RevCode Client - > Establishing a secure connection to server over SSL (443)

All packets sent from RevCode Android Client are mainly formatted using JSON.

RevCode Android Client will try to establish a connection and then continuously retrieve tasks from: recv\_android.php, based on a returned interval. *See “Connection” for more information on this.*

Note: Diffie-Hellman handshake authentication is required in order to establish a valid connection. If client isn’t already authenticated, then server is expected to reply with “-1”. See *Handshake* below.

**Output packet structure for tasks:**

All sent packets by the client are sent using HTTP Post.

All packets include the following variables (see exceptions on footnote 3):

* “t\_id” – Task ID
* “key” – *{CONFIG\_KEY} refers to a configuration key generated by a third party (the builder) and is extracted from the client SETTINGS\_DATA during runtime.*
* “mode” – Task mode
* “uid” – *{SYS\_UID) Unique identifiable hardware ID obtained by the client during runtime.*
* “enc” – Encryption; 1=Encryption used, 0=Encryption not used
* “cmp” – Compression; 1=Compression used, 0=Compression not used

1. *Note: If only encryption was used, then Base64 decoding must be applied before decrypting values.*
2. ***Decoding:*** *If a combination of encryption and compression has been used, then pseudo decoding would be: raw\_value =* 
   * 1. *decompress(decrypt(Base64Decode(encoded\_value)))*
3. ***Note:*** *Handshake (mode=”keyauth” and Connection (mode=connect”) may not include all of the above listed variables.*

**Important:** A task should be assumed to have failed whenever the **“v0”** POST variable equals “0”.

**Handshake:**

Client will attempt a handshake by posting the following variables without any encryption or compression:

key={CONFIG\_KEY}

uid=”{SYS\_UID)

mode=”keyauth”

data={BASE64(PUBLIC\_KEY|UID|dP|dG|dPub)}

**Important:** Should the client ever receive “-1” during runtime, then a new authentication (handshake) and connection should be performed.

**Encryption information:**

Encryption algorithm used is AES-CBC without any IV.

Key size is 128 bits, as well as 128 bits block size.

Note: Manual padding of nulls must be applied to plain data should its length not be divisible by 16 (block size).

**Compression information:**

Compression algorithm used is GZIP/Deflate.

**Connection:**

Client will proceed to establish a connection instantly upon a successful handshake by posting the following variables:

key={CONFIG\_KEY}

uid=”{SYS\_UID}

cmp={1 or 0, depending on client settings}

cmp={1 or 0, depending on client settings}

mode=”connect”

v0=LOCAL IP

v1=PHONE MODEL

v2=CONNECTION TYPE

v3=RADIO TYPE

v4=BATTERY LEVEL

v5=ANDROID VERSION

v6=NETWORK OPERATOR NAME

v7=IS ROOT/ADMIN

v8=IDLE TIME

* The interval for which the client will continuously query tasks is based on the response of the POST above. A numeric value, seconds, is expected.

**Input packet structure for tasks:**

All received tasks by the client need to be identified by their header (mode) followed by necessary parameters.

The Client will attempt to split parameters by the delimiter “|”.

Incoming packet structure (string): BASE64Encode(encrypt({MODE}|{TASK\_STATUS}|{TASK\_ID}|{Params[split by “|” if several parameters are required]})).

For several tasks, client will split the available tasks by a new line as delimiter and process each task consecutively.

**Note:** This scheme does not apply to responses from Handshake (mode=”keyauth” and Connection (mode=connect”), but solely to when client queries tasks (mode=”get\_tasks”).

