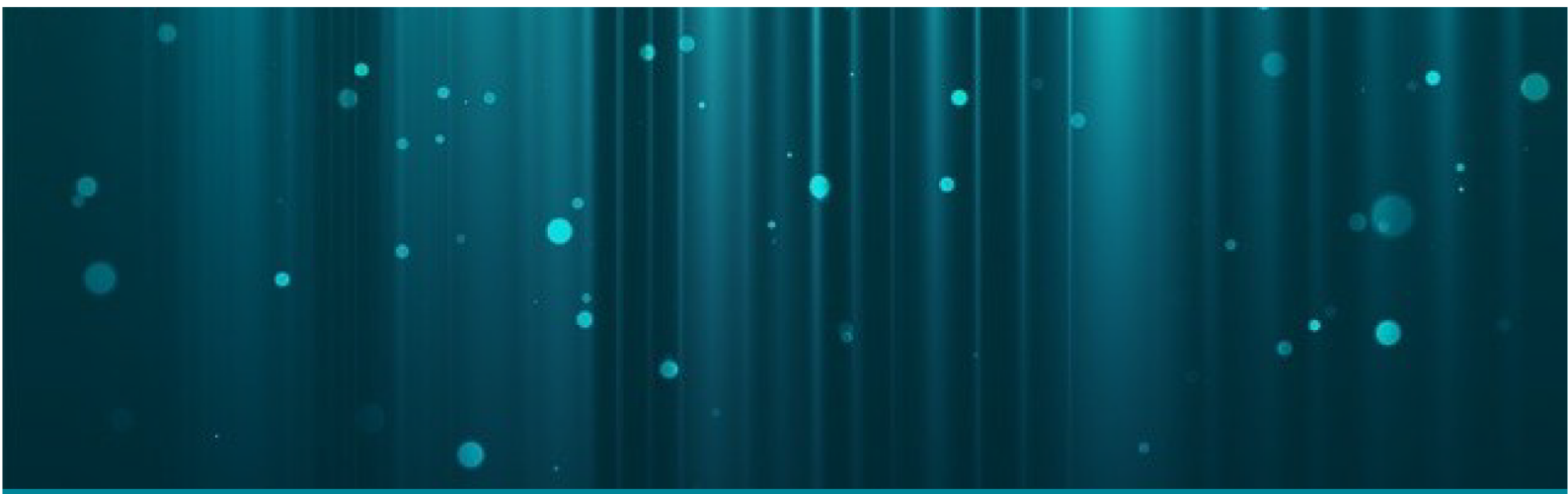
OneSpan



Notification SDK

Integration Guide

Version: 4.30

Copyright Notice

Copyright © 2013–2022 OneSpan North America, Inc. All rights reserved.

Trademarks

OneSpan™, DIGIPASS® and CRONTO® are registered or unregistered trademarks of OneSpan North America Inc., OneSpan NV and/or OneSpan International GmbH (collectively "OneSpan") in the U.S. and other countries.

OneSpan reserves all rights to the trademarks, service marks and logos of OneSpan and its subsidiaries.

All other trademarks or trade names are the property of their respective owners.

Intellectual Property

OneSpan Software, documents and related materials (“Materials”) contain proprietary and confidential information. All title, rights and interest in OneSpan Software and Materials, updates and upgrades thereof, including software rights, copyrights, patent rights, industrial design rights, trade secret rights, sui generis database rights, and all other intellectual and industrial property rights, vest exclusively in OneSpan or its licensors. No OneSpan Software or Materials may be downloaded, copied, transferred, disclosed, reproduced, redistributed, or transmitted in any form or by any means, electronic, mechanical or otherwise, for any commercial or production purpose, except as otherwise marked or when expressly permitted by OneSpan in writing.

Disclaimer

OneSpan accepts no liability for the accuracy, completeness, or timeliness of content, or for the reliability of links to and content of external or third party websites.

OneSpan shall have no liability under any circumstances for any loss, damage, or expense incurred by you, your company, or any third party arising from the use or inability to use OneSpan Software or Materials, or any third party material made available or downloadable. OneSpan will not be liable in relation to any loss/damage caused by modification of these Legal Notices or content.

Reservation

OneSpan reserves the right to modify these Notices and the content at any time. OneSpan likewise reserves the right to withdraw or revoke consent or otherwise prohibit use of the OneSpan Software or Materials if such use does not conform to the terms of any written agreement between OneSpan and you, or other applicable terms that OneSpan publishes from time to time.

Contact us

Visit our website: **https://www.onespan.com**
Resource center: **https://www.onespan.com/resource-center**
Technical support and knowledge base: **https://www.onespan.com/support**

If there is no solution in the knowledge base, contact the company that supplied you with the OneSpan product.

Date: 2022-10-24

Contents

1 Introduction 1

1.1 OneSpan Mobile Security Suite Documentation 2

2 What's new in the Notification SDK 4

2.1 Version 4.30.0 5

2.2 Version 4.29.2 6

3 Usage of the Notification SDK 7

3.1 The Push Notification Process 8

3.2 Overview of the OneSpan Notification SDK 11

3.3 Migration from Version 4.18.x and Previous Versions 14

4 Notification SDK Client 16

4.1 Integrate the Notification SDK Client 17

5 Notification SDK Server 24

5.1 Integrate the Notification SDK Server Sample 25

5.2 Migration from Version 4.24.x and Previous Versions on .NET 27

A Appendix: Platform-Specific Steps to Integrate the OneSpan Notification SDK 29

A.1 Enable Push Notification in a Google project 30

A.2 Enable Push Notification in an iOS project with token-based authentication scheme 32

