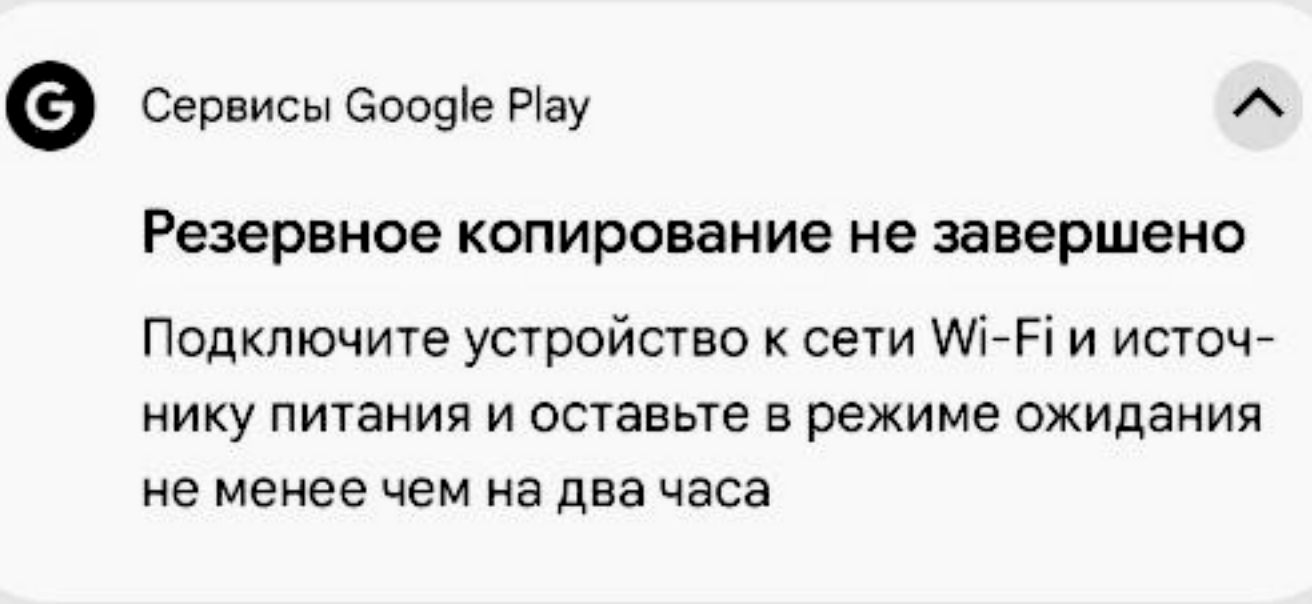
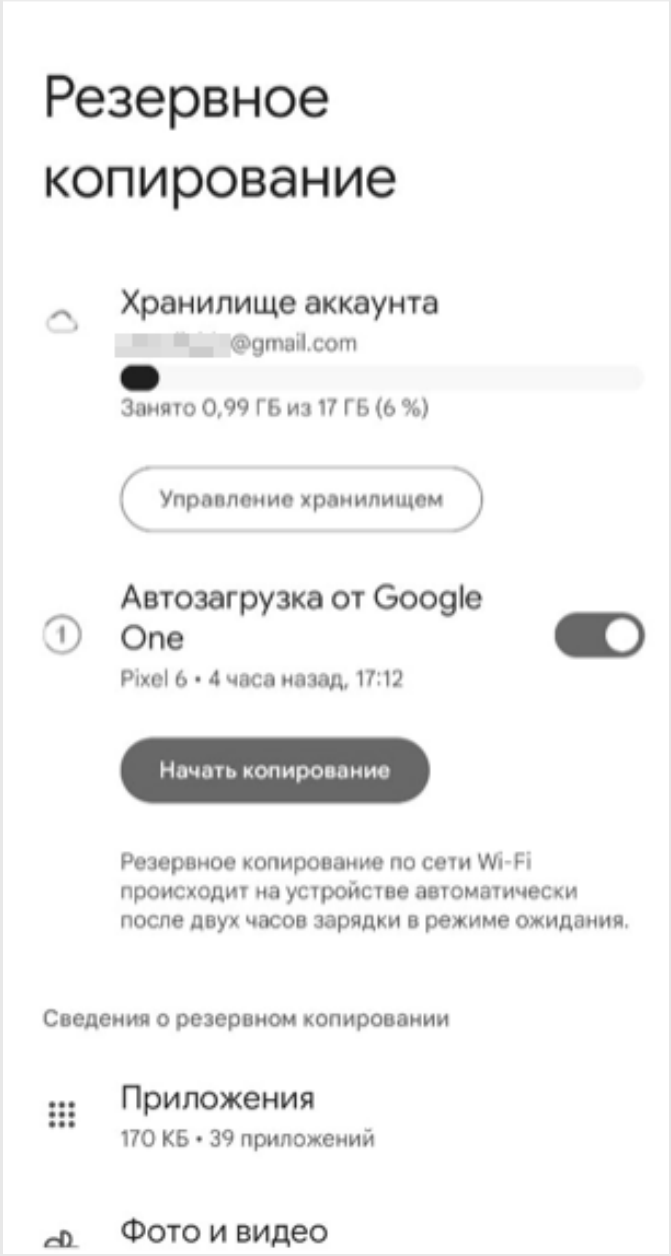
Backup в Android

Или как сэкономить бизнесу миллионы

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
```


Сервисы Google Play
Резервное копирование не завершено
Подключите устройство к сети Wi-Fi и источнику питания и оставьте в режиме ожидания не менее чем на два часа


Резервное копирование
Хранилище аккаунта
@gmail.com
Занято 0,99 ГБ из 17 ГБ (6 %)
Управление хранилищем
Автозагрузка от Google One
Pixel 6 • 4 часа назад, 17:12
Начать копирование
Резервное копирование по сети Wi-Fi происходит на устройстве автоматически после двух часов зарядки в режиме ожидания.
Сведения о резервном копировании
Приложения
170 КБ • 39 приложений
Фото и видео

Backup в Android

Или как сэкономить бизнесу миллионы

Agenda

1. Контекст бизнеса: зачем?

2. Детально про реализацию: как?
   - key-value backup
   - autobackup
   - blockstore
   - тестирование

3. Результаты: что в итоге? 💵


ID

Бизнес-метрики

Снижаем траты
на авторизацию

Увеличиваем % быстрых /
бесшовных авторизаций

Почему этот доклад будет полезен?

>1 500 000₽

Ежемесячная экономия

1%

Юзеров переустанавливают приложение

~1.5 - 2.5 ₽

Цена одной смс

Вы не используете эту технологию

Почему мне можно верить?

> Разрабатываю фичи, на которых бизнес зарабатывает кучу денег

> Нанимающие менеджеры очень любят рассказы про эту фичу

> Веду профессиональный блог. Помогаю айтишникам увеличивать доход и прокачивать харды

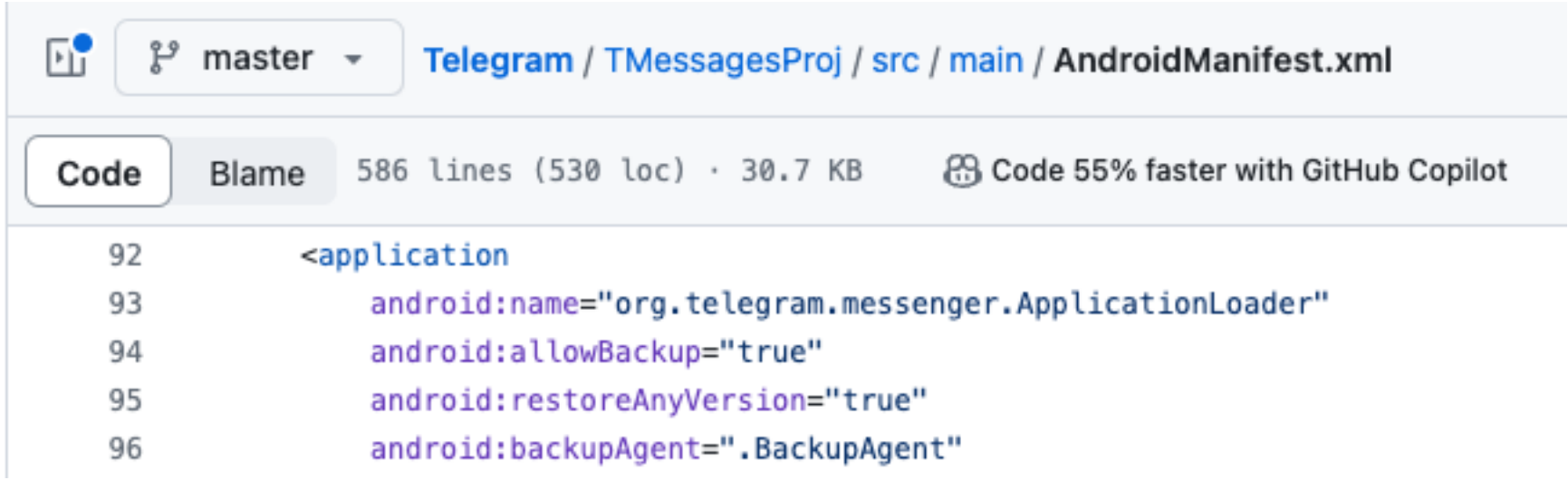
Подпишись на меня

Как обнаружил технологию?

backup_rules.xml | AndroidManifest.xml | data_extraction_rules.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<full-backup-content>
    <!--
    <include domain="sharedpref" path="."/>
    <exclude domain="sharedpref" path="device.xml"/>
-->
</full-backup-content>
```

Как обнаружил технологию?



Как обнаружил технологию?

