, but without including the pn-prid SIP URI parameter. Later, once the UA has received a response to the REGISTER request, with the push notification information from the network, if the UA wants to receive push notifications, the UA will send a binding refresh REGISTER request, including all pn- SIP URI parameters required by the specific PNS, following the procedures in Section 4.1.

5. SIP Proxy Behavior

5.1. PNS Provider

The type of PNS is identified by the pn-provider SIP URI parameter.

The protocol and format used for the push notification requests are PNS-specific, and the details for constructing and sending a push notification request are outside the scope of this specification.

5.2. Trigger Periodic Binding Refresh

In order to request push notifications towards a SIP UA that will trigger the UA to send binding refresh SIP REGISTER requests, the SIP proxy needs to have information about when a registration will expire. The proxy needs to be able to retrieve the information from the registrar using some mechanism. Such mechanisms are outside the scope of this document.

When the proxy receives an indication that the UA needs to send a binding refresh REGISTER request, the proxy requests a push notification towards the UA.

Note that the push notification needs to be requested early enough, in order for the associated binding refresh REGISTER request to reach the registrar before the registration expires. It is RECOMMENDED that the proxy requests the push notification at least 120 seconds before the registration expires.

If the UA has indicated, using the 'sip.pnsreg' media feature tag, that it is able to wake using a non-push mechanism for sending binding refresh REGISTER requests, if the proxy does not receive a REGISTER request prior to 120 seconds before the registration expires, the proxy MAY request a push notification towards the UA, to trigger the UA to send a REGISTER request.

NOTE: As described in Section 4.2, a SIP UA might send a REGISTER request without including a pn-prid SIP URI parameter, in order to retrieve push notification capabilities from the network before the UA expects to receive push notifications from the network. A proxy will not request push notifications towards a UA that has not provided a pn-prid SIP URI parameter.

If the proxy receives information that a registration has expired, the proxy MUST NOT request further push notifications towards the UA.

5.3. SIP Requests

5.3.1. REGISTER

The procedures in this section apply when the SIP proxy receives a SIP REGISTER request (initial REGISTER request for a registration, or a binding refresh REGISTER request) that contains a pn-provider SIP URI parameter identifying a type of PNS. If the proxy receives a REGISTER request that does not contain a pn-provider SIP URI parameter, or that removes the registration, the proxy MUST NOT request push notifications towards the UA associated with the REGISTER request, or perform any other procedures in this section.

As described in Section 4.2, a SIP UA might send a REGISTER request without including a pn-prid SIP URI parameter in the Contact header field URI, in order to retrieve push notification capabilities from the network before the UA the proxy will request push notifications towards the UA. The procedures in this section apply to both the case when the REGISTER request contains a pn-prid SIP URI parameter, and when it does not.

When the proxy receives a REGISTER request, if the REGISTER request contains a Feature-Caps header field with a 'sip.pns' feature-capability indicator, it indicates that an upstream proxy supports, and will request (if the Contact header field URI of the REGISTER request contains a pn-prid SIP URI parameter), push notifications towards the UA. The proxy MUST skip the rest of the procedures in this section, and process the REGISTER request using normal SIP procedures.

If the proxy considers the requested registration expiration interval [RFC3261] to be too short, the proxy MUST either send a 423 (Interval Too Brief) response to the REGISTER request, or skip the rest of the procedures in this section and process the REGISTER request using normal SIP procedures. If the proxy sends a 423 (Interval Too Brief) response, the proxy SHOULD insert a Feature-Caps header field with a 'sip.pns' feature-capability indicator in the response, identifying each type of PNS that the proxy supports. Similarly, when the proxy receives a 2xx response to the REGISTER request (see below), if the proxy considers the registration expiration interval indicated by the registrar too short, the proxy MUST NOT insert a Feature-Caps header field with a 'sip.pns' feature-capability indicator in the response, and the proxy MUST NOT request push notifications associated with the registration. A registration expiration interval MUST be considered too short if the interval is smaller than the time prior to expiration that the proxy would request a push notification. The proxy MAY consider the interval too small based on its own policy so as to reduce load on the system.

