|  |
| --- |
| **Сервис доставки push-сообщений** |
| **Руководство по интеграции библиотеки PushLib 3.3.0 Lite в приложение на базе ОС Android** |

Оглавление

[Термины и сокращения 3](#_Toc69376019)

[Введение 4](#_Toc69376020)

[1. Интеграция библиотеки в приложение 5](#_Toc69376021)

[1.1 Требования к интеграции 5](#_Toc69376022)

[1.2 Порядок интеграции библиотеки 5](#_Toc69376023)

[1.2.1 Обновление файла AndroidManifest.xml 5](#_Toc69376024)

[1.2.2 Добавление зависимостей в Gradle 6](#_Toc69376025)

[1.2.3 Код интеграции в приложение 7](#_Toc69376026)

[1.2.4 Подключение Firebase 10](#_Toc69376027)

[3. Настройка сборки приложения в консоли edna 12](#_Toc69376028)

[4. Подключение облака HUAWEI 13](#_Toc69376029)

[4.1 Добавление зависимостей в Gradle 13](#_Toc69376030)

[4.2 Инициализация Huawei 14](#_Toc69376031)

[4.3 Возврат статуса доставки уведомления через облако Huawei с помощью upstream 14](#_Toc69376032)

[4.4 Уведомления из прочих источников 17](#_Toc69376033)

[5. Описание методов SDK 18](#_Toc69376034)

[5.1 Класс PushBroadcastReceiver 18](#_Toc69376035)

[5.2 Структура сообщения PushMessage 20](#_Toc69376036)

[5.3 Класс PushController 21](#_Toc69376037)

[6. Дополнительные настройки 22](#_Toc69376038)

[7. Коды ошибок 23](#_Toc69376039)

# Термины и сокращения

| Термин | Полная форма | Описание |
| --- | --- | --- |
| appPackage |  | Уникальный код приложения заказчика в магазинах приложений Google Play и App Store.  Используется при регистрации приложения на push-сервере |
| deviceUid |  | Уникальный идентификатор мобильного приложения, установленного на конкретном устройстве. Формируется таким образом, что является постоянным при обновлениях и переустановках приложения на конкретном аппарате клиента |
| deviceAddress |  | Уникальный адрес мобильного приложения, установленного на конкретном устройстве. Формируется по заданному алгоритму при каждом запуске приложения на стороне push-сервера. Может меняться при изменении push-адреса приложения в PNS. Это зависит от настроек, заданных для приложения на push-сервере |
| PNS | Push Notification Services | Провайдеры push-уведомлений GCM (Google Cloud Messaging) и HCM (Huawei Cloud Messaging), обеспечивают доставку push-уведомлений в приложение на устройстве |
| Push-адрес |  | Уникальный адрес приложения в PNS, может изменять значение по инициативе PNS |
| Push-сервер |  | Программно-аппаратный комплекс, который реализует сервис доставки push-сообщений. В частности, механизм регистрации приложений на push-сервере, отправку push-уведомлений и доставку содержимого push-сообщений в приложение на устройстве, резервирование доставки с помощью sms-сообщений, хранение статусов доставки push-сообщений |
| Push-сообщение |  | Сообщение от заказчика, включающее push-уведомление и содержимое сообщения в текстовом формате. В качестве содержимого заказчик может передавать: тексты, а также бинарные данные в формате Base64 |
| Push-уведомление |  | Короткое уведомление, которое push-сервер отправляет в приложение посредством сервисов PNS. Push-уведомления, доставленное в приложение, инициирует процедуру получения содержимого push-сообщения с push-сервера |
| Библиотека | Push-библиотека | Компонент мобильного приложения, участвует в интеграции с приложением. Обеспечивает регистрацию приложения на push-сервере, привязку дополнительных идентификаторов, доставку push-уведомлений и содержимого push-сообщений в приложение |
| Заказчик |  | Издатель мобильного приложения, является инициатором отправки push-сообщений |
| Приложение | Мобильное приложение | Программное обеспечение заказчика, установленное на мобильном устройстве клиента, в которое интегрирована push-библиотека |

# Введение

Документ представляет собой руководство разработчика, описывающее порядок работы по интеграции push-библиотеки в приложение заказчика. В документе приводится необходимая информация по интеграции, настройке и администрированию библиотеки.

Для работы с push-сервисом edna предоставляются библиотека для интеграции в приложение на базе ОС Android. Высокое качество доставки уведомлений через push-сервис edna в Android приложения обеспечивается за счет взаимодействия приложения, библиотеки и push-сервера.

Библиотека предназначена для решения следующих задач в рамках сервиса доставки push-сообщений:

* Идентификация устройства.
* Регистрация устройства в облаке APNs.
* Регистрация устройства на push-сервере.
* Обработка push-уведомлений.
* Подтверждение доставки push-уведомлений.
* Загрузка контента push-сообщений с push-сервера.
* Подтверждение загрузки контента push-сообщений.
* Передача информации о прочтении сообщений пользователем.

Процедура интеграции библиотеки в приложение заказчика является частью процесса внедрения сервиса доставки push-сообщений.

# Интеграция библиотеки в приложение

## 1.1 Требования к интеграции

**Внимание!** Минимальная версия compileSdkVersion и targetSdkVersion – 28.

Библиотека поддерживает работу с PNS-сервисом Firebase Cloud Messaging (FCM) 20.3.0.

## 1.2 Порядок интеграции библиотеки

Интеграция библиотеки в приложение производится в следующей последовательности:

1. Обновить файл AndroidManifest.xml.
2. Добавить зависимости в файлы build.gradle.
3. Реализовать класс-наследник PushBroadcastReceiver.
4. Инициализировать Firebase.
5. Написать код инициализации библиотеки.
6. Настроить сборку приложения в консоли администрирования.

