Session Initiation Protocol (SIP) Event Package for OMA Content Push Delivery
draft-mdolly-dispatch-oma-push-00

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 28, 2010.

Copyright Notice

Copyright (c) 2009 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 in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
Abstract

This document specifies a new event package for OMA Push-based service over SIP. The purpose is to allow an OMA application or a UA to subscribe to updates to its own OMA application events containing either content or references to the content.

This document further describes how content can be pushed out to an application by the use of OMA Push-based events.

Table of Contents

1. Introduction ........................................... 4
2. Definitions and Document Conventions .................... 4
3. Overview ............................................. 4
   3.1. OMA Push ........................................ 4
   3.2. HTTP Long Polling ................................ 5
4. Use Cases ............................................. 5
5. Motivation Scenarios .................................... 6
6. Push Delivery Framework ................................ 7
   6.1. Discovery of Push AS ............................... 7
   6.2. Push delivery stages ............................... 7
7. The push event package (Normative) ...................... 8
   7.1. Event Package Definition ........................... 8
   7.2. Event Package Name ................................ 8
   7.3. Event Package Parameters ........................... 8
   7.4. Event parameters .................................. 9
      7.4.1. Parameters format ............................. 9
      7.4.2. Dev-cap ....................................... 9
      7.4.3. Event-app-id ................................. 10
8. Push AS Processing of SUBSCRIBE Requests ............... 10
9. SUBSCRIBE Bodies ..................................... 10
10. Subscription Duration ................................... 10
11. Push Enrollment ....................................... 10
   11.1. Initiation of a Push Enrollment .................... 11
   11.2. Profile Enrollment Confirmation .................... 13
   11.3. Push Content Delivery ............................. 14
   11.4. Use of the REFER Method ........................... 15
12. NOTIFY Bodies ......................................... 16
13. Handling of Forked Requests ............................. 17
14. Rate of Notifications .................................. 17
15. IANA Considerations ................................... 17
   15.1. SIP Event Package ................................. 17
16. Security Considerations ................................ 18
17. References ........................................... 18
   17.1. Normative References .............................. 18
   17.2. Informative References ............................ 19
1. Introduction

Push services are usually based on information preference expressed in advance, in a sort of Publish/Subscribe model. A client might "subscribe" to various information "channels", or establish readiness to receive network-initiated events from an application server. Whenever new content or an application event is available the server would push that information to the client. Push delivery enables applications to depend upon asynchronous delivery of content on a scheduled or unscheduled basis rather than a trial and error basis as provided by a pull model. For example, Push enables real-time information delivery for services such as stock market updates or earthquake alerts.

This specification defines use case scenarios and related requirements supporting use of SIP for OMA Push-based services. In particular it shows what is necessary to define/extend in SIP to let applications on the client subscribe to events in the network, how those events can be delivered and what type of content can be delivered by them.

2. Definitions and Document Conventions

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

3. Overview

This section provides an overview of push methodology, and how other protocols or technology provide a similar solution.

3.1. OMA Push

OMA Push, which includes WAP (Wireless Application Protocol) Push [OMA-WAP] and SIP Push (using SIP MESSAGE and SIP INVITE + MSRP) [SIP-Push-1.0], is widely supported by mobile devices, as the basis for network-initiated delivery of content and OMA enabler protocol events to mobile clients e.g. browsers, IM, MMS, location services, device management etc.

In the OMA Push architecture, the originator of push events is called a Push Initiator (PI), and may be any server or OMA enabler system dependent upon OMA Push. The PI communicates with a Push Proxy Gateway (PPG) using the OMA Push Access Protocol (PAP). The PPG uses
one of the Push Over-The-Air (Push-OTA) [OMA-OTA] protocol variants (WAP1, WAP2, or SIP) supported by the target device to deliver the push content to the client. PAP is based on standard Internet protocols; XML is used to express the delivery instructions, and the push content can be any MIME media type. OMA Push has been successful due to its efficient use of resources, its variety of supported network bearers and features supporting context-adaptive delivery of push content, and its independence from the content to be delivered.

3.2. HTTP Long Polling

Several server push programming mechanisms have been implemented in recent years, in the wired Web environment, to enable asynchronous bidirectional HTTP or server-initiated communication from a server to a client as well as from a client to a server. Those mechanisms have been implemented to avoid the need for an HTTP client to send regular requests to the Web server in order to "pull" any available events or data [I-D.loreto-http-bidirectional].