TMessagesProj/src/main/java/org/telegram/messenger/BackupAgent.java

```
16  import java.io.IOException;
17  import java.util.ArrayList;
18
19  public class BackupAgent extends BackupAgentHelper {
20
21      private static BackupManager backupManager;
22
```

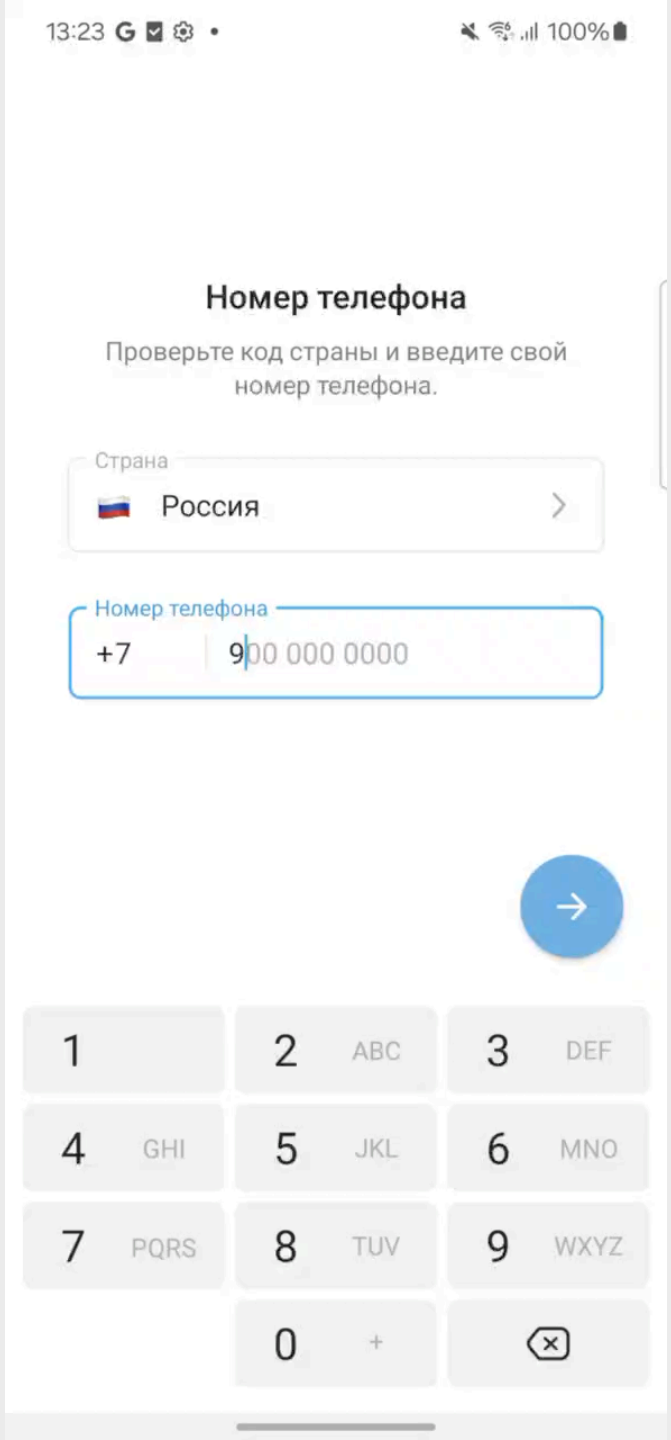
Show 1 more match

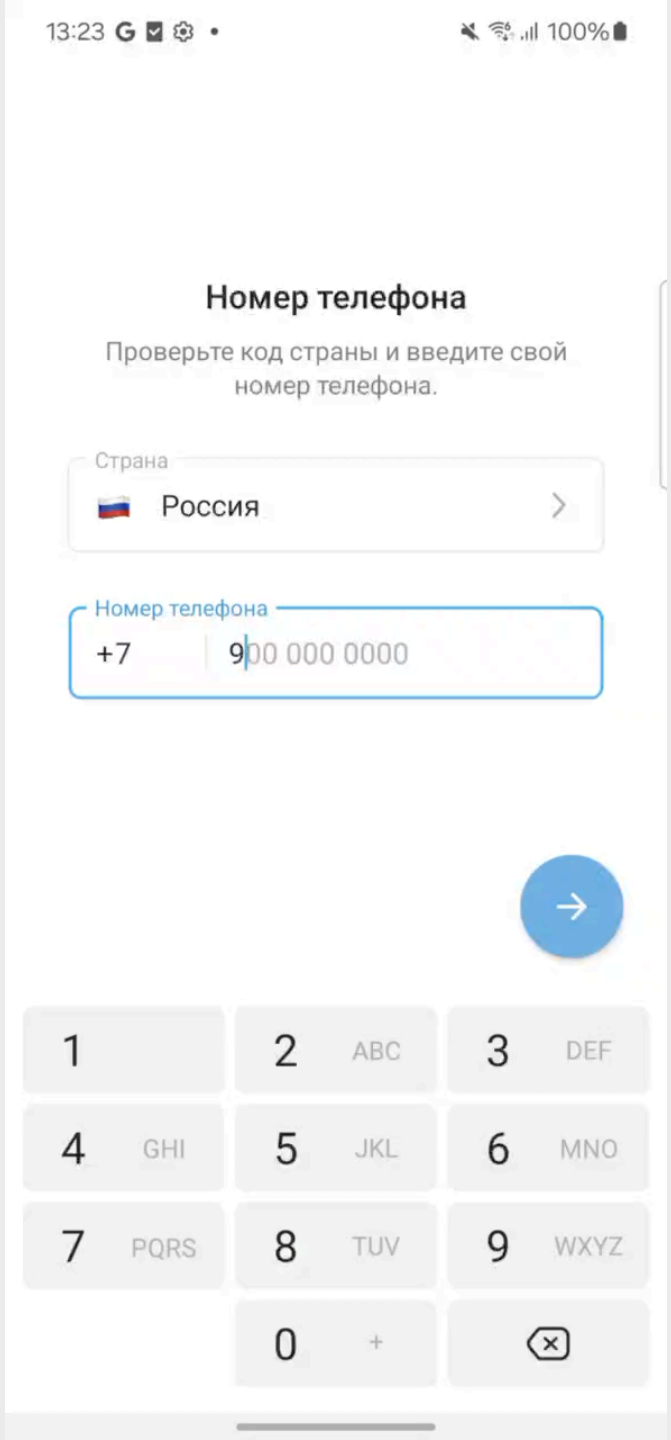
TMessagesProj/src/main/java/org/telegram/messenger/AuthTokensHelper.java

```
53              editor.apply();
54              //    BackupAgent.requestBackup(ApplicationLoader.applicationContext);
55          }

110             editor.apply();
111             BackupAgent.requestBackup(ApplicationLoader.applicationContext);
112         }
```

Show 1 more match


13:23
100%
Номер телефона
Проверьте код страны и введите свой номер телефона.
Страна
Россия
Номер телефона
+7
900 000 0000
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0 +


13:23
100%
Номер телефона
Проверьте код страны и введите свой номер телефона.
Страна
Россия
Номер телефона
+7
900 000 0000
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0 +

Виды авторизационных токенов

Access Token

Refresh Token

Виды авторизационных токенов

Access Token

Refresh Token

Виды авторизационных токенов

Access Token

- Авторизует все запросы => дает доступ к ресурсам от лица пользователя

Refresh Token

Виды авторизационных токенов

Access Token

- Авторизует все запросы => дает доступ к ресурсам от лица пользователя
- Живет очень мало (пару часов)

Refresh Token

Виды авторизационных токенов

Access Token

- Авторизует все запросы => дает доступ к ресурсам от лица пользователя
- Живет очень мало (пару часов)
- Получаем через Refresh Token

Refresh Token

Виды авторизационных токенов

Access Token

- Авторизует все запросы => дает доступ к ресурсам от лица пользователя
- Живет очень мало (пару часов)
- Получаем через Refresh Token

Refresh Token

- Выписывает Access Token

Виды авторизационных токенов

Access Token

- Авторизует все запросы => дает доступ к ресурсам от лица пользователя
- Живет очень мало (пару часов)
- Получаем через Refresh Token

Refresh Token

- Выписывает Access Token
- Живет долго (месяцы или бесконечно)

Виды авторизационных токенов

Access Token

- Авторизует все запросы => дает доступ к ресурсам от лица пользователя
- Живет очень мало (пару часов)
- Получаем через Refresh Token

Refresh Token

- Выписывает Access Token
- Живет долго (месяцы или бесконечно)
- Требует усиленной защиты

Виды авторизационных токенов

Access Token

- Авторизует все запросы => дает доступ к ресурсам от лица пользователя
- Живет очень мало (пару часов)
- Получаем через Refresh Token

Refresh Token

- Выписывает Access Token
- Живет долго (месяцы или бесконечно)
- Требует усиленной защиты
- Инвалидация его приводит к инвалидации AT

Trusted Hash — производный токен от Refresh Token

Trusted Hash

- Производный от RT => выдает AT
- Привязан к device_id
- Живет сильно меньше (1 неделя максимум)
- Одноразовый
- Инвалидация RT = инвалидация TH

Agenda

1. Контекст бизнеса: зачем?