### 1.2.1 Обновление файла AndroidManifest.xml

Для настройки интеграции библиотеки в приложение необходимо обновить файл AndroidManifest.xml, добавив в него приведенный ниже код:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  <uses-permission android:name="android.permission.INTERNET"/>  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>  <permission  android:name="[APP\_PACKAGE\_NAME].permission.pushserver.RECEIVE"  android:protectionLevel="signature" />  <uses-permission android:name="[ APP\_PACKAGE\_NAME].permission.pushserver.RECEIVE" />  <application>  <!-- Setting up Push Lib -->  <receiver  android:name="[PUSH\_BROADCAST\_RECEIVER]"  android:exported="false"  android:permission="[ APP\_PACKAGE\_NAME].permission.pushserver.RECEIVE">  <intent-filter>  <action android:name="com.pushserver.android.NEW\_PUSH\_MESSAGE\_EVENT" />  <action android:name="com.pushserver.android.LONG\_MESSAGES\_RECEIVED" />  <action android:name="com.pushserver.android.MESSAGES\_WERE\_READ" />  <action android:name="com.pushserver.android.RECEIVER\_CHANGED\_EVENT" />  <action android:name="com.pushserver.android.DEVICE\_ADDRESS\_CHANGED\_EVENT" />  <action android:name="com.pushserver.android.HAS\_SECURED\_MESSAGE" />  <action android:name="com.pushserver.android.ERROR\_EVENT" />  </intent-filter>  </receiver>  </application>  </manifest> |

Используемые параметры:

* [PUSH\_BROADCAST\_RECEIVER] – путь к классу-наследнику PushBroadcastReceiver, который будет обрабатывать события библиотеки;
* [APP\_PACKAGE\_NAME] – полное наименование пакета приложения.

### 1.2.2 Добавление зависимостей в Gradle

Необходимо обновить корневой build.gradle и файл *app*/build.gradle приложения, добавив следующие зависимости:

1. Корневой build.gradle:

|  |
| --- |
| allprojects {  repositories {  google()  jcenter()  maven {  url 'https://maven-pub.edna.ru/repository/maven-public'  }  }  } |

1. app/build.gradle приложения:

|  |
| --- |
| apply plugin: 'com.google.gms.google-services'  …  android {  compileSdkVersion 28  …  defaultConfig {  …  minSdkVersion 17  targetSdkVersion 28  …  }  …  }  dependencies {  debugImplementation 'com.edna.android:push-lite-debug:3.3.0  releaseImplementation 'com.edna.android:push-lite-release:3.3.0  } |

Важно! Debug и Release версия библиотеки используют разные сервера, поэтому для сборок приложений, которые выкладываются в Google Play Market, необходимо использовать строго push-lite-release.

### 1.2.3 Код интеграции в приложение

Библиотекой предоставляются следующие основные классы для интеграции:

Класс PushController – используется для доступа к функциональности библиотеки.

Класс PushBroadcastReceiver – используется для получения событий библиотеки.

**Код инициализации библиотеки**

Для инициализации библиотеки необходимо вызвать метод PushController.getInstance(this).init(). Инициализацию требуется проводить в методе Application onCreate при запуске приложения.

Библиотека выполнит вызов callback-метода onDeviceAddressChanged. После этого станет доступна процедура отправки push-сообщений на мобильное устройство по предоставленному deviceAddress.

**Обработка событий библиотеки**

Для интеграции библиотеки в приложение требуется реализовать класс-наследник от PushBroadcastReceiver. В методах наследника требуется реализовать обработку событий библиотеки.

Основные методы класса для реализации:

onDeviceAddressChanged – вызывается при изменении deviceAddress приложения. deviceAddress является уникальным идентификатором приложения на push-сервере. По данному идентификатору производится отправка уведомлений в приложение, через push-сервис edna. При авторизации пользователя в приложении, deviceAddress необходимо передать на бэкенд приложения и привязать к пользователю, для дальнейшей отправки сообщений.

onShortPushRecieved – вызывается при поступлении уведомления в приложение и предназначен для обработки и отображения уведомления приложением.

Другие методы обработчика PushBroadcastReceiver описаны в разделе Описание методов SDK.

**Автоматическое отображение уведомлений библиотекой**

С версии 3.3.0 библиотеки метод onShortPushRecieved имеет реализацию по умолчанию в библиотеке. Если метод не переопределён в наследнике класса, или в методе наследника происходит вызов super-метода – уведомление будет автоматически отображено библиотекой.

Отрисовка уведомлений библиотекой позволяет использовать все возможности Android по отображению уведомлений. Например, при использовании сервиса Красивый Чек автоматически будет отображен переданный сервисом логотип. Также, управлять отображением можно с помощью набора дополнительных параметров, передаваемых вместе с уведомлением.

Библиотекой при отображении может быть обработан следующий набор дополнительных параметров:

Таблица 1. Дополнительные параметры обрабатываемые библиотекой

|  |  |
| --- | --- |
| Наименование параметра | Описание |
| title | Заголовок уведомления. |
| logoURL | URI иконки для отображения с уведомлением. Допускается использование изображений в форматах jpg и png. Рекомендуемое соотношение сторон - 1:1. |
| bigContentText | Альтернативный текст, для отображения в расширенном виде уведомления. На некоторых устройствах отображается вместо основного текста для обоих видов уведомления. |
| bigContentTitle | Альтернативный заголовок, для отображения в расширенном виде уведомления. На некоторых устройствах отображается вместо основного заголовка для обоих видов уведомления. |
| bigContentLogoURL | URI изображения, для отображения в расширенном виде уведомления. Допускается использование изображений в форматах jpg и png. Рекомендованное соотношение сторон – 2:1. При другом соотношении сторон, изображение будет обрезано до него на большей части устройств. При использовании кнопок в уведомлении, изображение будет дополнительно обрезано по высоте. |
| NotificationChannel | Канал уведомлений Android, в котором должно быть отображено уведомление. При отсутствии указанного канала, он будет создан при отображении уведомления. Если не передан, используется канал по умолчанию – “Default channel”. |
| deeplink | Строка, передаваемая в Intent при переходе по уведомлению. |
| actions | Массив наименований кнопок, добавляемых к уведомлению, и информации, передаваемой при переходе по ним. Должен быть передан строкой содержащей JSON-массив |
| sound | Звуковой файл используемый для уведомления/канала уведомлений. Файл должен быть добавлен в папку raw ресурсов приложения. Наименование файла передаётся без указания расширения. |
| vibrate | Шаблон вибрации. Целочисленный массив. В массиве поочерёдно указываются продолжительности активности и бездействия вибромотора в миллисекундах, начиная с бездействия. |
| lights | Цвет светового индикатора. Передаётся как цвет RGB в шестнадцатеричном формате. При ограниченной палитре индикатора, системой будет подобран самый близкий доступный цвет. При отсутствии индикатора – проигнорирован. |

Важно! Параметры sound, vibrate и lights устанавливаются для каждого сообщения на Android 7 и ниже. На Android 8 и выше, параметры используются при создании канала уведомлений NotificationChannel. После создания канала уведомлений изменить настройки нельзя. Все последующие сообщения направленные в данный канал будут отображены с установленными при создании канала настройками. Для отображения уведомлений с новым набором параметров необходимо создать новый канал.

