SIPCORE Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 27, 2019

C. Holmberg
Ericsson
M. Arnold
Metaswitch Networks

March 26, 2019

Push Notification with the Session Initiation Protocol (SIP)
draft-ietf-sipcore-sip-push-29

Abstract

This document describes how a Push Notification Service (PNS) can be used to wake a suspended Session Initiation Protocol (SIP) User Agent (UA) with push notifications, and also describes how the UA can send binding-refresh REGISTER requests and receive incoming SIP requests in an environment in which the UA may be suspended. The document defines new SIP URI parameters to exchange PNS information between the UA and the SIP entity that will then request that push notifications be sent to the UA, and to trigger such push notification requests. The document also defines new feature-capability indicators that can be used to indicate support of this mechanism.

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 27, 2019.

Table of Contents

1. Introduction

In order to save resources such as battery life, some devices (especially mobile devices) and operating systems will suspend an application that is not in use. A suspended application might not be able to wake itself with internal timers and might not be awakened by incoming network traffic. In such an environment, a Push Notification Service (PNS) is used to wake the application. A PNS is a service that sends messages requested by other applications to a user application in order to wake the user application. These messages are called push notifications. Push notifications might contain payload data, depending on the application. An application can request that a push notification be sent to a single user application or to multiple user applications.

Typically each operating system uses a dedicated PNS. Different PNSs exist today. Some are based on the standardized mechanism defined in [RFC8030], while others are proprietary. For example, Apple iOS devices use the Apple Push Notification service (APNs) while Android

devices use the Firebase Cloud Messaging (FCM) service. 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.

When a Session Initiation Protocol (SIP) User Agent (UA)[RFC3261] is suspended in such an environment, it is unable to send binding-refresh SIP REGISTER requests, unable to receive incoming SIP requests, and might not be able to use internal timers to wake itself. A suspended UA will not be able to maintain connections e.g., using the SIP Outbound Mechanism [RFC5626] because it cannot send periodic keep-alive messages. A PNS is needed to wake the SIP UA so that the UA can perform these functions.

This document describes how a PNS can be used to wake a suspended UA, using push notifications, so that the UA can 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 that push notifications are sent to the UA, and to request such push notification requests.

NOTE: Even if a UA is able to be awakened by means other 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 UA 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 with a PNS (Figure 1), 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, which will then request that push notifications are sent to the UA.

When the SIP proxy receives a SIP request for a new dialog or a stand-alone SIP request addressed towards a UA, or when the SIP proxy determines that the UA needs to send a binding-refresh REGISTER request, the SIP proxy will send a push request containing the PRID of the UA to the PNS, which will then send a push notification to the UA. Once the UA receives the push notification, it will be able to send a binding-refresh REGISTER request. The proxy receives the REGISTER request from the UA and forwards it to the SIP registrar [RFC3261]. After accepting the REGISTER request, the SIP registrar sends a 2xx response to the proxy, which forwards the response to the UA. If the push notification request was triggered by a SIP request

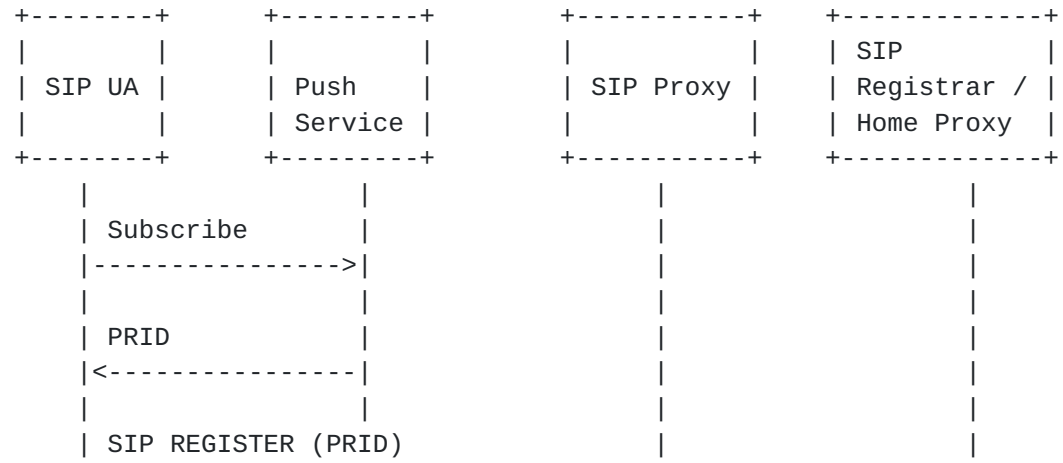
addressed towards the UA, the proxy can then forward the SIP request to the UA using normal SIP routing procedures. In some cases the proxy can forward the SIP request without waiting for the SIP 2xx response to the REGISTER request from the SIP registrar. Note that this mechanism necessarily adds delay to responding to requests requiring push notification. The consequences of that delay are discussed in Section 5.6.2.

If there are Network Address Translators (NATs) between the UA and the proxy, the REGISTER request sent by the UA will create NAT bindings that will allow the incoming SIP request that triggered the push notification to reach the UA.

NOTE: The lifetime of any NAT binding created by the REGISTER request only needs to be long enough in order for the SIP request that triggered the push notification to reach the UA.

Figure 1 shows the generic push notification architecture supported by the mechanism in this document.

The SIP proxy MUST be in the signalling path of REGISTER requests sent by the UA towards the registrar, and 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. 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.