A.3 Enable Push Notification in an iOS project with certificate-based authentication scheme 33

B Appendix: FAQ 34

Index 36

Figures

Figure 1: Registration process iOS 9

Figure 2: Notification delivery process (iOS) 10

Figure 3: Registration process with OneSpan Notification SDK 12

Figure 4: Notification delivery process with OneSpan Notification SDK 13

Figure 5: Integrate the Notification SDK Client (Android) 22

Figure 6: Integrate the Notification SDK Client (iOS) 23

Tables

Table 1: Properties in the onespan-notifications.properties file 19

Procedures

To use the Notification SDK Client in your Android project 17

To use the Notification SDK Client in your iOS project 22

To use the Notification SDK Server sample in your Java project 25

To use the Notification SDK Server in your .NET project 26

To update the Notification SDK 27

To get the Android project key 30

To get a google-services.json file 30

To migrate from Google Cloud Messaging (GCM) to Firebase Cloud Messaging with the Notification SDK 31

To use the Push Notification feature in your iOS application with a token-based authentication scheme 32

To use the Push Notification feature in your iOS application with a certificate-based authentication scheme 33

Introduction

1

Welcome to the Notification SDK Integration Guide! This guide describes how to integrate the Notification SDK into any application.

This guide provides information about:

- Functions of the Notification SDK, with parameter and method descriptions
- Integration steps

1.1 OneSpan Mobile Security Suite Documentation 2

1.1 OneSpan Mobile Security Suite Documentation

The OneSpan Mobile Security Suite product documentation comprises the following guides:

- *OneSpan Mobile Security Suite Product Guide*: Provides an overview of OneSpan Mobile Security Suite and its components.
- *Biometric Sensor SDK Integration Guide*: Provides instructions to integrate the Biometric Sensor SDK.
- *Client Device Data Collector SDK Integration Guide*: Provides instructions to integrate the Client Device Data Collector SDK.
- *Device Binding SDK Integration Guide*: Provides instructions to integrate the Device Binding SDK.
- *DIGIPASS SDK Integration Guide*: Provides instructions to integrate the DIGIPASS SDK.
- *Digipass Software Advanced Provisioning Protocol SDK Integration Guide*: Provides instructions to integrate the Digipass Software Advanced Provisioning Protocol SDK.
- *FIDO Authentication Solution Guide*: Provides instructions to integrate the OneSpan FIDO Authenticator Specific Modules into a FIDO application and a FIDO Server. It also provides information on how to configure a FIDO Server.
- *Image Generator SDK Integration Guide*: Provides instructions to integrate the Image Generator SDK.
- *Image Scanner SDK Integration Guide*: Provides instructions to integrate the Image Scanner SDK.
- *Notification SDK Integration Guide*: Provides instructions to integrate the Notification SDK.
- *Orchestration SDK Integration Guide*: Provides instructions to integrate the Orchestration SDK.
- *Root Detection SDK Integration Guide*: Provides instructions to integrate the Root Detection SDK.

- *Secure Messaging SDK Client Integration Guide*: Provides instructions to integrate the Secure Messaging SDK Client.
- *Secure Messaging SDK Server Integration Guide*: Provides instructions to integrate the Secure Messaging SDK Server.
- *Secure Storage SDK Integration Guide*: Provides instructions to integrate the Secure Storage SDK.
- *White-Box Cryptography SDK Integration Guide*: Provides instructions to integrate the White-Box Cryptography SDK.

What's new in the Notification SDK 2

This section provides an overview of changes introduced in the Notification SDK to facilitate the integration of the SDK and provide information on backward compatibility.

2.1 Version 4.30.0 5

2.2 Version 4.29.2 6

2.1 Version 4.30.0

2.1.1 Android

Target API increased to Android 12 (API 31)

To meet new requirements for apps published on the Google Play Store, the target version of the SDK has been increased to Android 12 with API 31 or higher. This change is needed to avoid APK rejection caused by security issues found with the older API versions.

Minimum supported version increased to Android 6 (API 23)

The minimum supported version has been increased to fully support new features and devices. Deprecated code has been replaced and simplified to be compatible with API 23 or higher.

2.1.2 iOS

Increased the minimum supported version to iOS 13

The minimum supported version has been increased to fully support all ARM64 devices and SwiftUI features. Deprecated code has been removed and replaced with code fully supporting iOS 13 or higher.

Fixed internal error conversion issues

Error codes converted between Objective-C and Swift need to have the `NSCustomError` setting implemented to display the correct value. Without this setting, the wrong codes could be displayed from the enum. The setting has been implemented in all the remaining error handling objects.

2.2 Version 4.29.2

2.2.1 iOS

Change of framework name to *MSSNotificationClient*

The framework name has been changed from *NotificationClientSDK* to *MSSNotificationClient*. To use the new Objective-C API, replace all previous `NotificationClientSDK` imports with `#import <MSSNotificationClient/MSSNotificationClient.h>`.

API updates – NSErrors API support for Objective-C added

The Objective-C API points no longer throw an `NSException`. All API points that previously would throw such an `NSException` now require an `NSError` pointer. If an error occurs, it will be attached to the pointer that is provided as a parameter. Refer to the Objective-C sample included in the product package for full code examples.

Usage of the Notification SDK 3

This section contains an overview of the Notification SDK and information about the functions of the SDK.

3.1 The Push Notification Process 8

3.2 Overview of the OneSpan Notification SDK 11

3.3 Migration from Version 4.18.x and Previous Versions 14

3.1 The Push Notification Process

The OneSpan Notification SDK allows you to send **Push notification**[1] messages to your mobile applications.