Все указанные параметры являются необязательными. При отсутствии части параметров, соответствующие возможности уведомлений не будут использованы.

Пример передачи всех доступных параметров сообщения в формате JSON:

|  |
| --- |
| {  "title": "Заголовок",  "logoURL": " https://.../logo.png",  "bigContentText": "Альтернативный текст уведомления",  "bigContentTitle": "Альтернативный заголовок уведомления",  "bigContentLogoURL": "https://.../picture.png",  "NotificationChannel": "Имя канала уведомлений",  "deeplink": "test://deeplink\_action",  "actions": "[{\"title\":\"Кнопка 1\", \"action\":\"test://button1\_action\"}, {\"title\":\"Кнопка 2\", \"action\":\"test://button2\_action\"}]",  "sound": "ring",  "vibrate": "[200,500,200,500,200]",  "lights": "#00ff00"  } |

При клике на уведомлении или на одной из кнопок в уведомлении, пользователь будет направлен в приложение. В стартовое Activity приложения будет направлен Intent с данными переданными в уведомлении. Необходимо обработать полученную информацию в методах onCreate или onNewIntent. В данных Intent будет передана либо строка, переданная с уведомлением как deeplink, если пользователь кликнул по уведомлению. Либо будут переданы наименование кнопки и строка, переданная для неё. Пример кода обработки перехода пользователя по уведомлению:

|  |
| --- |
| override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  // deeplink – строка действия уведомления, по которому пользователь перешёл в приложение  // actionTitle – наименование кнопки, по которой пользователь перешёл в приложение  // action – строка действия, переданная с кнопкой, по которой пользователь перешёл в приложение  val action = intent.action  val actionTitle = intent.extras?.getString(action)  val deeplink = intent.extras?.getString("deeplink")  if (!deeplink.isNullOrEmpty()) {  // TODO: Дальнейшая навигация клиента с учётом переданной строки (параметр deeplink)  } else if (!action.isNullOrEmpty()) {  // TODO: Дальнейшая навигация клиента с учётом переданной строки (параметр action)  }  } |

### 1.2.4 Подключение Firebase

Для инициализации Firebase необходимо воспользоваться соответствующим ассистентом. Для этого нужно зайти в пункт меню Android Studio –> Tools -> Firebase -> Cloud Messaging -> Set up Firebase Cloud Messaging. В открывшейся панели необходимо нажать кнопку Add FCM to your app и принять изменения. Это добавит необходимые зависимости в ваш проект.

Кроме этого необходимо поместить в корневую папку приложения (рядом с build.gradle приложения) файл google-services.json. Файл можно сформировать самостоятельно в консоли Firebase или запросить у своего консультанта по внедрению.

Для регистрации приложения в Firebase доступны два варианта:

1. **В учетной записи edna:**

* Получить SHA256 fingerprint сертификата, используемого в приложении. Например: keytool -list -v -keystore «файл сертификата», Certificate fingerprints.
* Передать в edna appPackage приложения и SHA256 fingerprint сертификата.
* Получить от edna файл google-services.json.
* Включить в Android Studio вид Project.
* Скопировать файл google-services.json, в корневую папку приложения.

1. **В собственной учетной записи:**

* Зарегистрировать учетную запись Firebase.
* Добавить проект мобильного приложения.
* Передать в edna appPackage, Идентификатор отправителя и Ключ сервера с вкладки Cloud Messaging настроек проекта.

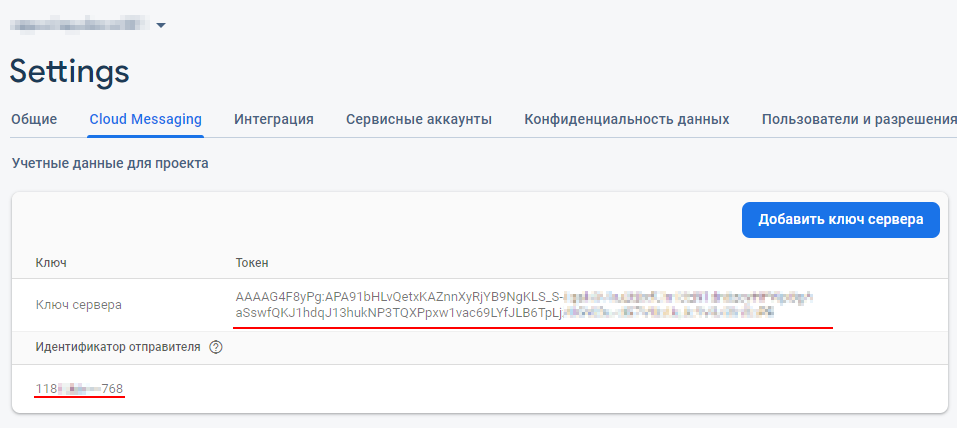


Рис. 1. Параметры приложения Sender ID и API Key в консоли Firebase

#### 1.2.4.1 Уведомления из прочих источников

В данном разделе описаны действия, необходимые в приложениях, которые принимают уведомления не только от edna Push Service, но и из других источников. Данное описание позволяет:

* перехватить событие обновления Firebase Token-а;
* перехватить событие входящего уведомления.

