РУКОВОДСТВО РАЗРАБОТЧИКА

Часть 2

Подсистема Сервис уведомлений

для операционной системы Android

Версия 1.1

Листов 34

2026

АННОТАЦИЯ

Настоящий документ является второй частью руководства разработчика подсистемы Сервис уведомлений (ПСУ), входящей в состав прикладного программного обеспечения «Аврора Центр» (далее – ППО) релиз 5.7.0.

ПРИМЕЧАНИЯ:

✓ Подробная информация о составе и назначении ППО, а также требования к условиям выполнения приведены в документе «Руководство администратора»;

✓ Подробная информация об особенностях резервного копирования приведена в документе «Рекомендации по резервному копированию».

Настоящий документ содержит инструкции для разработчика приложений, выполняющих отправку push-уведомлений на устройства, функционирующие под управлением операционной системы (ОС) Android.

СОДЕРЖАНИЕ

1. Общая информация 4
1.1. Лицензионное соглашение 4
1.2. Получение доступа к тестовому Серверу приложений ПСУ 5
1.3. Работа с тестовым Сервером приложений ПСУ 6
2. Описание работы 8
2.1. Разработка приложения 8
2.1.1. Добавление Aurora Center Services SDK в проект Android 8
2.1.2. Получение регистрационного токена 9
2.1.3. Тестовое сообщение 9
2.1.4. Пример тестового приложения 12
2.2. Отправка push-уведомлений 12
2.2.1. Запрос токена авторизации 13
2.2.2. Подпись токена авторизации 15
2.2.3. Оправка push-уведомлений 16
2.2.4. Получение информации о проекте push-уведомлений 19
2.2.5. Рекомендации использования запроса 23
2.2.6. Автоматическая смена ключа безопасности проекта 23
2.3. Пример кода сервера внешнего приложений 28
2.3.1. Код на языке Python 28
3. Диагностика и устранение неполадок 29
3.1. Проверка известных проблем 29
3.2. Проверка счетчика принятых push-уведомлений 30
3.3. Диагностика при отсутствии известных проблем 30
3.4. Локализация проблемы, если используемый push-сервер развернут в составе Аврора Центр 31
Перечень терминов и сокращений 32

1. ОБЩАЯ ИНФОРМАЦИЯ

Если на момент отправки push-уведомления устройство недоступно, приложение получит push-уведомление после подключения к сети, обеспечивающей доступ к Серверу приложений ПСУ.

Для использования push-уведомлений в своих системах разработчикам необходимо расширить функциональность приложений и сервера внешнего приложения, отправляющего push-уведомления.

ПРИМЕЧАНИЕ. Не рекомендуется с помощью push-уведомлений осуществлять передачу конфиденциальной и чувствительной информации в незащищенных сетях, т.к. протокол взаимодействия устройства с Сервером приложений ПСУ не подразумевает передачу конфиденциальной информации.

При разработке приложений доступно подключение к тестовому серверу для тестирования функциональности push-уведомлений.

1.1. Лицензионное соглашение

Программное обеспечение (ПО) Aurora Center Services SDK является интеллектуальной собственностью ООО «Открытая мобильная платформа» и лицензируется на основе Лицензионного соглашения с конечным пользователем (EULA).

Лицензионное соглашение с конечным пользователем является приложением к настоящему руководству разработчика и располагается по адресу: https://maven.omp.ru/repository/filestorage/sdk/EULA.

1.2. Получение доступа к тестовому Серверу приложений ПСУ

ПРИМЕЧАНИЕ. Для тестирования сервера внешнего приложения, который отправляет push-уведомления, и приложения, которое получает push-уведомления, необходимо использовать тестовый Сервер приложений ПСУ.

Для получения настроек подключения к тестовому Серверу приложений ПСУ необходимо обратиться к Администратору Аврора Центр и получить адрес и порт, а также файл с настройками Сервера приложений ПСУ и настройками приложений.

Настройки необходимо указать в конфигурации сервера внешнего приложения, который после прохождения аутентификации получает токен доступа и отправляет push-уведомления через Сервер приложений ПСУ.

Список настроек для подключения сервера внешнего приложения к тестовому Серверу приложений ПСУ:

– `project_id` – уникальный идентификатор (ID) проекта;

– `push_public_address` – адрес тестового Сервера приложений ПСУ, на который будут отправляться запросы;

– `api_url` – адрес API тестового Сервера приложений ПСУ;

– `client_id` – аккаунт учетной записи разработчика в Подсистеме безопасности тестового Сервера приложений ПСУ, обычно совпадает с ID проекта;

– `scopes` – список действий, которые могут быть разрешены приложению;

– `audience` – предполагаемый потребитель токена, как правило, это сервер ресурсов (API), к приложению которого необходимо получить доступ;

– `token_url` – адрес получения токена авторизации для запросов к тестовому Серверу приложений ПСУ;

– `key_id` – идентификатор приватного ключа RSA для передачи сообщений между сервером внешнего приложения и Сервером приложений ПСУ;

– `private_key` – приватный ключ RSA для передачи сообщений между сервером внешнего приложения и тестовым Сервером приложений ПСУ. Рекомендации по ротации приватного ключа проекта приведены в п. 2.2.6.

Для приложений доступна настройка следующих параметров:

– `application_id` – уникальный идентификатор приложений, который необходимо добавить в приложение;

– `project_id` – уникальный идентификатор проекта.

ПРИМЕЧАНИЕ. `Application_id` необходимо использовать для регистрации в push-демоне на устройстве. Каждому `application_id` может соответствовать только 1 приложение.

1.3. Работа с тестовым Сервером приложений ПСУ

Для того, чтобы использовать в разработке тестовый Сервер приложений ПСУ для отправки push-уведомлений, необходимо выполнить следующие шаги:

1) Установить приложение «Аврора Центр» на устройство и активировать его с помощью сканирования QR-кода.