The push notification process involves the following entities:

- The client device
- Your back-end server
- The push notification provider (e.g. Apple, Google, Microsoft)

> **CAUTION:** The delivery of notifications relies on the availability of the push notification provider. Thus, the notification process does not include any warranty of delivery or delay. Notifications should not be used as a safe end-to-end delivery system.

3.1.1 Client registration

During application start, the client device is registered with the push notification provider. When registration is completed, the client application gets a unique notification identifier to send (usually via HTTP) to the back-end server for later use. **Figure 1** illustrates the registration process on iOS.

[1]Push notifications are clickable pop-up messages that are displayed outside an app. They are pushed from the server the app uses to the user's device.

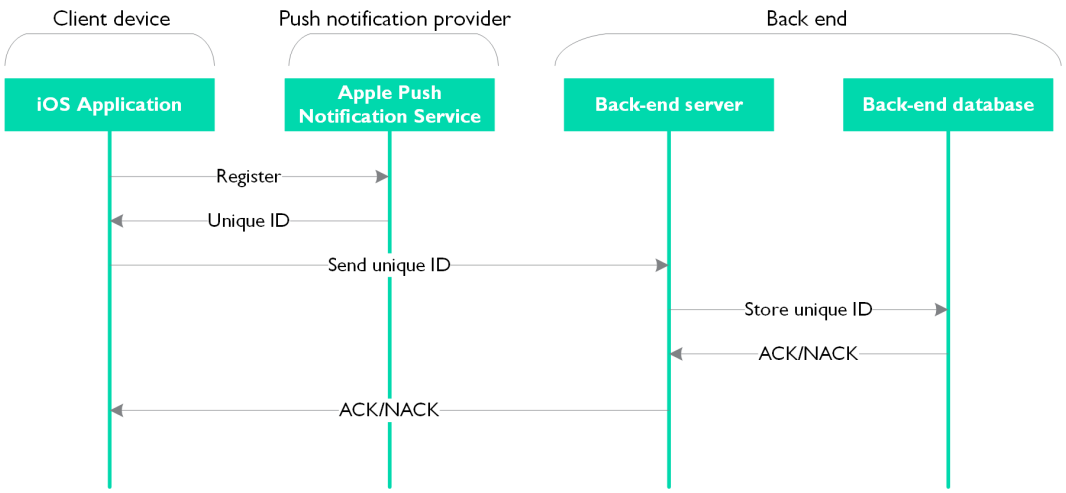


Figure 1: Registration process iOS

> **NOTE:** The registration process varies depending on the push notification provider. A specific implementation is required for each targeted platform.

3.1.2 Notification delivery

When the server needs to send a notification to a specific client device, the server:

1. Retrieves the unique identifier from the database.
2. Creates a notification with the specific content.
3. Sends the notification to the push notification provider. The server uses the appropriate credentials together with the unique identifier.

The push notification provider delivers the notification to the client device. The client device or application then creates the notification that is displayed to the user. **Figure 2** illustrates the registration process on iOS.

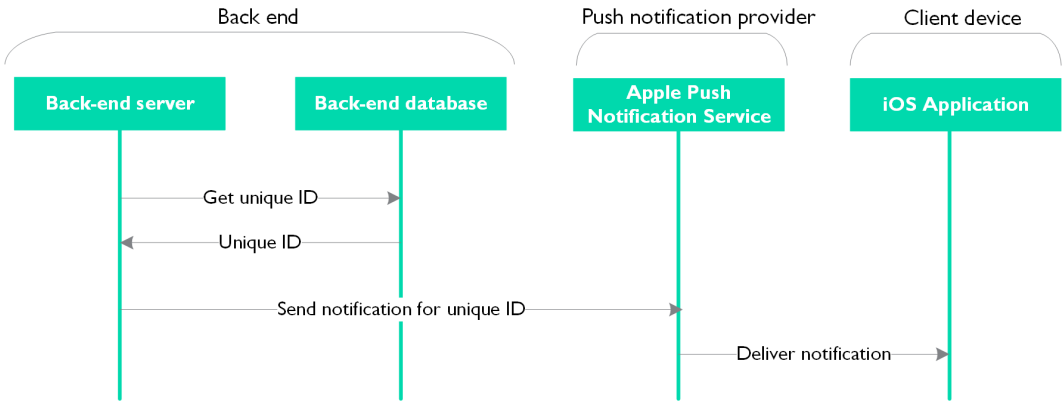


Figure 2: Notification delivery process (iOS)

Notification delivery is immediate in most cases. However, depending on the behavior of the push notification provider and the reachability of the client device, delivery may also be deferred.

NOTE: The notification delivery process varies depending on the push notification provider. A specific implementation is required for each targeted platform.

3.2 Overview of the OneSpan Notification SDK

The OneSpan Notification SDK provides an abstraction layer that allows notification support for mobile applications. On the client-side, the registration process is platform-dependent. On the server-side, you can send notifications with a simple function, independent of the target platform.

The Notification SDK Client can be used on a variety of devices and supports the following platforms:

Android:

- Minimum: Android 6 (API 23) or higher
- Target: Android 12 (API 31)

iOS

- Minimum: iOS 13 or higher
- Swift 5.0 or higher
- Xcode 13 or higher

3.2.1 Registration process

The Notification SDK Client handles registration with a unified registration function. When the registration process is completed, the client application and the back end receive a **Notification Identifier**[1]. **Figure 3** illustrates the registration process with the SDK.

[1]Unique hexadecimal string with a maximum length of 2064 characters. The Notification SDK assigns this identifier to the client application - it is unique to the client device, regardless of the platform. It must be sent to the back end for later use.

