Configuring and Using Ensemble Push Notifications

Version 2017.2
2020-06-26

InterSystems Corporation   1 Memorial Drive   Cambridge MA 02142   www.intersystems.com

Configuring and Using Ensemble Push Notifications
Ensemble Version 2017.2 2020-06-26

Table of Contents

List of Figures

About This Book

This book describes how configure and use the Push Notifications feature to send push notifications to mobile devices using iOS Push Notifications and Google Cloud Messaging. It consists of the following chapters:

- Push Notifications Overview
- Configuring and Using the Push Notifications

For a detailed outline, see the table of contents.

For general information, see *Using InterSystems Documentation*.

In this document, app refers to the mobile app that the user being notified is running on a mobile device and application refers to the Ensemble production that is sending the push notification to the mobile device.

1

Push Notifications Overview

This chapter introduces Push Notifications. It contains the following sections:

- Push Notifications Background
- Push Notification Operations
- Push Notifications Identity Manager
- Sending Push Notifications from Caché

1.1 Push Notifications Background

The Ensemble Push Notifications feature make it possible for an Ensemble production to send push notification messages to iOS and Google devices. This feature makes it possible to easily send notifications to a user with an app running on multiple devices. Your code can send notifications without having to be aware of the number of devices or whether the devices are iOS or Google devices.

Push notifications are typically used to:

- Inform the user that an asynchronous request has finished and data is available.
- Inform the user that a workflow reached a point which requires the user to take an action.
- Raise awareness and ask the user to use the app or a feature of the app.

Ensemble supports the following push notification protocols:

- iOS Push Notifications (APNS)
- Google Cloud Messaging (GCM)

The Push Notifications feature allows an Ensemble production to initiate a notification that notifies the user on a mobile device. For example, an app can provide mobile device access to a system built on Ensemble. In most cases, the mobile app user is initiating actions, either querying to get information or adding or updating data. But in some cases, the Ensemble production initiates the action to notify the mobile device user of some event or condition. The Push Notifications feature enables the Ensemble code to initiate the action and does not depend on the user actively using the app at the time the push notification is sent..

The push notifications protocols do not guarantee that a message will be delivered. Consequently, you should not use these protocols to send important data, but rather to notify the user that the data is available. The user is responsible for initiating an action based on the notification.

In order to enable push notifications, the mobile app registers with the APNS or GCM server and gets a token that identifies the app running on a specific device. The mobile app must also enable push notifications with the device operating system. The details for these operations are different for the APNS and GCM protocols. You can develop the app using any technology available on the device. You can but do not have to use InterSystem's Zen Mojo to create mobile applications. Note that to use push notifications, the mobile app needs to either be a native app or be a hybrid web app and have a native component. A pure web app cannot use push notifications. For information on hybrid applications in Zen Mojo, see Kinds of Zen Mojo Applications in *Using Zen Mojo*.

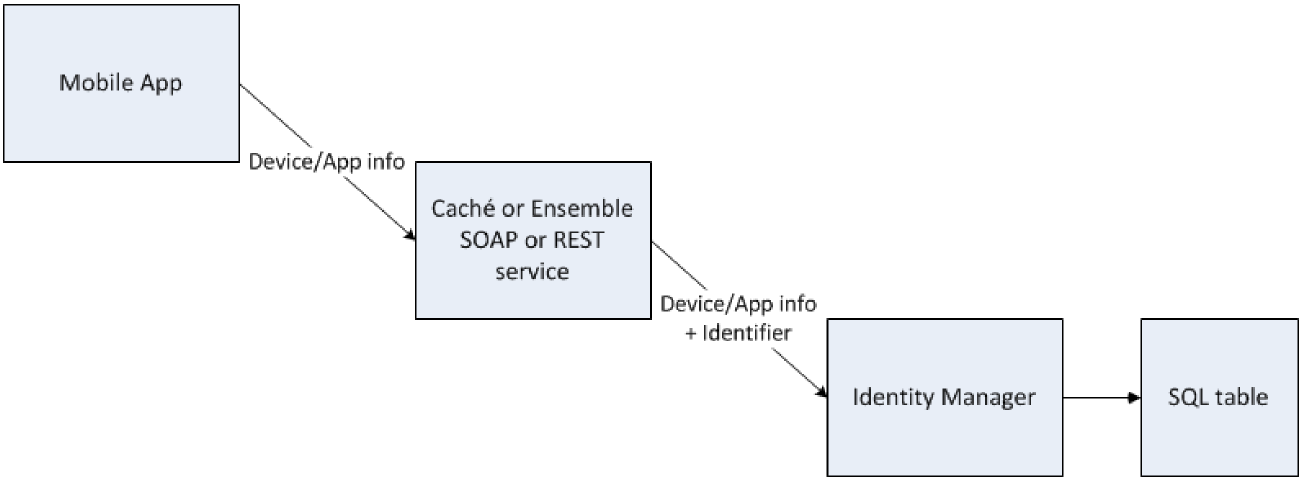
Ensemble's Push Notification allows your Ensemble production to push notifications to a user. If the user has multiple devices the notifications will be sent to all devices. Your code sending the notifications does not need to be aware of whether the user has multiple devices and whether the devices use the APNS or GCM protocols. Your code can send a single notifications to multiple devices for a user. The Push Notifications Identity Manager makes this possible.