With the HTTP Long Polling mechanism, the server attempts to "hold open" (not immediately reply to) each HTTP request, responding only when there are events to deliver. In this way, there is always a pending request available to use for delivering events as they occur, thereby minimizing the latency in message delivery. This mechanism is the one commonly implemented by the XMLHttpRequest used for Ajax web Applications.

The problem with this solution is that it requires a TCP connection to be open for each application expecting a push to be delivered at sometime in the future. This can result in resource problems in the network, especially for mobile devices, e.g. due to battery impact, the cost of data usage for TCP session keep-alive transactions, and the impact on the public IP address space of network Operators due to extended holding of NAT ports for long-lived TCP connections.

4. Use Cases

This section lists several use cases where an application takes advantage subscribing to an OMA Push-based services. Once an application has correctly subscribed the OMA Push-based service delivers content directly by imbedding small content into the notification itself or indirectly by using content indirection.
An Application, that needs to receive push events to work correctly, subscribes to a Push Server to receive either content or references to the content.

An Application MAY need push events from more than one Push Server, e.g. in the case that a particular Push Server supports one type of event, and others support different events.

5. Motivation Scenarios

SIP currently provides no direct support for content delivery paradigms in which clients subscribe to services for which the network, not the client, has the knowledge about content availability and thereby takes actions to notify the client of the content availability.

In the mobile environment OMA Push has been very successful due to strong support for the standard, enabling any mobile application to leverage this generic push capability. In the fixed environment does such a standard not exist, instead do the market relies on varying implementations of delayed HTTP response or pure pull models. A common push model for fixed and mobile environment will benefit the market.

The proposed push specification in this document will enable Service or Content Providers to deliver push-based content updates to applications via a new event package, specified to facilitate opaque delivery of content. This will thus become an enabler of new services which can be designed to use this subscribe/notify facility for delivery of arbitrary content or application events.

This proposal provides a push model based on SIP SUBSCRIBE / NOTIFY, as this will provide the best model to ensure the required functionality. This push model will complement the existing OMA Push specification [OMA-OTA] which defines the use of SIP MESSAGE and SIP INVITE+MSRP for delivery of push content. In addition, the SUBSCRIBE/NOTIFY will have advantages in the support it can provide for capability negotiation and dialog management.

The proposal will cover the push delivery mechanism, and how push resources (identifying the subscribing applications) can be added and removed from client/server dialogs, but not any application-specific services.

This will allow for example for:
SIP applications to get content updates directly or indirectly without the need to implement other protocol.

Non SIP application that don’t want or can’t have a TCP connection (reverse Ajax) open during the whole time the terminal is connected to the network.

6. Push Delivery Framework

This section specifies the push delivery framework. It provides the requirements push delivery stages and presents the associated security requirements. It supports two push delivery stages - enrollment and content delivery.

6.1. Discovery of Push AS

How Push AS are discovered by an UA is out of the scope of this document. Push AS are expected to be discovered using unspecified mechanisms like:

- Preconfiguration of the application as installed on the device.
- Discovery of the Push AS through application-layer features.
- Configuration of the Push AS through device management, e.g. OMA DM.

6.2. Push delivery stages

The framework specified in this document requires an application to initiate push delivery by explicit subscription to Push events. It also requires one or more Push Servers which provide the push content. The processes that lead an application to obtain push content can be explained in three stages, termed the profile delivery stages.

Push Enrollment: the process by which an application subscribes to push content delivery, and enrolls with one or more Push Servers capable of providing push content.

A successful enrollment is indicated by a notification with the accepted push resources (content/event types) that will be delivered in the push events as embedded or indirectly referenced content. Depending upon the request, this could also eventually results in a subscription to notification of profile changes.

Push Content Delivery: the process by which a application receives
push content, if the push enrollment was successful.

Change Notification: the process by which an application is notified of any changes to an enrolled profile. This may provide the application with modified profile data or content indirection information.

[To clarify] the need for this Change Notification feature was not anticipated by OMA.

7. The push event package (Normative)

The "push" Event Package defines a Push-based service, and allows for SIP applications to subscribe to specific push event deliveries from an application server. The "push" event package specified in this document proposes and specifies an event package according to [RFC3265]

The Push Client uses the application profile type to subscribe to content for Push Applications. The "oma-app" profile allows a Push Client to indicate its desire for application-specific content.

The oma-app profile type MUST follow the steps of Profile Enrollment and Profile Content Retrieval as defined in [OMA-UAProf]. Profile Enrollment is the process by which the Push Client subscribes to a Push Application accessed via a Profile Delivery Server, and the Profile Content Retrieval is the process by which an application on a device receives profile content.