ПРИМЕЧАНИЕ. Процесс установки приложения «Аврора Центр» и активации устройства может быть выполнен Администратором Платформы управления;

2) Получить доступ к тестовому Серверу приложений ПСУ:

– направить запрос Администратору Платформы управления;

– получить ключи для доступа к тестовому Серверу приложений ПСУ;

– использовать ключи при взаимодействии сервера внешнего приложения с тестовым Сервером приложений ПСУ;

3) Создать и установить приложение на устройстве:

– передать в приложение `applicationId`, полученный от Администратора Платформы управления;

– зарегистрироваться в push-демоне с помощью `applicationId`, получив в ответ на запрос `registrationId` для тестового Сервера приложений ПСУ;

4) Организовать передачу push-уведомлений сервера внешнего приложения через тестовый Сервер приложений ПСУ:

– передать `registrationId` серверу внешнего приложения доверенным способом от приложения. Это позволит серверу внешнего приложения отправлять таргетированные push-уведомления к своим приложениям на конкретном устройстве;

– получить токен доступа на тестовом Сервере приложений ПСУ через сервер внешнего приложения;

– передать через сервер внешнего приложения push-уведомления, используя токен доступа и идентификатор связки приложение – устройство `registrationId`.

Процесс регистрации приложений приведен на рисунке (Рисунок 1).

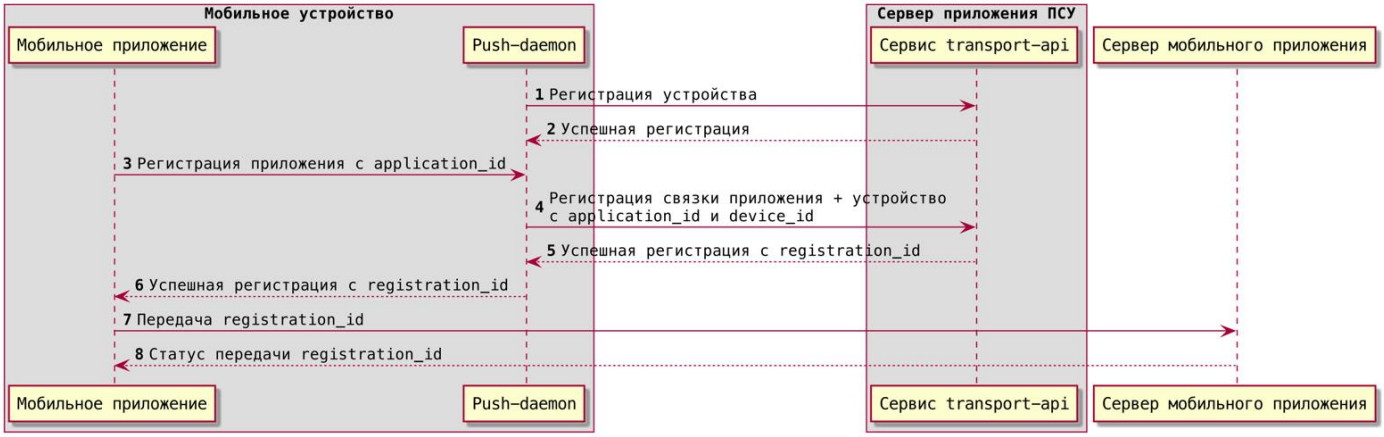


Рисунок 1

Запрос на регистрацию приложений должен происходить при каждом его запуске, т.к. `registrationId` имеет срок жизни, не зависящий от приложений.

ПРИМЕЧАНИЕ. При переходе с тестового Сервера приложений ПСУ на основой необходимо получить новый проект push-уведомлений и добавить его в свое приложение.

2. ОПИСАНИЕ РАБОТЫ

2.1. Разработка приложения

2.1.1. Добавление Aurora Center Services SDK в проект Android

Для интеграции приложения с Aurora Center Services SDK необходимо использовать Android Studio IDE и язык разработки Java.

Необходимо выполнить следующие шаги:

1) Получить проект в Сервисе уведомлений от Администратора Аврора Центр (см. подраздел 1.2). В результате будет получен `applicationId`;

2) Запомнить и сохранить полученный `applicationId`. Он понадобится в дальнейшем при интеграции с Aurora Center Services SDK;

3) Добавить `maven` репозиторий `omp` в файл `settings.gradle`:

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        maven {
            url "https://maven.omp.ru/repository/maven/"
        }
    }
}
```

4) Добавить `aurora-center-services-push` в зависимости в `build.gradle/build.gradle.kts` (в соответствии с настройками проекта):

```
dependencies {
    ...

    implementation "ru.omp.aurora-center-services:aurora-center-ser-
vices-push:1.1.5"
}
```

2.1.2. Получение регистрационного токена

Чтобы получить регистрационный токен (`registrationId`), необходимо вызвать метод `getToken` у `PushSdk`, передав `Context` и `applicationId`, полученный выше (см. п. 2.1.1):

```
PushSdk.getInstance().getToken(getApplicationContext(), m_appId)
                .addOnCompleteListener(new OnCompleteListener() {
                    @Override
                    public void onComplete(@NonNull Task task) {
                        if (!task.isSuccessful()) {
                            Log.w(TAG, "Fetching registration token
failed: " + task.getResult());
                            return;
                        }

                        String token = task.getResult();
                        Log.d(TAG, "token: " + token);
                    }
                });
```

2.1.3. Тестовое сообщение

2.1.3.1. Настройки получения тестового сообщения

Чтобы принять тестовое сообщение, необходимо:

1) Отредактировать манифест приложения, добавив секцию `service`:

```
<service
    android:name=".MyPushService"
    android:enabled="true"
    android:exported="false">
    <intent-filter>
      <action android:name="ru.omp.services.REGISTRATION_STATUS" />
      <action android:name="ru.omp.services.PUSH_AVAILABLE" />
      <action android:name="ru.omp.services.PUSH_MESSAGES" />
    </intent-filter>
</service>
```

2) Унаследовать класс `Service` (переопределив метод `onMessageReceived`):

```
public class MyPushService extends PushService {

@Override
    public void onMessageReceived(PushMessage message) {
        Log.d(TAG, "onMessageReceived: " + message.getMessage());
    }
}
```

ПРИМЕЧАНИЕ. Полученное в приложении push-уведомление должно быть обработано логикой приложений:

1) Тихий push без отображения пользователю;

2) Вывести уведомление пользователю в UI (Notification tray).

2.1.3.2. Отправление тестового сообщения в приложение

Используя полученный выше токен, необходимо отправить тестовое сообщение в приложение.

Для этого можно воспользоваться эмулятором сервера внешнего приложения на языке программирования Python или отправить push-уведомление со своего сервера внешнего приложения.

Эмулятор доступен на ресурсе https://hub.mos.ru/auroraos/tools/PushSenderPython.

Программа считывает настройки проекта из YAML-файла, указанного в параметре `config`, делает запрос на получение токена доступа и в зависимости от параметра `action` выполняет ту или иную логику, доступную через API.

Использование:

1) Задать конфигурацию одним из двух способов:

– приоритет – загрузить файл YAML сервера из настроек проекта через Консоль администратора ПСУ и сохранить его рядом с `main.py` под именем `push-server.yml`;

– если файл `push-server.yml` не найден, то в конфигурационный файл вставляется вся информация, полученная из настроек проекта от Администратора Аврора Центр;

2) Установить зависимости:

```
python3 -m venv venv
source ./venv/bin/activate
pip install -r requirements.txt
```

3) Запустить:

```
python ./main.py \
      --action one of:
            * send - send message (default)
```

```
            * get-project - get information about the project
            * update-key - generate a new key pair, update the public
key for the service account and send a test message (if the --regID
parameter is specified)
      --config <path to app_server_xxx.yaml, downloaded from the Push
project settings (push-server.yml by default)>
      --regID <registration ID received in the push daemon or emula-
tor during registration (default e3bbd700-6a76-4ad6-a699-ac284da5b1ab
for TARGET NOT FOUND check)>
      --messageText <message for push notification body ("some text"
by default)>
```

Например, чтобы отправить сообщение, необходимо выполнить следующую команду:

```
python main.py --action send --config ~/configs/push/push-
feat/app_server_kaa_smoke_111_c5kq2iiq44trvsk5lsm0.yaml --regID
22be010d-a5cc-4f9d-96be-366d4c5432ca
```

После успешной отправки программа выведет на экран результат.

Пример:

```
Project options:
    push_public_address: "http://ocs-push-
dev.devel.pro:8009/push/public"
    project_id: "novyi_proekt_btpivpvl5abfanlsq8r0"
    client_id: "novyi_proekt_btpivpvl5abfanlsq8r0"
    audience: "ocs-auth-public-api-gw ocs-push-public-api-gw"
    scope: "openid offline message:update"
    private_key_id: "public:sVnj2ES7UQ"
    private_key: "b'-----BEGIN RSA PRIVATE KEY-----
                  \nMIIJ...RYJhsSqPoLA==\n-----END RSA PRIVATE KEY-
----\n'"
Got token_endpoint:
    "http://ocs-push-dev.ompcloud.ru/auth/public/oauth2/token"
Result:
    response: {'access_token': 'eyJhbGci...fojLY', 'expires_in':
3599, 'scope': 'openid offline message:update', 'token_type': 'bear-
er', 'expires_at': 1601966682}
    access_token: eyJhbGci...fojLY
Send push message result:
    response: "{'expiredAt': '2020-10-06T07:44:43.967576Z', 'id':
'1526c09c-1ca4-48ea-8357-e1b8aa2f3fc3', 'notification': {'data':
{'action': 'command', 'another_key': 'value'}, 'message': 'somem
message', 'title': 'some title'}, 'status': '', 'string': '', 'tar-
get': '7cefc0d1-b262-4bcc-a959-497c7bfd38f4', , 'type': 'device'}"
    id: "1526c09c-1ca4-48ea-8357-e1b8aa2f3fc3"
```

Команда для получения информации о проекте:

```
python main.py --action get-project --config ~/configs/push/push-feat/app_server_kaa_smoke_111_c5kq2iiq44trvsk5lsm0.yaml
```

Команда для перевыпуска ключей учетной записи службы:

```
python main.py --action update-key --config ~/configs/push/push-feat/app_server_kaa_smoke_111_c5kq2iiq44trvsk5lsm0.yaml
```

Если требуется сразу проверить, что сообщение отправлено успешно с новым ключом, необходимо указать идентификатор регистрации в качестве параметра:

```
python main.py --action update-key --config ~/configs/push/push-feat/app_server_kaa_smoke_111_c5kq2iiq44trvsk5lsm0.yaml --regID 22be010d-a5cc-4f9d-96be-366d4c5432ca
```

2.1.4. Пример тестового приложения

В качестве примера было подготовлено тестовое приложение «QuickStart Push».

Исходный код с примером интеграции приложения «QuickStart Push» с Aurora Center Services SDK возможно получить по адресу: https://github.com/omprussia/quick-start-push.

2.2. Отправка push-уведомлений

Push-уведомления отправляются с сервера внешнего приложения через API Сервера приложений ПСУ, который защищен протоколом OAUTH2 OpenID Connect согласно JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication. При этом используется метод аутентификации `private_key_jwt`.

При регистрации проекта на Сервере приложений ПСУ разработчику будут направлены конфигурационные настройки приложения и сервера внешнего приложения. В конфигурации представлены параметры, необходимые для настройки клиента протокола авторизации OpenID Connect и формирования URL для отправки push-уведомлений, приведенные в подразделе 1.2.

Этапы отправки push-уведомлений:

1) Получение токена, позволяющего в течение времени его жизни отправлять push-уведомления (п. 2.2.1);

2) Непосредственно отправка push-уведомления (п. 2.2.3).

2.2.1. Запрос токена авторизации

Для получения токена необходимо выполнить неавторизованный запрос по адресу из параметра `token_url` (подраздел 1.2).

Запрос:

```
POST {token_url}
```

Пример:

```
POST http://example.ru/auth/public/oauth2/token
```

Заголовок `Content-Type` должен иметь значение `application/json`.

Обязательные параметры тела запроса приведены в таблице (Таблица 1).

Таблица 1

Обязательность	Параметр	Тип	Описание
Да	`scope`	`String`	Список действий, которые могут быть разрешены приложению. Значения из файла с настройками проекта (подраздел 1.2). Пример: `openid offline message:update project:read`
Да	`audience`	`String`	Предполагаемый потребитель токена, как правило, это сервер ресурсов (API), к которому приложению необходимо получить доступ. Значения из файла с настройками проекта (подраздел 1.2). Пример: `http://example.ru/auth/public http:// example.ru /push/public`

Обязательность	Параметр	Тип	Описание
Да	`grant_type`	`String`	Тип авторизации сервера внешнего приложения в Подсистеме безопасности Push-сервера. Передать значение: «`client_credentials`»
Да	`client_assertion_type`	`String`	Передать значение, как строку: «`urn:ietf:params:oauth:client-assertion-type:jwt-bearer`»
Да	`client_assertion`	`String`	Сформированный самостоятельно JWT, содержащий информацию для аутентификации клиента. Он должен быть подписан цифровой подписью с использованием закрытого ключа, значение которого находится в параметре `private_key` из файла с настройками проекта (подраздел 1.2). Требования к формированию токена JWT можно посмотреть в таблице (Таблица 2)

JWT должен содержать обязательные параметры, приведенные в таблице (Таблица 2).

Таблица 2

Обязательность	Параметр	Тип	Описание
Да	`iss`	`String`	Issuer. Должен содержать `client_id` клиента OAuth
Да	`sub`	`String`	Subject. Должен содержать `client_id` клиента OAuth
Да	`aud`	`String`	Audience. Предполагаемый потребитель токена, как правило, это сервер ресурсов (API), к которому приложению необходимо получить доступ. Сервер авторизации должен подтвердить, что он является целевой аудиторией для токена. Значение должно представлять собой URL конечной точки сервера авторизации

Обязательность	Параметр	Тип	Описание
Да	`jti`	`String`	JWT ID. Уникальный идентификатор токена, который может быть использован для предотвращения повторного использования токена. Такие маркеры должны использоваться только один раз, если только условия повторного использования не были оговорены между сторонами; любые такие переговоры выходят за рамки данной спецификации
Да	`exp`	`String`	Время истечения срока действия токена, по истечении которого JWT не должен приниматься к обработке

2.2.2. Подпись токена авторизации

JWT должен быть подписан цифровой подписью с использованием закрытого ключа в асимметричной криптографии RS256.

Клиент, использующий метод аутентификации, должен заранее зарегистрировать свой открытый ключ (параметр `key_id`) на сервере авторизации, чтобы сервер мог проверить `assertion`.

Проверить корректность выпущенного токена можно, к примеру, на сайте https://jwt.io/.

Ответ на запрос получения токена приведен в таблице (Таблица 3).

Таблица 3

Обязательность	Параметр	Тип	Описание
Да	`access_token`	`String`	Токен авторизации, с которым необходимо выполнять запросы к серверу
Да	`expires_in`	`String`	Время действия токена авторизации
Да	`scope`	`String`	Список действий, которые могут быть разрешены приложениям
Да	`token_type`	`String`	Тип токена
Да	`expires_at`	`String`	Время окончания действия токена авторизации

Из ответа следует считать атрибуты, которые возвращают токен, и время его действия.

Пример кода на Python для получения токена доступа приведены в подразделе 2.3.

В случае, если вышел срок действия `access_token`, необходимо инициировать процедуру получения токена (п. 2.2.1).

2.2.3. Оправка push-уведомлений

2.2.3.1. Общая информация

Для отправки push-уведомления необходимо сформировать запрос на Сервер приложений ПСУ, указанный в `api_url` (подраздел 1.2).