2. Детально про реализацию: как?
   - key-value backup
   - autobackup
   - blockstore
   - тестирование

3. Результаты: что в итоге? 💵

Key = уникальный ключ блока данных

Value = массив байт для этого ключа

5 МБ

Ограничение по данным на одно приложение

Включаем бэкап в манифесте

```
<application
    android:allowBackup="true"
    android:backupAgent="com.vk.VkBackupAgent">
```

Включаем бэкап в манифесте

```
<application
    android:allowBackup="true"
    android:backupAgent="com.vk.VkBackupAgent">

    <meta-data android:name="com.google.android.backup.api_key"
        android:value="unused" />
```

Создаем базового наследника

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onCreate() {
        super.onCreate()
        ...
    }
```

Добавляем префы для бэкапирования

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onCreate() {
        super.onCreate()

        addHelper(
            keyPrefix = "prefs_key",
            helper = SharedPreferencesBackupHelper(
                context = this,
                prefGroups = arrayOf("prefs_trusted_hash")
            )
        )

        ...
    }
```

Добавляем файлы для бэкапирования

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onCreate() {
        super.onCreate()

        addHelper(
            keyPrefix = "prefs_key",
            helper = SharedPreferencesBackupHelper(
                context = this,
                prefGroups = arrayOf("prefs_trusted_hash")
            )
        )

        addHelper(
            keyPrefix = "files_key",
            helper = FileBackupHelper(
                context = this,
                files = arrayOf("file_auth.txt")
            )
        )

    }
```

Шифрование бэкапа

> Android 9+

> Включен пин-код
на устройстве

Проверяем наличие шифрования

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onBackup(
        oldState: ParcelFileDescriptor?,
        data: BackupDataOutput?,
        newState: ParcelFileDescriptor?
    ) {
        val isClientSideBackupEncryptionEnabled = Build.VERSION.SDK_INT >= Build.VERSION_CODES.P // android 9
                && data != null
                && (data.transportFlags and FLAG_CLIENT_SIDE_ENCRYPTION_ENABLED) != 0

        if (isClientSideBackupEncryptionEnabled) {
            super.onBackup(oldState, data, newState)
        }
    }
```

Проверяем д2д перенос + аналитика

```
class VkBackupAgent : BackupAgentHelper() {

        override fun onBackup(
            oldState: ParcelFileDescriptor?,
            data: BackupDataOutput?,
            newState: ParcelFileDescriptor?
        ) {
        super.onBackup(oldState, data, newState)

        val isClientSideBackupEncryptionEnabled = Build.VERSION.SDK_INT >= Build.VERSION_CODES.P // android 9
                && data != null
                && (data.transportFlags and FLAG_CLIENT_SIDE_ENCRYPTION_ENABLED) != 0

        val isD2DBackup = Build.VERSION.SDK_INT >= Build.VERSION_CODES.P // android 9
                && data != null
                && (data.transportFlags and FLAG_DEVICE_TO_DEVICE_TRANSFER) != 0

        EventBuilder().event(
            SchemeStat.Item(
                key = EventKey.BACKUP,
                value = isClientSideBackupEncryptionEnabled,
                value2 = isD2DBackup
            )
        ).buildAndSend()
    }
```

Уведомляем систему о необходимости забэкапить наши данные

```
override fun afterSuccessAuth(authResult: AuthResult) {
    val prefs = context.getSharedPreferences("prefs_trusted_hash", Context.MODE_PRIVATE)
    prefs.edit {
        putString("trusted_hash_key", authResult.trustedHash)
    }

    BackupManager(context).dataChanged()
}
```

Пост-процессинг после восстановления данных

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onRestoreFinished() {
        File(filesDir, "onRestoreFinished.txt")
            .writeText(System.currentTimeMillis().toString())
    }
```

onRestoreFinished запускается в ограниченном режиме

> Content Providers не инициализируются

> MainActivity не запускается

> Используется базовый Application класс

Потоко-безопасность

Создаем объект-монитор или мютекс на блокировку

```
class AuthRepository {
    companion object {
        const val AUTH_FILE = "auth.txt"
        val lock = Any()
    }

    fun save(userId: Long) {
        synchronized(lock) {
            File(context.filesDir, AUTH_FILE).writeText(userId.toString())
        }
    }
}
```

Потокобезопасность

```
class AuthRepository {
    companion object {
        const val AUTH_FILE = "auth.txt"
        val lock = Any()
    }

    fun save(userId: Long) {
        synchronized(lock) {
            File(context.filesDir, AUTH_FILE).writeText(userId.toString())
        }
    }
}
```

Потокобезопасность

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onCreate() {
        super.onCreate()
        addHelper(
            "files_key",
            FileBackupHelper(this, AuthRepository.AUTH_FILE)
        )
    }

    override fun onBackup(
        oldState: ParcelFileDescriptor?,
        data: BackupDataOutput?,
        newState: ParcelFileDescriptor?
    ) {
        synchronized(AuthRepository.lock) {
            super.onBackup(oldState, data, newState)
        }
    }

    override fun onRestore(
        data: BackupDataInput?,
        appVersionCode: Int,
        newState: ParcelFileDescriptor?
    ) {
        synchronized(AuthRepository.lock) {
            super.onRestore(data, appVersionCode, newState)
        }
    }
```

Кастомная имплементация бэкапа

Зачем?