7.1. Event Package Definition

This section defines a new SIP event package [RFC3265]. The purpose of this event package is to send to subscribers notification of content updates to the subscribed push resources via content indirection [RFC4483] or directly in the body of the NOTIFY.

7.2. Event Package Name

The name of this package is "oma-push".

This value appears in the Event header field present in SUBSCRIBE and NOTIFY requests for this package as defined in [RFC3265].

7.3. Event Package Parameters

This package defines the "event-app-id" and "dev-cap" as new parameters for the event Header. The event-app-id parameter is used
to indicate the token name of the push application(s) to which the Push Client wishes to subscribe, and is referred to in this specification as the "Application Resource Identifier".

7.4. Event parameters

The following table shows the use of Event header parameters in SUBSCRIBE requests for the oma-push event package:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>event-app-id</td>
<td>mandatory</td>
</tr>
<tr>
<td>dev-cap</td>
<td>optional</td>
</tr>
<tr>
<td>extension</td>
<td>optional</td>
</tr>
</tbody>
</table>

The Push UA and the Push AS MAY support extensions to the push event. Extensions MUST be registered via IANA. The Push UA and Push AS MUST ignore extensions that they do not support.

7.4.1. Parameters format

A Push Receiver or Application MUST use the following format for oma-app: In the following ABNF, SEMI, EQUAL and token are defined in [RFC3261].

```
EVENT-APP-ID SEMI DEV-CAP

EVENT-APP-ID = "event-app-id" EQUAL event-app-id-list
DEV-CAP = "dev-cap" EQUAL quoted-string

event-app-id-list = DQUOTE app *"","" app) DQUOTE
app = 1*(&x21 / &x23-2B / &x2D-7E)
DQUOTE = &x22 ;as per section 6.1 of [RFC2234]
```

7.4.2. Dev-cap

This parameter is useful to the Push AS to affect the content provided. In some scenarios it is desirable to provide different services based upon this parameter. e.g., feature property X in a service may work differently on two versions of the same user agent. This gives the Push AS the ability to compensate for, or take advantage of, the differences.

The dev-cap parameter is a parameter that provides an optional method...
of getting the device capabilities.

Use of the dev-cap parameter is described in Section 11.1.

7.4.3. Event-app-id

The event-app-id parameter provides the token name of the push application(s) to which the Push Client wishes to subscribe.

The value of a event-app-id MUST be a URN [SIP-Push-1.0].

8. Push AS Processing of SUBSCRIBE Requests

Upon receipt of a SUBSCRIBE request, the notifier applies authorization according to local policy.

A successful SUBSCRIBE request results in a NOTIFY with either profile content or a pointer to it (via Content Indirection).

9. SUBSCRIBE Bodies

This package definition includes use of the SUBSCRIBE request body to carry an indirect reference to the OMA User Agent Profile [OMA-UAProf] of the Push UA, if any, as described in Section 11.1.

10. Subscription Duration

The durations of push event subscriptions, and in general the conditions under which they are terminated, vary widely with the nature of the applications as they are assumed to be application-specific criteria.

The subscriber is responsible for selecting an appropriate expiration time.

In the absence of an expires value in a subscription, the notifier assumes a default expiration period according to local policy. It would not be unreasonable for a server to assume a default expiration value of 86400 seconds (one day).

11. Push Enrollment
11.1. Initiation of a Push Enrollment

To initiate Push Enrollment the Push UA SHALL send a SIP SUBSCRIBE with the event header set to "push".

The SUBSCRIBE request SHALL be generated according to the rules and procedures of [RFC3265], and in addition:

1. The Push UA SHALL set the Request-URI to either the user Address of Record (AoR) identifying the current user, or a SIP URI identifying the Push AS’s Public Service Identity (PSI), based on local policy or configuration.

2. The Push UA MAY insert the URI of the Push UA in the P-Preferred-Identity header according to the rules and procedures of [RFC3265].

3. If a GRUU [I-D.ietf-sip-gruu], is supported, and has been obtained during the registration process, the Push UA SHALL include the GRUU in the Contact header of the SIP SUBSCRIBE message, and include a Supported header containing the option tag "gruu";

4. In the Contact header, the Push UA SHALL include the Application Resource Identifier including values for all push applications supported by the Push UA.

5. The Push UA MAY include the Application Resource Identifier in the Accept-Contact header per [RFC3841], indicating the push applications to which the Push UA seeks subscription from the Push AS.

6. The Push UA SHALL include the dev-cap parameter if a [OMA-UAProf] is available

7. If [OMA-UAProf] information is not available for a device, then the model, vendor, and version parameters SHALL be included in the dev-cap parameter.