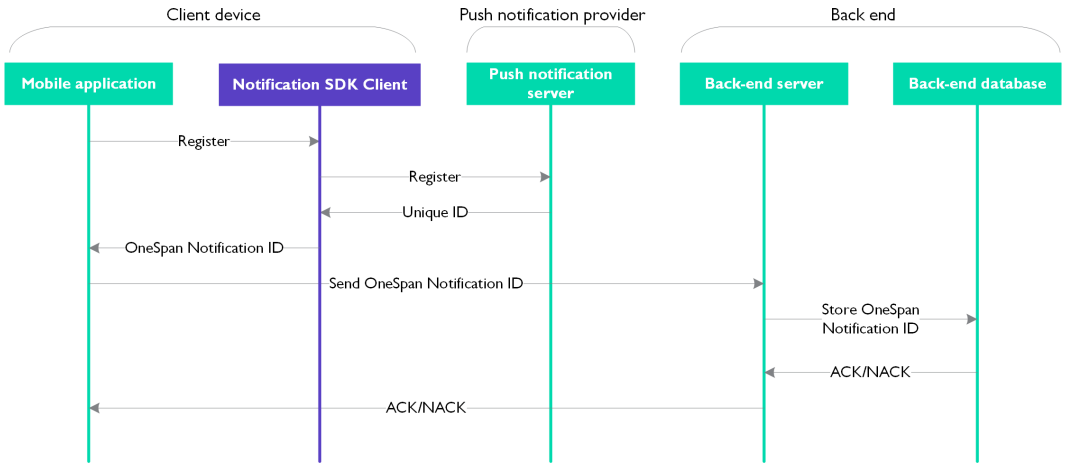


Figure 3: Registration process with OneSpan Notification SDK

OneSpan Notification Identifier

The OneSpan Notification Identifier is unique to the client device, regardless of the platform (i.e. Android, iOS etc.). It is a unique hexadecimal string with a maximum length of 2064 characters that the Notification SDK Client assigns to the client application. It must be sent to the back end for later use.

When the Notification SDK Server sends a notification, it uses the OneSpan Notification Identifier to address the correct client device on the dedicated **Push notification**[1] provider.

Notification content

The payloads the OneSpan Notification SDK transmits are strings. Depending on the target platform, different size limitations may apply. For more information about these limitations, refer to the respective platform documentation.

Notification delivery

The Notification SDK Server uses the OneSpan Notification Identifier to recognize the target device platform and apply the correct sending procedure. This comprises the following:

[1]Push notifications are clickable pop-up messages that are displayed outside an app. They are pushed from the server the app uses to the user's device.

1. Connect to the correct push notification provider with the appropriate credential.
2. Create a notification based on specific push notification provider requirements.
3. Send the notification.

On the client-side, the Notification SDK Client provides a unified way to receive and parse incoming notifications.**Figure 4** illustrates the notification delivery process.

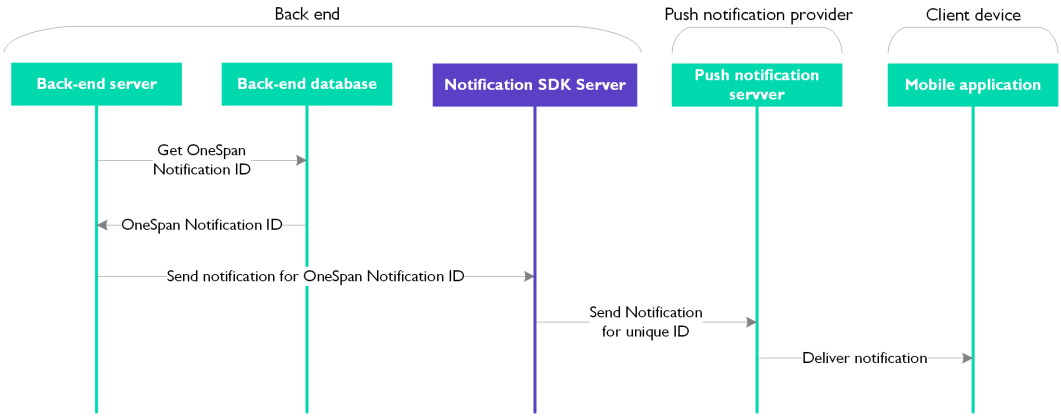


Figure 4: Notification delivery process with OneSpan Notification SDK

3.3 Migration from Version 4.18.x and Previous Versions

In April 2018, Google deprecated Google Cloud Messaging (GCM). The GCM server and corresponding client APIs are deprecated and were removed in April 2019. This requires a migration of GCM apps to Firebase Cloud Messaging (FCM).

3.3.1 Before you begin

Before you start the migration, note:

- GCM and FCM SDKs cannot co-exist within the same mobile application.
- GCM tokens retrieved via `GoogleCloudMessaging.register()` or `InstanceID.getToken()` continue to work in FCM without any modification or renewal.

Currently, the Notification SDK uses the FCM URLs to send and receive notifications on Android. Those modifications only affect the Android side on both the client- and the server.

> **NOTE:** FCM can manage both GCM and FCM tokens; therefore, the server-side must be migrated *before* the client side.

3.3.2 Android server-side

There are two ways to migrate the server-side:

- **Via the project number equivalent or legacy key.** This is the easiest way to complete the migration, and requires less effort for the migration. To migrate the server-side, update the server `.jar` dependencies. The new library connects automatically to the FCM back end to send the notification.
- **Via OAuth and the Firebase account JSON file.** This is the quickest way to connect to the Firebase back end, and the cleanest solution if you do not have an existing code base.