Otherwise, if the pn-provider SIP URI parameter identifies a type of PNS that the proxy does not support, or if the REGISTER request does not contain all additional information required for the specific type of PNS, the proxy MUST either forward the request (e.g., if the proxy knows that a downstream proxy supports the type of PNS) or send a SIP 555 (Push Notification Service Not Supported) response to the REGISTER request. If the proxy sends a SIP 555 (Push Notification Service Not Supported) response Section 7.1, the proxy SHOULD insert a Feature-Caps header field with a 'sip.pns' feature-capability indicator in the response, identifying the type of each PNS that the proxy supports. The decision whether to forward the request, or to send a response, is done based on local policy.

If the proxy supports the type of PNS identified by the pn-provider SIP URI parameter, the proxy MUST insert a Feature-Caps header field with a 'sip.pns' feature-capability indicator, identifying the type of PNS, in the REGISTER request before forwarding the REGISTER request towards the registrar. This will inform downstream proxies that the proxy supports, and will request (if the Contact header field URI of the REGISTER request contains a pn-prid SIP URI parameter), push notifications towards the UA.

If the proxy inserted a Feature-Caps header field with a 'sip.pns' feature-capability indicator in the REGISTER request (see above), if the proxy receives a 2xx response to the REGISTER request, the proxy MUST insert a Feature-Caps header field with a 'sip.pns' feature-capability indicator in the response, identifying the type of PNS. This will inform the UA that the proxy supports, and will request (if the Contact header field URI of the REGISTER request contained a pn-prid SIP URI parameter), push notifications towards the UA. The proxy MUST only indicate support of the same PNS that was identified in the pn-provider SIP URI parameter in the REGISTER request. If the proxy receives a receives a SIP 555 (Push Notification Service Not Supported) response, the proxy SHOULD insert a Feature-Caps header field with a 'sip.pns' feature-capability indicator in the response, identifying the type of each PNS that the proxy supports.

In addition, if the proxy receives a 2xx response to the REGISTER request:

- o if the proxy supports, and will use, the VAPID mechanism, the proxy MUST insert a Feature-Caps header field with a 'sip.vapid' feature-capability indicator in the response. The header field parameter contains the public key identifying the proxy [RFC8292]. The proxy MUST be able to determine whether the PNS supports the VAPID mechanism before it inserts the feature-capability indicator.

- o if the proxy received a 'sip.pnsreg' media feature tag in the REGISTER request, the proxy SHOULD include a 'sip.pnsreg' feature-capability indicator with an indicator value bigger than 120 in the response, unless the proxy always want to request push notifications to trigger the UA to send a REGISTER request.

5.3.2. Initial Request for Dialog or Stand-Alone Request

The procedures in this section apply when the SIP proxy has indicated that it supports, and will request, push notifications towards the SIP UA.

When the proxy receives a SIP request for a new dialog (e.g., a SIP INVITE request) or a stand-alone SIP request (e.g., a SIP MESSAGE request) addressed towards a SIP UA, if the Request-URI of the request contains a pn-provider, a pn-prid and a pn-param (if required for the specific PNS provider) SIP URI parameter, the proxy requests a push notification towards the UA, using the PRID included in the pn-prid SIP URI parameter and the PNS identified by the pn-provider SIP URI parameter.

The push notification will trigger the UA to send a binding refresh REGISTER request. The proxy will process the REGISTER request and the associated response as described in Section 5.3.1. In case of a 2xx response to the REGISTER request, once the proxy has forwarded the REGISTER response towards the UA, if the contact of the SIP REGISTER request associated with the REGISTER response matches the Request-URI of the SIP request to be forwarded, and the contact was also present (and has not expired) in the REGISTER response, the proxy can forward the SIP request towards the UA, using normal SIP procedures. If the contact of the REGISTER request does not match the Request-URI of the SIP request to be forwarded, or if the contact was not present in the REGISTER response, the proxy MUST reject the SIP request with a 404 (Not Found) response. This can happen if the UA sends a binding refresh REGISTER request with a new contact at the same time the registrar forwards a SIP request towards a UA using the previously registered contact in the Request-URI.