The app running on the mobile device makes a call to your Ensemble SOAP or REST service, which registers the app and device with the Identity Manager. It associates a string identifier with the app on the mobile device. You can choose any kind of identifier. For example, you can use the account that identifies the user on your Ensemble production application. You can register multiple devices using the same identifier. When you register the device, you must provide the information required by the APNS or GCM servers to push a notification to the device.

Once one or more devices have been registered with an identifier, your Ensemble code can push a notification by sending a notification message to the Identity Manager with the user's identification. The Identity Manager sends the notifications through a router to one or more Push Notifications operations.

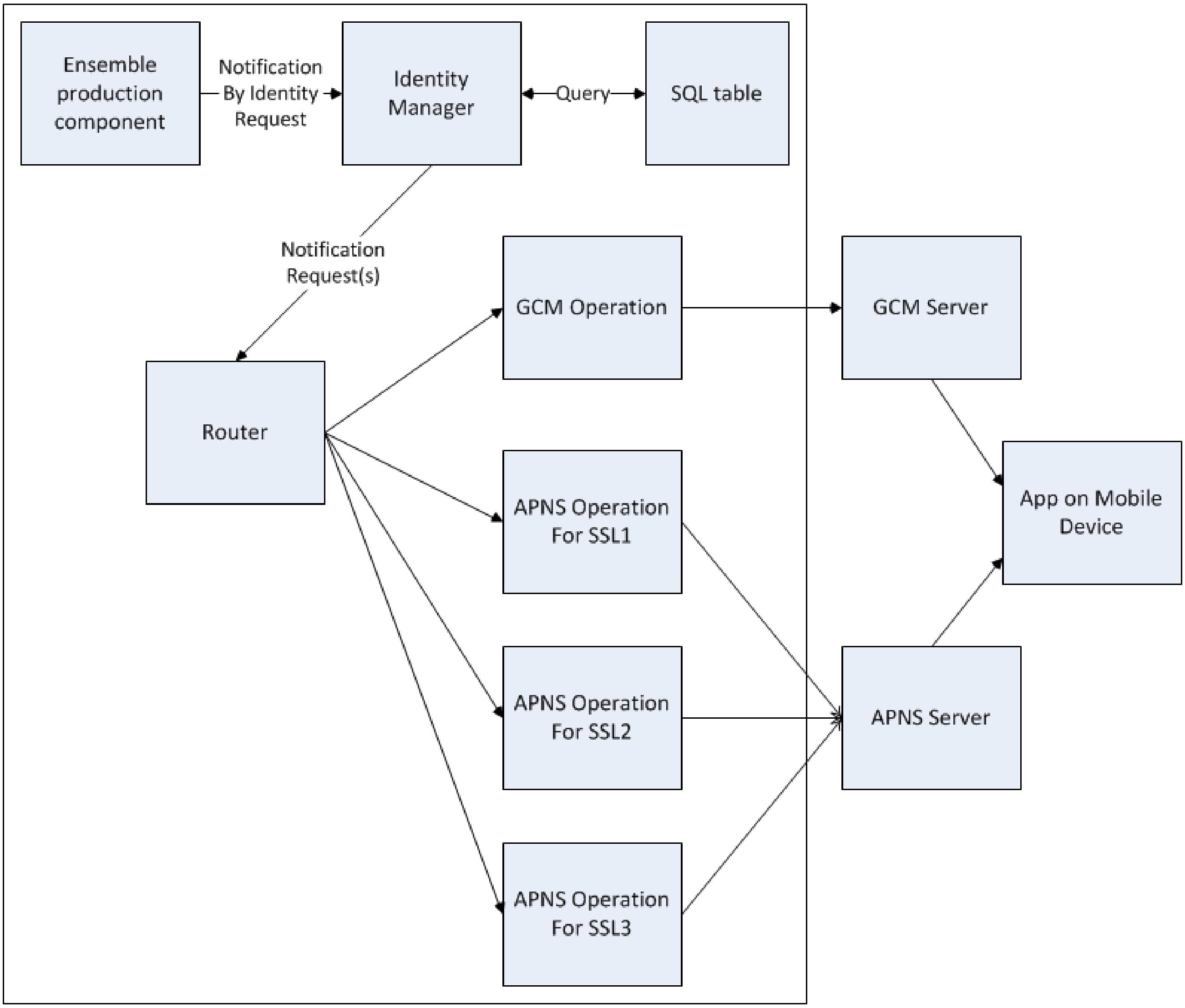
The following illustrates how an app on a mobile device registers with the Push Notifications Identity Manager.