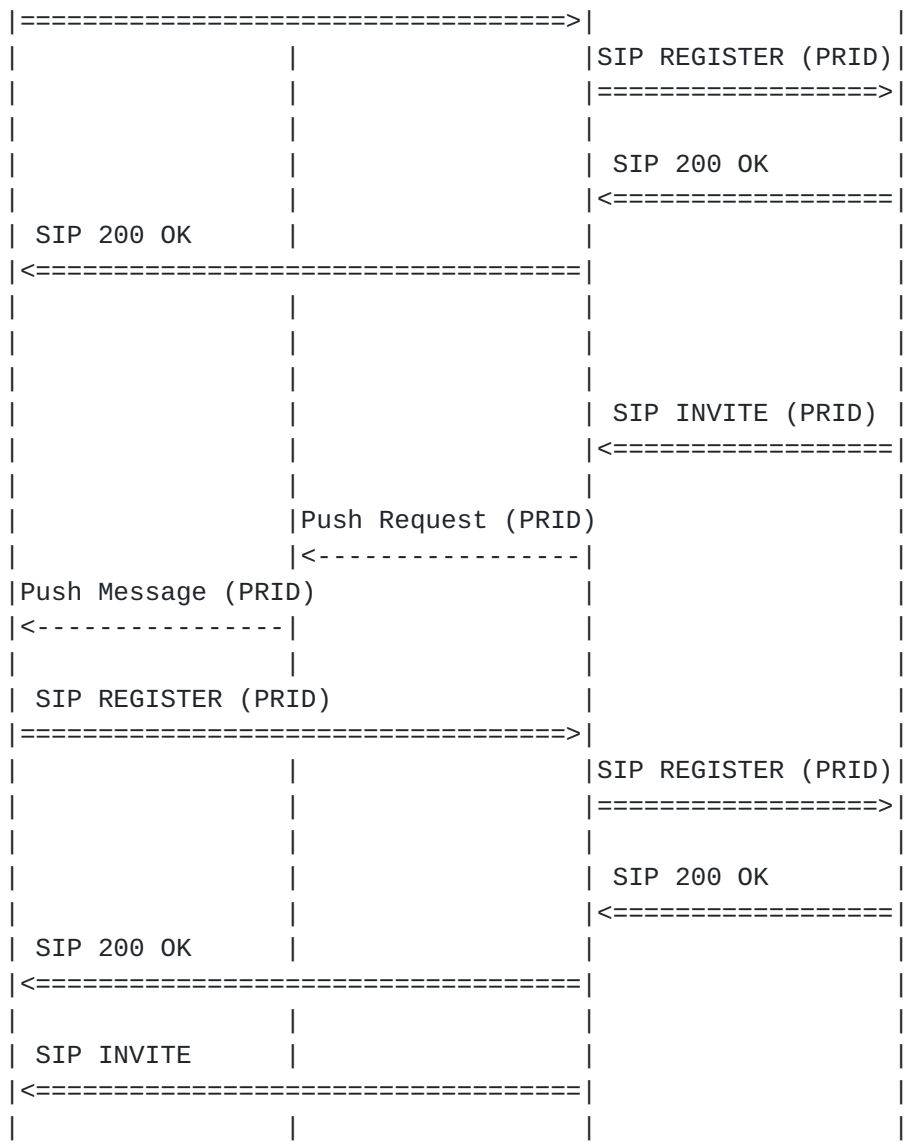


Figure 1: SIP Push Information Flow

Example of a SIP REGISTER request in the flow above:

```
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 2: SIP REGISTER Example

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 varies 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. REGISTER

This section describes how a SIP UA sends SIP REGISTER requests (either an initial REGISTER request for a binding or a binding-refresh REGISTER request) in order to request and disable push

notifications from a SIP network, and to query the types of PNSs supported by the SIP network.

Unless specified otherwise, the normal SIP UA registration procedures [RFC3261] apply. The additional procedures described in this section apply when the REGISTER request contains a pn-provider SIP URI parameter in the Contact header field URI (Figure 2).

The procedures in this section apply to individual bindings [RFC3261]. If a UA creates multiple bindings (e.g., one for IPv4 and one for IPv6) the UA needs to perform the procedures for each binding.

NOTE: Since a push notification will trigger the UA to refresh all bindings, if a SIP UA has created multiple bindings, it is preferable if one can ensure that all bindings expire at the same time to help prevent some bindings from being refreshed earlier than needed.

For privacy and security reasons, a UA MUST NOT insert the SIP URI parameters (except for the pn-purr parameter) defined in this specification in non-REGISTER requests in order to prevent the PNS information associated with the UA from reaching the remote peer. For example, the UA MUST NOT insert the pn-prid SIP URI parameter in the Contact header field URI of an INVITE request. REGISTER requests will not reach the remote peer, as they will be terminated by the registrar of the UA. However, the registrar MUST still ensure that the parameters are not sent to other users, e.g., using the SIP event package for registrations mechanism [RFC3680]. See Section 13 for more information.

4.1.1. Request Push Notifications

This section describes the procedures that a SIP UA follows to request push notifications from the SIP network. The procedures assume that the UA has retrieved a PRID from a PNS. The procedures for retrieving the PRID from the PNS are PNS-specific and outside the scope of this specification. See PNS-specific documentation for more details.

This specification does not define a mechanism to explicitly request push notifications from the SIP network for usages other than triggering binding-refresh REGISTER requests (e.g., for sending periodic subscription-refresh SUBSCRIBE requests [RFC6665]), nor does it describe how to distinguish push notifications associated with such usages from the push notifications used to trigger binding-refresh REGISTER requests. If a SIP UA wants to use push notifications for other usages, the UA can perform actions associated with such usages (in addition to sending a binding-refresh REGISTER

request) whenever it receives a push notification by using the same refresh interval that is used for the binding-refreshes.

To request push notifications from the SIP network, the UA MUST insert 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 indicates the type of PNS to be used for the push notifications.

If the UA receives a 2xx response to the REGISTER request that contains a Feature-Caps header field [RFC6809] with a 'sip.pns' feature-capability indicator, with an indicator value identifying the same type of PNS that was identified by the pn-provider URI parameter in the REGISTER request, it indicates that another SIP Proxy in the SIP network will request that push notifications are sent to the UA. In addition, if the same Feature-Caps header field contains a 'sip.vapid' feature-capability indicator, it indicates that the proxy supports use of the Voluntary Application Server Identification (VAPID) mechanism [RFC8292] to restrict push notifications to the UA.

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

If the UA receives a non-2xx response to the REGISTER, or if the UA receives a 2xx response that does not contain a Feature-Caps header field [RFC6809] with a 'sip.pns' feature-capability indicator, the UA MUST NOT assume the proxy will request that push notifications are sent to the UA. The actions taken by the UA in such cases are outside the scope of this document.

If the PRID is only valid for a limited time then the UA is responsible for retrieving a new PRID from the PNS and sending a binding-refresh REGISTER request with the updated pn- parameters. If a PRID is no longer valid, and the UA is not able to retrieve a new PRID, the UA MUST disable the push notifications associated with the PRID (Section 4.1.2).

4.1.2. Disable Push Notifications

When a UA wants to disable previously requested push notifications, the UA SHOULD remove the binding [RFC3261], unless the UA is no longer able to perform SIP procedures (e.g., due to a forced shutdown of the UA) in which case the registrar will remove the binding once it expires. When the UA sends the REGISTER request for removing the binding, the UA MUST NOT insert the pn-prid SIP URI parameter in the Contact header field URI of the REGISTER request, in order to inform the SIP network that the UA no longer wants to receive push notifications associated with the PRID.

4.1.3. Receive Push Notifications

When a UA receives a push notification, the UA MUST send a binding-refresh REGISTER request. The UA MUST insert the same set of pn- SIP URI parameters in the SIP Contact header field URI of the REGISTER request that it inserted when it requested push notifications (Section 4.1.1). Note that, in some cases the PNS might update the PRID value, in which case the UA will insert the new value in the pn-prid SIP URI parameter of the binding-refresh REGISTER request.

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 proxy to request that the push notification was sent to the UA. 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, the UA MUST send a binding-refresh REGISTER request for each of those bindings when it receives a push notification.

This specification does not define any usage of push notification payload. 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.

4.1.4. Sending Binding-Refresh Requests Using Non-Push Mechanism

If a 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 of each REGISTER request.

If the UA receives a 2xx response to the REGISTER request that contains a Feature-Caps header field with a 'sip.pnsreq' feature-capability indicator, the UA MUST send a binging-refresh REGISTER request prior to binding expiration. The indicator value indicates the minimum time (given in seconds), prior to the binding expiration when the UA needs to send the REGISTER request.

If the UA receives a 2xx response to the REGISTER request that does not contain a Feature-Caps header field with a 'sip.pnsreq' feature-capability indicator, the UA SHOULD only send a binding-refresh REGISTER request when it receives a push notification (even if the UA is able to use a non-push mechanism for sending binding-refresh 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.

Even if the UA is able to to send binding-refresh REGISTER requests using a non-push mechanism, the UA MUST still send a binding-refresh REGISTER request whenever it receives a push notification (Section 4.1.3).

NOTE: If the UA uses a non-push mechanism to wake and send binding-refresh REGISTER requests, such REGISTER requests will update the binding expiration timer, and the proxy does not need to request that a push notification be sent to the UA in order to wake the UA. The proxy will still request that a push notification be sent to the UA when the proxy receives a SIP request addressed towards the UA (Section 5.6.2). 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.

Example of a SIP REGISTER request including a 'sip.pnsreg' media feature tag:

```
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>;
  +sip.pnsreg
Expires: 7200
Content-Length: 0
```

Example of a SIP REGISTER response including a 'sip.pnsreg' media feature tag and a 'sip.pnsreq' feature-capability indicator:

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP alicemobile.example.com:5060;branch=z9hG4bKnashds7
To: Alice <sip:alice@example.com>;tag=123987
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>;
  +sip.pnsreg
Feature-Caps: *;+sip.pns="acme";+sip.pnsreg="121"
Expires: 7200
Content-Length: 0
```

Figure 3: SIP REGISTER When Using Non-Push Mechanism Example

4.1.5. Query Network PNS Capabilities

This section describes how a SIP UA can query the types of PNSs supported by a SIP network, and PNS-related capabilities (e.g., support of the VAPID mechanism). When a UA performs a query, it does not request push notifications from the SIP network. Therefore, the UA can perform the query before it has registered to a PNS and received a PRID.

In order to perform a query, the UA MUST insert a pn-provider SIP URI parameter in the Contact header field URI of the REGISTER request:

- If the UA inserts a pn-provider parameter value, indicating support of a type of PNS, the SIP network will only inform the UA whether that type of PNS is supported.
- If the UA does not insert a pn-provider parameter value (i.e., it inserts an "empty" pn-provider parameter) the SIP network will inform the UA about all types of PNSs supported by the network. This is useful e.g., if the UA supports more than one type of PNS. Note that it is not possible to insert multiple parameter values in the pn-provider parameter.

The UA MUST NOT insert a pn-prid SIP URI parameter in the Contact header field URI of the REGISTER request.

If the UA receives a 2xx response to the REGISTER request, the response will contain one or more Feature-Caps header fields with a 'sip.pns' feature-capability indicator, indicating the types of PNSs supported by the SIP network. If the UA inserted a pn-provider SIP URI parameter value in the REGISTER request, the response will only indicate whether the SIP network supports the type of PNS supported by the UA.

If the UA receives a 555 (Push Notification Service Not Supported) response to the REGISTER request, and if the UA inserted a pn-provider SIP URI parameter in the REGISTER request, the response indicates that the network does not support the type of PNS that the UA indicated support of. If the UA did not insert a pn-provider parameter in the REGISTER request, the response indicates that the network does not support any type of PNS, while still supporting the 555 (Push Notification Service Not Supported) response.

NOTE: It is optional for a UA to perform a query before it requests push notifications from the SIP network.

5. SIP Proxy Behavior

5.1. PNS Provider

The type of PNS is identified by the pn-provider SIP URI parameter. In some cases there might only be one PNS provider for a given type of PNS, while in other cases there might be multiple providers. The pn-param SIP URI parameter will provide more details associated with the actual PNS provider to be used.

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. SIP Request Push Bucket

When a SIP proxy receives a SIP request addressed towards a UA, that will trigger the proxy to request that a push notification be sent to the UA. The proxy will place the request in storage referred to as the SIP Request Push Bucket, and the proxy starts a timer (referred to as the Bucket Timer) associated with the transaction. A SIP request is removed from the bucket when one of the following has occurred: the proxy forwards the request towards the UA, when the proxy sends an error response to the request, or when the Bucket Timer times out. The detailed procedures are described in the sections below.

Exactly how the SIP Request Push Bucket is implemented is outside the scope of this document. One option is to use the PRID as a key to search for SIP requests in the bucket. Note that mid-dialog requests (Section 6) do not carry the PRID in the SIP request itself.

5.3. SIP URI Comparison Rules

By default, a SIP proxy uses the URI comparison rules defined in [RFC3261]. However, when a SIP proxy compares the Contact header field URI of a 2xx response to a REGISTER request with a Request-URI of a SIP request in the SIP Request Push Bucket (Section 5.2), the proxy uses the URI comparison rules with the following additions: the pn-prid, pn-provider and pn-param SIP URI parameters MUST also match. If a pn- parameter is present in one of the compared URIs but not in the other URI, there is no match.

If only the pn- SIP URI parameters listed above match, but other parts of the compared URIs do not match, a proxy MAY still consider the comparison successful based on local policy. This can occur in a race condition when the proxy compares the Contact header field URI of a 2xx response to a REGISTER request with a Request-URI of a SIP request in the SIP Request Push Bucket (Section 5.2), if the UA had modified some parts of the Contact header field URI in the REGISTER request but the Request-URI of the SIP request in the SIP Request Push Bucket still contains the old parts.