When matching the Request-URI of the SIP request to be forwarded with a contact of a REGISTER request, the proxy uses the URI comparison rules in [RFC8292], with the following addition: the pn-prid SIP URI parameter MUST also match. If the parameter is not present in the Request-URI of the SIP request, or in the contact of the REGISTER, there is no match.

The reason the proxy needs to wait for the REGISTER response before forwarding the SIP request is to make sure that the REGISTER request has been accepted by the registrar, and that the UA which initiated

the REGISTER request is authorized to receive messages for the Request-URI. However, if the proxy is able to authorize the sender of the REGISTER request, it does not need to wait for the associated 2xx response before it forwards the SIP request towards the UA. The mechanism for authorizing the UA is outside the scope of this document.

As both the SIP request to be forwarded towards the UA, and the binding refresh REGISTER request triggered by the push notification request, will convey pn- SIP URI parameters associated with the SIP registration, those can be used to match the SIP request with the binding refresh REGISTER request (even if the most recent contact and the Request-URI of the SIP request do not match).

NOTE: The proxy needs to store (or be able to retrieve) the contact of the most recent REGISTER 2xx response, to be able to compare it with the Request-URI of the request to be forwarded towards the UA.

In case of non-2xx response to the REGISTER request, the proxy MUST reject the SIP request with a 404 (Not Found) response.

If the push notification request fails (see PNS-specific documentation for details), the proxy MUST reject the SIP request with a 480 (Temporarily Unavailable) or a 556 (Push Notification Failed) response.

NOTE; Before sending a 556 (Push Notification Failed) response, the proxy operator needs to determine whether it could have privacy implications.

If the proxy does not receive the REGISTER request from the UA within a given time after the proxy has requested the push notification, the proxy MUST reject the request with a 480 (Temporarily Unavailable) response. The time value is set based on local policy.

As dicussed in [RFC4320] and [RFC4321], non-INVITE transactions must complete immediately or risk losing race that results in stress on intermediaries and state misalignment at the endpoints. The mechanism defined in this document inherently delays the final response to any non-INVITE request that requires a push notification. In particular, while waiting for the push notification request to succeed, and the associated REGISTER request to arrive from the SIP UA, the proxy needs to take into consideration that the transaction associated with the SIP request will eventually time out at the sender of the request (UAC), and the sender will consider the transaction a failure. If the proxy forwards the SIP request towards the SIP UA, the SIP UA accepts the request and the transaction times out at the sender before it receives the successful response, this

will cause state misalignment between the endpoints (the sender will consider the transaction a failure, while the receiver will consider the transaction a success). The SIP proxy needs to take this into account when deciding for how long to wait before it considers the transaction associated with the SIP request a failure, to make sure that the error response reaches the sender before the transaction times out. If the accumulated delay of this mechanism combined with any other mechanisms in the path of processing the non-INVITE transaction is not kept short, this mechanism should not be used. For networks encountering such conditions, an alternative (left for possible future work) would be for the proxy to immediately return an new error code meaning "wait at least the number of seconds specified in this response, and retry your request" before initiating the push notification.

NOTE: While this work on this document was ongoing, implementation test results showed that the time it takes for a proxy to receive the REGISTER request, from when the proxy has requested a push notification, is typically around 2 seconds. However, the time might vary depending on the characteristics and load of the SIP network and the PNS.

The proxy MUST NOT include the SIP request as payload in the requested push message.

If the proxy has knowledge that the UA is wake, and that the UA is able to receive the SIP request without first sending a REGISTER request, the proxy MAY choose to not request a push notification towards the UA (and wait for the associated REGISTER request and 2xx response) before it tries to forward the SIP request towards the UA. The mechanisms for getting such knowledge might be dependent on implementation or deployment architecture, and are outside the scope of this document. Similarly, if the Request-URI of the SIP request only contains any pn-provider SIP URI parameter, but no other pn- SIP URI parameters, e.g., because the SIP UA has not included them in a REGISTER request (Section 4.2), the proxy is not able to request a push notification towards the UA. If the proxy has knowledge that the UA is wake, and that the UA is able to receive the SIP request, the proxy MAY forwards the request towards the UA. Otherwise the proxy MUST reject the SIP request with a 480 (Temporarily Unavailable) or a 556 (Push Notification Failed) response.