Для настройки необходимо реализовать класс наследник com.edna.android.push\_lite.FcmPushService и реализовать в нем методы onNewToken и onMessageReceived следующим образом:

**import com.google.firebase.messaging.RemoteMessage;  
import com.edna.android.push\_lite.fcm.FcmPushService;**

**public class** PushFcmIntentCustomService **extends** FcmPushService

{  
 **public void** onNewToken(String s) {  
 *// Обработка токена в прочих целях* **super**.onNewToken(s);  
 }  
 **public void** onMessageReceived(RemoteMessage message)  
 {  
 **boolean** isEdnaPush = **true**;  
 *// Если Push относится к edna* **if** (isEdnaPush && NotificationManagerCompat.*from*(**this**).areNotificationsEnabled())

{  
 **super**.onMessageReceived(message);  
 }  
 **else** {  
 *//Обработка уведомления в прочих целях* }  
 }  
}

**Важно!** Сообщение, отправленное через платформу edna, может быть передано в одном из двух форматов – JSON или Base64-encoded JSON, в зависимости от канала передачи. Для уведомлений Huawei сообщения будут передаваться в виде Base64-encoded строки. Следует предусмотреть оба варианта при обработке сообщений.

Данный класс необходимо оформить в виде сервиса в Android.manifest - в раздел application добавить строки:

<service android:name=".push.EdnaPushFcmIntentService" >  
 <intent-filter>  
 <action android:name="com.google.firebase.MESSAGING\_EVENT" />  
 </intent-filter>  
</service>

# Настройка сборки приложения в консоли edna

Вместе с этим руководством предоставляется доступ в тестовую консоль администрирования push-сервера, который необходим, чтобы проверить корректность интеграции библиотеки в приложение.

Консоль администрирования позволяет:

* настраивать сборки приложений;
* просматривать список устройств;
* отправлять тестовые push-сообщения в свое мобильное приложение;
* смотреть их статусы доставки.

Перед запуском приложения необходимо настроить сборку приложения (вы можете попросить сделать это у вашего консультанта по внедрению). Для этого необходимо в консоли администрирования зайти в меню «Приложения – Добавить приложение» и заполнить следующие поля (далее приведены значения полей для тестовых целей):

* Платформа приложения – Android.
* Добавить используемые приложением сервисы – FCM и/или HCM (Firebase и Huawei)
* Google SenderId – идентификатор отправителя, полученный из консоли Firebase.
* Google API Key – ключ доступа, используемый для отправки push, полученный из консоли Firebase.
* Huawei Application Id– идентификатор отправителя, полученный из консоли Huawei.
* Huawei App Secret – ключ доступа, используемый для отправки push, полученный из консоли Huawei.
* Для Gcm отмена резервной SMS (Upstream Message) – Включено.
* Для Gcm отмена резервной SMS (Delivery Reciepts) – Отключено.
* Количество попыток переотправки при ошибке – 5.
* Значение badge – Количество непрочитанных.
* Блокируемые версии ОС (regexp) – оставить пустым.
* Отправлять push даже если истек StopTime – Отправлять.
* Предельный TTL для быстрой отправки – 20.
* Динамические таймауты отправки резервных SMS – оставить пустым.
* JSON-параметры приложения - оставить пустым.
* Ключи доступа – необходимо добавить ключ (интерфейс подставит по умолчанию правильное значение).

Для отправки тестового сообщения, необходимо найти целевое устройство в списке зарегистрированных устройств. Для этого рекомендуется использовать поиск по deviceAddress, полученному в приложении. В карточке устройства будет доступна отправка сообщений. При отправке можно задать заголовок и текст сообщения, а также дополнительные параметры уведомления.

# Подключение облака HUAWEI

Облако HUAWEI является дополнительным каналом отправки push-уведомлений на устройства с установленным Huawei Mobile Service. Уведомление отправляется сразу через два облака, что позволяет улучшить доставку.

Библиотека подключается к уже имеющемуся Android-приложению.

**ВНИМАНИЕ!!!** Библиотека поддерживает приложения на базе ОС Android версии 4.2 и выше (minSdkVersion 17).

## 3.1 Добавление зависимостей в Gradle

Необходимо обновить корневой build.gradle и файлы build.gradle приложения, добавив следующие зависимости:

1. Корневой build.gradle:

|  |
| --- |
| buildscript {  repositories {  google()  jcenter()  maven {  url **'http://developer.huawei.com/repo/'**  }  }  dependencies {  classpath 'com.huawei.agconnect:agcp:1.4.1.300'  }  }  allprojects {  repositories {  google()  jcenter()  maven {  url **'http://developer.huawei.com/repo'**  }  } } |

1. build.gradle приложения:

|  |
| --- |
| apply plugin: 'com.huawei.agconnect'  …  dependencies {  debugImplementation "com.edna.android:push-lite-huawei-debug:3.3.0"  releaseImplementation "com.edna.android:push-lite-huawei-release:3.3.0"  } |

1. Добавить правила в proguard-rules.pro:

|  |
| --- |
| -keep class com.edna.android.push\_lite.\*\*{\*;}  -keep class com.hianalytics.android.\*\*{\*;}  -keep class com.huawei.updatesdk.\*\*{\*;}  -keep class com.huawei.hms.\*\*{\*;} |

## 3.2 Инициализация Huawei

Для регистрации приложения в HUAWEI доступны два варианта:

1. **В учетной записи edna:**

* Получить SHA256 fingerprint сертификата, используемого в приложении. Например: keytool -list -v -keystore «файл сертификата», Certificate fingerprints.
* Передать в edna appPackage приложения и SHA256 fingerprint сертификата.
* Получить от edna файл agconnect-services.json.
* Включить в Android Studio вид Project.
* Скопировать файл agconnect-services.json, в корневую папку приложения.

1. **В собственной учетной записи:**

* Зарегистрировать учетную запись на странице <https://developer.huawei.com/consumer/en/>.
* Добавить мобильное приложение согласно инструкциям HUAWEI <https://developer.huawei.com/consumer/en/codelab/HMSPreparation/>.
* Для проекта в AppGallery Connect обязательно подключить Push Kit.
* Передать в edna appPackage, ID приложения и Секрет приложения.