For more detailed information on those steps, see **4.1.1 Integrate the Notification SDK Client with Android** and **A.1 Enable Push Notification in a Google project**.

3.3.3 Android client-side

Since GCM and FCM SDKs cannot co-exist within the same mobile application, the migration requires code modifications.

Notification SDK Client

4

The Notification SDK Client provides numerous client functions to integrate the Notification SDK on various platforms. It serves to receive the notifications and obtain the unique identifier the server requires to send the notification to the intended application on the correct mobile device.

4.1 Integrate the Notification SDK Client 17

4.1 Integrate the Notification SDK Client

This chapter provides an overview of the Notification SDK Client functions and instructions to integrate the SDK on various platforms.

4.1.1 Integrate the Notification SDK Client with Android

The following procedure allows you to integrate and use the Notification SDK Client in your Android project.

▸ To use the Notification SDK Client in your Android project

1. Copy `NotificationSDKClient.aar` from the OneSpan Mobile Security Suite package to your project.
2. Add the Google Play services project as a dependent library.

   For more information, refer to **https://firebase.google.com/docs/android/setup**.
3. Configure your manifest to be able to receive a notification.

EXAMPLE: `AndroidManifest.xml`

```
<application>
  <activity android:name="ActivityName">
  ...
<!-- This intent filter is used to select the activity that
will be launched when the user clicks on a notification.
The MIME type is customizable. Make sure it matches the
“notificationMIMEType” property in the assets/onespan-
notifications.properties
file. -->
    <intent-filter>
```

```
10.       <action android:name="android.intent.action.VIEW"/>
11.       <category android:name=
           "android.intent.category.DEFAULT"/>
12.       <data android:mimeType=
           "application/vnd.com.vasco.notification.NOTIFICATION_
            ACTIVITY"/>
13.     </intent-filter>
14.   </activity>
15.   <meta-data
       android:name="com.google.android.gms.version"
       android:value="@integer/google_play_services_version"/>
16. </application>
```

NOTE: If you target Android 13, you need to declare the **POST_NOTIFICATIONS** permission in the `AndroidManifest.xml` of the application.

NOTE: As of version 4.17.2 of the OneSpan Notification SDK, the library is delivered as an AAR file.

The AAR file embeds an `AndroidManifest.xml` file; thus less information is required in the `AndroidManifest.xml` of the application.

4. Create a `onespan-notifications.properties` file in the `assets` folder.

   This file must contain several properties, each of which has a *key/value* pair. The *key* and *value* are separated by an equal sign (=). For example:

   ```
   notificationMIMEType=application/vnd.com.vasco.notification.NOTIFICATION_ACTIVITY
   ```

   Table 1 lists the mandatory and optional properties to include in the file:

Table 1: Properties in the `onespan-notifications.properties` file

Key	Mandatory	Description
`notificationMIMEType`	✓	Mime type that is used when a notification is received. This must be the same value as the one specified in the manifest.
`notificationIconName`	✓	Icon displayed in the notification center when a notification is received. This icon file must be in a `.png` format and must be added as a resource.
`notificationIconNameLollipop`		Icon displayed in the notification center when a notification is received. This property is used on Android 5.0 and later. The icon can only be white and transparent, must be in a `.png` format and must be added as a resource.

Table 1: Properties in the onespan-notifications.properties file (continued)

Key	Mandatory	Description
`notificationIconBackgroundColorLollipop`		Background color used for the notification icon when a notification is displayed on the notification drawer. This property is used on Android 5.0 and later. Supported formats are: • #RRGGBB • #AARRGGBB, • or you can use the following color terms: `red, blue, green, black, white, gray, cyan, magenta, yellow, lightgray, darkgray, grey, lightgrey, darkgrey, aqua, fuchsia, lime, maroon, navy, olive, purple, silver, teal.`
`notificationOpenInForeground`		Optional Boolean switching behavior of the notifications that are received while the application is in foreground. Default value: **false**.

EXAMPLE: `onespan-notifications.properties`:

```
notificationMIMEType=application/vnd.com.vasco.notification.NOTIFICATION_ACTIVITY
notificationIconName=ic_notification.png
notificationIconNameLollipop=ic_notification_lollipop.png
notificationIconBackgroundColorLollipop=#848484
notificationOpenInForeground=true
```

5. Retrieve `google-services.json` from your Firebase project. This file is mandatory to enable FCM from your Android project. For more information about obtaining this file, see **A.1 Enable Push Notification in a Google project**. This file must be downloaded from your Firebase project:
   - Go to **Console** -> **Select Project** -> **Project Settings** and open the **General** tab.
   - Retrieve `google-services.json` and replace the `.json` file inside the app sample project repository.

You are now ready to use the Notification SDK Client.