6. Support Of Longlived SIP Dialogs

Some SIP dialogs might have a long lifetime, with little activity. For example, when the SIP event notification mechanism [RFC6665] is used, there might be a long period between mid-dialog SIP NOTIFY requests are sent. Because of this a SIP UA might get suspended, and

needs to be awaken in order to be able to receive mid-dialog requests.

When the proxy receives a SIP request for a new dialog, or a stand-alone SIP request, addressed towards a UA, the request will contain information (pn- SIP URI parameters) that allows proxy to request push notifications towards the UA Section 5.3.2. However, this information will not be present in mid-dialog requests towards the UA. Instead, the proxy need to support a mechanism where it stores the information needed to request a push notification towards the UA and be able to find and retrieve that information when it receives a mid-dialog request. This section defines such mechanism. The UA and proxy procedures in this section are applied in addition to the generic procedures defined in this specification.

6.1. SIP UA Behavior

6.1.1. Initial Request for Dialog

When the UA sends in initial request for a dialog, or a 2xx response to such requests, if the UA is willing to receive push notifications triggered by incoming mid-dialog requests, the UA MUST include a 'pn-purk' SIP URI parameter in the Contact header of the request or response. The UA MUST include a parameter value identical to the the last 'sip.pnspurk' feature-capability indicator that it received in a REGISTER response.

The UA decision whether it is willing to receive push notifications triggered by incoming mid-dialog requests is done based on local policy. Such policy might be based on the type of SIP dialog, the type of media (if any) negotiated for the dialog [RFC3264], etc.

6.2. SIP Proxy Behavior

6.2.1. REGISTER

When the proxy receives an initial REGISTER request for a registration from the UA, if the proxy supports requesting push notifications, triggered by mid-dialog requests, towards the registered UA, the proxy MUST store the information (the pn- SIP URI parameters) needed to request push notifications associated with the registration towards the UA. In addition the proxy MUST generate a unique (within the context of the proxy) value, referred to as the PURK (Proxy Unique Registration Key), that is unique within the context of the proxy and that can be used as a key to retrieve the information. When the proxy receives the associated 2xx REGISTER response, it adds a 'sip.pnspurk' feature-capability indicator with the PURK value to the associated 2xx REGISTER response. When the

proxy receives a binding refresh REGISTER request, it MUST add a 'sip.pnspurk' feature-capability indicator with the previously generated key as the value to the associated 2xx REGISTER response.

The PURK value MUST be generated in such a way so that it cannot be used to retrieve information about the user or associate it with registrations. It can be generated e.g., by utilizing a cryptographically secure random function.

In order to prevent client fingerprinting, the proxy MUST periodically generate a new PURK value (even if pn- parameters did not change) and provide the new value to the UA in a 2xx binding refresh REGISTER response. However, as long as there are ongoing dialogs associated with the old value, the proxy MUST store it so that it can request push notifications towards the UA when it receives a mid-dialog request addressed towards the UA.

6.2.2. Initial Request for Dialog

When the proxy receives an initial request for a dialog from the UA, and if the request contains a 'pn-purk' SIP URI parameter in the Contact header field with a PURK value that the proxy has generated Section 6.2.2, the proxy MUST add a Record-Route header to the request, to insert itself in the dialog route [RFC3261].

When the proxy receives an initial request for a dialog addressed towards the UA, and if the proxy has generated a PURK value associated with the pn- parameters included in the SIP-URI of the request Section 6.2.2, the proxy MUST add a Record-Route header to the request, to insert itself in the dialog route [RFC3261].