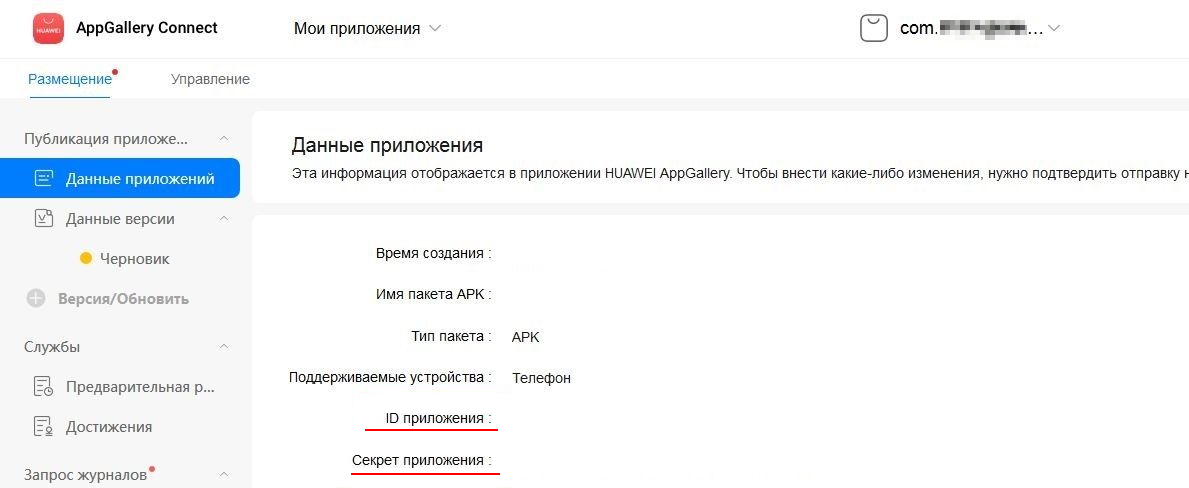


Рис. 2. Параметры приложения App ID и App Secret в AppGallery Connect

**ВНИМАНИЕ!** Если в приложении используется несколько productFlavors, то для каждого имени нужно добавлять новое приложение в консоль HUAWEI. Для каждого имени будет свой файл agconnect-services.json, который нужно добавить к соответствующему productFlavor.

## 3.3 Возврат статуса доставки уведомления через облако Huawei с помощью upstream

Получение сообщений от устройств через upstream позволяет получать информацию о доставке сообщения на устройство с использованием средств облака Huawei. Использование upstream позволяет резервировать основной способ передачи статуса доставки уведомления, уменьшая количество случаев отсутствия статуса доставки при успешном получении уведомления приложением.

Настройка

Для подключения upstream в мобильном приложении необходимо использовать мобильные библиотеки для Android версии 3.1.2 или выше. Библиотеки автоматически при получении сообщения, передают необходимую информацию в сервисы Huawei на устройстве.

Для подключения передачи upstream в настройках приложения в консоли разработчика HUAWEI нужно выполнить следующие шаги:

1. Перейти в окно управления приложением AppGallery Connect.
2. Перейти в Продвижение – Push Kit.
3. Перейти на вкладку Настройки.
4. Перейти к включению «Получения сообщений от устройств».

