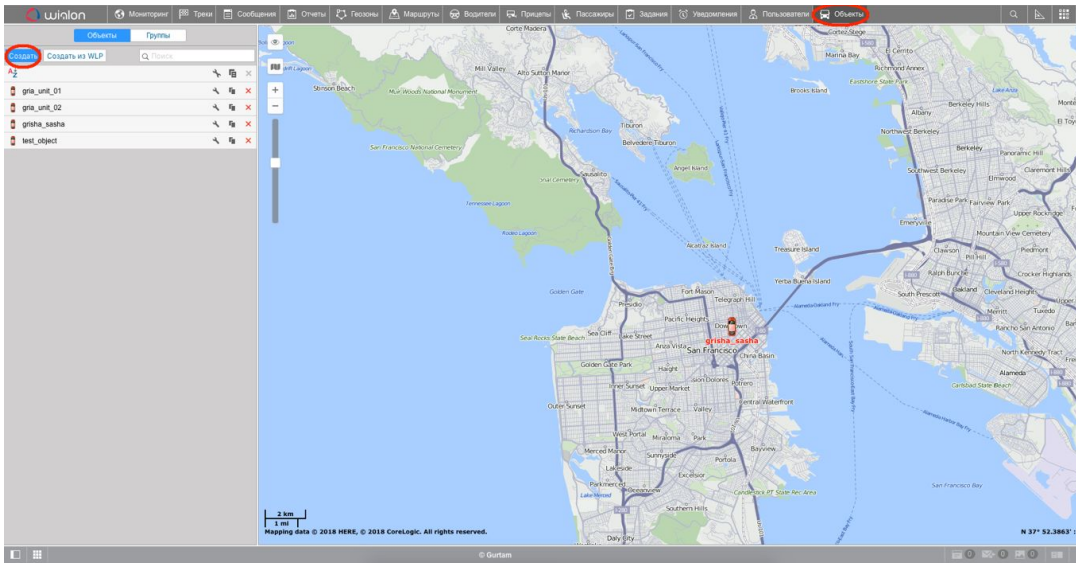
Gurtam выпустили библиотеку WiaTagKit, которая позволяет мобильным устройствам «общаться» с платформой Wialon. Это значит, что теперь любое ваше приложение можно легко превратить в трекер или вообще написать новое.

Почему следует воспользоваться новой библиотекой WiaTagKit?

- **Новый закрытый протокол**. Более защищенный, современный и компактный (по сравнению с GPSTag), а значит он потребляет меньше трафика и его сложнее взломать.
- **Вы получаете готовый компонент** и экономите время и деньги на разработке. Таким образом ваши решения появляются на рынке быстрее, и вам не нужно тратить силы на реализацию протокола связи с Wialon.

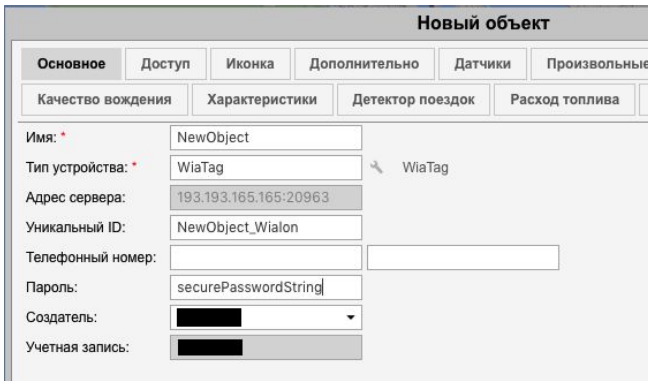
Руководство для Android

1. Создаем объект, который будет отправлять сообщения в Wialon: заходим в учетную запись, открываем вкладку **«Объекты»**, нажимаем кнопку **«Создать»**.



2. Настраиваем объект в появившемся диалоге: задаем **«Имя»**, выбираем **«Тип устройства»** WiaTag, заполняем поля **«Уникальный ID»** и **«Пароль».**

Важно! Любой, кто знает ID и пароль объекта, может отправлять в Wialon сообщения от его имени. Поэтому убедитесь, что пароль знают только доверенные лица.



Для отправки сообщения нам понадобится содержимое полей **«Адрес сервера»**, **«Уникальный ID»** и **«Пароль»**.