6.2.3. Mid-Dialog Request

When the proxy receives a mid-dialog request addressed towards the UA, if the request contains a 'pn-purk' SIP URI parameter and if the proxy is able to retrieve the stored information needed to request a push notification towards the UA (Section 6.2.1), the proxy MUST request a push notification towards the UA. Once the proxy has received the triggered REGISTER request, and the associated successful response, the proxy can forward the mid-dialog request towards the UA.

As described in Section 5.3.2, while waiting for the push notification request to succeed, and the associated REGISTER request to arrive from the SIP UA, the proxy needs to take into consideration that the transaction associated with the NOTIFY request will eventually time out at the sender of the request (UAC), and the sender will consider the transaction a failure.

7. Grammar

7.1. 555 (Push Notification Service Not Supported) Response Code

The 555 response code is added to the "Server-Error" Status-Code definition. 555 (Push Notification Service Not Supported) is used to indicate that the server did not support the push notification service identified in a 'pn-provider' SIP URI parameter.

The use of the SIP 555 response code is defined for SIP REGISTER responses.

7.2. 556 (Push Notification Failed) Response Code

The 556 response code is added to the "Server-Error" Status-Code definition. 556 (Push Notification Failed) is used to indicate that the server failed to request a push notification from the push notification service identified in a 'pn-provider' SIP URI parameter.

The use of the SIP 556 response code is defined for responses to SIP requests initiating dialogs, and responses to stand-alone SIP requests.

7.3. sip.pns Feature-Capability Indicator

The sip.pns feature-capability indicator, when included in a Feature-Caps header field of a SIP REGISTER request or a SIP 2xx response to a REGISTER request, indicates that the entity associated with the indicator supports, and will use, the SIP push mechanism and the push notification service identified by the indicator value. When included in a 555 (Push Notification Service Not Supported) response to a REGISTER request, the the indicator indicates that the entity associated with the indicator supports the SIP push mechanism, and the push notification service(s) identified by the indicator value. The values defined for the pn-provider SIP URI parameter are used as indicator values.

```
pns-fc          = "+sip.pns" EQUAL LDQUOT pns-list RDQUOT
pns-list        = pns *(COMMA pns)
pns             = tag-value

; tag-value as defined in RFC 3840
```

7.4. sip.vapid Feature-Capability Indicator

The sip.vapid feature-capability indicator, when included in a SIP 2xx response to a SIP REGISTER request, indicates that the entity associated with the indicator supports, and will use, the Voluntary Application Server Identification (VAPID) [RFC8292] mechanism when requesting push notifications towards the SIP UA associated with the SIP registration. The indicator value is a public key identifying the entity, that can be used by a SIP UA to restrict subscriptions to that entity.

```
vapid-fc          = "+sip.vapid" EQUAL LDQUOT vapid RDQUOT
vapid             = tag-value

; tag-value as defined in RFC 3840
```

7.5. sip.pnsreg Feature-Capability Indicator

The sip.pnsreg feature-capability indicator, when included in a SIP 2xx response to a SIP REGISTER request, indicates that the entity associated with the indicator expects to receive binding refresh REGISTER requests from the SIP UA associated with the registration before the registration expires, without the entity having to request push notifications towards the SIP UA in order to trigger the REGISTER requests. The indicator value is the minimum value (given in seconds) before the registration expiration when the entity expects to receive the REGISTER request.

```
pns-fc          = "+sip.pnsreg" EQUAL LDQUOT reg RDQUOT
reg             = 1*DIGIT

; DIGIT as defined in RFC 3261
```

7.6. sip.pnsreg Media Feature Tag

The sip.pnsreg media feature tag, when included in the SIP Contact header field of a SIP REGISTER request, indicates that the SIP UA associated with the tag is able to send binding refresh REGISTER requests associated with the registration without being awaken by push notifications. The media feature tag has no values.

```
pnsreg-mt          = "+sip.pnsreg"
```

7.7. SIP URI Parameters

The section defines new SIP URI parameters, by extending the grammar for "uri-parameter" as defined in [RFC3261]. The ABNF is as follows:

```
uri-parameter   =/ pn-provider / pn-param / pn-prid
pn-provider     = "pn-provider" EQUAL pvalue
pn-param        = "pn-param" EQUAL pvalue
pn-prid         = "pn-prid" EQUAL pvalue

; pvalue as defined in RFC 3261
; EQUAL as defined in RFC 3261
; COLON as defined in RFC 3261
```

The format and semantics of pn-prid and pn-param are specific to the pn-provider value.