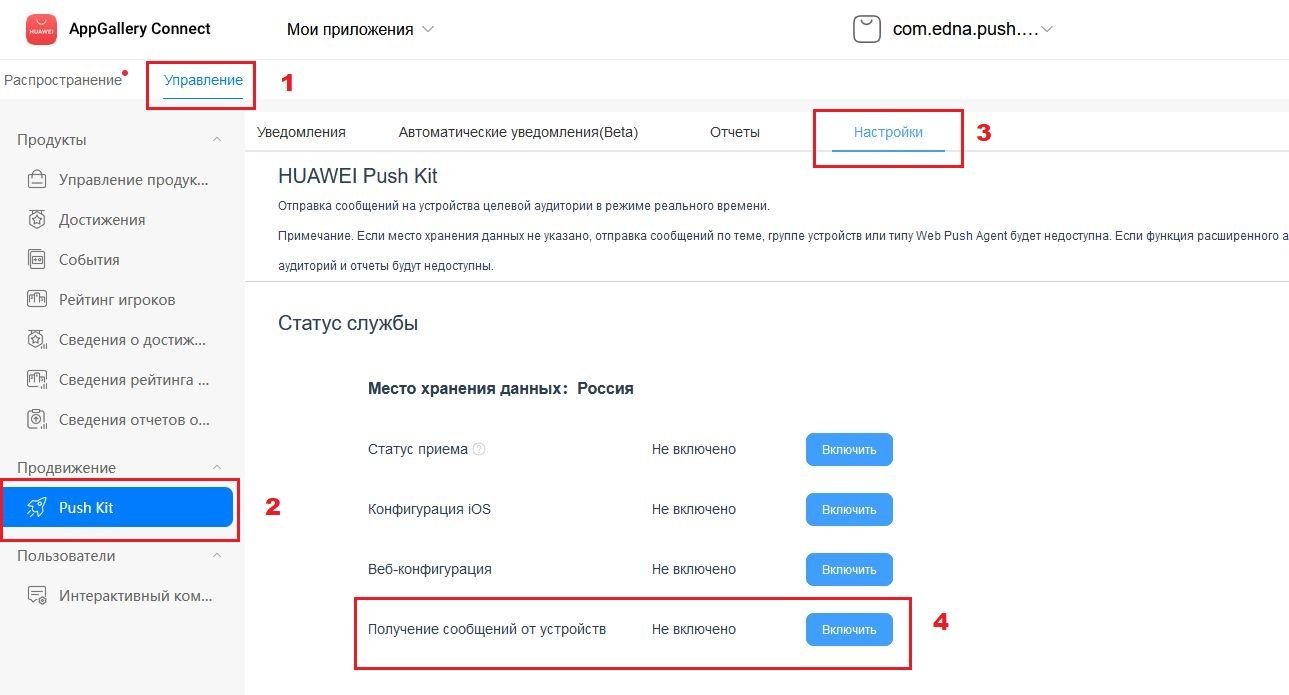


Рис. 3. Окно управления приложением AppGallery Connect

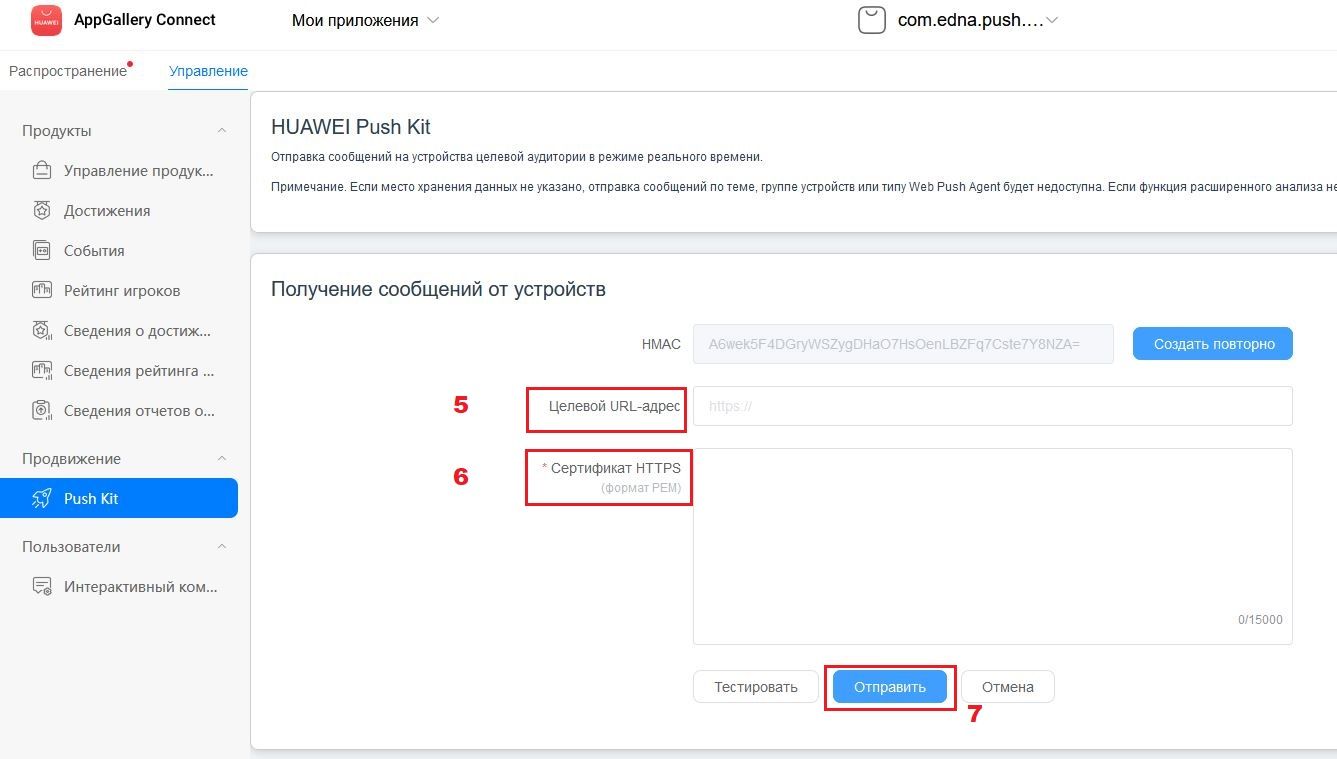


Рис. 4. Экран настройки получения сообщений от устройств

1. В качестве целевого URL-адреса установить следующий:

https://pushserver.edna.ru/banks/service/messages/pushUpstreamReceived

1. Указать следующий сертификат HTTPS (актуален на 12.11.2020, действует до 16.11.2021):

|  |
| --- |
| -----BEGIN CERTIFICATE-----  MIIF/DCCBOSgAwIBAgIMOai+SBaI/Uadwr39MA0GCSqGSIb3DQEBCwUAMFAxCzAJ  BgNVBAYTAkJFMRkwFwYDVQQKExBHbG9iYWxTaWduIG52LXNhMSYwJAYDVQQDEx1H  bG9iYWxTaWduIFJTQSBPViBTU0wgQ0EgMjAxODAeFw0yMDEwMTUxMzE2MzlaFw0y  MTExMTYxMzE2MzlaMFUxCzAJBgNVBAYTAlJVMQ8wDQYDVQQIEwZNb3Njb3cxDzAN  BgNVBAcTBk1vc2NvdzEQMA4GA1UEChMHTExDIE9TSzESMBAGA1UEAwwJKi5lZG5h  LnJ1MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAp7g5EW5ZtRZ7UGL8  YRdNy2u2nJW6Lv2REiccVEBfLLrAwvqFyC6GRlCCTAjFDIhJrgemRud12Ph8U0gT  sISo0D+iTB/uIOvS7TRmw36CUxQD1Qvg+8b5g4ESaAoF75nVx7ust5V8c7YmXOQ8  gRYAQ08CLJKM25f35/fQtr0LhDNOMKmFuIiDMhgwj0kolfxgeJjlC62TrIPPzvVB  4aciCIDhUvUjjPkRRRXbRFjkLRv06l/WRSXX7ihSM8sxUY3s8W877JyE9/3ZUP+H  CjwQkv1r0voEVsxH94FsEFpX8fZroS/BQN64xEMjx/o1S1TeSaQD7SsdpIOmOKYk  QFInxQIDAQABo4ICzzCCAsswDgYDVR0PAQH/BAQDAgWgMIGOBggrBgEFBQcBAQSB  gTB/MEQGCCsGAQUFBzAChjhodHRwOi8vc2VjdXJlLmdsb2JhbHNpZ24uY29tL2Nh  Y2VydC9nc3JzYW92c3NsY2EyMDE4LmNydDA3BggrBgEFBQcwAYYraHR0cDovL29j  c3AuZ2xvYmFsc2lnbi5jb20vZ3Nyc2FvdnNzbGNhMjAxODBWBgNVHSAETzBNMEEG  CSsGAQQBoDIBFDA0MDIGCCsGAQUFBwIBFiZodHRwczovL3d3dy5nbG9iYWxzaWdu  LmNvbS9yZXBvc2l0b3J5LzAIBgZngQwBAgIwCQYDVR0TBAIwADA/BgNVHR8EODA2  MDSgMqAwhi5odHRwOi8vY3JsLmdsb2JhbHNpZ24uY29tL2dzcnNhb3Zzc2xjYTIw  MTguY3JsMB0GA1UdEQQWMBSCCSouZWRuYS5ydYIHZWRuYS5ydTAdBgNVHSUEFjAU  BggrBgEFBQcDAQYIKwYBBQUHAwIwHwYDVR0jBBgwFoAU+O9/8s14Z6jeb48kjYjx  hwMCs+swHQYDVR0OBBYEFMPTkna03wR8D+3WHmU0tL4zHd8cMIIBBAYKKwYBBAHW  eQIEAgSB9QSB8gDwAHUAb1N2rDHwMRnYmQCkURX/dxUcEdkCwQApBo2yCJo32RMA  AAF1LGfTvwAABAMARjBEAiAZldJkQWwoCs48yO+KfvJchE9NMN00OD1sI5CWBtHL  oAIgCKQrECDeIVsNBGgQ9TX4EOEwNEdZbhREBNzPQU5DEiEAdwD2XJQv0XcwIhRU  GAgwlFaO400TGTO/3wwvIAvMTvFk4wAAAXUsZ9aBAAAEAwBIMEYCIQDCOOIByQFq  6nC4qlbku79b3N+AyAMnxAKO51rn7ypjzgIhAN6d4RiaH76UN9BcNFX30JoSNAei  49EWoQECHVwBY1QAMA0GCSqGSIb3DQEBCwUAA4IBAQCZU2iMdoBRfvNB+sWDcXQw  eR/eAetLm8W2JWu7luW35KrHGYJ7qGf1/qaUcgnQk7biAIZUrRGkl4bbcf9ITkdm  xeT2/uycjH4VQRykLZudbaBEaHWcM84G7erG10Sry7LAaGpmh727n+IMxpco58f4  aTA1OBaEK+LTOFVLOhi1UtF8OwYsP5KIP6HaMszF/we/cl9fe6pbKlUqsjCoiMlh  D6EPax7zhdUDPxtBcGvekRJn8cnzPvQUZRaxxrdSqMa5puN5h2NFRXc2cwfcfW6G  SwkuP8eAGMy9zQROl8dAPBdba1QmdQiPx24njzGPr88q0q64bUxH+F5IxOwIWp0I  -----END CERTIFICATE----- |