> Бэкапирование БД

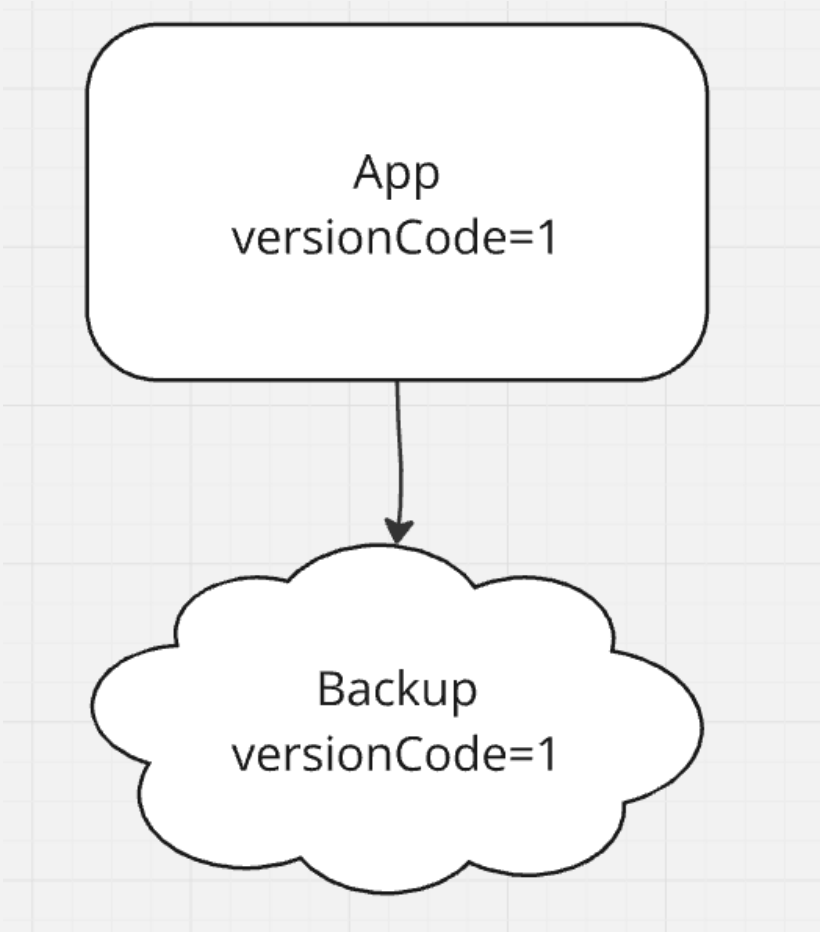
> Перекрестные проверки appVersion

> Бэкапирование части данных, не всего файла

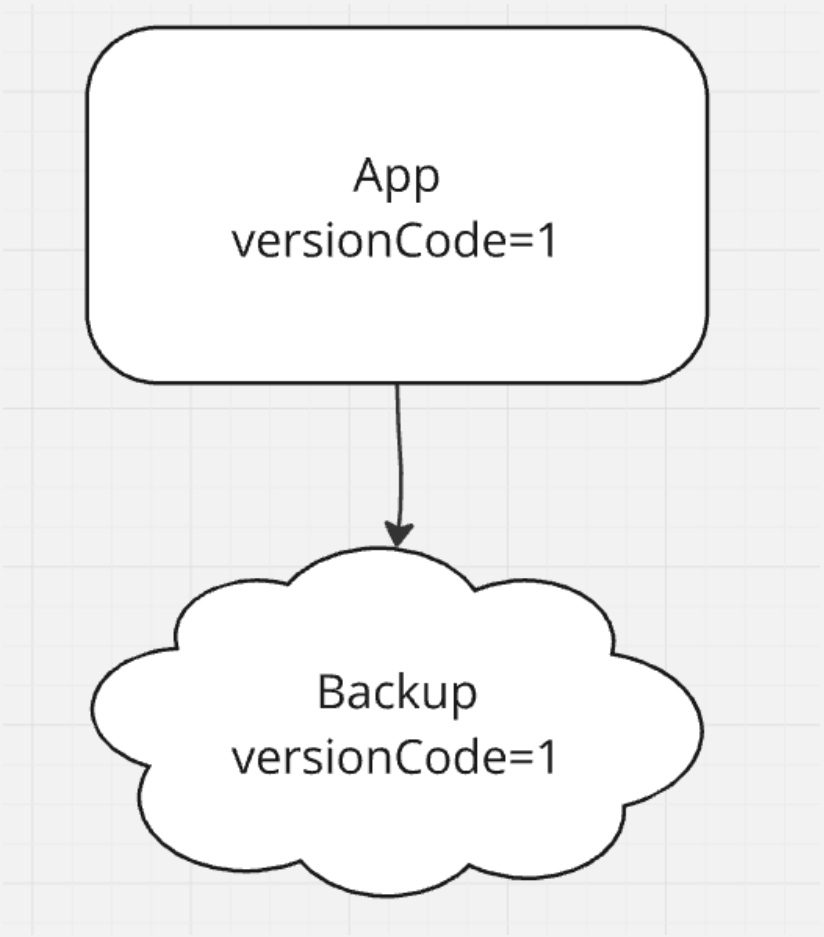
Кастомная имплементация

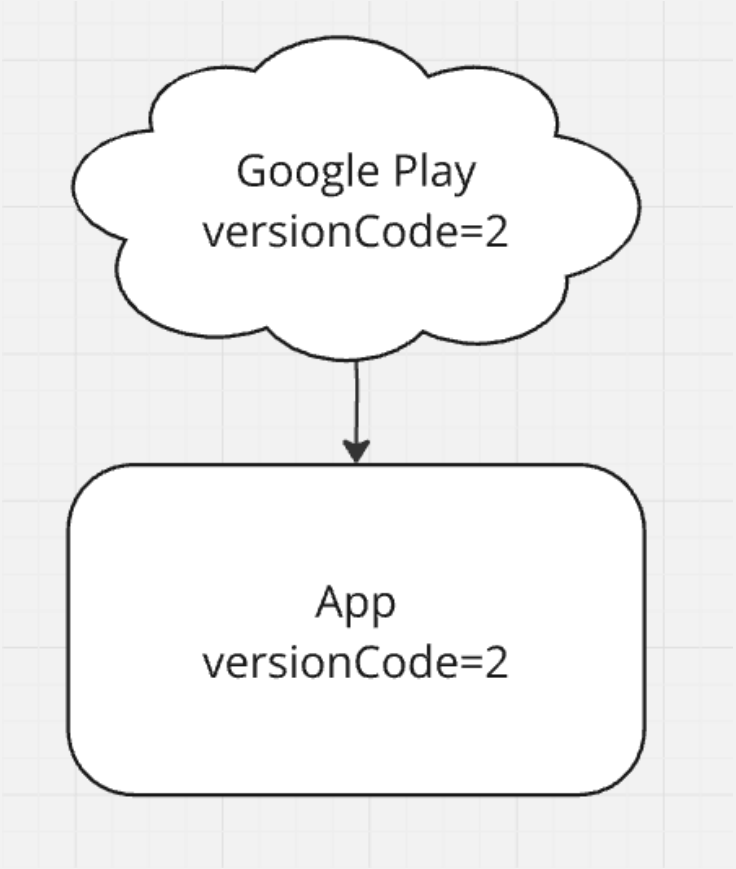
github.com/iartr/backup

Версионирование бэкапа

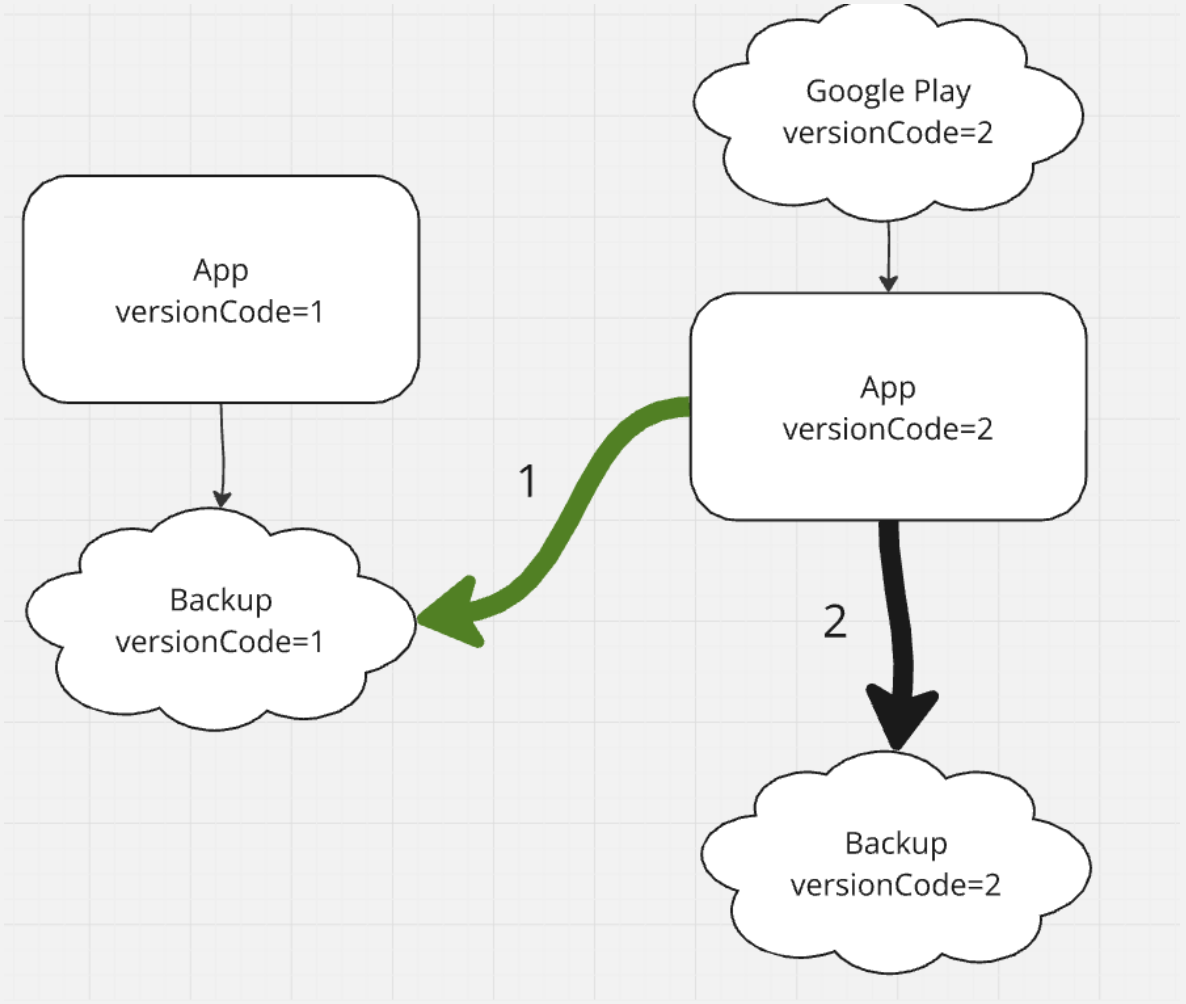


Версионирование бэкапа

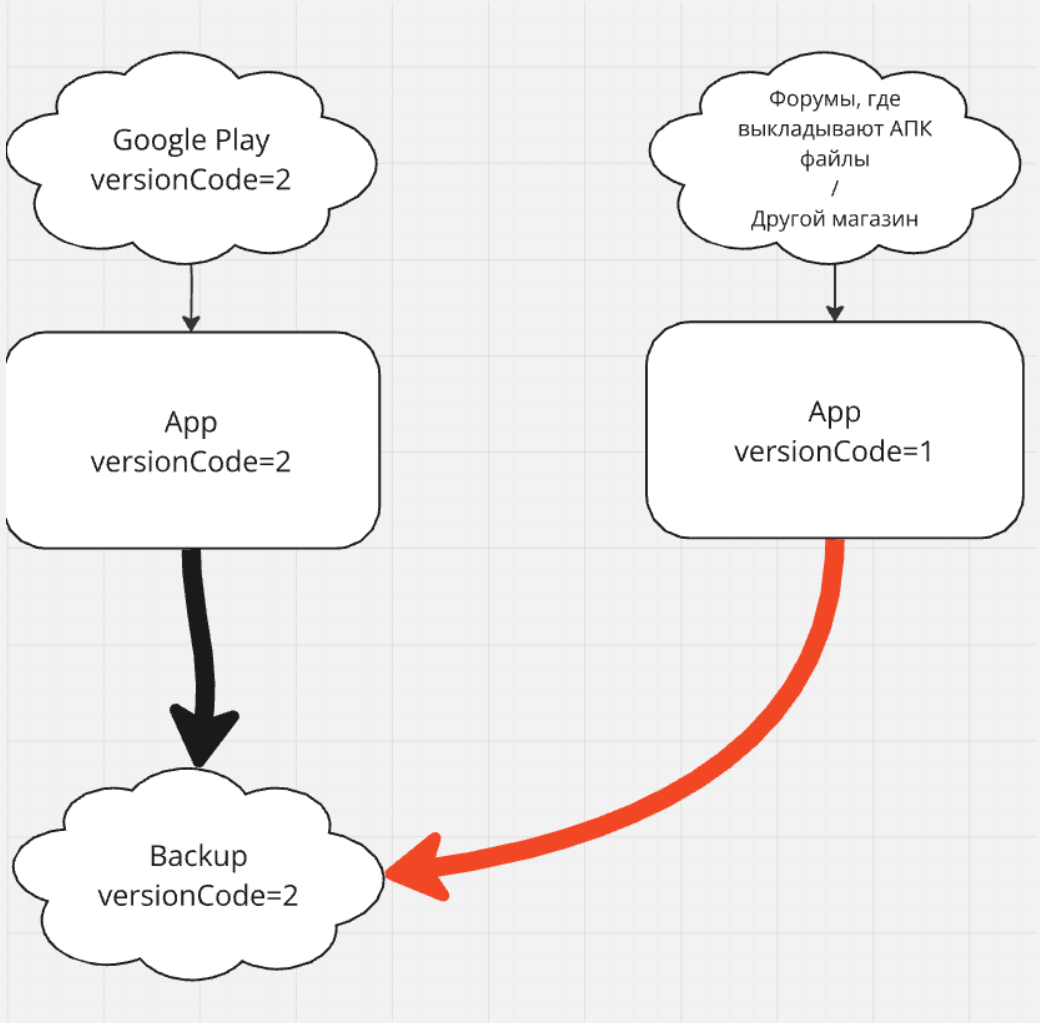




Версионирование бэкапа



Новая версия не понравилась, откатил обратно



version backup > version app

По умолчанию невозможно

```
<!-- restoreAnyVersion=false по умолчанию -->
<application
    android:allowBackup="true"
    android:backupAgent="com.vk.VkBackupAgent"
    android:restoreAnyVersion="false"
```

version backup > version app

```
<!-- restoreAnyVersion=false по умолчанию -->
<application
    android:allowBackup="true"
    android:backupAgent="com.vk.VkBackupAgent"
    android:restoreAnyVersion="true"
```

Backup Privacy

User privacy

At Google, we are keenly aware of the trust users place in us and our responsibility to protect users' privacy. Google securely transmits backup data to and from Google servers in order to provide backup and restore features. Google treats this data as personal information in accordance with Google's Privacy Policy.