5.4. Indicate Support of Type of PNS

A SIP proxy uses feature-capability indicators [RFC6809] to indicate support of types of PNSs and additional features (e.g., VAPID) associated associated with the type of PNS. A proxy MUST use a

separate Feature-Cap header field for each supported type of PNS. A feature-capability indicator that indicates support of an additional feature associated with a given type of PNS MUST be inserted in the same Feature-Caps header field that is used to indicate support of the type of PNS.

This specification defines the following feature-capability indicators that a proxy can use to indicate support of additional features associated with a given type of PNS: 'sip.vapid', 'sip.pnsreg' and 'sip.pnspurr'. These feature-capability indicators MUST only be inserted in a Feature-Caps header field that also contains a 'sip.pns' feature-capability indicator.

5.5. Trigger Periodic Binding Refresh

In order to request that a push notification be sent to a SIP UA, a SIP proxy needs to have information about when a binding will expire. The proxy needs to be able to retrieve the information from the registrar using some mechanism, or run its own registration timers. Such mechanisms are outside the scope of this document, but could be implemented e.g., using the SIP event package for registrations mechanism [RFC3680].

When the proxy receives an indication that the UA needs to send a binding-refresh REGISTER request, the proxy will request that a push notification be sent to the UA.

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

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

NOTE: As described in Section 4.1.5, a 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 that push notifications are sent to a UA that has not provided a pn-prid SIP URI parameter (Section 5.6.2).

If the proxy receives information that a binding associated with a PRID has expired, or that a binding has been removed, the proxy MUST NOT request that further push notifications are sent to the UA using that PRID.

5.6. SIP Requests

5.6.1. REGISTER

This section describes how a SIP proxy processes SIP REGISTER requests (initial REGISTER request for a binding, or a binding-refresh REGISTER request).

The procedures in this section apply when the REGISTER request contains a pn-provider SIP URI parameter in the Contact header field URI. In other cases the proxy MUST skip the procdeures in this section, and process the REGISTER request using normal SIP procedures.

5.6.1.1. Request Push Notifications

This section describes the SIP proxy procedures when a SIP UA requests push notifications from the SIP network.

The procedures in this section apply when the SIP REGISTER request contains, in addition to the pn-provider SIP URI parameter, a pn-prid SIP URI parameter in the Contact header field URI of the request.

When a proxy receives a REGISTER request that contains a Feature-Caps header field with a 'sip.pns' feature-capability indicator, it indicates that another proxy between this proxy and the UA supports the type of PNS supported by the UA, and will request that push notifications are sent to the UA. In such case, the proxy MUST skip the rest of the procedures in this section, and process the REGISTER request using normal SIP procedures.

When a proxy receives a REGISTER request that does not contain a Feature-Caps header field with a 'sip.pns' feature-capability indicator, the proxy processes the request according to the procedures below:

- If the proxy does not support the type of PNS supported by the UA, or if the REGISTER request does not contain all information required for the type of PNS, the proxy SHOULD forward the request towards the registrar and skip the rest of the procedures in this section. If the proxy knows (by means of local configuration) that no other proxies between itself and the registrar support the type of PNS supported by the UA, the proxy MAY send a SIP 555

(Push Notification Service Not Supported) response instead of forwarding the request.

- If the proxy supports the type of PNS supported by the UA, but considers the requested binding expiration interval [RFC3261] to be too short (see below), the proxy MUST either send a 423 (Interval Too Brief) response to the REGISTER request or forward the request towards the registrar and skip the rest of the procedures in this section.
- If the proxy supports the type of PNS supported by the UA, the proxy MUST indicate support of that type of PNS (Section 5.4) in the REGISTER request before it forwards the request towards the registrar. This will inform proxies between the proxy and the registrar that the proxy supports the type of PNS supported by the UA, and that the proxy will request that push notifications are sent to the UA.

A binding expiration interval MUST be considered too short if the binding would expire before the proxy can request that a push notification be sent to the UA to trigger the UA to send a binding-refresh REGISTER request. The proxy MAY consider the interval too short based on its own policy so as to reduce load on the system.

When a proxy receives a 2xx response to the REGISTER request, if the proxy indicated support of a type of PNS in the REGISTER request (see above), the proxy performs the following actions:

- If the proxy considers the binding expiration interval indicated by the registrar too short (see above), the proxy forwards the response towards the UA and MUST skip the rest of the procedures in this section.
- The proxy MUST indicate support of the same type of PNS in the REGISTER response. In addition:
  - If the proxy supports the VAPID mechanism [RFC8292], the proxy MUST indicate support of the mechanism, using the 'sip.vapid' feature-capability indicator, in the REGISTER response. The indicator value contains the public key identifying the proxy. The proxy MUST determine whether the PNS provider supports the VAPID mechanism before it indicates support of it.
  - If the proxy received a 'sip.pnsreg' media feature tag in the REGISTER request, the proxy SHOULD insert a 'sip.pnsreg' feature-capability indicator with an indicator value bigger than 120 in the response, unless the proxy always wants to request that push notifications are sent to the UA in order to trigger the UA to send a binding-refresh REGISTER request.

5.6.1.2. Query Network PNS Capabilities

This section describes the SIP proxy procedures when a SIP UA queries about the push notification support in the SIP network (Section 4.1.5).

The procedures in this section apply when the REGISTER request contains a pn-provider SIP URI parameter, but does not contain a pn-prid SIP URI parameter in the Contact header field URI of the REGISTER request.

When a proxy receives a REGISTER request that contains a pn-provider SIP URI parameter indicating the type of PNS supported by the UA, the proxy MUST perform the following actions:

- If the proxy supports the type of PNS supported by the UA, the proxy MUST indicate support of that type of PNS (Section 5.4) in the REGISTER request before it forwards the request towards the registrar. This will inform any other proxies between the proxy and the registrar that the proxy supports the type of PNS supported by the UA.
- If the proxy does not support the type of PNS supported by the UA, and if the REGISTER request contains Feature-Caps header fields indicating support of one or more types of PNSs, the proxy forwards the request towards the registrar.
- If the proxy does not support the type of PNS supported by the UA, and if the REGISTER request does not contain Feature-Caps header fields indicating support of one or more types of PNSs, the proxy MUST either forward the request towards the registrar, or send a SIP 555 (Push Notification Service Not Supported) response towards the UA. The proxy MUST NOT send a SIP 555 (Push Notification Service Not Supported) response unless it knows (by means of local configuration) that no other proxy supports any of the types of PNSs supported by the UA.