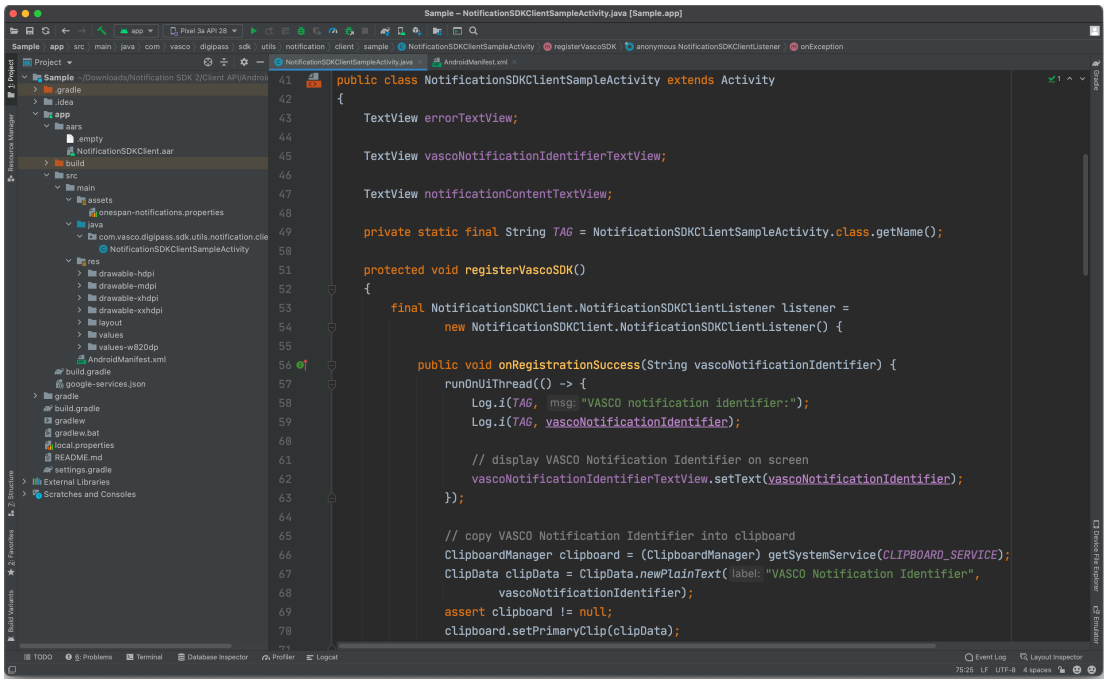


Figure 5: Integrate the Notification SDK Client (Android)

4.1.2 Integrate the Notification SDK Client with iOS

The following procedure allows you to integrate and use the Notification SDK Client in your iOS project.

▸ To use the Notification SDK Client in your iOS project

1. Link `MSSNotificationClient.xcframework` from the OneSpan Mobile Security Suite package to your Xcode project (linked framework and libraries).
2. In your `.plist` file, add the *App downloads content in response to push notifications* mode (raw value: *<remote-notification>*) to the list associated with the *Required background modes* property (raw value: *UIBackGroundModes*) .
3. Make sure your application bundle identifier matches that in the provisioning profile that has been created for this specific application with the *Remote notifications* capability.

You are now ready to use the Notification SDK Client.

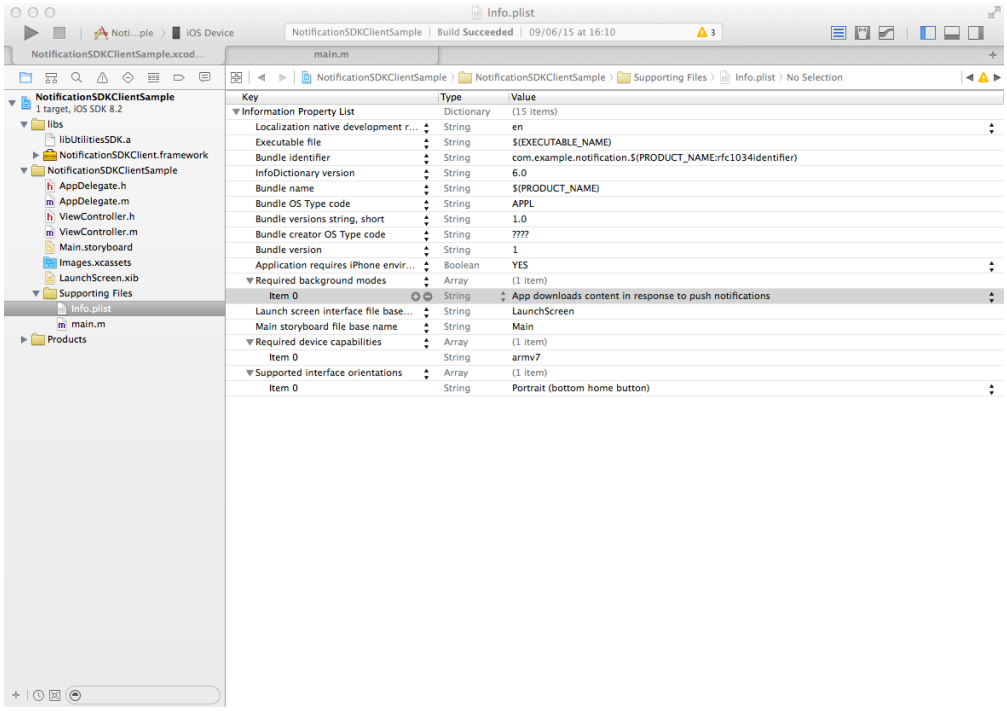


Figure 6: Integrate the Notification SDK Client (iOS)

Notification SDK Server

5

The Notification SDK Server provides numerous server functions to integrate the Notification SDK on various platforms. It is used to send raw-data messages to the Notification SDK Client based on a previously received unique identifier.

5.1 Integrate the Notification SDK Server Sample 25

5.2 Migration from Version 4.24.x and Previous Versions on .NET 27

5.1 Integrate the Notification SDK Server Sample

This chapter provides instructions to integrate the Notification SDK Server sample in Java and .NET.

> **NOTE:** The Notification SDK Server requires **Java 8**.

5.1.1 Integrate the Notification SDK Server in a Java project

The following procedure allows you to integrate and use the Notification SDK Server in your Java project.