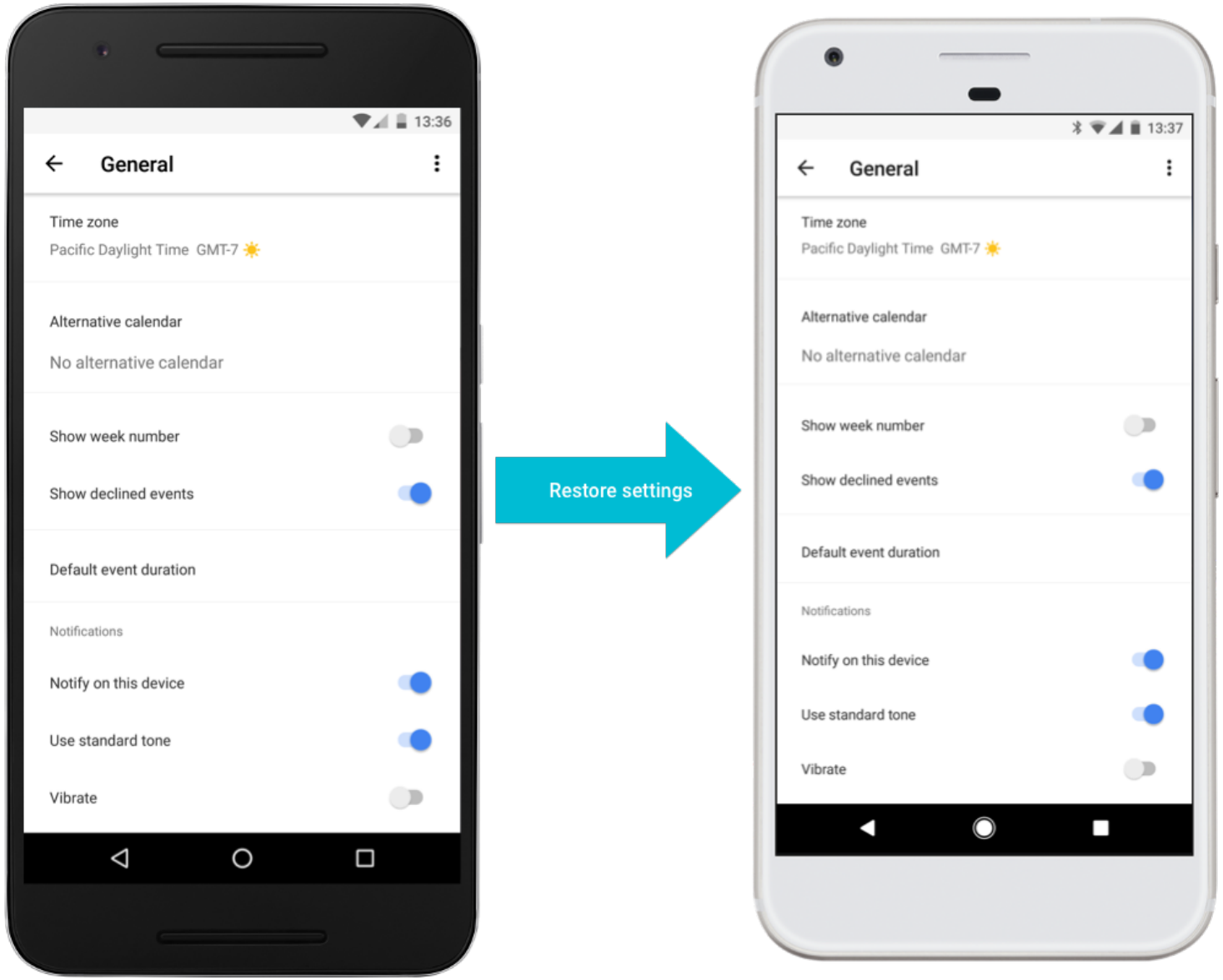
In addition, users can disable data backup functionality through the Android system's backup settings. When a user disables backup, Android Backup Service deletes all saved backup data. A user can re-enable backup on the device, but Android Backup Service will not restore any previously deleted data.

Backup может работать без гуглосервисов

Зависит от вендора

Что бэкапить?

Пользовательские настройки



Что бэкапить?

> Авторизационные секреты

> Токены аутентификации

> Примеры: telegram, whatsapp, vk

Что НЕ бэкапить?

URIs

Резюме

key-value бэкап — это просто

Agenda

1. Контекст бизнеса: зачем?

2. Детально про реализацию: как?
   - key-value backup
   - autobackup
   - blockstore
   - тестирование

3. Результаты: что в итоге? 💵

25 МБ

Ограничение по данным на одно приложение

Google drive

Место хранения

Каталоги резервного копирования

Почти все, что назначено вашему приложению

/shared_preferences

/databases

/files

/external_storage

/custom_dirs

Каталоги резервного копирования

Почти все, что назначено вашему приложению

/shared_preferences /databases

/files /external_storage

/custom_dirs

❌ Не включено:

/caches /code_cache /no_backup

Включаем бэкап в манифесте

```
<application
    android:allowBackup="true"
    android:backupAgent="@null">
```

Android 12 и выше + allowBackup=false

Будет выполнен device-to-device перенос данных

Контроль бэкапируемых данных

```
<application
    android:allowBackup="true"
    android:fullBackupContent="@xml/backup_android11_lower"
    android:dataExtractionRules="@xml/backup_android12_above">
```

Android 11 и ниже

```
<full-backup-content>
    <include domain="sharedpref" path="."/>
    <exclude domain="sharedpref" path="device.xml"/>

    <include domain="database" path="2fa.db"
        requireFlags="clientSideEncryption" />
</full-backup-content>
```

Android 11 и ниже

```
<full-backup-content>
    <include domain="sharedpref" path="."/>
    <exclude domain="sharedpref" path="device.xml"/>

    <include domain="database" path="2fa.db"
        requireFlags="clientSideEncryption" />
</full-backup-content>
```

Android 11 и ниже

```
<full-backup-content>
    <include domain="sharedpref" path="."/>
    <exclude domain="sharedpref" path="device.xml"/>

    <include domain="database" path="2fa.db"
        requireFlags="clientSideEncryption" />
</full-backup-content>
```

Android 11 и ниже

```
<full-backup-content>
    <include domain="sharedpref" path="."/>
    <exclude domain="sharedpref" path="device.xml"/>

    <include domain="database" path="auth_secrets.db"
        requireFlags="clientSideEncryption" />
</full-backup-content>
```

Android 11 и ниже

```
<full-backup-content>
    <include domain="sharedpref" path="."/>
    <exclude domain="sharedpref" path="device.xml"/>

    <include domain="database" path="2fa.db"
        requireFlags="clientSideEncryption" />

    <include domain="database" path="clips_drafts.db"
        requireFlags="deviceToDeviceTransfer" />
</full-backup-content>
```

Android 12 и выше

```
<data-extraction-rules>
    <cloud-backup>
    </cloud-backup>
    <device-transfer>
    </device-transfer>
</data-extraction-rules>
```

Android 12 и выше

```
<cloud-backup disableIfNoEncryptionCapabilities="true">
    <include domain="sharedpref" path="."/>
    <exclude domain="sharedpref" path="device.xml"/>

    <include domain="database" path="2fa.db"/>
</cloud-backup>
```

Android 12 и выше

```
<cloud-backup disableIfNoEncryptionCapabilities="true">
    <include domain="sharedpref" path="."/>
    <exclude domain="sharedpref" path="device.xml"/>

    <include domain="database" path="2fa.db"/>
</cloud-backup>
```

Android 12 и выше

```
<device-transfer>
    <include domain="database" path="clips_drafts.db" />
</device-transfer>
```

Настройка BackupAgent

Настройка BackupAgent

```
<application
    android:allowBackup="true"
    android:backupAgent="com.vk.VkBackupAgent">
```

При указании агента система
считает, что это key-value backup

Настройка BackupAgent

```
<application
    android:allowBackup="true"
    android:backupAgent="com.vk.VkBackupAgent"
    android:fullBackupOnly="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules">
```

Настройка BackupAgent

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onRestoreFinished() {
        val json = JSONObject().put(KEY_TIMESTAMP, System.currentTimeMillis()).toString()
        File(noBackupFilesDir, "onRestoreFinished").writeText(json)
    }

    override fun onQuotaExceeded(backupDataBytes: Long, quotaBytes: Long) {
        val json = JSONObject()
            .put("backupDataBytes", backupDataBytes)
            .put("quotaBytes", quotaBytes)
            .toString()
        File(noBackupFilesDir, "onQuotaExceeded").writeText(json)
    }
```

Настройка BackupAgent

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onRestoreFinished() {
        val json = JSONObject().put(KEY_TIMESTAMP, System.currentTimeMillis()).toString()
        File(noBackupFilesDir, "onRestoreFinished").writeText(json)
    }

    override fun onQuotaExceeded(backupDataBytes: Long, quotaBytes: Long) {
        val json = JSONObject()
            .put("backupDataBytes", backupDataBytes)
            .put("quotaBytes", quotaBytes)
            .toString()
        File(noBackupFilesDir, "onQuotaExceeded").writeText(json)
    }
```

Резюме

autobackup — это еще проще

Сравнение способов

	key-value	autobackup
Версии Android	Android 2+	Android 6+

Сравнение способов

	key-value	autobackup
Версии Android	Android 2+	Android 6+
Имплементация	Через BackupAgent	Через XML