Приступаем к написанию кода

Создаем проект нового приложения:

1. Запускаем **Android Studio**.
2. Создаем новый проект в диалоге **«Welcome to Android Studio»** или выбираем File -> New -> New Project в панели меню.
3. Вводим имя приложения, домен компании, путь к проекту.
4. Выбираем форм-факторы приложения. Если вы не до конца уверены, какие форм-факторы вам необходимы, просто выберите **«Phone»** или **«Tablet»**.
5. Выбираем **«Empty Activity»** в диалоге **«Add an activity to Mobile»**.
6. Вводим имя **«Activity»**, название макета и заголовок. Значения по умолчанию можно не менять.

Подключаем библиотеку wiatag-kit

1. Открываем файл **build.gradle** из каталога модуля приложения.

Важно! Проекты в Android Studio содержат высокоуровневый файл **build.gradle** и файлы **build.gradle** каждого модуля. Убедитесь, что редактируете файл для модуля вашего приложения.

2. Добавляем зависимость для последней версии **wiatag-kit**.

```
apply plugin: 'com.android.application'
...

dependencies {
    implementation "com.gurtam:wiatag-kit:0.1.4"
}
```

Важно! Убедитесь, что ваш высокоуровневый build.gradle содержит ссылку на репозиторий jcenter().
3. Сохраняем изменения и нажимаем «**Sync Project with Gradle Files»** в панели инструментов.

Далее, добавляем разрешение в **Android Manifest**. Чтобы осуществлять сетевые операции в вашем приложении, **Android Manifest** должен содержать следующие разрешения.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Отправляем сообщение в Wialon

1. Инициализируем **MessageSender,** используя **Адрес сервера»**, **Уникальный ID»** и **Пароль»** объекта в Wialon;
2. Инициализируем объект Message;
3. Отправляем **Message**, используя соответствующий метод **MessageSender;**

Меняем класс **MainActivity**.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;

import com.gurtam.wiatagkit.Message;
import com.gurtam.wiatagkit.MessageSender;
import com.gurtam.wiatagkit.MessageSenderListener;

import java.util.Date;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        String host = "193.193.165.165";
        int port = 20963;
        String unitId = " NewObject_Wialon";
        String password = "securePasswordString";
        MessageSender.initWithHost(host,port,unitId,password);
        Message message = new Message().time(new Date().getTime());
        MessageSender.sendMessage(message,new MessageSenderListener() {
            @Override
            protected void onSuccess() {
                super.onSuccess();
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(MainActivity.this,"Message
Sent",Toast.LENGTH_LONG).show();
                    }
                });
            }
            @Override
            protected void onFailure(byte errorCode) {
                super.onFailure(errorCode);
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
```

```
Toast.makeText(MainActivity.this,"Failure",Toast.LENGTH_LONG).show();
                    }
                });
            }
        });
    }
}
```

Важно! Значения **host**, **port**, **unitId** и **password** нужно заменить на **Адрес сервера» (IP, порт)**, **Уникальный ID»** и **Пароль»** вашего объекта типа WiaTag.

Запускаем проект

Если вы все сделали правильно, появится всплывающее сообщение

Message Sent

Поздравляем, вы отправили первое пустое сообщение на сервер.

Добавляем содержимое в сообщение WiaTag

В настоящий момент, сообщение может содержать следующую информацию:

1. Время создания сообщения в **миллисекундах**, истекших с 1 января 1970 года.
2. Местоположение. Для отправки местоположения используется класс Location.
3. Данные LBS. Для отправки данных LBS используется класс Lbs.
4. Флаг SOS-сообщения.
5. Изображение.
6. Текстовое сообщение.
7. Уровень заряда батареи.
8. Параметры. Имеют ключи (только текстовые) и значения (текстовые, бинарные и типов **Int**, **Long**, **Float**, **Double**). Нельзя отправить 2 параметра для одного и того же значения ключа.

Для примера создадим сообщение, которое отправляет время, SOS-сигнал, позицию, текстовое сообщение и параметр-число:

```
Message message = new Message()
.time(new Date().getTime())
.Sos()
.location(new Location(53.90582,27.45697,290,2F,(short)15,(byte)8));
.text("This is my text message!")
.addParam("int value", 3);
```

Для получения результата отправки нужно передать в функцию sendMessage класса MessageSender *реализацию интерфейса* MessageSenderListener

В случае успеха вызовется метод onSucces, иначе – OnFailure с кодом ошибки:

```
FAILED_TO_CONNECT = 3;
FAILED_TO_SEND = 4;
INVALID_UNIQUE_ID = 5;
INCORRECT_PASSWORD = 6;
INCORRECT_MESSAGE = 8;
```

Пример проекта для отправки одного или нескольких сообщений доступен на GitHub.

Это один из примеров работы с **WiaTagKit.** По любым вопросам использования новой библиотеки обращайтесь на development@gurtam.com.