APNs Provider API

Apple Push Notification service includes the APNs Provider API that allows you to send remote notifications to your app on iOS, tvOS, and OS X devices, and to Apple Watch via iOS. This API is based on the HTTP/2 network protocol. Each interaction starts with a POST request, containing a JSON payload, that you send from your provider server to APNs. APNs then forwards the notification to your app on a specific user device.

Your APNs certificate, which you obtain as explained in Creating a Universal Push Notification Client SSL Certificate in *App Distribution Guide*, enables connection to both the APNs Production and Development environments.

You can use your APNs certificate to send notifications to your primary app, as identified by its bundle ID, as well as to any Apple Watch complications or backgrounded VoIP services associated with that app. Use the `( 1.2.840.113635.100.6.3.6 )` extension in the certificate to identify the topics for your push notifications. For example, if you provide an app with the bundle ID `com.yourcompany.yourexampleapp`, you can specify the following topics in the certificate:

```
Extension ( 1.2.840.113635.100.6.3.6 )
Critical NO
Data com.yourcompany.yourexampleapp
Data app
Data com.yourcompany.yourexampleapp.voip
Data voip
Data com.yourcompany.yourexampleapp.complication
Data complication
```

Connections

The first step in sending a remote notification is to establish a connection with the appropriate APNs server:

- **Development server:** `api.development.push.apple.com:443`
- **Production server:** `api.push.apple.com:443`

Note: You can alternatively use port `2197` when communicating with APNs. You might do this, for example, to allow APNs traffic through your firewall but to block other HTTPS traffic.

Your provider server must support TLS 1.2 or higher when creating the connection to the APNs servers. In addition, the connection must be authenticated with your provider client certificate, which you obtain from Member Center, as described in Creating a Universal Push Notification Client SSL Certificate.

The APNs server allows multiple concurrent streams for each connection. The exact number of streams is based on server load, so do not assume a specific number of streams. The APNs servers ignore HTTP/2 `PRIORITY` frames, so do not send them on your streams.

Best Practices for Managing Connections

Keep your connections with APNs open across multiple notifications; don't repeatedly open and close connections. APNs treats rapid connection and disconnection as a denial-of-service attack. You

should leave a connection open unless you know it will be idle for an extended period of time—for example, if you only send notifications to your users once a day it is ok to use a new connection each day.

You can establish multiple connections to APNs servers to improve performance. When you send a large number of remote notifications, distribute them across connections to several server endpoints. This improves performance, compared to using a single connection, by letting you send remote notifications faster and by letting APNs deliver them faster.

You can check the health of your connection using an HTTP/2 `PING` frame.

Termination

If APNs decides to terminate an established HTTP/2 connection, it sends a `GOAWAY` frame. The `GOAWAY` frame includes JSON data in its payload with a `reason` key, whose value indicates the reason for the connection termination. For a list of possible values for the `reason` key, see Table 6-6.

Normal request failures do not result in termination of a connection.

Notification API

The APNs Provider API consists of a request and a response that you configure and send using an HTTP/2 `POST` command. You use the request to send a push notification to the APNs server and use the response to determine the results of that request.

Request

Use a request to send a notification to a specific user device.

Table 6-1 HTTP/2 request

Name	Value
`:method`	`POST`
`:path`	`/3/device/`*<device-token>*

For the *<device-token>* parameter, specify the hexadecimal bytes of the device token for the target device.

APNs requires the use of HPACK (header compression for HTTP/2), which prevents repeated header keys and values. APNs maintains a small dynamic table for HPACK. To help avoid filling up the APNs HPACK table and necessitating the discarding of table data, encode headers in the following way—especially when sending a large number of streams:

- The `:path` header should be encoded as a literal header field without indexing,
- The `apns-id` and `apns-expiration` headers should be encoded differently depending on initial or subsequent POST operation, as follows:
  - The first time you send these headers, encode them with incremental indexing to allow the header names to be added to the dynamic table
  - Subsequent times you send these headers, encode them as literal header fields without indexing

Encode all other headers as literal header fields with incremental indexing. For specifics on header encoding, see tools.ietf.org/html/rfc7541#section-6.2.1 and tools.ietf.org/html/rfc7541#section-

6.2.2.

APNs ignores headers other than the ones in Table 6-2.

Table 6-2 Request headers

Header	Description
`apns-id`	A canonical UUID that identifies the notification. If there is an error sending the notification, APNs uses this value to identify the notification to your server. The canonical form is 32 lowercase hexadecimal digits, displayed in five groups separated by hyphens in the form 8-4-4-4-12. An example UUID is as follows: `123e4567-e89b-12d3-a456-42665544000` If you omit this header, a new UUID is created by APNs and returned in the response.
`apns-expiration`	A UNIX epoch date expressed in seconds (UTC). This header identifies the date when the notification is no longer valid and can be discarded. If this value is nonzero, APNs stores the notification and tries to deliver it at least once, repeating the attempt as needed if it is unable to deliver the notification the first time. If the value is `0`, APNs treats the notification as if it expires immediately and does not store the notification or attempt to redeliver it.
`apns-priority`	The priority of the notification. Specify one of the following values: ▪ `10`–Send the push message immediately. Notifications with this priority must trigger an alert, sound, or badge on the target device. It is an error to use this priority for a push notification that contains only the `content-available` key. ▪ `5`—Send the push message at a time that takes into account power considerations for the device. Notifications with this priority might be grouped and delivered in bursts. They are throttled, and in some cases are not delivered. If you omit this header, the APNs server sets the priority to `10`.
`apns-topic`	The topic of the remote notification, which is typically the bundle ID for your app. The certificate you create in Member Center must include the capability for this topic. If your certificate includes multiple topics, you must specify a value for this header. If you omit this header and your APNs certificate does not specify multiple topics, the APNs server uses the certificate's Subject as the default topic.