Сравнение способов

	key-value	autobackup
Версии Android	Android 2+	Android 6+
Имплементация	Через BackupAgent	Через XML
Частота выполнения	Каждые несколько часов	Примерно раз в день

Сравнение способов

	key-value	autobackup
Версии Android	Android 2+	Android 6+
Имплементация	Через BackupAgent	Через XML
Частота выполнения	Каждые несколько часов	Примерно раз в день
Состояние сети	Wifi + Cellular	Wifi only

Сравнение способов

	key-value	autobackup
Версии Android	Android 2+	Android 6+
Имплементация	Через BackupAgent	Через XML
Частота выполнения	Каждые несколько часов	Примерно раз в день
Состояние сети	Wifi + Cellular	Wifi only
Состояние приложения в момент бэкапа	Может работать	Убивается

Сравнение способов

	key-value	autobackup
Версии Android	Android 2+	Android 6+
Имплементация	Через BackupAgent	Через XML
Частота выполнения	Каждые несколько часов	Примерно раз в день
Состояние сети	Wifi + Cellular	Wifi only
Состояние приложения в момент бэкапа	Может работать	Убивается
Хранилище бэкапа	Google сервера	Google drive юзера

Сравнение способов

	key-value	autobackup
Версии Android	Android 2+	Android 6+
Имплементация	Через BackupAgent	Через XML
Частота выполнения	Каждые несколько часов	Примерно раз в день
Состояние сети	Wifi + Cellular	Wifi only
Состояние приложения в момент бэкапа	Может работать	Убивается
Хранилище бэкапа	Google сервера	Google drive юзера
Наличие гугл аккаунта на устройстве	Обязательно	Обязательно

Agenda

1. Контекст бизнеса: зачем?

2. Детально про реализацию: как?
   - key-value backup
   - autobackup
   - blockstore
   - тестирование

3. Результаты: что в итоге? 💵

16 наборо байт

для добавления

4 КБ

на каждый ключ

Базируется на инфрастуктуре бэкапа

```
dependencies {
  implementation 'com.google.android.gms:play-services-auth-blockstore:16.2.0'
}
```

Параметры запроса на сохранение — ключ + набор байт

```
fun save(context: Context, credentials: UserData) {
    Blockstore.getClient(context)
        .storeBytes(
            StoreBytesData.Builder()
                .setKey(credentials.userId.toString())
                .setBytes(credentials.authSecret.toByteArray())
                .build()
        )
}
```

Отправляем запрос — получаем Task

```
fun save(context: Context, credentials: UserData) {
    val storeRequest = StoreBytesData.Builder()
        .setKey(credentials.userId.toString())
        .setBytes(credentials.authSecret.toByteArray())
       .build()

    val blockstore = Blockstore.getClient(context)

    val task: Task<Int> = blockstore.storeBytes(storeRequest)
}
```

Колбэки на Task api

```
fun save(context: Context, credentials: UserData) {
    val storeRequest = StoreBytesData.Builder()
        .setKey(credentials.userId.toString())
        .setBytes(credentials.authSecret.toByteArray())
        .setShouldBackupToCloud(true) // можно только при переустановках оставить
        .build()

    val blockstore = Blockstore.getClient(context)

    val task: Task<Int> = blockstore.storeBytes(storeRequest)
        .addOnSuccessListener { storedBytes: Int ->

        }
        .addOnFailureListener { th: Throwable ->

        }
        .addOnCompleteListener { task: Task<Int> ->

        }
}
```

Все колбэки на главном потоке, а сами операции не блокируют текущий поток

```
@WorkerThread
fun save(context: Context, credentials: UserData) {
    val storeRequest = StoreBytesData.Builder()
        .setKey(credentials.userId.toString())
        .setBytes(credentials.authSecret.toByteArray())
        .setShouldBackupToCloud(true) // можно только при переустановках оставить
        .build()

    val blockstore = Blockstore.getClient(context)

    val task: Task<Int> = blockstore.storeBytes(storeRequest)
        .addOnSuccessListener { storedBytes: Int ->
            // MAIN THREAD
        }
        .addOnFailureListener { th: Throwable ->
            // MAIN THREAD
        }
        .addOnCompleteListener { task: Task<Int> ->
            // MAIN THREAD
        }
}
```

Меняем пул потоков для колбэков

```
fun save(context: Context, credentials: UserData, callbackExecutor: Executor) {
    val storeRequest = StoreBytesData.Builder()
        .setKey(credentials.userId.toString())
        .setBytes(credentials.authSecret.toByteArray())
        .setShouldBackupToCloud(true) // можно только при переустановках оставить
        .build()

    val blockstore = Blockstore.getClient(context)

    val task: Task<Int> = blockstore.storeBytes(storeRequest)
        .addOnSuccessListener(executor) { /* other thread */ }
}
```

Привязываем колбэки к ЖЦ 👎

```
fun save(context: Context, credentials: UserData, activity: Activity) {
    val storeRequest = StoreBytesData.Builder()
        .setKey(credentials.userId.toString())
        .setBytes(credentials.authSecret.toByteArray())
        .setShouldBackupToCloud(true) // можно только при переустановках оставить
        .build()

    val blockstore = Blockstore.getClient(context)

    val task: Task<Int> = blockstore.storeBytes(storeRequest)
        .addOnSuccessListener(activity) { /* lifecycle-based */ }
}
```

Конвертация Task в Rx / Deferred

```
fun save(context: Context, credentials: UserData) {
    val storeRequest = StoreBytesData.Builder()
        .setKey(credentials.userId.toString())
        .setBytes(credentials.authSecret.toByteArray())
        .setShouldBackupToCloud(true) // можно только при переустановках оставить
        .build()

    val blockstore = Blockstore.getClient(context)

    blockstore.storeBytes(storeRequest)
        .toSingle()
        .subscribeOn(Schedulers.io())
        .observeOn(Schedulers.io())
        .subscribeSuccess { storedBytes ->

        }
}
```

Kotlin Coroutine

Usage

Add the following dependency to your project and use the code below to convert from a `Task`.

Gradle (module-level `build.gradle`, usually `app/build.gradle`)

```
// Source: https://github.com/Kotlin/kotlinx.coroutines/tree/master/integration/kotlinx-coroutines-play-se
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-play-services:1.7.3'
```

build.gradle

Snippet

```
import kotlinx.coroutines.tasks.await
// ...
  textView.text = simpleTask.await()
}
```

MainActivity.kt

developers.google.com/
android/guides/tasks

Проверка на доступность шифрования

```
blockstore.isEndToEndEncryptionAvailable
    .toSingle()
```

Достаем фиче-тоггл + доступность шифрования

```
blockstore.isEndToEndEncryptionAvailable
    .toSingle()
    .zipWith(metaInf.get()) { p1, p2 -> p1 to p2 }
    .flatMap { (isEncryptionAvailable, metaInf) ->

    }
```

Нет ни того ни того — завершаем операцию сохранения

```
blockstore.isEndToEndEncryptionAvailable
    .toSingle()
    .zipWith(metaInf.get()) { p1, p2 -> p1 to p2 }
    .flatMap { (isEncryptionAvailable, metaInf) ->
        if (!isEncryptionAvailable || !metaInf.isToggleEnabled) {
            return@flatMap Single.error(
                IllegalStateException(
                    """
                        isEncryptionAvailable=$isEncryptionAvailable,
                        isToggleEnabled=${metaInf.isToggleEnabled}
                    """.trimIndent()
                )
            )
        }
    }
```

Можно не отправлять данные в облако, оставить сохранение на переустановки

```
blockstore.isEndToEndEncryptionAvailable
    .toSingle()
    .zipWith(metaInf.get()) { p1, p2 -> p1 to p2 }
    .flatMap { (isEncryptionAvailable, metaInf) ->
        if (!isEncryptionAvailable || !metaInf.isToggleEnabled) {
            return@flatMap Single.error(
                IllegalStateException(
                    """
                        isEncryptionAvailable=$isEncryptionAvailable,
                        isToggleEnabled=${metaInf.isToggleEnabled}
                    """.trimIndent()
                )
            )
        }

        val storeBytesRequest = StoreBytesData.Builder()
            .setShouldBackupToCloud(false)
            .setKey(credentials.userId.toString())
            .setBytes(credentials.authSecret.toByteArray())
            .build()

        blockstore.storeBytes(storeBytesRequest).toSingle()
    }
    .subscribe()
```

Запрос на удаление по ключу или сразу всего

```
fun delete(context: Context, credentials: UserData) {
    val deleteRequest = DeleteBytesRequest.Builder()
        .setDeleteAll(false)
        .setKeys(listOf(credentials.userId.toString()))
        .build()

    Blockstore.getClient(context)
        .deleteBytes(deleteRequest)
        .addOnCompleteListener { /**/ }
}
```

```
fun delete(context: Context, credentials: UserData) {
    val deleteRequest = DeleteBytesRequest.Builder()
        .setDeleteAll(false)
        .setKeys(listOf(credentials.userId.toString()))
        .build()

    Blockstore.getClient(context)
        .deleteBytes(deleteRequest)
        .addOnCompleteListener { /**/ }
}
```

Разбираем кейс получения данных только при переустановках

```
fun getAllAsync(context: Context): Single<List<UserData>> {

    val blockstore = Blockstore.getClient(context)
    metaInf.get()
        .flatMap { metaInf ->
            if (!metaInf.isToggleEnabled) {
                return@flatMap Single.error(IllegalStateException("Toggle disabled"))
            }

        }

}
```

Запрос на получение всех ключей-значений

```
fun getAllAsync(context: Context): Single<List<UserData>> {

    val blockstore = Blockstore.getClient(context)

    metaInf.get()
        .flatMap { metaInf ->
            if (!metaInf.isToggleEnabled) {
                return@flatMap Single.error(IllegalStateException("Toggle disabled"))
            }

            val retrieveRequest = RetrieveBytesRequest.Builder()
                .setRetrieveAll(true)
                .build()

            blockstore.retrieveBytes(retrieveRequest).toSingle()
        }

}
```