8. The Push UA SHALL include all the SIP methods that it supports in the Allow header;

9. The Push UA SHOULD include a User-Agent header containing the model, vendor, and version of the Push Receiver Agent according to the rules and procedures of [RFC3261];

10. The Push UA SHALL, if a User Agent Profile as defined by [OMA-UAProf] is supported, include a Content-Type header per
A. the MIME type message/external-body;
B. "URL" in the ACCESS_TYPE parameter;
C. the expiration parameter according to [RFC4483];
D. the HTTP URL of the [OMA-UAProf] document in the URL parameter.

11. The Push UA SHALL, if a User Agent Profile as defined by [OMA-UAProf] is supported, include a body according to [RFC4483] containing:
   A. application/rdf+xml in the Content-Type;
   B. "URL" in the ACCESS_TYPE parameter;
   C. "attachment" in the Content-Disposition;
   D. a Content-ID header per [RFC4483].

[NOTE] If the User Agent Profile as defined by [OMA-UAProf] is not supported by the Push UA, the Push AS can only obtain information about the Push UA from the User-Agent header.

Upon receiving a SUBSCRIBE request for the "push" event, the Push AS SHALL follow the rules and procedures of [RFC3261], [RFC3265], [RFC3841], [OMA-UAProf], and [RFC4483] as applicable, and in addition:

1. The Push AS SHALL verify that a P-Asserted-Identity header is present and the URI in P-Asserted-Identity header is trusted. If the authorization check fails, the Push AS SHALL return a SIP 403 "Forbidden" response per [RFC3261], and consider the enrollment unsuccessful.

If enrollment is successful:

1. The Push AS SHALL generate a SIP 200 "OK" response per [RFC3261], and in addition:
   A. The Push AS SHALL include an Application Resource Identifier identifying the supported push applications;
   B. The Push AS SHOULD include an Accept header containing the MIME types supported by the Push AS including message/
external-body and application/rdf+xml, and any other MIME types supported by the Push AS as part of the push applications it will support for the subscription, per [RFC3261];

C. The Push AS SHALL include all the SIP methods that the Push AS supports in the Allow header;

2. The Push AS SHALL provide a profile enrollment confirmation for each push application as indicated by the Application Resource Identifier if any, or as applicable by default, per the push applications supported by the Push AS for the requesting Push UA.

11.2. Profile Enrollment Confirmation

The Push AS SHALL deliver profile enrollment confirmations via a NOTIFY request.

The NOTIFY request SHALL be generated per [RFC3265], and in addition:

1. The Push AS SHALL include one event-app-id parameter, identifying the enrollment being confirmed.

2. The Push AS MAY include application content in the notification.

3. If content is to be included in the notification, the Push AS SHALL either embed the content in the NOTIFY, or provide a content reference per [RFC4483].

The Push UA response to the SIP NOTIFY request SHALL be handled in accordance to rules and procedures of [RFC3265].

As examples:

The Push UA only subscribes to one application (app1), and supports [OMA-UAProf]. The Push AS supports app1.

SUBSCRIBE sip:user-aor@example.com SIP/2.0:
Event: oma-push;event-app-id="app1";
   dev-cap= "http://wap.company.com/UAProf/model.xml"

NOTIFY
Event: oma-push; event-app-id ="app1"

The Push UA only subscribes to one application (app1), and does not support [OMA-UAProf]. The Push AS supports app1.
The Push UA subscribes to multiple applications (app1, app2 and app3), and supports [OMA-UAProf]. The Push AS supports app1, app2, and app3.

The Push UA subscribes to multiple applications (app1, app2 and app3), and does not support [OMA-UAProf]. The Push AS supports app1, app2.

11.3. Push Content Delivery

A successful Push Enrollment may result in a series of notifications to the Push UA, delivered as applicable for the subscribed push applications.

The Push AS SHALL deliver notifications via a NOTIFY request.

The NOTIFY request SHALL be generated according to the rules and procedures of [RFC3265], and in addition:
1. The Push AS SHALL include one event-app-id parameter, identifying the enrollment being confirmed.

2. In the case of a user having multiple registered terminals with a Push AS, the Push AS MAY enforce a delivery model including a GRUU value according to rules and procedures in [I-D.ietf-sip-gruu] in order to select the explicit terminal(s) to receive the NOTIFY.

3. The Push AS MAY include application content in the notification.

4. If content is to be included in the notification, the Push AS SHALL either embed the content in the NOTIFY, or provide a content reference per [RFC4483].