When a proxy receives a REGISTER request, and the pn-provider SIP URI parameter does not contain a parameter value, the proxy MUST indicate support of each type of PNS supported by the proxy before it forwards the request towards the registrar.

When a proxy receives a 2xx response to the REGISTER request, if the proxy had indicated support of one or more types of PNSs in the REGISTER request (see above), the proxy MUST indicate support of the same set of types of PNSs in the response. In addition, if the proxy supports the VAPID mechanism for one or more types of PNSs, the proxy MUST indicate support of the mechanism for those PNSs in the response.

5.6.2. Initial Request for Dialog or Stand-Alone Request

The procedures in this section apply when a SIP proxy has indicated that the it will request that push notifications are sent to 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 that a push notification be sent to the UA using the information in the pn- SIP URI parameters. The proxy then places the SIP request in the SIP Request Push Bucket. The push notification will trigger the UA to send a binding-refresh REGISTER request that the proxy will process as described in Section 5.6.1. In addition, the proxy MUST store the Contact URI of the REGISTER request during the lifetime of the REGISTER transaction.

NOTE: If the proxy receives a SIP request that does not contain the pn- SIP URI parameters listed above, the proxy processing of the request is based on local policy. If the proxy also serves requests for UAs that do not use the SIP push mechanism, the proxy can forward the request towards the UA. Otherwise the proxy can reject the request.

When the proxy receives a 2xx response to the REGISTER request, the proxy performs the following actions:

- The proxy processes the REGISTER response as described in Section 5.6.1.
- The proxy checks whether the SIP Request Push Bucket contains a SIP request associated with the REGISTER transaction, by comparing (Section 5.3) the Contact header field URI in the REGISTER response with the Request-URIs of the SIP requests in the bucket. If there is a match, the proxy MUST remove the SIP request from the bucket and forward it towards the UA.

The reason the proxy needs to wait for the REGISTER response before forwarding a SIP request towards a UA is to make sure that the REGISTER request has been accepted by the registrar, and that the UA that initiated the REGISTER request is authorized to receive messages for the Request-URI.

If the proxy receives a non-2xx response to the REGISTER request, the proxy compares the Contact URI stored from the REGISTER request (see above) with the Request-URIs of the SIP requests in the SIP Request Push Bucket. If there is a match, the proxy SHOULD remove the

associated request from the bucket and send an error response to the request. It is RECOMMENDED that the proxy sends either a 404 (Not Found) response or a 480 (Temporarily Unavailable) response to the SIP request, but other response codes can be used as well. However, if the REGISTER response is expected to trigger a new REGISTER request from the UA (e.g., if the registrar is requesting the UA to perform authentication) the proxy MAY keep the SIP request in the bucket.

If the push notification request fails (see PNS-specific documentation for details), the proxy MUST remove the SIP request from the bucket and send an error response to the SIP request. It is RECOMMENDED that the proxy sends either a 404 (Not Found) response or a 480 (Temporarily Unavailable) response, but other response codes can be used as well.

After the proxy has requested that a push notification be sent to a UA, if the proxy does not receive a REGISTER response with a Contact URI that matches the Request-URI of the SIP request before the Bucket Timer (Section 5.2) associated with the SIP request times out, the proxy MUST remove the SIP request from the SIP Request Push Bucket (Section 5.2) and send a 480 (Temporarily Unavailable) response. The Bucket Timer time-out value is set based on local policy, taking the guidelines below into consideration.

As discussed in [RFC4320] and [RFC4321], non-INVITE transactions must complete immediately or risk losing a race, which 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, if the proxy forwards the SIP request towards the SIP UA, the SIP UA accepts the request, but the transaction times out at the sender before it receives the successful response, this will cause state misalignment between the endpoints (the sender considers the transaction a failure, while the receiver considers the transaction a success). The SIP proxy needs to take this into account when it sets the value of the Bucket Timer associated with the transaction, to make sure that the error response (triggered by a Bucket Timer time out) 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 cannot be 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.

In addition to the procedures described above there are two cases where a proxy, as an optimization, can forward a SIP request towards a UA without waiting for a 2xx response to a REGISTER request, or without even requesting that a push notification be sent to the UA:

- If the proxy is able to authenticate the sender of the REGISTER request and verify that it is allowed by authorization policy, the proxy does not need to wait for the 2xx response before it forwards the SIP request towards the UA. In such cases, the proxy will use the Contact URI of the REGISTER request when comparing it against the Request-URIs of the SIP requests in the SIP Request Push Bucket.
- If the proxy has knowledge that the UA is awake, and that the UA is able to receive the SIP request without first sending a binding-refresh REGISTER request, the proxy does not need to request that a push notification be sent to the UA (the UA will not send a binding-refresh REGISTER request) before it forwards 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.

