VTAP

Application Note – Using the VTAP Android USB Serial Comms SDK

Firmware from v2.2.7.0

VTAP50 and VTAP100

Revised February 2025 v2.03

DOT ORIGIN

If you need help to set up or use your VTAP reader, beyond what is contained in this Application Note, then please contact our support team.

Email: **vtap-support@dotorigin.com**

Download the latest documentation and firmware from **https://vtapnfc.com**

Telephone UK and Europe: +44 (0) 1428 685861

Telephone North America and Latin America: +1 (562) 262-9642

If you have any feedback on setting up or using your VTAP reader or this documentation, then please contact our support team. The product is constantly being reviewed and improved and we value feedback about your experience.

Dot Origin Ltd
Unit 7, Coopers Place Business Park, Combe Lane, Wormley
Godalming GU8 5SZ United Kingdom
+44 (0) 1428 685861

Contents

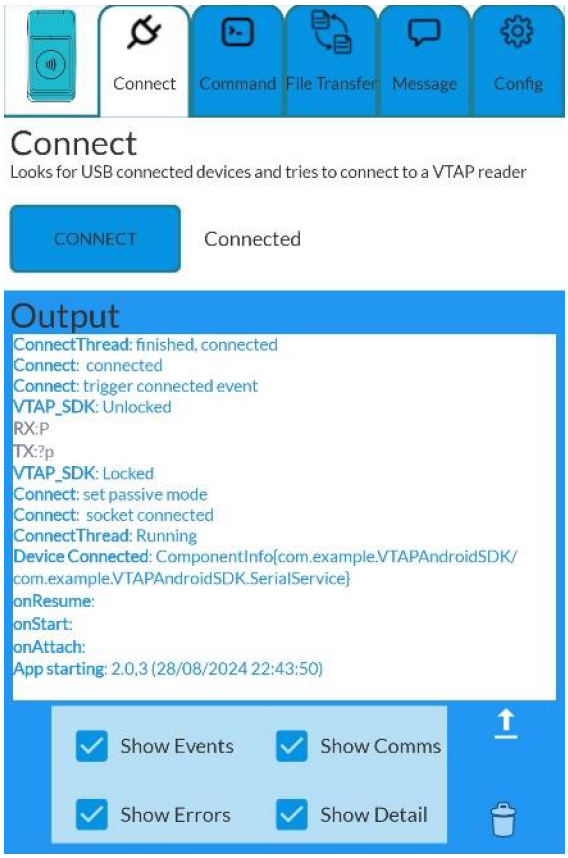
1 Using the VTAP Android USB serial communications SDK

1.1 Overview

This package is designed to help you to write Android applications to control a VTAP reader over a USB serial interface. It includes:

- Java classes that provide a lightweight interface to the VTAP reader's Command Interface over the USB virtual COM port.
- An example Android app that demonstrates the use of these classes, both as
  - a pre-built apk file, and
  - a source code project for use with Android Studio.

The example application provides easy access to the full functionality of the VTAP Command Interface and the ability to send and receive files to/from the VTAP reader file system using the ZModem protocol.



The VTAP Command Interface allows for direct control of the VTAP reader as well as allowing updates to configuration, private keys and firmware to be performed over the USB serial, RS-232, RS-485 or TTL serial interfaces (depending on the VTAP reader product or configuration in use).

1.2 Functionality and Implementation

The VTAP reader Command Interface is a simple ASCII command line protocol designed to be used interactively via a terminal emulator or driven programmatically. It consists of commands to:

- get VTAP reader serial number and device information;
- poll for pass or tag data and type information;
- control the LEDs and buzzer;
- send messages to any of the VTAP reader peripheral interfaces;
- and to set or get configuration values from the VTAP reader `config.txt` file.

Full details of these commands can be found in the VTAP Commands Reference Guide.

A high-level API for easy use of these different groups of commands can be found in the vtap100 Java class, described overleaf:

vtap100.java

This class provides a lightweight wrapper to make it easier to use the VTAP reader command interface programmatically. Despite being named vtap100, it is used for all VTAP reader models. It contains the following public methods:

```
public void connect(SerialSocket port)
public void disconnect()
```

Initialise and release the serial interface to the VTAP reader. The connect() method should be called before any of the other vtap100 class methods.

```
public boolean isConnected()
```

Returns true if the VTAP reader is connected via USB.

```
public void setPassiveMode()
```

This sends the `?p` command to enter VTAP reader passive mode on the USB COM port. This is useful when using the command interface in order to prevent pass or tag data from being sent actively over the serial interface. When in passive mode, your application will need to poll the VTAP Command Interface (using `?r`) to read pass or tag data.

```
public int SendCommand(final String command, StringBuffer response)
```

Send a command to the VTAP reader and receives it response (where applicable). This allows sending any of the commands that start with `?` on the command line terminal. The `?` starting the command and newline ending it should not be included in the command. Use this to enter passive mode, poll for pass payload or tag data, control the LEDs or buzzer, or reboot the VTAP reader.

