The Remote Notification Payload

Each remote notification includes a payload. The payload contains information about how the system should alert the user as well as any custom data you provide. The maximum size allowed for a notification payload depends on which provider API you employ. When using the HTTP/2 provider API, maximum payload size is 4096 bytes. Using the legacy binary interface, maximum payload size is 2048 bytes. Apple Push Notification service (APNs) refuses any notification that exceeds the maximum size.

Payload Keys

For each notification, compose a JSON dictionary object (as defined by RFC 4627). This dictionary must contain another dictionary identified by the `aps` key. The `aps` dictionary can contain one or more properties that specify the following user notification types:

- An alert message to display to the user
- A number to badge the app icon with
- A sound to play

To support silent remote notifications, add the `remote-notification` value to the `UIBackgroundModes` array in your `Info.plist` file. To learn more about this array, see UIBackgroundModes.

If the target app isn't running when the notification arrives, the alert message, sound, or badge value is played or shown. If the app is running, the system delivers the notification to the app delegate as an `NSDictionary` object. The dictionary contains the corresponding Cocoa property-list objects (plus `NSNull`).

Providers can specify custom payload values outside the Apple-reserved `aps` dictionary. Custom values must use the JSON structured and primitive types: dictionary (object), array, string, number, and Boolean. You should not include customer information (or any sensitive data) as custom payload data. Instead, use it for such purposes as setting context (for the user interface) or internal metrics. For example, a custom payload value might be a conversation identifier for use by an instant-message client app or a timestamp identifying when the provider sent the notification. Any action associated with an alert message should not be destructive—for example, it should not delete data on the device.

> **Important:** Delivery of notifications is a "best effort", not guaranteed. It is not intended to deliver data to your app, only to *notify* the user that there is new data available.

Table 5-1 lists the keys and expected values of the `aps` payload.

Table 5-1 Keys and values of the `aps` dictionary

Key	Value type	Comment
`alert`	string or dictionary	If this property is included, the system displays a standard alert or a banner, based on the user's setting. You can specify a string or a dictionary as the value of `alert`. ▪ If you specify a string, it becomes the message text of an alert with two buttons: Close and View. If the user taps View, the app launches. ▪ If you specify a dictionary, refer to Table 5-2 for descriptions of the keys of this dictionary. The JSON `\U` notation is not supported. Put the actual UTF-8 character in the alert text instead.
		The number to display as the badge of the app icon.

badge	number	If this property is absent, the badge is not changed. To remove the badge, set the value of this property to 0.
sound	string	The name of a sound file in the app bundle or in the Library/Sounds folder of the app's data container. The sound in this file is played as an alert. If the sound file doesn't exist or default is specified as the value, the default alert sound is played. The audio must be in one of the audio data formats that are compatible with system sounds; see Preparing Custom Alert Sounds for details.
content-available	number	Provide this key with a value of 1 to indicate that new content is available. Including this key and value means that when your app is launched in the background or resumed, application:didReceiveRemoteNotification:fetchCompletionHandler: is called.
category	string	Provide this key with a string value that represents the identifier property of the UIMutableUserNotificationCategory object you created to define custom actions. To learn more about using custom actions, see Registering Your Actionable Notification Types.

Table 5-2 lists the keys and expected values for the alert dictionary.