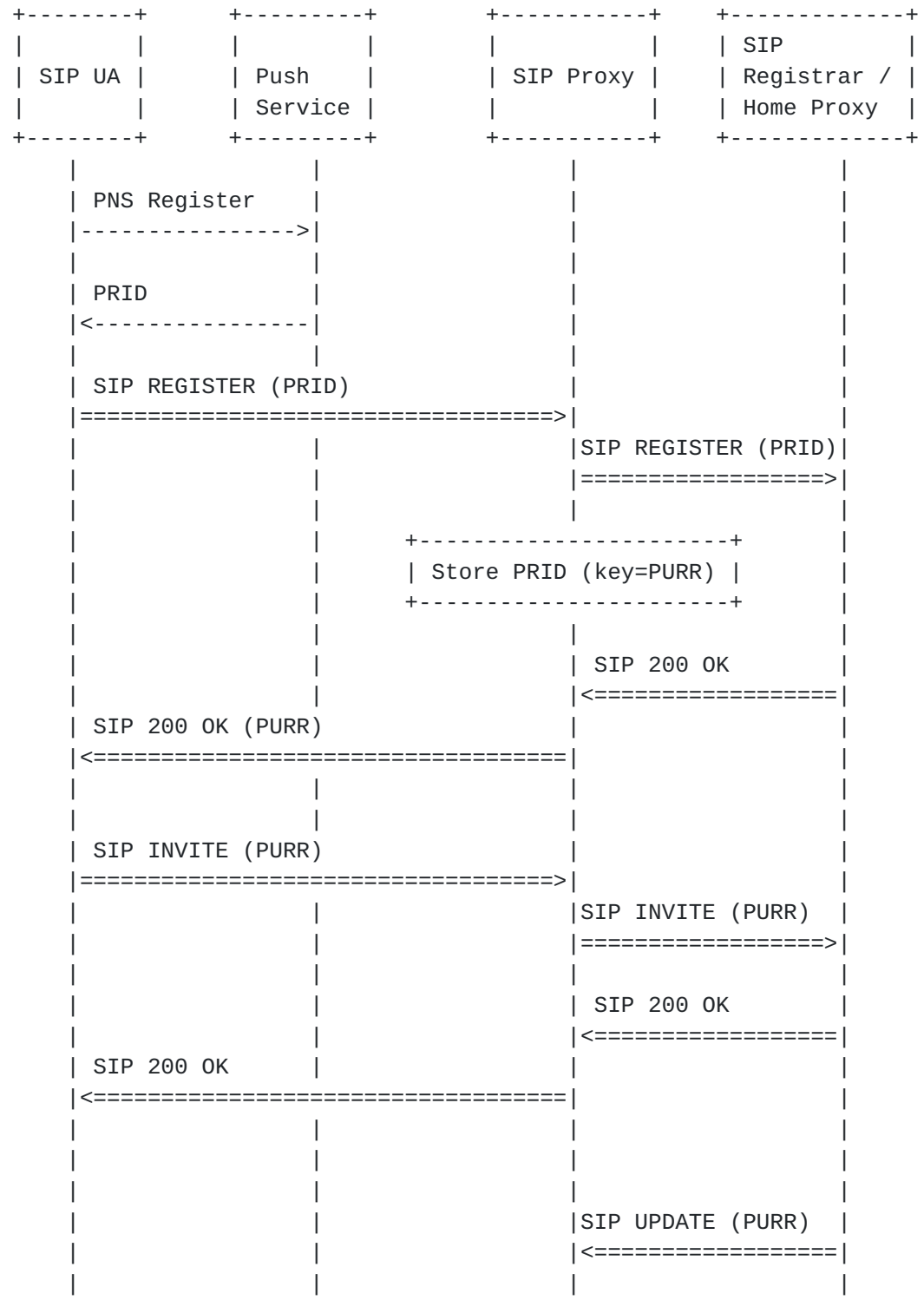
Some PNS providers allow payload in the push notifications. This specification does not define usage of such payload (in addition to any payload that might be required by the PNS itself).

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 the sending of mid-dialog requests. Because of this, a SIP UA may be suspended, and may need to be awakened in order to be able to receive mid-dialog requests.

SIP requests for a new dialog and stand-alone SIP requests addressed towards a UA will pn- SIP URI parameters that allows the proxy to request that a push notification be sent to the UA Section 5.6.2. However, pn- SIP URI parameters will not be present in mid-dialog requests addressed towards the UA. Instead, the proxy needs to support a mechanism to store the information needed to request that a push notification be sent to the UA, and to be able to retrieve that information when it receives a mid-dialog request addressed towards

the UA. This section defines such mechanism. The SIP UA and SIP proxy procedures in this section are applied in addition to the generic procedures defined in this specification.



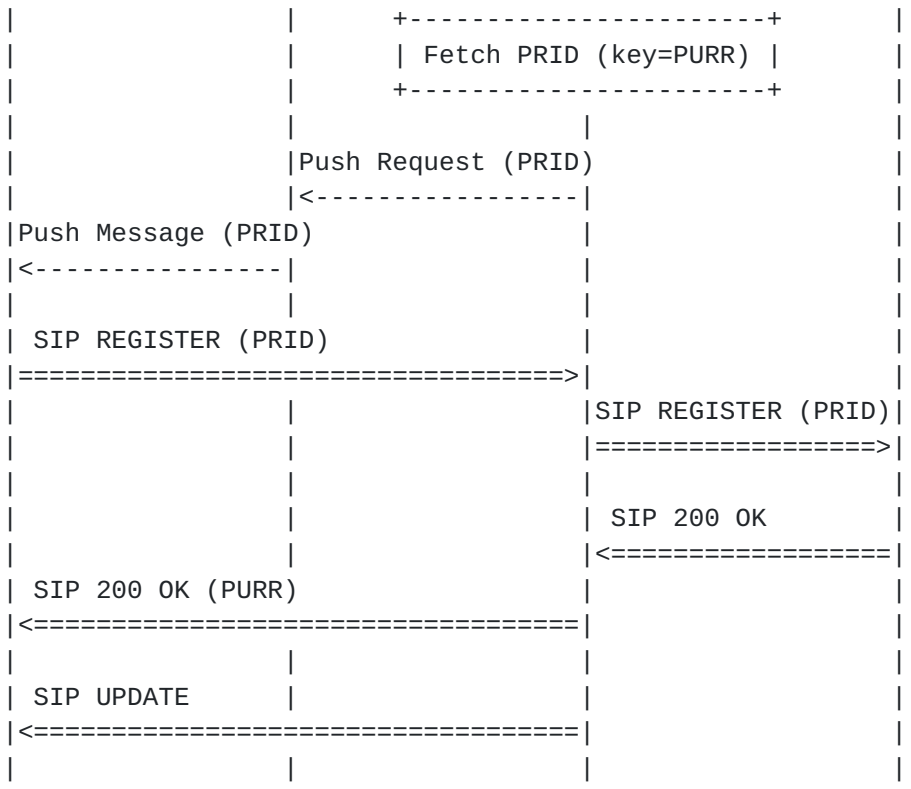


Figure 4: SIP Push Longlived Dialog Flow

6.1. SIP UA Behavior

6.1.1. Initial Request for Dialog

If the UA is willing to receive push notifications when a proxy receives a mid-dialog request addressed towards the UA, the UA MUST insert a 'pn-purr' SIP URI parameter (Section 6.2.1) in the Contact header field URI of the initial request for a dialog or the 2xx response to such requests. XXX The UA MUST insert a parameter value identical to the last 'sip.pnspurr' feature-capability indicator (Section 6.2.1) that it received in a REGISTER response. If the UA has not recived a 'sip.pnspurr' feature-capability indicator, the UA MUST NOT insert a 'pn-purr' SIP URI parameter in a request or response.

The UA makes the decision to receive push notifications triggered by incoming mid-dialog requests 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.

NOTE: As the 'pn-purr' SIP URI parameter only applies to a given dialog, the UA needs to insert a 'pn-purr' parameter in the Contact header field URI of the request or response for each dialog in which the UA is willing to receive push notifications triggered by incoming mid-dialog requests.