Figure 1–1: Using the Identity Manager to associate an app with an identifier


Mobile App
Device/App info
Caché or Ensemble SOAP or REST service
Device/App info + Identifier
Identity Manager
SQL table

The following illustrates how push notifications are sent to the app on the mobile device. The Identity Manager, GCM Operation, and APNS Operation are provided as built-in components. You can define a router using the Routing Engine and a routing rule.

Figure 1–2: Sending Push Notifications to an app on a mobile device



1.2 Push Notifications Operations

Ensemble provides built-in business operations for Push Notifications. These operations enable you to push a notification to an app on an iOS or Google device. The operations are:

- EnsLib.PushNotifications.APNS.Operation—Business operation that sends the push notifications to the iOS Push Notifications server. Each target app requires an SSL certificate. The operation maintains a connection to the APNS server for that certificate. Calls the underlying %Net.PushNotifications.APNS class to send the motification to the APNS server.
- EnsLib.PushNotifications.GCM.Operation—Business operation that sends the push notification to the Google Cloud Messaging server. Calls the underlying %Net.PushNotifications.GCM class to send the notification to the GCM server. The GCM operation can send a single notification to multiple devices for the same user and app.

1.3 Push Notifications Identity Manager

The Identity Manager allows you to send a push notification to a user without knowing the number or kind of devices that the user has. The Push Notifications Identity Manager maintains a table associating a single ID for a user of a mobile app and associates it with all of the user’s mobile devices. The Identity Manager business process receives messages from other Ensemble production components. It, typically, sends messages to a router that forwards all GCM messages to the GCM operation and forwards each APNS message to the APNS operation configured to handle the specified SSL certificate.

1.4 Sending Push Notifications from ObjectScript

If you are sending Push Notifications from an Ensemble production component, you can send the notification to the Identity Manager or directly to one of the Push Notifications operations. However, if you are generating your Push Notifications in ObjectScript outside of the Ensemble production environment, you can use the Push Notifications AppService to get the notification into the Ensemble environment. This allows you to use the capabilities of the Identity Manager. Note that if you are not using the Identity Manager, your code could also use the %Net.PushNotifications.APNS and %Net.PushNotifications.GCM classes directly.

2

Configuring and Using Push Notifications

This chapter describes how to use Push Notifications services, processes, and operations. This chapter contains the following sections:

- Push Notifications Message Types
- Pushing a Notification Using the APNS Operation
- Pushing a Notification Using the GCMOperation
- Registering Devices with the Identity Manager
- Pushing Notifications Using the Identity Manager
- Using the AppService to Push Notifications

2.1 Push Notifications Message Types

Push Notifications use the following message types:

- EnsLib.PushNotifications.NotificationInfo—contains the information sent to the mobile device.
- EnsLib.PushNotifications.NotificationRequest—contains the NotificationInfo plus routing information that allows the operation to send it to the correct device.
- EnsLib.PushNotifications.NotificationResponse—contains the response from the notification server.
- EnsLib.PushNotifications.IdentityManager.NotificationByIdentityRequest—contains the NotificationInfo plus identity information so that the Identity Manager can find the device information.
- EnsLib.PushNotifications.IdentityManager.NotificationByIdentityResponse—contains the response from the Identity Manager.