For example, use a command of `?b` to get the VTAP reader boot information: the hardware/firmware and status information that is also found in the VTAP reader`boot.txt` file.

```
public int SetConfig(@NonNull final String name, @NonNull final String value)
public int SetConfig(@NonNull final String[] name, @NonNull final String[] value)
```

Set a config option `name=value` pair in the VTAP reader`config.txt` file. The new setting will take effect immediately in most cases (only a couple of options will require restarting as noted in the VTAP Commands Reference Guide).

The second form of this method allows multiple `name=value` options to be set with a single call.

```
public int GetConfig(final String name, @NonNull StringBuffer value)
```

Reads the value of a named option from the `config.txt` file. If name is empty then all config names and values will be returned.

public int SendMessage(final char dest, final char type, @NonNull String message)

Sends message data to one of the VTAP reader peripherals. This can be used to directly deliver the given message data over the VTAP keyboard emulation, Wiegand interface (where applicable) or serial (USB, RS-232, RS-485) interfaces in the same way as pass or tag payloads can be delivered.

public int SendFile(@NonNull final String sourcePath,@NonNull final String filename)

Transfers a file to the VTAP reader file system using the ZModem file transfer protocol. The sourcePath specifies the location of the file in the Android file system. The filename will be used to find the source file and as the name of the destination file on the VTAP reader filesystem. You can use this to update the entire `config.txt` file, update `privateX.pem` private key files or update the VTAP reader firmware (`vtapware.dat`).

public void setListener(vtap100Listener listener)

Set the listener to receive event messages created by the vtap100 class. See the vtap100Listener class description overleaf.

vtap100.java - vtap100Listener

This is a sub class of vtap100 that allows event messages to be sent to a listener. This defines the following methods that may be overridden by the listener.

void onLock()
void onUnlock()

When the vtap100 class is sending messages to the VTAP reader, it will lock other transactions from happening. These events are sent on the lock and unlock events respectively.

void onResult(int result)
void onResult(String result)

When a vtap100 interface command returns an integer or string result, the appropriate event handler will be called with the result.

void onLog(String header, String message, int level)

Sends a log message that the listener can optionally write to its own log facility. Sends a header (usually the function or operation), the message (a description) and a log level.

Zmodem.java

File transfers to and from the VTAP reader's file system are implemented over all VTAP serial communications ports using the ZModem protocol. The Zmodem class provides an implementation of this protocol which is optimised for use with VTAP readers. The ZModem protocol used is a standard for simple, recoverable, low overhead file transfers that can also be found in many terminal emulator software packages.

```
public static Boolean Zmodem_send(vtap100 dev, String path, char[ ] file_name)
public static Boolean Zmodem_receive(vtap100 dev, String path)
```

The public methods defined in the Zmodem class, Zmodem_send() and Zmodem_receive(), can be called directly from your application (passing them a connected vtap100 object) or can be used through the vtap100.SendFile() method described **above**.
The Zmodem_receive() method is not currently used by the vtap100 class.

FIL.java

This class is used by the vtap100 and Zmodem classes to read, write and manage files.

CustomProber.java

Used to find the VTAP reader's vendor ID and product ID.

SerialSocket.java

Used to write serial comms

SerialListener.java

Used to pass through the serial events

VTAP_CharBuffer.java

Char buffer for serial comms

1.3 USB connection and intent filters

When importing code into your app you may need to ensure you have the USB filters and intents set up correctly to detect the VTAP reader when it is plugged in. They will need the vendor ID and product ID to identify the VTAP reader. The following codes or settings may be required in your app.

in AndroidManifest.xml

To launch your app when a VTAP reader is detected

```
<intent-filter>
<action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
</intent-filter>
<meta-data
android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
android:resource="@xml/usb_device_filter" />
```

in usb_device_filter.xml:

In the example, this file is in `app/src/main/res/xml` and can contain multiple USB filters. The two used most commonly for VTAP readers are:

```
<resources>
<!-- VTAP -->
<usb-device vendor-id="1003" product-id="9249" /><!-- 0x03EB/0x2421: VTAP100 v4a -->
<usb-device vendor-id="12346" product-id="16391" /><!-- 0x303A/0x4007: VTAP50/100 Gen2 -->
</resources>
```

The vendor and product IDs set in the filters must match those in the implementation code. `CustomProber.java` also contains these values:

in CustomProber.java:

```
customTable.addProduct(0x03EB, 0x2421, CdcAcmSerialDriver.class); // VTAP100
customTable.addProduct(0x303A, 0x4007, CdcAcmSerialDriver.class); // VTAP100 Gen2 / VTAP50
```