Parameter value characters that are not part of pvalue needs to be escaped, as defined in RFC 3261.

8. PNS Registration Requirements

When a new value is registered to the PNS Sub-registry, a reference to a specification which describes the PNS associated with the value is provided. That specification MUST contain the following information:

- o The value of the pn-provider SIP URI parameter.
- o How the pn-prid SIP URI parameter value is retrieved and set by the SIP UA.
- o How the pn-param SIP URI parameter (if required for the specific PNS provider) value is retrieved and set by the SIP UA.

9. pn-provider, pn-param and pn-prid URI Parameters for Apple Push Notification service

When the Apple Push Notification service (APNs) is used, the PNS-related SIP URI parameters are set as described below.

The value of the pn-provider URI parameter is "apns".

Example: pn-provider = apns

The value of the pn-param URI parameter is the APNs App ID, which is encoded by two values, separated by a period (.): Team ID and Bundle ID. The Team ID is provided by Apple and is unique to a development team. The Bundle ID is unique to a development team, and is a string that will match a single application or a group of applications.

Example: pn-param = DEF123GHIJ.com.yourcompany.yourexampleapp

The value of the pn-prid URI parameter is the device token, which is a unique identifier assigned by Apple to a specific app on a specific device.

Example: pn-prid = 00fc13adff78512

For more information on the APNs App ID:

https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/AppID.html

For more information on the APNs device token:

https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html#//apple_ref/doc/uid/TP40008194-CH8-SW13

10. pn-provider, pn-param and pn-prid URI Parameters for Google Firebase Cloud Messaging (FCM) push notification service

When Firebase Cloud Messaging (FCM) is used, the PNS related URI parameters are set as described below.

The value of the pn-provider URI parameter is "fcm".

The value of the pn-param URI parameter is the Project ID.

The value of the pn-prid URI parameter is the Registration token, which is generated by the FCM SDK for each client app instance.

For more information on the Project ID and Registration token:

https://firebase.google.com/docs/cloud-messaging/concept-options

11. pn-provider, pn-param and pn-prid URI Parameters for RFC 8030 (Generic Event Delivery Using HTTP Push)

When Generic Event Delivery Using HTTP Push is used, the PNS related URI parameters are set as described below.

The value of the pn-provider URI parameter is "webpush".

The value of the pn-param URI parameter MUST NOT be used.

The value of the pn-prid URI parameter is the push subscription URI.

See RFC 8030 for more details:

Note that encryption for web push [RFC8291] is not used, therefore parameters for message encryption are not defined in this specification. Web push permits the sending of a push message without a payload without encryption.

12. Security Considerations

Different mechanisms exist for authenticating and authorizing devices and users registering with a PNS. The mechanisms for authorizing and authenticating the users are PNS-specific, and are outside the scope of this document. In addition to the information that needs to be exchanged between a device and the PNS in order to establish a push notification subscription, the mechanism defined in this document does not require any additional information to be exchanged between the device and the PNS.

Typically, the PNS also requires the SIP proxy requesting push notifications to be authenticated and authorized by the PNS. In some cases the PNS also require the SIP application (or the SIP application developer) to be identified in order for the application to request push notifications.

Operators MUST ensure that the SIP signalling is properly secured, e.g., using encryption, from malicious middlemen, unless they are sure that the signalling cannot be accessed and used maliciously (e.g., to trigger push notifications towards a device) by a middleman.

[RFC8292] defines a mechanism which allows a proxy to identity itself to a PNS, by signing a JWT sent to the PNS using a key pair. The public key serves as an identifier of the proxy, and can be used by devices to restrict push notifications to the proxy associated with the key.

The mechanism in this document does not require a proxy to include any payload (in addition to possible payload used for the PNS itself) when requesting push notifications.