▸ To use the Notification SDK Server sample in your Java project

1. Link the libraries from the OneSpan Mobile Security Suite package to your project's `classpath`.
2. In `NotificationSDKServerSample.java`, replace the following variables according to your device:

   #### Android

   - Adjust the `ANDROID_VASCO_NOTIFICATION_IDENTIFIER` variable to set up the OneSpan Notification Identifier for your device. You can retrieve it from your client sample integration after the notification service registration.
   - Adjust the `ANDROID_PROJECT_KEY` variable to set up your Android project key (API key). You can retrieve it from the Google API Console.

   #### iOS

   - Adjust the `IOS_VASCO_NOTIFICATION_IDENTIFIER` variable to set up the **Notification Identifier**[1] for your device. You can retrieve it from your client sample integration after the notification service registration.

[1]Unique hexadecimal string with a maximum length of 2064 characters. The Notification SDK assigns this identifier to the client application - it is unique to the client device, regardless of the platform. It must be sent to the back end for later use.

- Update the Boolean variable `IOS_USE_DEVELOPMENT_SERVER` as needed. Note that this also depends on the used PKCS #8 key.
- Adjust the `IOS_P8_KEY_PATH` variable to set up the path of your PKCS #8 key.
- Adjust the `IOS_KEY_ID` variable to set up the password of your PKCS #8 key.
- Adjust the `IOS_TEAM_ID` variable to set up the identifier of the team that is responsible for the creation of the PKCS #8 key.

You are now ready to use the Notification SDK Server.

5.1.2 Integrate the Notification SDK Server in a .NET project

The Notification SDK Server contains a solution sample to demonstrate how to integrate the SDK in your .NET project. describes the necessary steps for the integration.

▸ To use the Notification SDK Server in your .NET project

1. Copy `NotificationSDKServer.dll` from the OneSpan Mobile Security Suite package in the `lib` folder.

> **NOTE:** The token-based authentication scheme is the only available method to send push notifications to iOS devices.

You are now ready to use the Notification SDK Server.

5.2 Migration from Version 4.24.x and Previous Versions on .NET

Apple updated the Push Notification service, and as of November 2020, the Apple Push Notification service (APNs) no longer supports the legacy binary protocol. Apple recommends an update to the HTTP/2-based APNs provider API. When you use HTTP/2, you must also update the framework to use PKCS #8 because of constraints when PKCS #12 is used with HTTP/2: It is not possible to use a PKCS #12 client certificate for notifications on an HTTP/2 connection in .NET, and to deliver a Windows library that forces the usage of this certificate. Apple Push Notification service (APNs) requires to only use cipher methods that are available from Windows Server 2016.

5.2.1 Update the Notification SDK

> **NOTE:** This only concerns the Notification SDK .NET server, the client is not affected.

To use the new library, you must add references to the Microsoft libraries and the usage of the .NET NuGet package manager.

▸ To update the Notification SDK

1. Modify the integrating project. From the package sample solution,
    a. Add every `dependentAssembly` element from the app configuration to the app configuration file of your project.
    b. Add every `package` element from the packages configuration file to the package configuration file of your project.
    c. Add every `Reference` element from the `csproj` file to the `csproj` file of your project.
2. Install the following libraries:
    a. `System.Buffers` version 4.5.0
    b. `System.Diagnostics.DiagnosticSource` version 4.3.0

c. `System.Memory` version 4.5.3

d. `System.Net.Http.WinHttpHandler` version 4.7.0

Appendix: Platform-Specific Steps to Integrate the OneSpan Notification SDK

A

Depending on your platform, you need to complete additional platform-specific steps when you integrate the OneSpan Notification SDK.

A.1 Enable Push Notification in a Google project

OneSpan proposes two solutions to enable **Push Notification**[1] on Android. To connect to Firebase Cloud Messaging (FCM), you can use one of the following:

- Android project key
- Google services account.

After the connection is set, you can migrate the server- and client sides of your project to FCM.

▸ To get the Android project key

1. In the Firebase console (**https://console.firebase.google.com**), import your Google project.
2. From the project settings, retrieve the *server key* for cloud messaging.
3. Use this server key for the content of the Android project key.
4. Use the *sender ID* in your mobile application (`ANDROID_PROJECT_NUMBER` constant in the Android sample).

For more information about the creation and import of projects in the Firebase console, refer to the Firebase documentation at **https://firebase.google.com/docs**.

▸ To get a `google-services.json` file

- In the Firebase console (**https://console.firebase.google.com**), import your Google project.
- Download the `google-services.json` file; this file should replace the existing configuration file in the SDK sample.

[1]Push notifications are clickable pop-up messages that are displayed outside an app. They are pushed from the server the app uses to the user's device.

▸ To migrate from Google Cloud Messaging (GCM) to Firebase Cloud Messaging with the Notification SDK

1. Migrate the server-side of your project.

   This step is required to comply with Google's announcement about the end of life for Google Cloud Messaging (GCM) as of April 2019. When you update the Notification SDK, the applications that are enabled for GCM as well as the future mobile application can be notified. The Firebase server-side can handle both identifier types.

   > **NOTE:** If you still have a legacy server key in your old project, it can be used but we recommend using the server key available on FCM.

   - The server-side can be migrated with the previous settings. You can use either the Android project key or the Firebase service account credentials mechanism.

     > **NOTE:** The Firebase service account credentials option allows an OAuth connection to the Firebase back end.

2. Migrate the client-side of your project.

   This step is necessary for a complete migration to Firebase Cloud Messaging (FCM). When you update the mobile application, the previous version will still work and receive notifications with the updated server library.

A.2 Enable Push Notification in an iOS project with token-based authentication scheme

With the token-based authentication scheme, a private key is used to authenticate to the Apple Push Notification Service. It is simpler to use this authentication scheme because:

- The same key can be used for development and production apps.
- The same key can be used for all your apps as referenced in your Apple developer account.
- The key does not expire.