Проходимся по всем ключам

```
fun getAllAsync(context: Context): Single<List<UserData>> {
    metaInf.get()
        .flatMap { metaInf ->
            if (!metaInf.isToggleEnabled) {
                return@flatMap Single.error(IllegalStateException("Toggle disabled"))
            }

            val retrieveRequest = RetrieveBytesRequest.Builder()
                .setRetrieveAll(true)
                .build()

            blockstore.retrieveBytes(retrieveRequest).toSingle()
        }
        .map { response: RetrieveBytesResponse ->
            val data: Map<String, RetrieveBytesResponse.BlockstoreData> = response.blockstoreDataMap
            data.mapNotNull { (userIdKey, bytes) ->
                val userId = userIdKey.toLongOrNull() ?: return@mapNotNull null
                val authSecret = String(bytes.getBytes()).takeIf(String::isNotBlank) ?: return@mapNotNull null
                UserData(userId, authSecret)
            }
        }
}
```

Конвертим ключ в userId, а байты — в строчки

```
fun getAllAsync(context: Context): Single<List<UserData>> {
    metaInf.get()
        .flatMap { metaInf ->
            if (!metaInf.isToggleEnabled) {
                return@flatMap Single.error(IllegalStateException("Toggle disabled"))
            }

            val retrieveRequest = RetrieveBytesRequest.Builder()
                .setRetrieveAll(true)
                .build()

            blockstore.retrieveBytes(retrieveRequest).toSingle()
        }
        .map { response: RetrieveBytesResponse ->
            val data: Map<String, RetrieveBytesResponse.BlockstoreData> = response.blockstoreDataMap
            data.mapNotNull { (userIdKey, bytes) ->
                val userId = userIdKey.toLongOrNull() ?: return@mapNotNull null
                val authSecret = String(bytes.getBytes()).takeIf(String::isNotBlank) ?: return@mapNotNull null
                UserData(userId, authSecret)
            }
        }
}
```

Грязная функция с записью данных из блокстора во внутреннее хранилище

```
fun getAllAsync(context: Context): Single<List<UserData>> {
    ...
        .map { response: RetrieveBytesResponse ->
            val data: Map<String, RetrieveBytesResponse.BlockstoreData> = response.blockstoreDataMap
            data.mapNotNull { (userIdKey, bytes) ->
                val userId = userIdKey.toLongOrNull() ?: return@mapNotNull null
                val authSecret = String(bytes.getBytes()).takeIf(String::isNotBlank) ?: return@mapNotNull null
                UserData(userId, authSecret)
            }
        }
        .doOnSuccess { usersList ->
            usersList.forEachSafe { originalStore.save(context, it) }
        }
}
```

Пост-процессинг — говорим, что операция завершена

```
fun getAllAsync(context: Context): Single<List<UserData>> {

    ...
        .doOnSuccess { usersList ->
            usersList.forEachSafe { originalStore.save(context, it) }
        }
        .doAfterTerminate {
            File(context.noBackupFilesDir, "BlockstoreCompleted.txt")
                .writeText("completed")
        }
}
```

В начале делаем проверку на то, что пост-процессинг уже был сделан

```
fun getAllAsync(context: Context): Single<List<UserData>> {

    if (File(context.noBackupFilesDir, "BlockstoreCompleted.txt").exists()) {
        return Single.just(emptyList())
    }

    return ...
        .doAfterTerminate {
            File(context.noBackupFilesDir, "BlockstoreCompleted.txt")
                .writeText("completed")
        }
}
```

Резюме

blockstore — это просто

Нужно написать немного кода

Сравнение способов

	Blockstore	Backup
Что можно сохранять?	Наборы байт	Любые файлы

Сравнение способов

	Blockstore	**Backup**
Что можно сохранять?	Наборы байт	Любые файлы
Версии андроида	Android 6+	Android 2+ Android 6+

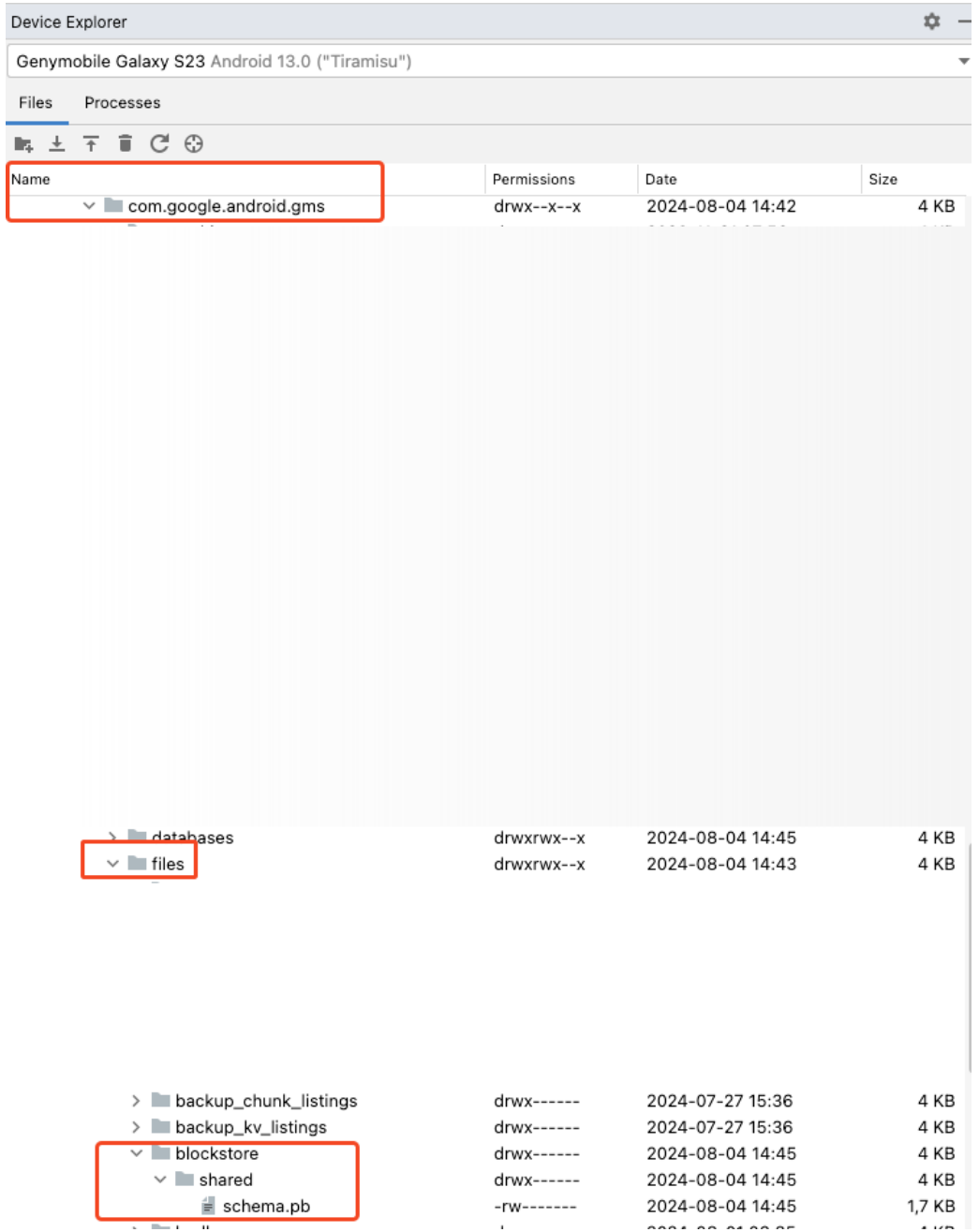
Сравнение способов

	Blockstore	**Backup**
Что можно сохранять?	Наборы байт	Любые файлы
Версии андроида	Android 6+	Android 2+ Android 6+
Сторонние зависимости	GMS Lib	Отсутствуют

Сравнение способов

	Blockstore	**Backup**
Что можно сохранять?	Наборы байт	Любые файлы
Версии андроида	Android 6+	Android 2+ Android 6+
Сторонние зависимости	GMS Lib	Отсутствуют
Место хранения данных	**backupToCloud=false** => приватная гугловая директория на устройстве **backupToCloud=true** => Google сервера	Google сервера Google drive

Как выглядит blockstore на девайсе?