2.1.1 NotificationInfo

The EnsLib.PushNotifications.NotificationInfo class contains the information that is in the notification that is pushed to the device.

Property	Description
AlertNotification	String to be displayed to the user as a notification.

Property	Description
BadgeNotification	Integer that is displayed to the user in an icon. For example, this can be used to indicate the number of unread messages.
SoundNotification	String that identifies a sound to be played on the device.
ExpiresUTC	Timestamp that indicates when the notification expires. If the notification is not delivered to the device by this time, the server will not send it to the device.
Data	String containing name-value pairs in JSON form. Data contains the names and values that are used by the notification handling code in the mobile app.
UrlNotification	Reserved for future use.
CollapseKey	Reserved for future use.

2.1.2 NotificationRequest

The EnsLib.PushNotifications.NotificationRequest class includes the EnsLib.PushNotifications.NotificationInfo class and, additionally, specifies the device(s) to receive the notification.

Property	Description
AppIdentifier	String that is only used for GCM notifications and identifies the App that the notification is associated with.
Identifiers	String that specifies the device that are to get the notification. For APNS, the device is specified by a device token. For GCM, the device is specified as a registration ID.
Service	Identifies whether the device is a GCM or APNS device. Service has a value of “GCM” or “APNS”.

2.1.3 NotificationResponse

The EnsLib.PushNotifications.NotificationResponse class contains the information returned from the APNS or GCN server.

Property	Description
DeliveredAtUTC	Timestamp that specifies the time that the notification was sent to the APNS or GCM server.
MessageIds	String that contains a list of message Ids returned by the server.
MulticastId	String that contains the multicast Id returned by GCM server.

2.1.4 NotificationByIdentityRequest

The EnsLib.PushNotifications.IdentityManager.NotificationByIdentityRequest class includes the EnsLib.PushNotifications.NotificationInfo class and, additionally, specifies the identity registered in the Identity Manager that is to receive the notification.

Property	Description
AssociatedIdentity	String that specifies the identity to receive the notification.

The AssociatedIdentity can be any string as long as it is unique in the table. Typically, this is a login name or some other string that identifies a user in the application.

2.1.5 NotificationByIdentityResponse

The EnsLib.PushNotifications.IdentityManager.NotificationByIdentityResponse class returns the number of NotificationRequest messages sent to the target. This is the same as the number of devices registered for the AssociatedIdentity.

Property	Description
NotificationResponse	Integer that specifies the number of NotificationRequest messages sent to the target of the Identity Manager.

2.2 Pushing a Notification Using the APNS Operation

The EnsLib.PushNotifications.APNS.Operation sends the notification request to the APNS server to forward to the specified device. The APNS server pushes the notification to one device for each call. It does not return any information other than if an error occurs, an indication that the error has occurred.

You configure the following settings on the APNS Operation:

Setting	Description
PushServerAddress	Hostname of the Apple Push Notification Server. Value is: `gateway.sandbox.push.apple.com`
PushServerPort	Port for the Apple Push Notification Server interface. Value is: `2195`
SSLConfig	Configuration name in the Caché table of SSL configurations. The SSL is associated with the app and must be one provided by Apple for the APNS service. For example, can have the configuration name value: `MyAppAppleSSL`
ConnectTimeout	SSL Connection timeout period in seconds. The connection is terminated after this period of inactivity. Default value: `30`
ResponseTimeout	Time period in seconds to wait for a response to a connection request. Default value: `5`
NotificationProtocol	Specifies the APNS notification protocol. Has one of the following values: • `Simple`—Simple Notification Protocol, which does not return any value and does not notify you if the protocol message causes an error. • `Enhanced`— Enhanced Notification Protocol, which returns values if the protocol message contains an error. This protocol does not return a value for messages without errors. • `Modern`—Reserved for future use. The APNS operation does not support the Modern Notification Protocol.