1. Сохранить настройки, нажав кнопку «Отправить»

## 3.4 Уведомления из прочих источников

В данном разделе описаны действия необходимые в приложениях, которые принимают уведомления не только от edna Push Service, но и из других источников. Данное описание позволяет:

* перехватить событие обновления HUAWEI Token-а;
* перехватить событие входящего уведомления.

Для настройки необходимо выбрать класс-наследник com.edna.android.push\_lite.HcmPushService и применить в нем методы onNewToken и onMessageReceived следующим образом:

**import com.edna.android.push\_lite.huawei.HcmPushService;  
import com.huawei.hms.push.RemoteMessage;**

**public class** PushHcmIntentCustomService **extends** HcmPushService

{  
 **public void** onNewToken(String s) {  
 *// Обработка токена в прочих целях* **super**.onNewToken(s);  
 }  
 **public void** onMessageReceived(RemoteMessage message)  
 {  
 **boolean** isEdnaPush = **true**;  
 *// Если Push относится к edna* **if** (isEdnaPush && NotificationManagerCompat.*from*(**this**).areNotificationsEnabled())

{  
 **super**.onMessageReceived(message);  
 }  
 **else** {  
 *//Обработка уведомления в прочих целях* }  
 }  
}

**Важно!** Сообщение, отправленное через платформу edna, может быть передано в одном из двух форматов – JSON или Base64-encoded JSON, в зависимости от канала передачи. Для уведомлений Huawei сообщения будут передаваться в виде Base64-encoded строки. Следует предусмотреть оба варианта при обработке сообщений.

Данный класс необходимо оформить в виде сервиса в Android.manifest - в раздел application добавить строки:

<service android:name=".push.PushHcmIntentCustomService" >  
 <intent-filter>  
 <action android:name=" com.huawei.push.action.MESSAGING\_EVENT" />  
 </intent-filter>  
</service>

# Описание методов SDK

## 4.1 Класс PushBroadcastReceiver

Для обработки событий библиотеки предоставлен класс PushBroadcastReceiver. Методы класса представлены в Таблице 1.

Таблица 2. Методы класса PushBroadcastReceiver

|  |
| --- |
| **onShortPushReceived** |
| Метод вызывается при получении нового PushNotification от FCM. В качестве параметра передается текст, полученный из shortMessage при отправке PNS (также Bundle с содержимым push-сообщения). Необходимо отобразить локальное уведомление в области уведомлений ОС.  **Метод:**  protected void onShortPushReceived  (  android.content.Context context  java.lang.String messageId  java.lang.String alert  android.os.Bundle bundle  )  **Параметры**:   * *context* – контекст; * *messageId* – идентификатор push-сообщения. Используется при дальнейшем взаимодействии с библиотекой по данному сообщению; * *alert* – текст push-сообщения, для отображения пользователю; * *bundle* – содержимое push-сообщения (полный список атрибутов уведомления) |
| **onLongPushReceived** |
| Метод вызывается при получении списка новых PushMessages c push-сервера.  **Метод:**  protected boolean onLongPushReceived  (  android.content.Context context  final List<PushMessage> pushMessages  )  **Параметры**:   * *context* – контекст; * *pushMessages* – список сообщений;   **Возвращаемые значения**:  *true –* при успешном сохранении сообщений, иначе *false.* |

|  |
| --- |
| **messagesWereRead** |
| Метод вызывается при получении списка идентификаторов прочитанных push-сообщений.  **Метод:**  protected void messagesWereRead(List<String> messageIds);  **Параметры**:  *messageIds* – список идентификаторов прочитанных push-сообщений. |
| **onStatusChanged** |
| Метод вызывается при изменении push-адреса.  **Метод:**  protected abstract void onStatusChanged  (  android.content.Context context,  java.lang.String registrationId  )  **Параметры**:   * *context* – контекст; * *registrationId* – идентификатор приложения в push-сети |
| **onDeviceAddressChanged** |
| Метод вызывается после изменения значения deviceAddress мобильного устройства.  **Метод:**  protected abstract void onDeviceAddressChanged  (  android.content.Context context,  java.lang.String newDeviceAddress  )  **Параметры**:   * *context* – контекст; * *newDeviceAddress* – новое значение deviceAddress |
| **onDeviceAddressProblems** |
| Метод вызывается при наличии проблем, препятствующих генерации deviceAddress.  **Метод:**  protected abstract void onDeviceAddressProblems  (  android.content.Context context,  java.lang.String errorCode  )  **Параметры**:   * *context* – контекст; * *errorCode* – описание ошибки |
| **onError** |
| Метод вызывается при возникновении ошибки взаимодействия с сервером.  **Метод:**  protected abstract void onError  (  android.content.Context context,  java.lang.String errorCode  )  **Параметры**:   * *context* – контекст; * *errorCode* – описание ошибки |

## 4.2 Структура сообщения PushMessage

Структура сообщения PushMessage (является параметром в методе onLongPushReceived) представлена в Таблице 3.