13. IANA considerations

13.1. SIP URI Parameters

This section defines new SIP URI Parameters that extend the "SIP/SIPS URI Parameters" sub-registry [RFC3969] under the sip-parameters registry: http://www.iana.org/assignments/sip-parameters.

13.1.1. pn-provider

Parameter Name: pn-provider

Predefined Values: No

Reference: RFC XXXX

13.1.2. pn-param

Parameter Name: pn-param

Predefined Values: No

Reference: RFC XXXX

13.1.3. pn-prid

Parameter Name: pn-prid

Predefined Values: No

Reference: RFC XXXX

13.2. SIP Response Codes

13.2.1. 555 (Push Notification Service Not Supported)

This section defines a new SIP response code that extends the "Response Codes" sub-registry [RFC3261] under the sip-parameters registry: http://www.iana.org/assignments/sip-parameters.

Response Code Number: 555

Default Reason Phrase: Push Notification Service Not Supported

13.2.2. 556 (Push Notification Failed)

This section defines a new SIP response code that extends the "Response Codes" sub-registry [RFC3261] under the sip-parameters registry: http://www.iana.org/assignments/sip-parameters.

Response Code Number: 556

Default Reason Phrase: Push Notififcation Failed

13.3. SIP Global Feature-Capability Indicator

13.3.1. sip.pns

This section defines a new feature-capability indicator that extends the "SIP Feature-Capability Indicator Registration Tree" sub-registry [RFC6809] under the sip-parameters registry: http://www.iana.org/assignments/sip-parameters.

Name: sip.pns

Description: This feature-capability indicator, when included in a Feature-Caps header field of a SIP REGISTER request or a SIP 2xx response to a REGISTER request, indicates that the entity associated with the indicator supports, and will use, the SIP push mechanism and the push notification service identified by the indicator value. When included in a 555 (Push Notification Service Not Supported) response to a REGISTER request, the the indicator indicates that the entity associated with the indicator supports the SIP push mechanism, and the push notification service(s) identified by the indicator value.

Reference: [RFCXXXX]

Contact: IESG (iesg@ietf.org)

13.3.2. sip.vapid

This section defines a new feature-capability indicator that extends the "SIP Feature-Capability Indicator Registration Tree" sub-registry [RFC6809] under the sip-parameters registry: http://www.iana.org/assignments/sip-parameters.

Name: sip.vapid

Description: This feature-capability indicator, when included in a SIP 2xx response to a SIP REGISTER request, indicates that the entity associated with the indicator supports, and will use, the Voluntary Application Server Identification (VAPID) mechanism when requesting push notifications towards the SIP UA associated with the SIP registration. The indicator value is a public key identifying the entity, that can be used by a SIP UA to restrict subscriptions to that entity.

Reference: [RFCXXXX]

Contact: IESG (iesg@ietf.org)

13.3.3. sip.pnsreg

This section defines a new feature-capability indicator that extends the "SIP Feature-Capability Indicator Registration Tree" sub-registry [RFC6809] under the sip-parameters registry: http://www.iana.org/assignments/sip-parameters.

Name: sip.pnsreg

Description: This feature-capability indicator, when included in a SIP 2xx response to a SIP REGISTER request, indicates that the entity associated with the indicator expects to receive binding refresh REGISTER requests from the SIP UA associated with the registration before the registration expires, without the entity having to request push notifications towards the SIP UA in order to trigger the REGISTER requests. The indicator value is the minimum value (given in seconds) before the registration expireation when the entity expects to receive the REGISTER request.

Reference: [RFCXXXX]

Contact: IESG (iesg@ietf.org)

13.4. SIP Media Feature Tag

13.4.1. sip.pnsreg

This section defines a new media feature tag that extends the "SIP Media Feature Tag Registration Tree" sub-registry [RFC3840] under the Media Feature Tag registry: https://www.iana.org/assignments/media-feature-tags/media-feature-tags.xhtml.