schema.pb

Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)

```
L
FFaXNFbmFibGVkDC2<
"SOHs9***R*DC4*-*&**ESCETB*,**x*>/**DDC2SYN
FFcom.vk.callsDLESOHSUBEOTBSNULDLEEOTDC2FFcom.vk.callsJ*ENQ
-jzMbwk5zgjlxYhY95YMN3GISyTNSiqDQfZIrsV3Vb1U=
DC2*ENQ
I
FFaXNFbmFibGVkDC29
"SOHs9**N*q>U*P**5*FFNUL*7***z*/*
J*>*}DC2DC3
    com.vk.imDLESOHSUBEOTBSNULDLEEOT
*STX
FFODYxOTQ1NDgwDC2*STX
*SOHSOHs9**F*zI]****RSSO&*fSUB;SUB*K@*62ACK>qo**.y;**=US**ETXu**nSa5*EOT**|C*}**HSYNhf*ENQ2K{GSax*SOH*
*z@V/7,*)*SIVT*S[**.*EPANAK***[-*o***&$**NULSUBEOTHRTCjL]K*!.)s0;L*****kDC2DC3
    com.vk.imDLESOHSUBEOTBSNULDLEEOT
*STX
FFODU2OTE1ODI5DC2*STX
*SOHSOHs9***ae-***NAK
**DEL2***NM**Ai**^`z**}*c*7*SYN<**:*;BS%*bKDEL'1*RS&*Z&wv1*gSTX*|]**UScDC1**RS}**?nvF0M'A**T*;*EOTEOTr*S
*T*(EM**V*Y*QZ  q***R*CB*CAN%***i,ETB*M**$:**DC2DC3
    com.vk.imDLESOHSUBEOTBSNULDLEEOTDC2  com.vk.imJ*SOH
-eVBAE4M67jT1Dz835pGg/YrjIx6l95yOxNBgAOumdms=
DC2~
a
DLEdXNlcl9pZF9rZXk=DC2M
&SOHs9**F*}yETBW*-****s**a*$****7*3e*+*VEM*DC2#
EMcom.example.myapplicationDLESOHSUBEOTBSNULDLEEOTDC2EMcom.example.myapplicationJ*ETX
-XUvh4ppG75shx2NF3VdP/g+BIFwdxkx7r7lnHbrTV8Q=
DC2*ETX
U
FFaXNFbmFibGVkDC2E
"SOHs9**7**'L**o*l*f*BEL**V****r*DC2US
NAKcom.vkontakte.androidDLESOHSUBEOTBSNULDLEEOT
*STX
FFODYxOTIwNjYzDC2*STX
*SOHSOHs9*****&o#**7DC4**]JSI******F*rNULDC1EOT*(/*BS**VT**;b*.rE6*X[**G*DLE*Gs*SYNk*H*DLE*1*~***@v*ZGS*
a*v*^*;1**ñDEL***Ï***x*%_c*ç*F*STX***SIGSœ-,*DC2US
NAKcom.vkontakte.androidDLESOHSUBEOTBSNULDLEEOTDC2NAKcom.vkontakte.androidJ*ETX
-un31qrR50yh8IsatSzRyuNpDWzqh3s3w2RsAhoNkdwA=
DC2*ETX
a
FFbWV0YUluZg==DC2Q
.SOHs9*塌ACKE**T~*ØØZ*A,4*2}*mE****(v*XFSBSq*R***DC2US
NAKcom.vkontakte.androidDLENULSUBEOTBSNULDLEEOT
*STX
FFODY2NzA1ODE5DC2*STX
*SOHSOHs9****]SOH**>4**ñ**SOHFF**BS*J*YK*****>****5******,DELmAf**O**ħ*g'N`fBEL****xACK*&*DC2qG**&NAKg0¿
NAKcom.vkontakte.androidDLENULSUBEOTBSNULDLEEOTDC2NAKcom.vkontakte.android
```

Как выглядит blockstore на девайсе?

> **Protobuf формат данных**

> **Ключи хранятся в base64**

На скриншоте:

1) "metaInf" ключ — включен ли тоггл

2-3) userIds VK ID

> **Данные в зашифрованном виде**

Ключ = пин-код

Agenda

1. Контекст бизнеса: зачем?
2. Детально про реализацию: как?
   - key-value backup
   - autobackup
   - blockstore
   - тестирование
3. Результаты: что в итоге? 💵

Bash скрипты в деле

Перед началом тестирования

> Android 6+

> Залогиниться в гугл акк

> В настройках включить резервное копирование

Скрипт для тестирования: бэкапирование

```
#!/bin/bash -eu
: "${1?"Usage: $0 package name"}"
```

Скрипт для тестирования: бэкапирование

```
#!/bin/bash -eu
: "${1?"Usage: $0 package name"}"

# Initialize and create a backup
adb shell bmgr enable true
```

Скрипт для тестирования: бэкапирование

```
#!/bin/bash -eu
: "${1?"Usage: $0 package name"}"

# Initialize and create a backup
adb shell bmgr enable true
adb shell bmgr transport com.android.localtransport/.LocalTransport
```

Скрипт для тестирования: бэкапирование

```
#!/bin/bash -eu
: "${1?"Usage: $0 package name"}"

# Initialize and create a backup
adb shell bmgr enable true
adb shell bmgr transport com.android.localtransport/.LocalTransport
adb shell bmgr backupnow "$1" | grep -F "Package $1 with result: Success" || (echo "Backup failed"; exit 1)
```

Скрипт для тестирования: переустановка

```
apk_path_list=$(adb shell pm path "$1")
OIFS=$IFS
IFS=$'\n'
apk_number=0
for apk_line in $apk_path_list
do
    (( ++apk_number ))
    apk_path=${apk_line:8:1000}
    adb pull "$apk_path" "myapk${apk_number}.apk"
done
IFS=$OIFS
adb shell pm uninstall --user 0 "$1"
apks=$(seq -f 'myapk%.f.apk' 1 $apk_number)
adb install-multiple -t --user 0 $apks
```

Скрипт для тестирования: выбор транспорта по умолчанию

```
adb shell bmgr transport com.google.android.gms/.backup.BackupTransportService
rm $apks

echo "Done"
```

Скрипт для d2d тестирования

Скрипт для d2d тестирования

```
#!/bin/bash -eu
: "${1?"Usage: $0 package name"}"

adb shell bmgr enable true
adb shell settings put secure backup_enable_d2d_test_mode 1
adb shell bmgr transport com.google.android.gms/.backup.migrate.service.D2dTransport
adb shell bmgr init com.google.android.gms/.backup.migrate.service.D2dTransport
adb shell bmgr backupnow "$1" | grep -F "Package $1 with result: Success" || (echo "Backup failed"; exit 1)
```

Скрипт для d2d тестирования

```
#!/bin/bash -eu
: "${1?"Usage: $0 package name"}"

adb shell bmgr enable true
adb shell settings put secure backup_enable_d2d_test_mode 1
adb shell bmgr transport com.google.android.gms/.backup.migrate.service.D2dTransport
adb shell bmgr init com.google.android.gms/.backup.migrate.service.D2dTransport
adb shell bmgr backupnow "$1" | grep -F "Package $1 with result: Success" || (echo "Backup failed"; exit 1)
```

Скрипт для d2d тестирования

```
#!/bin/bash -eu
: "${1?"Usage: $0 package name"}"

adb shell bmgr enable true
adb shell settings put secure backup_enable_d2d_test_mode 1
adb shell bmgr transport com.google.android.gms/.backup.migrate.service.D2dTransport
adb shell bmgr init com.google.android.gms/.backup.migrate.service.D2dTransport
adb shell bmgr backupnow
```

Проблемы при тестировании: таймаут (!)

```
12-05 18:59:02.033  1910  2251 D BackupManagerService:
   awaiting agent for ApplicationInfo{5c7cde0 com.vkontakte.android}

12-05 18:59:12.117  1910  2251 W BackupManagerService:
   Timeout waiting for agent ApplicationInfo{5c7cde0 com.vkontakte.android}

12-05 18:59:12.117  1910  2251 W BackupManagerService:
   Can't find backup agent for com.your.app.package
```

Проблемы при тестировании: превыше объем данных

I/PFTBT: Transport rejected backup of <PACKAGE>, skipping

--- or ---

I/PFTBT: Transport quota exceeded for package: <PACKAGE>

Проблемы при тестировании: не указаны данные

I Backup : [FullBackupSession] Package com.your.app.package doesn't have any backup data.

I Backup : [D2dTransport] Package com.your.app.package doesn't have any backup data.

Как закрывать под тогглами?

С key-value backup можно обратиться на диск и не бэкапировать

```
class VkBackupAgent : BackupAgentHelper() {

    override fun onBackup(
        oldState: ParcelFileDescriptor?,
        data: BackupDataOutput?,
        newState: ParcelFileDescriptor?
    ) {
        if (VkFeatures.Backup.isEnabled()) {
            super.onBackup(oldState, data, newState)
        }
    }
```

Чтобы отключить тоггл — работаем с разными файлами

```
class AuthRepository {
    companion object {
        const val BASE_AUTH_FILE = "auth.txt"
        const val AUTH_FILE_AB = "auth_to_backup.txt"
    }

    fun save(credentials: UserCredentials) {
        val file1 = File(appContext.filesDir, BASE_AUTH_FILE)
        val file2 = File(appContext.filesDir, AUTH_FILE_AB)

        arrayOf(file1, file2).forEach { fileToWrite ->
            DataOutputStream(FileOutputStream(fileToWrite)).use { stream ->
                stream.writeLong(credentials.userId)
                stream.writeUTF(credentials.authSecret)
            }
        }
    }
}
```

Сохраняем в оба файла

```
class AuthRepository {
    companion object {
        const val BASE_AUTH_FILE = "auth.txt"
        const val AUTH_FILE_AB = "auth_to_backup.txt"
    }

    fun save(credentials: UserCredentials) {
        val file1 = File(appContext.filesDir, BASE_AUTH_FILE)
        val file2 = File(appContext.filesDir, AUTH_FILE_AB)

        arrayOf(file1, file2).forEach { fileToWrite ->
            DataOutputStream(FileOutputStream(fileToWrite)).use { stream ->
                stream.writeLong(credentials.userId)
                stream.writeUTF(credentials.authSecret)
            }
        }
    }
}
```

Читаем в зависимости от тоггла

```
class AuthRepository {
    companion object {
        const val BASE_AUTH_FILE = "auth.txt"
        const val AUTH_FILE_AB = "auth_to_backup.txt"
    }

    fun read(): UserCredentials {
        val fileToRead = if (VkcFeatures.Backup.hasFeatureEnabled()) AUTH_FILE_AB else BASE_AUTH_FILE
        val file = File(appContext.filesDir, fileToRead)

        return DataInputStream(FileInputStream(file)).use { stream ->
            val userId = stream.readLong()
            val authSecret = stream.readUTF()
            UserCredentials(userId, authSecret)
        }
    }
}
```

Резюме

Бэкап — это очень просто

И очень прибыльно

>1 500 000₽

Ежемесячная экономия

Подпишись на меня

Спасибо!