Serial over USB Library

The example code uses a 'serial over usb' freeware library from **https://github.com/mik3y/usb-serial-for-android** which needs to be called

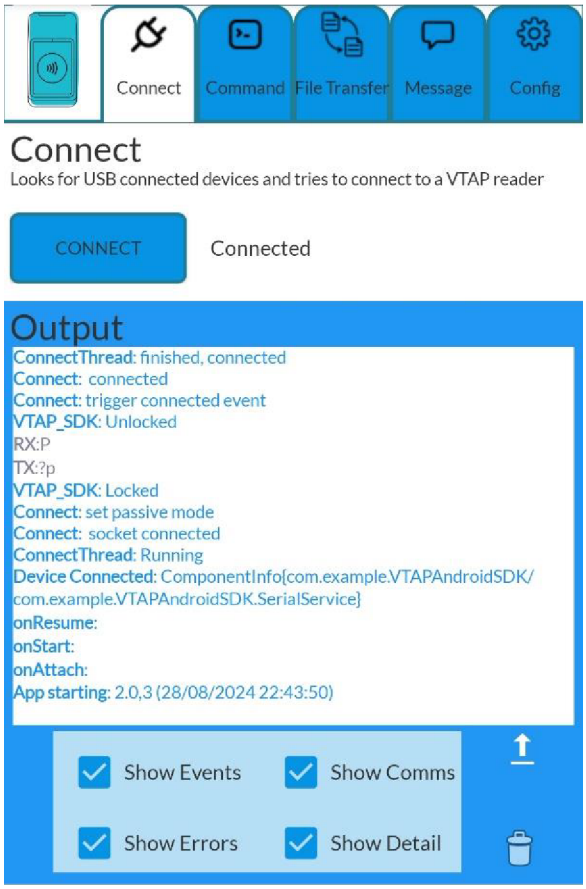
in AndroidManifest.xml:

```
dependencies {
 implementation 'com.github.mik3y:usb-serial-for-android:3.4.0' // maven jetpack
 ...
```

1.4 Example VTAP reader interface app

An example Android app is included, which demonstrates the use of the VTAP Command Interface. The full source code of this app is provided as a project for use with Android Studio.

To get started using a VTAP reader with Android, this app is also provided as a pre-built signed APK file, which can be sideloaded to an Android device. This file can be found in the 'built apk' folder of the VTAP Android SDK package.



When using a VTAP reader with an Android device, the VTAP reader's USB keyboard emulation may cause the Android device to not show its on-screen keyboard when an input field is selected. This is because Android detects that an external keyboard device is present and assumes that this should be used instead of the on-screen keyboard. To avoid this issue:

- Use an Android device setting to allow the on-screen keyboard to be used, even when it appears that there is an external keyboard. This setting can usually be found on an Android device under
  `Settings->Additional Settings->Language` then
  `input->Keyboard, mouse and track pad`
  (The exact location for this setting varies depending on the device and Android version). Ensure that the `Use on-screen keyboard` setting is checked.

- Alternatively, you could make use of the VTAP setting `KBEnable` (from VTAP firmware v2.2.6.0) to disable the USB keyboard device of the VTAP reader. This is a setting you can include in the VTAP reader's `config.txt` file.
  `KBEnable=0` ; will disable the USB keyboard device.

The example app source code includes the following files, as well as the VTAP Android SDK Java files described above:

- MainActivity.java

  Used to start the example application, and called when the file explorer is used or when a new intent is set.

- DeviceFragment.java

  Launches the TerminalFragment.

- TerminalFragment.java

  Sets up the user interface, serial service and listeners for the app. Uses the vtap100 class to control the connected VTAP reader.

- MyLog.java

  A thread-safe logging class for logging to LogCat, a log file and to a TextField within the app.

- SerialService.java

  Used to manage the serial connection. Creates notifications and queues serial data while activity is not in the foreground. The app uses the listener chain:
  `SerialSocket → SerialService → TerminalFragment`

- Constants.java

  Constant definitions used by the app

2 About Application Notes

Application Notes address topics of interest to small groups of customers, or topics around the use of a VTAP reader with third-party systems.

The main documents available to support your use of the VTAP50 and VTAP100 are the Installation Guide for your VTAP reader model and the VTAP Configuration Guide. You will find the latest versions of these, and other useful information at **https://vtapnfc.com**.

If you need further help do contact us by email to **vtap-support@dotorigin.com**, or by phone +44 (0) 1428 685861 from Europe and Asia, or +1 (562) 262-9642 from Northern and Latin America.