The body content of your message is the JSON dictionary object containing the notification data. The body data must not be compressed and its maximum size is 4KB (4096 bytes). For information about the keys and values to include in the body content, see The Remote Notification Payload.

Response

The response to a request has the format listed in Table 6-3.

Table 6-3 Response headers

Header name	Value
`apns-id`	The `apns-id` value from the request. If no value was included in the request, the server creates a new UUID and returns it in this header.
`:status`	The HTTP status code. For a list of possible status codes, see Table 6-4.

Table 6-4 lists the possible status codes for a request. These values are included in the `:status` header of the response.

Table 6-4 Status codes for a response

Status code	Description
200	Success
400	Bad request
403	There was an error with the certificate.
405	The request used a bad `:method` value. Only `POST` requests are supported.
410	The device token is no longer active for the topic.
413	The notification payload was too large.
429	The server received too many requests for the same device token.
500	Internal server error
503	The server is shutting down and unavailable.

For a successful request, the body of the response is empty. On failure, the response body contains a JSON dictionary with the keys listed in Table 6-5. This JSON data might also be included in the `GOAWAY` frame when a connection is terminated.

Table 6-5 JSON data keys

Key	Description
`reason`	The error indicating the reason for the failure. The error code is specified as a string. For a list of possible values, see Table 6-6.
`timestamp`	If the value in the `:status` header is `410`, the value of this key is the last time at which APNs confirmed that the device token was no longer valid for the topic. Stop pushing notifications until the device registers a token with a later timestamp with your provider.

Table 6-6 lists the possible error codes included in the `reason` key of a response's JSON payload.

Table 6-6 Values for the `reason` key

Reason	Description
`PayloadEmpty`	The message payload was empty. Expected HTTP/2 status code is `400`; see Table 6-4.
`PayloadTooLarge`	The message payload was too large. The maximum payload size is `4096` bytes.
`BadTopic`	The `apns-topic` was invalid.
`TopicDisallowed`	Pushing to this topic is not allowed.

BadMessageId	The apns-id value is bad.
BadExpirationDate	The apns-expiration value is bad.
BadPriority	The apns-priority value is bad.
MissingDeviceToken	The device token is not specified in the request :path. Verify that the :path header contains the device token.
BadDeviceToken	The specified device token was bad. Verify that the request contains a valid token and that the token matches the environment.
DeviceTokenNotForTopic	The device token does not match the specified topic.
Unregistered	The device token is inactive for the specified topic. Expected HTTP/2 status code is 410; see Table 6-4.
DuplicateHeaders	One or more headers were repeated.
BadCertificateEnvironment	The client certificate was for the wrong environment.
BadCertificate	The certificate was bad.
Forbidden	The specified action is not allowed.
BadPath	The request contained a bad :path value.
MethodNotAllowed	The specified :method was not POST.
TooManyRequests	Too many requests were made consecutively to the same device token.
IdleTimeout	Idle time out.
Shutdown	The server is shutting down.
InternalServerError	An internal server error occurred.
ServiceUnavailable	The service is unavailable.
MissingTopic	The apns-topic header of the request was not specified and was required. The apns-topic header is mandatory when the client is connected using a certificate that supports multiple topics.

Examples

Listing 6-1 shows a sample request constructed for a certificate.

Listing 6-1 Sample request for a certificate with a single topic

```
HEADERS
  - END_STREAM
  + END_HEADERS
  :method = POST
```

```
  :scheme = https
  :path = /3/device/00fc13adff785122b4ad28809a3420982341241421348097878e577c991de8f0
  host = api.development.push.apple.com
  apns-id = eabeae54-14a8-11e5-b60b-1697f925ec7b
  apns-expiration = 0
  apns-priority = 10
DATA
  + END_STREAM
    { "aps" : { "alert" : "Hello" } }
```

Listing 6-2 shows a sample request constructed for a certificate that contains multiple topics.

Listing 6-2 Sample request for a certificate with multiple topics

```
HEADERS
  - END_STREAM
  + END_HEADERS
  :method = POST
  :scheme = https
  :path = /3/device/00fc13adff785122b4ad28809a3420982341241421348097878e577c991de8f0
  host = api.development.push.apple.com
  apns-id = eabeae54-14a8-11e5-b60b-1697f925ec7b
  apns-expiration = 0
  apns-priority = 10
  apns-topic = <MyAppTopic>
DATA
  + END_STREAM
    { "aps" : { "alert" : "Hello" } }
```

Listing 6-3 shows a sample response for a successful push request.

Listing 6-3 Sample response for a successful request

```
HEADERS
  + END_STREAM
  + END_HEADERS
  :status = 200
```

Listing 6-4 shows a sample response when an error occurs.

Listing 6-4 Sample response for a request that encountered an error

```
HEADERS
  - END_STREAM
  + END_HEADERS
  :status = 400
  content-type = application/json
    apns-id: <a_UUID>
DATA
  + END_STREAM
```

```
{ "reason" : "BadDeviceToken" }
```