6.2. SIP Proxy Behavior

6.2.1. REGISTER

If the proxy supports requesting push notifications triggered by mid-dialog requests being sent to the registered UA, the proxy MUST store the information (the pn- SIP URI parameters) needed to request that push notifications are sent to the UA when a proxy receives an initial REGISTER request for a binding from the UA. In addition, the proxy MUST generate a unique (within the context of the proxy) value, referred to as the PURR (Proxy Unique Registration Reference), that can be used as a key to retrieve the information.

In order to prevent client fingerprinting, the proxy MUST periodically generate a new PURR value (even if pn- parameters did not change). However, as long as there are ongoing dialogs associated with the old value, the proxy MUST store it so that it can request that push notifications are sent to the UA when it receives a mid-dialog request addressed towards the UA. In addition, the PURR value MUST be generated in such a way so that it is unforgeable, anonymous, and unlinkable to entities other than the proxy. It must not be possible for an attacker to generate a valid PURR, to associate a PURR with a specific user, or to determine when two PURRs correspond to the same user. It can be generated e.g., by utilizing a cryptographically secure random function with an appropriately large output size.

Whenever the proxy receives a 2xx response to a REGISTER request, the proxy MUST insert a 'sip.pnspurr' feature-capability indicator with the latest PURR value (see above) in the response.

6.2.2. Initial Request for Dialog

When a proxy receives an initial request for a dialog from a UA that contains a 'pn-purr' SIP URI parameter in the Contact header field URI with a PURR value that the proxy has generated (Section 6.2.1), the proxy MUST add a Record-Route header to the request to insert itself in the dialog route [RFC3261] before forwarding the request.

When the proxy receives an initial request for a dialog addressed towards the UA, and the proxy has generated a PURR value associated with the pn- parameters inserted 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] before forwarding the request.

6.2.3. Mid-Dialog Request

When the proxy receives a mid-dialog SIP request addressed towards the UA that contains a 'pn-purr' SIP URI parameter, and the proxy is able to retrieve the stored information needed to request that a push notification be sent to the UA (Section 6.2.1), the proxy MUST place the SIP request in the SIP Request Push Bucket and request that a push notification be sent to the UA.

NOTE: The 'pn-purr' SIP URI parameter will either be carried in the Request-URI or in a Route header field [RFC3261] of the SIP request, depending on how the route set [RFC3261] of the mid-dialog SIP request has been constructed.

When the proxy receives a 2xx response to a REGISTER request, the proxy checks whether the SIP Request Push Bucket contains a mid-dialog SIP request associated with the REGISTER transaction. If the bucket contains such request the proxy MUST remove the SIP request from the SIP Request Push Bucket and forward it towards the UA.

Note that the proxy does not perform a URI comparison (Section 5.3) when processing mid-dialog requests, as a mid-dialog request will not contain the pn-prid, pn-provider and pn-param SIP URI parameters. The proxy only checks for a mid-dialog request that contains the PURR value associated with the REGISTER 2xx response.

As described in Section 5.6.2, while waiting for the push notification request to succeed, and then for the associated REGISTER request and 2xx response, the proxy needs to take into consideration that the transaction associated with the mid-dialog request will eventually time out at the sender of the request (UAC), and the sender will consider the transaction a failure.

When a proxy sends an error response to a mid-dialog request (e.g., due to a transaction time out), the proxy SHOULD select a response code that only impacts the transaction associated with the request [RFC5079].

7. Support Of SIP Replaces

[RFC3891] defines a mechanism that allows a SIP UA to replace a dialog with another dialog. A UA that wants to replace a dialog with another one will send an initial request for the new dialog. The Request-URI of the request will contain Contact header field URI of the peer.

If a SIP proxy wants to be able to request that a push notification be sent to a UA when it receives an initial request for a dialog that replaces an existing dialog, using the mechanism in [RFC3891], the proxy and the UA MUST perform the following actions:

- The proxy MUST provide a PURR to the UA during registration (Section 6.2.1).
- The UA MUST insert a 'pn-purr' SIP URI parameter in the Contact header field URI of the initial request for a dialog, or a 2xx response to such requests (Section 6.1.1). This includes dialogs replacing other dialogs, as those dialogs might also get replaced.
- The proxy MUST apply the mechanism defined in Section 6.2.3 to place and retrieve the request from the SIP Request Push Bucket.

In addition, the operator needs to make sure that the initial request for dialogs, addressed towards the UA using the contact of the replaced dialog, will be routed to the SIP proxy (in order to request that a push notification be sent to the UA). The procedures for doing that are operator specific, and are outside the scope of this specification.

8. Grammar

8.1. 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 does not support the push notification service identified in a 'pn-provider' SIP URI parameter.

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

8.2. sip.pns Feature-Capability Indicator

The sip.pns feature-capability indicator, when inserted 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 the SIP push mechanism and the type of push notification service indicated 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 RDQUOT
pns             = tag-value

tag-value = <tag-value defined in [RFC3840]>
```

8.3. sip.vapid Feature-Capability Indicator

The sip.vapid feature-capability indicator, when inserted in a SIP 2xx response to a SIP REGISTER request, denotes that the entity associated with the indicator supports the Voluntary Application Server Identification (VAPID) [RFC8292] mechanism when the entity requests that a push notification be sent to a SIP UA. 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 = <tag-value defined in [RFC3840]>
```

8.4. sip.pnsreg Feature-Capability Indicator

The sip.pnsreg feature-capability indicator, when inserted in a SIP 2xx response to a SIP REGISTER request, denotes that the entity associated with the indicator expects to receive binding-refresh REGISTER requests from the SIP UA associated with the binding before the binding expires, even if the entity does not request that a push notification be sent to the SIP UA in order to trigger the binding-refresh REGISTER requests. The indicator value conveys the minimum time (given in seconds), prior to the binding expiration when the UA MUST send the REGISTER request.