Table 5-2 Child properties of the alert property

Key	Value type	Comment
title	string	A short string describing the purpose of the notification. Apple Watch displays this string as part of the notification interface. This string is displayed only briefly and should be crafted so that it can be understood quickly. This key was added in iOS 8.2.
body	string	The text of the alert message.
title-loc-key	string or null	The key to a title string in the Localizable.strings file for the current localization. The key string can be formatted with %@ and %*n*$@ specifiers to take the variables specified in the title-loc-args array. See Localized Formatted Strings for more information. This key was added in iOS 8.2.
title-loc-args	array of strings or null	Variable string values to appear in place of the format specifiers in title-loc-key. See Localized Formatted Strings for more information. This key was added in iOS 8.2.
action-loc-key	string or null	If a string is specified, the system displays an alert that includes the Close and View buttons. The string is used as a key to get a localized string in the current localization to use for the right button's title instead of "View". See Localized Formatted Strings for more information.
loc-key	string	A key to an alert-message string in a Localizable.strings file for the current localization (which is set by the user's language preference). The key string can be formatted with %@ and %*n*$@ specifiers to take the variables specified in the loc-args array. See Localized Formatted Strings for more information.
loc-args	array of strings	Variable string values to appear in place of the format specifiers in loc-key. See Localized Formatted Strings for more information.
launch-		The filename of an image file in the app bundle, with or without the filename extension. The image is used as the launch image when users tap the action button or move the action slider. If this property is not specified, the system

`image`	string	either uses the previous snapshot,uses the image identified by the `UILaunchImageFile` key in the app's `Info.plist` file, or falls back to `Default.png`. This property was added in iOS 4.0.

Note: If you want the device to display the message text as-is in an alert that has both the Close and View buttons, then specify a string as the direct value of alert. *Don't* specify a dictionary as the value of `alert` if the dictionary only has the `body` property.

Configuring a Silent Notification

The `aps` dictionary can also contain the `content-available` property. The `content-available` property with a value of `1` lets the remote notification act as a silent notification. When a silent notification arrives, iOS wakes up your app in the background so that you can get new data from your server or do background information processing. Users aren't told about the new or changed information that results from a silent notification, but they can find out about it the next time they open your app.

For a silent notification, take care to ensure there is no `alert`, `sound`, or `badge` payload in the `aps` dictionary. If you don't follow this guidance, the incorrectly-configured notification might be throttled and not delivered to the app in the background, and instead of being silent is displayed to the user.

Localized Formatted Strings

You can display localized alert messages in two ways:

- The server originating the notification can localize the text; to do this, it must discover the current language preference selected for the device (see Passing the Provider the Current Language Preference (Remote Notifications)).
- The client app can store the alert-message strings in its bundle, translated for each localization it supports. The provider includes the `loc-key` and `loc-args` keys in the `aps` dictionary of the notification payload. (For title strings, it includes the `title-loc-key` and `title-loc-args` keys in the aps dictionary.) When the device receives the notification (assuming the app isn't running), it uses these `aps`-dictionary properties to find and format the string localized for the current language, which it then displays to the user.

Here's how that second option works in a little more detail.

An app can internationalize resources such as images, sounds, and text for each language that it supports, Internationalization collects the resources and puts them in a subdirectory of the bundle with a two-part name: a language code and an extension of `.lproj` (for example, `fr.lproj`). Localized strings that are programmatically displayed are put in a file called `Localizable.strings`. Each entry in this file has a key and a localized string value; the string can have format specifiers for the substitution of variable values. When an app asks for a particular resource—say a localized string—it gets the resource that is localized for the language currently selected by the user. For example, if the preferred language is French, the corresponding string value for an alert message would be fetched from `Localizable.strings` in the `fr.lproj` directory in the app bundle. (The app makes this request through the `NSLocalizedString` macro.)

Note: This general pattern is also followed when the value of the `action-loc-key` property is a string. This string is a key into the `Localizable.strings` in the localization directory for the currently selected language. iOS uses this key to get the title of the button on the right side of an alert message (the "action" button).

To make this clearer, let's consider an example. The provider specifies the following dictionary as the value of the alert property:

```
"alert" : {
    "loc-key" : "GAME_PLAY_REQUEST_FORMAT",
"loc-args" : [ "Jenna", "Frank"]
}
```

When the device receives the notification, it uses `"GAME_PLAY_REQUEST_FORMAT"` as a key to look up the associated string value in the `Localizable.strings` file in the `.lproj` directory for the current language. Assuming the current localization has a `Localizable.strings` entry such as this:

```
"GAME_PLAY_REQUEST_FORMAT" = "%@ and %@ have invited you to play Monopoly";
```

the device displays an alert with the message "Jenna and Frank have invited you to play Monopoly".