```
POST  {api_url}/projects/{project_id}/messages
```

Заголовок `Content-Type` должен иметь значение: `application/json`.

Заголовок `Authorization` должен иметь значение: `Bearer access_token`, полученный в запросе авторизации (п. 2.2.1).

К адресу Сервера приложений ПСУ необходимо добавить фрагмент с указанием `project_id`.

Параметры тела запроса приведены в таблице (Таблица 4).

Таблица 4

Обязательность	Параметр	Тип	Описание
Да	`target`	`String`	Указывается `registrationId`, полученный из приложения после его регистрации на Сервере приложений ПСУ
Да	`type`	`Enum (String)`	Поддерживается значение `device`
Да	`ttl`	`String`	Время жизни push-уведомления, например, 5h30m. Может быть задано только в следующих единицах: `s` — секунды, `h` — часы, `m` — минуты
Да	`notification`	`OBJ Notification`	Содержание push-уведомления

Параметры объекта `Notification` приведены в таблице (Таблица 5).

Таблица 5

Обязательность	Параметр	Тип	Описание
Нет	`title`	`string`	Заголовок (до 512 символов)
Нет	`message`	`string`	Тело сообщения (до 2048 символов)
Нет	`data`	`Object`	Объект для параметров ключ-значение, можно передавать любые необходимые приложению поля (до 1024 байт)
Нет	`action`	`string`	Информационное поле, в котором возможно передавать служебные сообщения для приложения. Поле должно содержать не более 255 символов. **ПРИМЕЧАНИЕ.** В будущих релизах возможно изменение поведения этого поля

Возможные коды ошибок приведены в таблице (Таблица 6).

Таблица 6

Код и статус ошибки	Описание	Действия для устранения ошибки
`400 Bad Request`	`ttl limit is exceeded`	Необходимо проверить допустимое время жизни push-уведомления в соответствии с настоящим документом
	`unsupported message type`	Необходимо проверить доступные типы доставки уведомлений в соответствии с настоящим документом
	`invalid notification title length`	Необходимо проверить доступные ограничения на создание уведомлений в соответствии с настоящим документом
	`invalid notification message length`	Необходимо проверить доступные ограничения на создание уведомлений в соответствии с настоящим документом

Код и статус ошибки	Описание	Действия для устранения ошибки
	`invalid target`	Необходимо проверить доступные значения в соответствии с настоящим документом
	`target not found` (указанный `registration_id` не найден)	`registration_id` необходимо получить после регистрации устройства и приложения на Сервере приложений Сервиса уведомлений
`401 Unauthorized`	Истек срок действия токена авторизации	Необходимо получить новый токен авторизации (п. 2.2.1). Дополнительно см. раздел 3
	`Client authentication failed, the provided client JSON Web key is expired` (не удалось выполнить аутентификацию. Срок действия предоставленного ключа истек)	Необходимо обратиться к Администратору Сервиса уведомлений с просьбой выпустить новый ключ безопасности проекта и прислать его, после чего следует установить новый ключ и повторить запрос
`403 Forbidden`	Отсутствуют необходимые права доступа для выполнения запроса	Необходимо обратиться к системному администратору для назначения прав
`413 Request Entity Too Large`	Общий размер JSON-объекта в теле сообщения превышает 4 КБ	Необходимо уменьшить размер JSON-объекта в теле сообщения
`429 Too Many Requests`	Достигнут лимит количества запросов в секунду	Необходимо сократить количество запросов в секунду для проекта
`500 Internal Server Error`	Внутренняя системная ошибка Сервера приложений ПСУ	Необходимо обратиться к системному администратору. Дополнительно см. раздел 3
`504 Gateway Timeout`	Инфраструктура Сервиса уведомлений недоступна	Необходимо обратиться к системному администратору

Пример запроса:

```
POST https://global-push.domain/api/projects/project-from-ui-
bthsc2iq44tso869omt0/messages
Content-Type: application/json
{
    "target": "442125d4-6041-4f72-ab4b-1e5fd3ad389a",
    "ttl": "2h",
    "type": "device",
    "notification": {
        "title": "some title",
        "message": "some message",
        "data": {
            "another_key": "value"
        },
                                                                "ac-
tion": "command"
    }
}
```

Пример ответа в случае успеха имеет следующий вид:

```
{
    "expiredAt": "2020-10-06T07:44:43.967576Z",
    "id": "1526c09c-1ca4-48ea-8357-e1b8aa2f3fc3",
    "notification": {
        "data": {
            "another_key": "value"
        },
        "message": "some message",
        "title": "some title",
                                                                "ac-
tion": "command"
    },
    "status": "",
    "string": "", "target": "442125d4-6041-4f72-ab4b-1e5fd3ad389a",
    "type": "device"
}
```

2.2.4. Получение информации о проекте push-уведомлений

Сервер приложений ПСУ предоставляет возможность получить следующую информацию:

- о доступности Сервера приложений ПСУ;
- об актуальности ключей безопасности проекта push-уведомлений;
- об активности проекта push-уведомлений.

Необходимо сформировать запрос на Сервер приложений ПСУ, указанный в `api_url` (подраздел 1.2):

```
GET  {api_url}/projects/{project_id}
```

Заголовок `Content-Type` должен иметь значение: `application/json`.

Заголовок `Authorization` должен иметь значение: `Bearer access_token`, полученный в запросе авторизации (п. 2.2.1).

При формировании URL для запроса вместо `{project_id}` необходимо указать идентификатор проекта из конфигурации.

Параметры ответа приведены в таблице (Таблица 7).

Таблица 7

Обязательность	Параметр	Тип	Описание
Да	`id`	`String`	Идентификатор проекта
Да	`name`	`String`	Название проекта
Да	`isActive`	`Boolean`	Активность проекта
Да	`serviceAccount`	`OBJ serviceAccount`	Объект с информацией о сервисном аккаунте проекта
Да	`createdAt`	`String`	Дата создания проекта
Да	`updatedAt`	`String`	Дата обновления проекта

Параметры объекта `serviceAccount` приведены в таблице (Таблица 8).

Таблица 8

Обязательность	Параметр	Тип	Описание
Да	`clientId`	`string`	Идентификатор сервисного аккаунта, совпадает с идентификатором проекта
Да	`clientName`	`string`	Название сервисного аккаунта, совпадает с названием проекта
Да	`publicKeys`	`Object`	Объект с информацией о сроке действия ключей безопасности проекта
Да	`scope`	`string`	Список действий, которые могут быть разрешены приложению
Да	`audience`	`Array[string]`	Предполагаемый потребитель токена, как правило, это сервер ресурсов (API), к которому приложению необходимо получить доступ

Параметры объекта `publicKeys` приведены в таблице (Таблица 9).

Таблица 9

Обязательность	Параметр	Тип	Описание
Да	`meta`	`Object`	Объект с информацией о ключе безопасности проекта. Содержит значение параметров: – `public:{значение}` – идентификатор публичного ключа; – `assigned_at` – дата назначения ключей безопасности проекта; – `expired_at` – дата окончания действия ключей безопасности проекта

Ответ будет представлен в следующем формате:


```
{
   "createdAt":"2023-09-13T21:43:29.712093+03:00",
   "id":"acs_1384_test_ck105k9pfb781l4a8n8g",
   "isActive":true,
   "name":"acs-1384-test",
   "serviceAccount":{
      "audience":[
         "http://example.ru/auth/public",
         "http://example.ru/push/public"
      ],
      "clientId":"acs_1384_test_ck105k9pfb781l4a8n8g",
      "clientName":"acs-1384-test",
      "publicKeys":{
         "meta":{
            "public:WcKEcj7wQQ":{
               "assigned_at":"2023-09-13T18:43:58Z",
               "expired_at":"2024-12-13T00:43:58Z"
            }
         }
      },
      "scope":"openid offline message:update project:read"
   },
   "updatedAt":"2023-09-13T21:44:43.73058+03:00"
}
```

Возможные коды ошибок приведены в таблице (Таблица 10).

Таблица 10

Код и статус ошибки	Описание	Действия для устранения ошибки
`401 Unauthorized`	Истек срок действия токена авторизации	Необходимо получить новый токен авторизации (п. 2.2.1). Дополнительно см. раздел 3
	`Client authentication failed, the provided client JSON Web key is expired` (не удалось выполнить аутентификацию. Срок действия предоставленного ключа истек)	Необходимо обратиться к Администратору Сервиса уведомлений с просьбой выпустить новый ключ безопасности проекта и прислать его, после чего следует установить новый ключ и повторить запрос
`403 Forbidden`	Доступ к ресурсу запрещен	Проверить корректность настроек переданного проекта. В случае повторения ошибки обратиться к Администратору Сервиса уведомлений
`404 Not Found`	Ресурс не найден	Проверить корректность настроек переданного проекта. В случае повторения ошибки обратиться к Администратору Сервиса уведомлений
`429 Too Many Requests`	Достигнут лимит количества запросов в секунду	Необходимо сократить количество запросов в секунду для проекта
`500 Internal Server Error`	Внутренняя системная ошибка Сервера приложений ПСУ	Необходимо обратиться к системному администратору. Дополнительно см. раздел 3
`504 Gateway Timeout`	Инфраструктура Сервиса уведомлений недоступна	Необходимо обратиться к системному администратору

2.2.5. Рекомендации использования запроса

Периодически, с частотой не менее одного раза в сутки, с сервера внешнего приложения необходимо выполнять запрос на Сервер приложений ПСУ, чтобы получать актуальные данные по проекту:

– атрибут `isActive` должен быть равен `true`. В случае нарушений связаться с Администратором Сервиса уведомлений;

– значение `expired_at` для используемого ключа должно быть с действующей датой. Рекомендуется получить новый ключ безопасности проекта, не дожидаясь даты окончания действия текущего. Процесс автоматической смены ключа безопасности проекта приведен в п. 2.2.6;

– эндпоинт должен возвращать статус `200 OK` – это значит, что Сервер приложений ПСУ физически доступен для отправки сообщений из сервера приложений. В случае нарушений связаться с Администратором Сервиса уведомлений.

2.2.6. Автоматическая смена ключа безопасности проекта

В случае, если при выполнении запроса на получение информации о проекте (п. 2.2.4), будет замечено, что срок действия ключа истекает, необходимо заблаговременно выполнить следующие действия:

– сгенерировать новую пару, состоящую из открытого и закрытого ключа;

– установить публичный ключ для аккаунта своего проекта;

– обновить конфигурацию своего сервера внешнего приложения для использования нового ключа при получении авторизационного токена.

Для генерации пары ключей необходимо выполнить запрос ниже:

```
POST  {api_url}/keyPairs
```

Заголовок `Content-Type` должен иметь значение: `application/json`.

Параметры тела запроса приведены в таблице (Таблица 11).

Таблица 11

Обязательность	Параметр	Тип	Описание
Да	`alg`	`String`	Алгоритм для подписи токена. Должно быть указано значение `RS256`
Да	`kid`	`String`	Идентификатор ключа. Строка длиной 10 символов, содержащая цифры и буквы латинского алфавита в нижнем и верхнем регистрах
Да	`use`	`String`	Целевое назначение алгоритма шифрования. Должно быть указано значение `sig`

Заголовок `Authorization` должен иметь значение: `Bearer access_token`, полученный в запросе авторизации (п. 2.2.1).

Коды ошибок аналогичны тем, что приведены в таблице (см. Таблица 6).

Для случая успеха параметры ответа приведены в таблице (Таблица 12).

Таблица 12

Обязательность	Параметр	Тип	Описание
Да	`private`	`OBJ private`	Объект с информацией о закрытом ключе
Да	`public`	`OBJ public`	Объект с информацией об открытом ключе

Параметры объекта `private` приведены в таблице (Таблица 13).

Таблица 13

Обязательность	Параметр	Тип	Описание
Да	`jwk`	`OBJ privateJWK`	Объект с описанием закрытого ключа
Да	`pem`	`String`	Строка с закрытым ключом в формате `.pem`

Параметры объекта `privateJWK` приведены в таблице (Таблица 14). Детальная информация доступна в RFC 7518.

Таблица 14

Обязательность	Параметр	Тип	Описание
Да	`alg`	`String`	Криптографический алгоритм, в котором предполагается использование ключа
Да	`d`	`String`	Экспонента RSA, участвующая в формировании закрытого ключа. Для подключения к ПСУ не используется
Да	`e`	`String`	Публичная экспонента RSA. Для подключения к ПСУ не используется
Да	`kid`	`String`	Идентификатор ключа. Строка длиной 10 символов, содержащая цифры и буквы латинского алфавита в нижнем и верхнем регистрах
Да	`kty`	`String`	Семейство алгоритмов шифрования, используемое при генерации ключа. Значение будет равно RSA
Да	`n`	`String`	Модуль RSA. Для подключения к ПСУ не используется
Да	`p`	`String`	Первый простой множитель. Для подключения к ПСУ не используется
Да	`q`	`String`	Второй простой множитель. Для подключения к ПСУ не используется
Да	`use`	`String`	Целевое назначение алгоритма шифрования. Должно быть указано значение `sig`

Параметры объекта `public` приведены в таблице (Таблица 15).

Таблица 15

Обязательность	Параметр	Тип	Описание
Да	`jwk`	`OBJ publicJWK`	Объект с описанием открытого ключа

Параметры объекта `publicJWK` приведены в таблице (Таблица 16). Детали доступны в RFC 7518.

Таблица 16

Обязательность	Параметр	Тип	Описание
Да	`alg`	`String`	Криптографический алгоритм, в котором предполагается использование ключа
Да	`e`	`String`	Публичная экспонента RSA. Для подключения к ПСУ не используется
Да	`kid`	`String`	Идентификатор ключа
Да	`kty`	`String`	Семейство алгоритмов шифрования, используемое при генерации ключа. Значение будет равно RSA
Да	`n`	`String`	Модуль RSA. Для подключения к ПСУ не используется
Да	`use`	`String`	Целевое назначение алгоритма шифрования. Должно быть указано значение `sig`

Пример ответа представлен ниже:

```
{
   "private":{
      "jwk":{
         "alg":"RS256",
         "d":"HD ..... PAHPkp_V6byeaf-G0", // сокращено для
отображения
         "e":"AQAB",
         "kid":"private:tt9jjwRJxx",
         "kty":"RSA",
         "n":"39a ....... LRs9J_ROgPUk", // сокращено для отображения
         "p":"8BU ....... fZW-RclDaPw", // сокращено для отображения
         "q":"7q3 ....... PdGvp62dw", // сокращено для отображения
         "use":"sig"
      },
    "pem":"-----BEGIN RSA PRIVATE KEY-----\nMIIJQ ..... jKqthG\n-----
END RSA PRIVATE KEY-----\n" // сокращено для отображения
   },
   "public":{
      "jwk":{
         "alg":"RS256",
         "e":"AQAB",
         "kid":"public:tt9jjwRJxx",
         "kty":"RSA",
         "n":"39a ...... BLRs9J_RogPUk", // сокращено для отображения
         "use":"sig"
      },
      "pem":""
   }
}
```

После генерации пары ключей необходимо применить публичный ключ для аккаунта своего проекта. Для этого необходимо сделать следующий запрос:

```
PUT
{api_url}/projects/{project_id}/serviceAccounts/{client_id}/publicKeys
```

Параметры `{project_id}` и `{client_id}` подставляются из конфигурационного файла проекта, полученного от Администратора Платформы управления.

Заголовок `Content-Type` должен иметь значение: `application/json`.

Параметры тела запроса приведены в таблице (Таблица 17).

Таблица 17

Обязательность	Параметр	Тип	Описание
Да	keys	Array of OBJ publicJWK	Массив с данными об открытом ключе, описанным в таблице (см. Таблица 16)

В тело запроса в массив `keys` должен быть добавлен объект `public.jwk` из ответа на запрос по генерации ключей.

Пример тела запроса:

```
{
   "keys":[
      {
         "alg":"RS256",
         "e":"AQAB",
         "kid":"public:tt9jjwRJxx",
         "kty":"RSA",
         "n":"39a ......... Rs9J_RogPUk", // сокращено для отображения
         "use":"sig"
      }
   ]
}
```

Заголовок `Authorization` должен иметь значение: `Bearer access_token`, полученный в запросе авторизации (п. 2.2.1).

Коды ошибок в ответе аналогичны тем, что приведены в таблице (см. Таблица 6).

В случае успеха пример ответа может выглядеть так:

```
{
   "audience":[
      "https://ocs-int.ompcloud.ru/auth/public",
      "https://ocs-int.ompcloud.ru/push/public"
   ],
   "clientId":"kushnarev_test_cla7thirvkjfc4j9g1e0",
   "clientName":"kushnarev-test",
   "scope":"openid offline message:update project:read keyPairs:create
serviceAccount:update"
}
```

На заключительном этапе необходимо обновить конфигурацию сервера внешнего приложения (и всех узлов кластера при наличии) для использования нового ключа при получении авторизационного токена.

При этом полученные ранее на основании старого ключа токены могут быть использованы до истечения срока их действия.

2.3. Пример кода сервера внешнего приложений

2.3.1. Код на языке Python

Тестовый пример на языке программирования Python демонстрирует все описанные ранее шаги. Пример полностью доступен на ресурсе https://hub.mos.ru/auroraos/tools/PushSenderPython. Подробная работа с программой приведена в пп. 2.1.3.2 (а также в сопровождающем код `README.md` файле).

Программа считывает настройки проекта из YAML-файла, указанного в параметре `config`, делает запрос на получение токена доступа и в зависимости от параметра `action` выполняет ту или иную логику, доступную через API.

3. ДИАГНОСТИКА И УСТРАНЕНИЕ НЕПОЛАДОК

В случае возникновения ошибок 401, 500, а также при отсутствии явных ошибок и одновременном отсутствии push-уведомлений рекомендуется выполнить подпункты, описанные ниже.

3.1. Проверка известных проблем

Для проверки известных проблем:

– используется корректный проект и приложение, а также их настройки (см.: 1.2, 1.3, 2.1.1, а также документ «Руководство пользователя. Часть 5. Подсистема Сервис уведомлений» подраздел «Регистрация проекта»);

– push-уведомление отправляется на актуальный `registration ID` (см.: 1.3, 2.2.3; Таблица 6 - код и статус ошибки «target not found»);

– push-уведомления в проекте включены (см.: 2.2.4, 2.2.5, атрибут `isActive`, а также документ «Руководство пользователя. Часть 5. Подсистема Сервис уведомлений» подраздел «Карточка проекта»;

– время серверов, на которых развернут push-сервер, синхронизировано со временем серверов, формирующих push-уведомления;

– используется действующий токен доступа (см.: 2.2.1, 2.2.3; Таблица 6 – код и статус ошибки «401 Unauthorized»);

– сервер, формирующий push-уведомления, использует актуальные ключи безопасности своего проекта (см.: 2.2.4 - 2.2.6, а также документ «Руководство пользователя. Часть 5. Подсистема Сервис уведомлений» подраздел «Настройки проекта»);

– доступны push-порт для устройства и порт 80/443 для сервера, формирующего push-уведомления (проверить, например, утилитой `telnet`).

3.2. Проверка счетчика принятых push-уведомлений

Если доступна карточка проекта, в блоке «Статистика» отображается поле «Общее количество push-уведомлений» за выбранный период (по умолчанию — текущий месяц). Если созданное push-уведомление было принято сервером, после обновления страницы значение счетчика увеличивается на единицу. Счетчик отражает только факт создания и приема сообщений сервером и не связан с доставкой push-уведомлений на устройство (см. документ «Руководство пользователя. Часть 5. Подсистема Сервис уведомлений» подраздел «Раздел «Статистика»).

3.3. Диагностика при отсутствии известных проблем

Если перечисленные выше известные проблемы отсутствуют, требуется дальнейший анализ и поиск проблем на стороне push-сервера и устройства, на котором возникают проблемы с получением push-уведомлений.

Для диагностики на стороне push-сервера требуется доступ к серверам и логам сервисов, поэтому необходимо обратиться к системному администратору (см.: Таблица 6 – код и статус ошибки «500 Internal Server Error»; Таблица 10, а также документ «Руководство пользователя. Часть 5. Подсистема Сервис уведомлений» раздел «Сообщения об ошибках»), а с устройство необходимо собрать логи.

На первом этапе рекомендуется выполнить следующие проверки:

1) Проверить статус сервисов push-сервера. Для этого требуется на сервере выполнить скрипт `consul_service_health_check.sh`;

2) Проверить отсутствие проблем с созданием партиций в БД. При наличии проблем в логах присутствуют сообщения вида:

```
ERROR: no partition of relation "notifications" found for row
DETAIL: Partition key of the failing row contains (created_at) =
(2025-05-13 05:26:44.94745+00)
```

Если указанные проверки не выявили причину, необходимо собрать логи с push-сервера и с проблемного устройства за период возникновения проблемы, чтобы передать в службу технической поддержки для дальнейшего анализа.

3.4. Локализация проблемы, если используемый push-сервер развернут в составе Аврора Центр

В случае, когда используемый push-сервер развернут в составе Аврора Центр и его интеграция с Аврора Центр работает корректно, но при этом проблемы с доставкой push-уведомлений возникают для сторонних приложений, тогда требуется обратиться к разработчику стороннего приложения. Дальнейшую диагностику имеет смысл сосредоточить на приложении, формирующем push-уведомления, и на устройстве-получателе.

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

Используемые в настоящем документе термины и сокращения приведены в таблице (Таблица 18).

Таблица 18

Термин/ Сокращение	Расшифровка
ОС	Операционная система
ПО	Программное обеспечение
ППО	Прикладное программное обеспечение «Аврора Центр»
Приложение	Приложением является мобильное приложение, функционирующее под управлением ОС Android
ПСУ	Подсистема Сервис уведомлений, реализующая логику управления жизненным циклом проекта push-уведомлений
Сервер внешнего приложения	Внешняя система, по отношению к ПСУ, реализующая бизнес-логику. Использует функционал Сервера приложения Сервиса уведомлений для доставки push-уведомлений на устройства
Сервер приложений ПСУ	Backend подсистемы Сервиса уведомлений предоставляет API для сервера приложений по созданию push-уведомления
Устройство	Под устройством подразумевается мобильное устройство, на котором функционируют соответствующие компоненты ППО
API	Application Programming Interface — программный интерфейс приложения, описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой
Aurora Center Services SDK	SDK для интеграции с Аврора Центр
IP	Internet Protocol — основной протокол сетевого уровня, использующийся в сети Интернет и обеспечивающий единую схему логической адресации устройств в сети и маршрутизацию данных
JSON	Текстовый формат обмена данными, основанный на JavaScript
JWT	JSON Web Token
Push-уведомления	Текстовые сообщения, предназначенные для оперативной (мгновенной) доставки на устройства пользователей

Термин/ Сокращение	Расшифровка
QR-код	Quick Response Code – код быстрого реагирования, предоставляющий информацию для быстрого ее распознавания с помощью камеры на мобильном устройстве
RSA	Криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел
UI	User Interface – пользовательский интерфейс
URL	Uniform Resource Locator — единообразный локатор (определитель местонахождения) ресурса

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

№ версии документа	Содержание изменения	ФИО	Дата
1.0	Начальная версия	Лебедев А.	07.09.2026 г.
1.1	Внесены изменения	Абрамов А. Пенджиев М.	28.09.2026 г.