The responses to the SIP NOTIFY request SHALL be handled in according to rules and procedures of [RFC3265].

11.4. Use of the REFER Method

The REFER method [RFC3515] is an extension to SIP [RFC3261]. The recipient of a REFER request, upon granting permission from the user, initiates a new SIP request to the resource provided in the REFER message.

The Push UA MUST support the REFER method as stated in [RFC3515]. In particular, the Push UA MUST be able to receive REFER requests, perform the requested action, and provide a response to the requester.

At any time, a Push AS may send a REFER request to the Push UA to trigger a profile enrollment by the Push UA to another Push AS.

A Push AS wishing to trigger refer a Push UA to another Push AS SHALL send a REFER request according to [RFC3515], and in addition:

1. The Push AS SHALL set the Request-URI to the public SIP URI identifying the destination Push UA.

2. The Push AS SHALL include a Refer-To header with the following values:

   A. the referred URI to the same value as the Request-URI, or to the SIP URI identifying a Push AS.

   B. "SUBSCRIBE" as the method parameter of the referred URI.
C. the Event header parameter in the referred URI, with the event package name set as "oma-push", the profile-type parameter value set to "oma-app", and including an Application Resource Identifier indicating the push applications for which the Push UA is referred to the new Push AS.

3. If the Push AS requires that Push UA explicitly terminate the existing subscription, the Push AS SHALL include the Replaces header per [RFC3891].

4. If the subscription was associated with a specific terminal via a GRUU, the Push AS SHALL direct the REFER request to the specific terminal via the GRUU.

5. If no forking can be guaranteed, the Push AS MAY include the "Refer-Sub" header set to "false" per [RFC4488], to suppress the implicit subscription.

Upon receiving a REFER request, the Push UA SHALL process the request as described in [RFC3515], with the following clarifications:

1. The Push UA SHALL return the SIP "489 Bad Event" error response per [RFC3265] and consider the REFER request as invalid, if the Refer-To header does not include a SUBSCRIBE to the "oma-push" event package.

For a valid REFER request:

1. The Push UA SHALL send a SUBSCRIBE request to the new Push AS as described in "11.1. Initiation of a Push Enrollment", including the Application Resource Identifier as indicated by the original Push AS in the REFER request.

2. The Push UA SHALL send a SIP "202 Accepted" per [RFC3515].

12. NOTIFY Bodies

As described in Section 11.3, NOTIFY bodies MAY include either embedded content or indirectly referenced content.

For minimal interoperability, the Push UA and Push AS MUST support the "http:" and "https:" URI schemes for content indirection. Other URI schemes MAY also be supported for content indirection. However as the security considerations in this document are defined for content indirection using HTTP and HTTPS only, use of other protocols for content indirection is out of scope of this document.
13. Handling of Forked Requests

This Event package allows the creation of only one dialog as a result of an initial SUBSCRIBE request as described in section 4.4.9 of [RFC3265]. It does not support the creation of multiple subscriptions using forked SUBSCRIBE requests.

14. Rate of Notifications

Push Clients have normative requirements to support a Lockout Timer for Push events if they exceed some reasonable number (decided by the vendor).

Overall OMA Push is assumed to be used within a trusted environment (client/server relationship or trusted network) thus throttling is not a key feature.

15. IANA Considerations

There are two IANA considerations associated with this document, SIPEvent Package and SIP configuration profile types. These are outlined in the following sub-sections.

15.1. SIP Event Package

This specification registers a new event package as defined in [RFC3265]. The following information required for this registration:

- Package Name: oma-push
- Type: package
- New event header parameters: profile-type
- Persons to Contact: Martin Dolly, mdolly@att.com
- Reference: RFC XXXX [Note to RFC Editor: replace with the RFC number for this specification]

This specification registers three new SIP header field parameters, defined by the following information which is to be added to the Header Field Parameters and Parameter Values sub-registry under http://www.iana.org/assignments/sip-parameters.
16. Security Considerations

The security considerations listed in SIP events [RFC3265], which the Push mechanism extends, apply in entirety. In particular, authentication and message integrity SHOULD be applied to subscriptions with the Push extension.

17. References

17.1. Normative References


17.2. Informative References


Authors’ Addresses

Martin Dolly
AT&T
200 Laurel Ave
Middletown, NJ,
US

Email: mdolly@att.com

Bryan Sullivan
AT&T
7277 164TH AVE NE BLDG 2
Redmond, WA,
US

Email: bryan.sullivan@att.com

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com

Kent Bogestam
Sweden

Email: kent.bogestam@cumbari.com