In addition to the format specifier `%@`, you can `%`*n*`$@` format specifiers for positional substitution of string variables. The *n* is the index (starting with 1) of the array value in `loc-args` to substitute. (There's also the `%%` specifier for expressing a percentage sign (%).) So if the entry in `Localizable.strings` is this:

```
"GAME_PLAY_REQUEST_FORMAT" = "%2$@ and %1$@ have invited you to play Monopoly";
```

the device displays an alert with the message "Frank and Jenna have invited you to play Monopoly".

For a full example of a notification payload that uses the `loc-key` and `loc-arg` properties, see Examples of JSON Payloads. To learn more about internationalization, see *Internationalization and Localization Guide*. String formatting is discussed in Formatting String Objects in *String Programming Guide*.

Note: You should use the `loc-key` and `loc-args` properties—and the `alert` dictionary in general—only if you absolutely need to. The values of these properties, especially if they are long strings, might use up more bandwidth than is good for performance. Many apps don't need these properties because their message strings are originated by users.

Examples of JSON Payloads

The following examples of the payload portion of notifications illustrate the practical use of the properties listed in Table 5-1. Properties with "acme" in the key name are examples of custom payload data.

Note: The examples are formatted with whitespace and line breaks for readability. In practice, omit whitespace and line breaks to reduce the size of the payload, improving network performance.

Example 1. "The following payload has an `aps` dictionary with a simple, recommended form for alert messages with the default alert buttons (Close and View). It uses a string as the value of `alert` rather than a dictionary. This payload also has a custom array property.

```
{
    "aps" : { "alert" : "Message received from Bob" },
    "acme2" : [ "bang",  "whiz" ]
}
```

Example 2. The payload in the example uses an `aps` dictionary to request that the device display an alert message with a Close button on the left and a localized title for the "action" button on the right side of the alert. In this case, "PLAY" is used as a key into the `Localizable.strings` file for the currently

selected language to get the localized equivalent of “Play”. The `aps` dictionary also requests that the app icon be badged with the number 5. On Apple Watch, the title key alerts the user to the new request.

```
{
    "aps" : {
        "alert" : {
            "title" : "Game Request",
            "body" : "Bob wants to play poker",
            "action-loc-key" : "PLAY"
        },
        "badge" : 5
    },
    "acme1" : "bar",
    "acme2" : [ "bang",  "whiz" ]
}
```

Example 3. The payload in this example specifies that the device should display an alert message with both Close and View buttons. It also requests that the app icon be badged with the number 9 and that a bundled alert sound be played when the notification is delivered.

```
{
    "aps" : {
        "alert" : "You got your emails.",
        "badge" : 9,
        "sound" : "bingbong.aiff"
    },
    "acme1" : "bar",
    "acme2" : 42
}
```

Example 4. The payload in this example uses the `loc-key` and `loc-args` child properties of the `alert` dictionary to fetch a formatted localized string from the app’s bundle and substitute the variable string values (`loc-args`) in the appropriate places. It also specifies a custom sound and includes a custom property.

```
{
    "aps" : {
        "alert" : {
            "loc-key" : "GAME_PLAY_REQUEST_FORMAT",
            "loc-args" : [ "Jenna", "Frank"]
        },
        "sound" : "chime.aiff"
    },
    "acme" : "foo"
}
```

Example 5. The payload in this example uses the `category` key to specify a group of notification actions (to learn more about categories, see Registering Your Actionable Notification Types).

```
{
   "aps" : {
      "category" : "NEW_MESSAGE_CATEGORY"
      "alert" : {
```

```
          "body" : "Acme message received from Johnny Appleseed",
          "action-loc-key" : "VIEW"
       },
       "badge" : 3,
       "sound" : “chime.aiff"
    },
    "acme-account" : "jane.appleseed@apple.com",
    "acme-message" : "message123456"
}
```