▸ To use the Push Notification feature in your iOS application with a token-based authentication scheme

1. Connect to your Apple Developer Account (**https://developer.apple.com/account**).
2. Select **Certificate, IDs & Profiles** and **Keys** (**https://developer.apple.com/account/ios/authkey**).
3. Select the **APNs** check box to create a new key.
4. Download the key and store it in a secure location.

A.3 Enable Push Notification in an iOS project with certificate-based authentication scheme

▸ To use the Push Notification feature in your iOS application with a certificate-based authentication scheme

1. Connect to your Apple Developer Account (**https://developer.apple.com/account**).
2. Locate or create your App ID that matches your bundle ID to enable the notification service. The bundle ID must be specific to your application (no wildcards are allowed for push notifications).
3. Generate a client SSL certificate for the connection between your notification server and the Apple Push Notification Service.

> **NOTE:** You can work with separate client SSL certificates for development and production. When you use a development certificate, you must use the sandbox gateway.

4. Verify that your provisioning profiles now include the Push Notification service.

Appendix: FAQ B

Question: How should I migrate from Google Cloud Messaging (GCM) to Firebase Cloud Messaging (FCM) with the Notification SDK?

Answer: The migration should be done in two steps :

1. Update the Notification SDK Server to version 4.19.0 or later, including its dependencies. If you want your server to use the OAuth mechanism to communicate with FCM, an additional configuration step is necessary.

2. Update your mobile application with the Notification SDK.

Question: Are previous versions of the Notification SDK Server compatible with the Android Firebase Cloud Messaging (FCM) version?

Answer: No, the Notification SDK Server from versions 4.18.x or earlier is not compatible with FCM. From version 4.19.x and later, the Notification SDK Server is able to handle FCM and GCM identifiers.

Question: On iOS, why is the notification permission not requested again when the application is re-installed?

Answer: For the notification permission to be again requested, the application must remain uninstalled for at least one day. For more information, refer to **iOS Technical Note TN2265**.

Question: On iOS, why does the client device not receive all the notifications that I send?

Answer: Depending on the device availability, if multiple notifications are sent, the Apple Push Notification service (APNs) may send only the last one. For more information, refer to **iOS Technical Note TN2265**.

Question: Why does the notification not arrive on the client device when it was successfully sent by the server?

Answer: You need to wait a few seconds after you sent the notification so that one of the success/failure callbacks is called.

Question: On iOS, why do I get an `Invalid token` error when a notification is sent?

Answer: This happens, for example, when the client device is connected to the development version of the Apple Push Notification service (APNs), while the server is connected to the APNs production version, or vice versa. Check `useSandboxGateway` in the `NotificationSDKServerCredentials` class and verify it is consistent with the used PKCS #12 certificate.

Question: On iOS, why is the icon badge not correctly incremented/decremented?

Answer: The icon badge can be set in the notification and is hence managed by the server. Value **0**clears the icon badge, a negative value does not affect the icon badge, and any positive value is displayed in the icon badge. Once the notification is received, it is up to the client application to handle the icon badge value as needed. Usually the badge value is decremented.

Question: On iOS, why are my notifications displayed under the old application name in the notification center?

Answer: If the client device is rebooted, the device data are updated, and the correct application name is displayed in the notification center.

Question: Why do I not receive notifications on my device when sending from the server seems to be working properly?

Answer: Check the network settings on the device, your network security settings and the firewall settings. The device must be connected to the public internet.

Index

.

.NET
integrate the SDK **26**

A

Android
enable Push Notification **30**
integrate the SDK **17**
integrate the SDK, example manifest configuration **17**
migrate from GCM to FCM, migration steps **31**
OneSpan Notification Identifier **25**
SDK server migration from version 4.18 **14**
Apple Push Notification service, migration from earlier version **27**

C

certificate-based authentication scheme, Push Notification **33**
client registration, push notification process **8**

E

Example Android manifest configuration **17**

F

Firebase account JSON file, server migration from earlier version **14**
Firebase Cloud Messaging, migrate apps **14**
Firebase Cloud Messaging, migration from Google Cloud Messaging **34**

G

Google Cloud Messaging, deprecated **14**

H

HTTP/2 update framework, SDK migration from earlier version **27**

I

integration steps
.NET **26**
Android **17**
iOS **22**
Java **25**
iOS
enable Push Notification, certificate-based authentication scheme **33**
enable Push Notification, token-based authentication scheme **32**

integrate the SDK 22
notification delivery process, illustration 9
OneSpan Notification Identifier 25
registration process, illustration 8
SDK migration from version 4.24.x 27
SDK migration, update steps 27

J

Java
integrate the SDK 25

L

legacy key, server migration from earlier version 14

M

Migration from earlier version, iOS 27

N

notification delivery 12
notification delivery, caution notice 8
notification properties configuration, example 21
Notification SDK Client
integration steps, Android 17
integration steps, iOS 22
Notification SDK Server
integration steps, .NET 26
integration steps, Java 25
notification. 17

O

OAuth, server migration from earlier version 14
OneSpan Notification Identifier 12

P

payloads 12
PKCS #8 framework update, SDK migration from earlier version 27
project number equivalent, server migration from earlier version 14
push notification
client registration 8
notification delivery 9
process overview 8
Push Notification, enable with Android project key or Google services account 30

R

registration process with the SDK 11

S

SDK server migration from earlier version, Android 14

T

token-based authentication scheme, Push Notification 32

U

unique identifier, Notification SDK Cli-

ent **16**
unique identifier, Notification SDK
Server **24**
unique notification identifier **8**