Таблица 3. Структура сообщения PushMessage

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Значение |
| messageId | String | Идентификатор сообщения |
| sentAt | Long | Дата и время формирования сообщения в формате DD.MM.YYYY hh:mm:ss Z, где Z – часовой пояс в формате RFC 822 |
| shortMessage | String | Сокращенная форма сообщения. Отправляется через push-сеть в виде системного push-уведомления (Alert). Предназначено для отображения в списке сообщений в виде заголовка |
| fullMessage | String | Полный текст push-сообщения (base64-encoded HTML) |
| secured | Boolean | Признак безопасного сообщения |
| read | Boolean | Статус прочтения сообщения |
| sessionKey | String | Сессионный ключ. Используется библиотекой для аутентификации устройства на push-сервер |

Пример сообщения:

|  |
| --- |
| {  messageId: "1:10",  sentAt: "24.10.2012 12:41:57",  shortMessage: "Уважаемый, ФИО! Изменились …",  fullMessage: "0KPQstCw0LbQsNC10LzRi9C5INC60LvQuNC10L3RgiEg0JjQt",  secured: true  read: true  sessionKey: sessionKey  } |

## 4.3 Класс PushController

Класс PushController реализует методы для предоставления доступа к функциональности библиотеки. Описание методов представлено в Таблице 4.

Таблица 4. Методы класса PushController

| Метод | Описание |
| --- | --- |
| public String getDeviceUid() | Метод возвращает текущий *deviceUid* |
| public void setDeviceUid(String deviceUuid) | Метод устанавливает внешнее значение для *deviceUid*. Применяется, если по каким-либо причинам встроенный механизм генерации *deviceUid* не подходит |
| public void resetCounterSync() throws PushServerErrorException | Сбрасывает счетчик непрочитанных сообщений. В случае ошибки push-сервера или сетевой ошибки генерирует исключение PushServerErrorException |
| public void notifyMessageUpdateNeeded() | Метод для принудительной загрузки сообщений с сервера |
| public void notifyMessageRead(String messageId) | Отмечает сообщение с указанным идентификатором как прочитанное. В качестве идентификатора может быть передан: *messageId* полученный в методе *onShortPushReceived*, *PushMessage.messageId* полученный в *onLongPushRecieved*, *Id* полученный при запросе истории сообщений на шлюзе. |
| public String getDeviceAddress() | Метод возвращает текущий *deviceAddress* |
| public String getVersion() | Получает текущую версию библиотеки |
| public void setLogPrefix(String logPrefix) | Позволяет установить произвольный префикс для логов библиотеки |
| sendMessage(String context, boolean isSystem) | Синхронная отправка (входящего) сообщения на push-сервер. isSystem = true, если сообщение системное (например typing). . В случае ошибки push-сервера или сетевой ошибки генерирует исключение PushServerErrorException |
| sendMessageAsync(String context, boolean isSystem) | Асинхронная отправка (входящего) сообщения на push-сервер. isSystem = true, если сообщение системное (например, typing) |

# Дополнительные настройки

В Android.manifest можно также задать некоторые дополнительные настройки (значения которых установлены по умолчанию). Рекомендуется делать это только по согласованию с вашим консультантом по внедрению. Список таких настроек приведен в Таблице 5.

Таблица 5. Дополнительные настройки

|  |  |
| --- | --- |
| Код настройки | Описание |
| com.pushserver.android.providerUid | Ключ доступа. По умолчанию является функцией App\_package. Аналогичное значение нужно задать на сервере |
| com.pushserver.android.logs | Признак необходимости логирования. По умолчанию true – для debug версии и false – для release версии |
| com.pushserver.android.serverUrlList | Список адресов push-серверов.  По умолчанию для debug версии:  { http://pushservertest.edna.ru/push-test, https://pushserver.edna.ru/banks }  и для release версии:  { https://pushserver.edna.ru/banks } |
| com.pushserver.android.serverIdList | Список идентификаторов push-серверов.  По умолчанию {1, 3} – для debug версии и { 3 } – для release версии |
| com.pushserver.android.serverPrimaryId | Идентификатор основного сервера. По умолчанию – 1 для debug версии и 3 – для release версии |
| com.pushserver.android.useDefaultDeviceUID | Необходимость автоматической генерации deviceUid. По умолчанию true |
| com.pushserver.android.appPackagePostfix | Строковая настройка, которая используется для изменения поведения BackEnd в новой версии (например, для смены senderId в Firebase) |
| com.pushserver.android.notificationChannelId | Строковая настройка, которая используется для определения канала уведомлений, в котором будет отображено уведомление о фоновой работе, если это будет необходимо при обработке входящих сообщений. |

# Коды ошибок

В метод, обрабатывающий ошибки взаимодействия с push-сервером, передаются коды ошибок. Описание кодов ошибок представлено в Таблице 6.

Таблица 6. Коды ошибок

|  |  |
| --- | --- |
| Код ошибки | Описание |
| DEVICE\_ADDRESS\_INVALID | Указано некорректное значение идентификатора deviceAddress |
| ACCESS\_DENIED | Недостаточно прав для выполнения операции |
| INTERNAL\_SERVER\_ERROR | Внутренняя ошибка сервера: требуется повторить запрос позже |
| BAD\_PARAMETERS | Указаны неверные параметры запроса. Данная ошибка возникает в случае неверного значения deviceAddress или его отсутствия на push-сервере |
| IO\_ERROR | Сетевая ошибка |
| JSON\_FORMING\_ERROR | Неверно сформировано JSON-сообщение |

**Примечание!** При возникновении ошибок сетевого взаимодействия или некорректной передаче параметров методам значение errorCode > 0. Если ошибок нет, тогда значение errorCode = nil, либо errorCode = 0; errorCode:Int32 = 0. Описание ошибки: errorDescription.