Media feature tag name: sip.pnsreg

Summary of the media feature indicated by this feature tag: This media feature tag, when included in the SIP Contact header field of a SIP REGISTER request, indicates that the SIP UA associated with the tag is able to send binding refresh REGISTER requests associated with the registration without being awaken by push notifications.

Values appropriate for use with this feature tag: none

Related standards or documents: [RFCXXXX]

Security considerations: This media feature tag does not introduce new security considerations, as it simply indicates support for a basic SIP feature. If an attacker manages to remove the media feature tag, push notifications towards the client will be requested.

Contact: IESG (iesg@ietf.org)

13.5. PNS Sub-registry Establishment

This section creates a new sub-registry, "PNS", under the sip-parameters registry: http://www.iana.org/assignments/sip-parameters.

The purpose of the sub-registry is to register SIP URI pn-provider values.

When a SIP URI pn-provider value is registered in the sub-registry, it needs to meet the "Specification Required" policies defined in [RFC8126].

This sub-registry is defined as a table that contains the following three columns:

Value: The token under registration

Description: The name of the Push Notification Service (PNS)

Document: A reference to the document defining the registration

This specification registers the following values:

Value	Description	Document
apns	Apple Push Notification service	[RFC XXXX]
fcm	Firebase Cloud Messaging	[RFC XXXX]
webpush	Generic Event Delivery Using HTTP Push	[RFC XXXX]

14. Acknowledgements

Thanks to Mickey Arnold, Paul Kyzivat, Dale Worley, Ranjit Avasarala, Martin Thomson, Mikael Klein, Susanna Sjoholm, Kari-Pekka Perttula, Liviu Chircu, Roman Shpount and Yehoshua Gev for reading the text, and providing useful feedback.

15. References

15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, DOI 10.17487/RFC3840, August 2004, <https://www.rfc-editor.org/info/rfc3840>.

[RFC3969] Camarillo, G., "The Internet Assigned Number Authority (IANA) Uniform Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)", BCP 99, RFC 3969, DOI 10.17487/RFC3969, December 2004, <https://www.rfc-editor.org/info/rfc3969>.

[RFC6809] Holmberg, C., Sedlacek, I., and H. Kaplan, "Mechanism to Indicate Support of Features and Capabilities in the Session Initiation Protocol (SIP)", RFC 6809, DOI 10.17487/RFC6809, November 2012, <https://www.rfc-editor.org/info/rfc6809>.

[RFC8030] Thomson, M., Damaggio, E., and B. Raymor, Ed., "Generic Event Delivery Using HTTP Push", RFC 8030, DOI 10.17487/RFC8030, December 2016, <https://www.rfc-editor.org/info/rfc8030>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8292] Thomson, M. and P. Beverloo, "Voluntary Application Server Identification (VAPID) for Web Push", RFC 8292, DOI 10.17487/RFC8292, November 2017, <https://www.rfc-editor.org/info/rfc8292>.

15.2. Informative References

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC4320] Sparks, R., "Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction", RFC 4320, DOI 10.17487/RFC4320, January 2006, <https://www.rfc-editor.org/info/rfc4320>.

[RFC4321] Sparks, R., "Problems Identified Associated with the Session Initiation Protocol's (SIP) Non-INVITE Transaction", RFC 4321, DOI 10.17487/RFC4321, January 2006, <https://www.rfc-editor.org/info/rfc4321>.

[RFC5626] Jennings, C., Ed., Mahy, R., Ed., and F. Audet, Ed., "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, DOI 10.17487/RFC5626, October 2009, <https://www.rfc-editor.org/info/rfc5626>.

[RFC6665] Roach, A., "SIP-Specific Event Notification", RFC 6665, DOI 10.17487/RFC6665, July 2012, <https://www.rfc-editor.org/info/rfc6665>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8291] Thomson, M., "Message Encryption for Web Push", RFC 8291, DOI 10.17487/RFC8291, November 2017, <https://www.rfc-editor.org/info/rfc8291>.

Authors' Addresses

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Michael Arnold
Metaswitch Networks
100 Church Street
Enfield EN2 6BQ
United Kingdom

Email: Michael.Arnold@metaswitch.com