The SSL is associated with the remote app, but the same SSL is used for every user of the app. If your application is pushing notifications to a single app, you only need one SSL and one APNS operation in your production. If your application is pushing notifications to multiple apps running on Apple devices, you need one SSL for each app and must have one corresponding APNS operation configured with that SSL for each different app. Note that if you are using a router, you must direct each notification to the correct APNS operation. For example, you could create a lookup table that relates the device token to its corresponding operation and then use that lookup table in a routing rule.

The APNS protocol does not return any values other than error statuses for the Enhanced protocol. Consequently, the only way to detect that an Enhanced protocol message has not failed is to wait to detect a return error response. If no error response is received in a reasonable time, the operation assumes that the protocol message has succeeded. You specify the time to wait for an error response in the **ResponseTimeout** setting. The value should be long enough to capture any error response from the APNS server, but since it cannot handle another request until after the time out, the value should not be any longer than needed. With the default value of 5 seconds, an operation instance can only handle one request every 5 seconds; consequently, you should set the pool size large enough to handle the number of requests that typically occur within the time out period.

If you are using the Simple protocol, it does not return any value even if there is an error. With the Simple protocol, there is no advantage in setting **ResponseTimeout** to anything other than the minimum value or 0 seconds.

The SSL certificate is supplied by the Apple server. Each app requires a unique SSL.

2.3 Pushing a Notification Using the GCM Operation

You configure the following settings on the GCM Operation:

Setting	Description
PushServer	URL for the Google Cloud Messaging REST interface. Value is: `https://android.googleapis.com/gcm/send`
SSLConfig	Configuration name in the Caché table of SSL configurations. For example, can have the value: `MyAppSSL`
Timeout	REST response timeout period in seconds. Default value: `30`
NotificationProtocol	Specifies the GCM notification protocol. Has one of the following values: • `HTTP`—HTTP REST protocol. • `XMPP`—Reserved for future use. The GCM operation does not support the XMPP always connected, bi-directional protocol.

2.4 Registering Devices with the Identity Manager

You can register one or more pairs of device tokens and applications with an Associated Identity string identifier. Your application can use this string identifier to push a notification to all the devices associated with the identifier. The EnsLib.PushNotifications.IdentityManager.DeviceTracking class provides the following methods:

- **AssociateDeviceWithAppToken**—Associates a device token, AppID, and service with an AssociatedIdentity string identifier.
- **DisassociateDeviceWithAppToken**—Removes the association of the device token, AppID, and service with the AssociatedIdentity string identifier.

- **FindDeviceByAppToken**—Finds all devices associated with the specified AssociatedIdentity string identifier. Returns a DeviceTracking object.
- **FindDeviceByDeviceAndAppIds**—Finds all devices with the specified device token and AppId. Returns a DeviceTracking object.

2.5 Pushing Notifications Using the Identity Manager

When the Identity Manager receives a NotificationByIdentityRequest message it does the following:

1. It queries the SQL table to find the devices associated with the AssociatedIdentity in the message.
2. If there are one or more devices returned by the query, it creates a NotificationMessage for each record returned by the query. In the NotificationMessage:
   - It sets NotificationInfo to the NotificationInfo in the incoming NotationByIdentityRequest.
   - It sets Identities to the device token in the record.
   - It sets AppId to the AppID in the record, if it is defined.
   - It sets Service to the Service in the record.
3. It sends each message to the target component.

Note: The IdentityManager receives one NotificationByIdentityRequest and creates one NotificationMessage for each associated device. Each NotificationMessage it sends has one device token in the Identities list. Although the GCM server allows you to push a single notification to multiple devices for a user in one call, the IdentityManager does not use this capability. You can send a single notification to multiple GCM devices using the lower-level calls, but not using the IdentityManager.

2.6 Using the AppService to Push Notifications

If you are generating your Push Notifications in ObjectScript outside of the Ensemble production environment, you can use the Push Notifications AppService to get the notification into the Ensemble environment. The AppService is a gateway into Ensemble. To call it, do the following:

1. Add the AppService to the Ensemble production.
2. Configure the target to send the message to the Identity Manager, a router, or directly to a Push Notifications operation.
3. In your code, create and populate an EnsLib.PushNotifications.NotificationRequest message.
4. Call the AppService **SendSync** method passing the NotificationRequest as the parameter.