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

DIGIT = <DIGIT defined in [RFC3261]>
```

8.5. sip.pnsreg Media Feature Tag

The sip.pnsreg media feature tag, when inserted in the 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 for the associated binding without being awakened by push notifications. The media feature tag has no values.

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

8.6. sip.pnspurr Feature-Capability Indicator

The sip.pnspurr feature-capability indicator, when inserted in a SIP 2xx response to a SIP REGISTER request, denotes that the entity associated with the indicator will store information that can be used to associate a mid-dialog SIP request with the binding information in the REGISTER request.

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

tag-value = <tag-value defined in [RFC3840]>
```

8.7. 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-purr
pn-provider     = "pn-provider" [EQUAL pvalue]
pn-param        = "pn-param" EQUAL pvalue
pn-prid         = "pn-prid" EQUAL pvalue
pn-purr         = "pn-purr" EQUAL pvalue

pvalue = <pvalue defined in [RFC3261]>
EQUAL = <EQUAL defined in [RFC3261]>
```

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 need to be escaped, as defined in RFC 3261.

9. PNS Registration Requirements

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

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

10. 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.

For detailed information about the parameter values:

https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CommunicatingwithAPNs.html [pns-apns])

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

Example: pn-provider=apns

The value of the pn-param URI parameter is a string that is composed of two values, separated by a period (.): Team ID and Topic. The Team ID is provided by Apple and is unique to a development team.

The Topic consists of the Bundle ID, which uniquely identifies an application, and a service value that identifies a service associated with the application, separated by a period (.). For VoIP applications the service value is "voip".

Example: pn-param=DEF123GHIJ.com.example.yourexampleapp.voip

NOTE: The Bundle ID might contain one or more periods (.). Hence, within the pn-param value, the first period will be separating the Team ID from the Topic, and within the Topic the last period will be separating the Bundle ID from the service.

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

11. 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.

For detailed information about the parameter values:

https://firebase.google.com/docs/cloud-messaging/concept-options [pns-fcm]

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.

12. 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 [RFC8030] 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.

13. Security Considerations

The security considerations for the use and operation of any particular PNS (e.g., how users and devices are authenticated and authorized) is out of scope for this document. [RFC8030] documents the security considerations for the PNS defined in that specification. Security considerations for other PNSs are left to their respective specifications.

Typically, the PNS requires the SIP proxy requesting push notifications to be authenticated and authorized by the PNS. In some cases the PNS also requires the SIP application (or the SIP application developer) to be identified in order for the application to request push notifications. Unless the PNS authenticates and authorizes the PNS, a malicious endpoint or network entity that managed to get access to the parameters transported in the SIP signalling might be able to request that push notifications are sent to a UA. Such push notifications will impact the battery life of the UA and trigger unnecessary SIP traffic.

[RFC8292] defines a mechanism that 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.

Operators MUST ensure that the SIP signalling is properly secured, e.g., using encryption, from malicious network entities. TLS MUST be used, unless the operators know that the signalling is secured using some other mechanism that provides strong crypto properties.

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.

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

Operators MUST ensure that the PNS-related SIP URI parameters conveyed by a user in the Contact URI of a REGISTER request are not sent to other users, or to non-trusted network entities. One way to convey contact information is by using the the SIP event package for registrations mechanism [RFC3680]. [RFC3680] defines generic security considerations for the SIP event package for registrations. As the PNS-related SIP URI parameters conveyed in the REGISTER request contain sensitive information, operators that support the event package MUST ensure that event package subscriptions are properly authenticated and authorized, and that the SIP URI parameters are not inserted in event notifications sent to other users, or to non-trusted network entities.

14. IANA considerations

[RFC EDITOR NOTE: Please replace all instances of RFCXXXX with the RFC number of this document.]

14.1. 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.

14.1.1. pn-provider

Parameter Name: pn-provider

Predefined Values: No

Reference: RFC XXXX

14.1.2. pn-param

Parameter Name: pn-param

Predefined Values: No

Reference: RFC XXXX

14.1.3. pn-prid

Parameter Name: pn-prid

Predefined Values: No

Reference: RFC XXXX

14.1.4. pn-purr

Parameter Name: pn-purr

Predefined Values: No

Reference: RFC XXXX

14.2. SIP Response Codes

14.2.1. 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

14.3. SIP Global Feature-Capability Indicator

14.3.1. 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 inserted in a Feature-Caps header field of a SIP REGISTER request or a SIP 2xx response to a REGISTER request, denotes that the entity associated with the indicator supports the SIP push mechanism and the type of push notification service conveyed by the indicator value.

Reference: [RFCXXXX]

Contact: IESG (iesg@ietf.org)

14.3.2. 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 inserted in a SIP 2xx response to a SIP REGISTER request, denotes that the entity associated with the indicator supports the Voluntary Application Server Identification (VAPID) mechanism when the entity requests that a push notifications be sent to a SIP UA. 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)

14.3.3. 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 inserted in a SIP 2xx response to a SIP REGISTER request, denotes that the entity associated with the indicator expects to receive binding-refresh REGISTER requests for the binding from the SIP UA associated with the binding before the binding expires, even if the entity does not request that a push notification be sent to the SIP UA in order to trigger the binding-refresh REGISTER requests. The indicator value conveys the minimum time (given in seconds), prior to the binding expiration when the UA MUST send the REGISTER request.

Reference: [RFCXXXX]

Contact: IESG (iesg@ietf.org)

14.3.4. sip.pnspurr

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.pnspurr

Description: This feature-capability indicator, when inserted in a SIP 2xx response to a SIP REGISTER request, conveys that the entity associated with the indicator will store information that can be used to associate a mid-dialog SIP request with the binding information in the REGISTER request. The indicator value is an identifier that can be used a key to retrieve the binding information.

Reference: [RFCXXXX]

Contact: IESG (iesg@ietf.org)

14.4. SIP Media Feature Tag

14.4.1. 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 inserted in the Contact header field of a SIP REGISTER request, conveys that the SIP UA associated with the tag is able to send binding-refresh REGISTER requests associated with the registration without being awakened 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 will not be requested to be sent to the client.

Contact: IESG (iesg@ietf.org)

14.5. 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]

15. Acknowledgements

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

16. References

16.1. 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>.

[RFC3891] Mahy, R., Biggs, B., and R. Dean, "The Session Initiation Protocol (SIP) "Replaces" Header", RFC 3891, DOI 10.17487/RFC3891, September 2004, <https://www.rfc-editor.org/info/rfc3891>.

[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>.

[RFC5079] Rosenberg, J., "Rejecting Anonymous Requests in the Session Initiation Protocol (SIP)", RFC 5079, DOI 10.17487/RFC5079, December 2007, <https://www.rfc-editor.org/info/rfc5079>.

[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>.

[pns-apns] Apple Inc, "Apple Push Notification Service", January 2019, <https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CommunicatingwithAPNs.html>.

[pns-fcm] Google Inc, "Firebase Cloud Messaging", January 2019, <https://firebase.google.com/docs/cloud-messaging/concept-options>.

16.2. 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>.

[RFC3680] Rosenberg, J., "A Session Initiation Protocol (SIP) Event Package for Registrations", RFC 3680, DOI 10.17487/RFC3680, March 2004, <https://www.rfc-editor.org/info/rfc3680>.

[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