**Performing tasks:**

* **Call logs:** Listing of all call logs
  + **Input:** 
    - Dataheader=**”**CALLS\_LOGS\_GET**”** (string)
    - No parameters required
  + Output:
    - Data header: mode=”calls\_logs\_get”
    - Data type: POST
    - Variables: **“v0”** – Contains JSON array -> Columns:
      * “contactName” - {Contact name (string)}
      * “phoneNumber” – {Phone number (string)}
      * “callType” – {Call type (string)}
      * “callDate” – {Call date (string)}
      * “callDuration” – {Call duration (string)}
* **SMS logs:** Listing of all SMS
  + **Input:** 
    - Dataheader=**”**SMS\_GET**”** (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”sms\_get”
    - Data type: POST
    - Variables: **“v0”** – Contains JSON array -> Columns:
      * “phonenumber” – {Phone number (string)}
      * “mailbox” – {Mailbox (string)}
      * “date” – {Date (string)}
      * “status” – {Read/Unread (string)}
      * “message” – {Message (string)}
* **Location:** Obtain current GPS position
  + **Input:** 
    - Dataheader=**“**LOCATION\_GET**”** (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”location\_get”
    - Data type: POST
    - Variables: POST with a range of variables (v0 to v5)
      * “v0” – {Latitude - Get the latitude, in degrees (double)}
      * “v1” – {Longitude - Get the longitude, in degrees (double)}
      * “v2” – {Time - Return the UTC time of this fix, in milliseconds since January 1, 1970 (long)}
      * “v3” – {Accuracy - Get the estimated horizontal accuracy of this location, radial, in meters (float)}
      * “v4” – {Speed - Get the speed if it is available, in meters/second over ground (float)}
      * “v5” – {Bearing - Get the bearing, in degrees (float)}
    - Notes: Client will wait until Android system GPS has been enabled in order to obtain a GPS position.
* **URL opener**: Open a URL in default browser
  + **Input:** 
    - Dataheader=“URL\_OPEN” (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”url\_open”
    - Data type: POST
    - Variable: POST with a single variable (v0)
      * “v0” – {Task status – 1=Success, 0=Failed (long)}
* **Browser bookmarks**: Obtain browser bookmarks
  + **Input:** 
    - Dataheader=“BROWSER\_BOOKMARKS\_GET” (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”browser\_bookmarks”
    - Data type: POST
    - Variable: **“v0”** – Contains JSON array -> Columns:
      * “title” – Title of web page
      * “url” – URL of web page
* **Device information**: Obtain device information
  + **Input:** 
    - Dataheader=“DEVICE\_INFO” (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”device\_info”
    - Data type: POST
    - Variable: **“v0”** - Contains JSON array -> Columns:
      * “imei” – {IMEI number (string)}
      * ”networkOperator” – {Returns the numeric name (MCC+MNC) of current registered operator. (string)}
      * ”networkCountry” – {Returns the ISO country code equivalent of the current registered operator's MCC (Mobile Country Code) (string)}
      * ”networkOperatorName” – { Returns the alphabetic name of current registered operator (string)}
      * ”batteryLevel” – {Battery level including percent-symbol (string)}
      * ”phoneNumber” – { Returns the phone number string for line 1, for example, the MSISDN for a GSM phone (string)}
      * ”radioType” – {Radio type ranging from GPRS to 4G (string)}
      * ”conType” – {Returns a human-readable name describing the type of the network, for example "WIFI" or "MOBILE" (string)}
      * ”deviceName” – {Returns manufacture followed by device model (string)}
      * ”localIP” – {Returns present local IP (string)}
      * ”wifiSSID” – {WiFi SSID – Only if connected (string)}
      * ”wifiIP” – {WiFi IP – Only if connected (string)}
      * ”appVersion” – {Application version (string)}
      * ”androidSDKVersion” – {Android SDK Version (string)}
      * ”androidReleaseVersion” – {Android release version (string)}
      * ”deviceModel” – {Device model (string)}
      * ”deviceBrand” – {Device brand (string)}
      * ”deviceProduct” – {Device product (string)}
      * ”deviceBuildID” – {Device Build ID (string)}
      * ”deviceHost” – {Device host (string)}
      * ”deviceSerial” – {Device serial number (string)}
      * ”deviceFingerprint” – {Device fingerprint data – Only if present (string)}
      * ”deviceHardware” – {Device hardware data (string)}
      * ”deviceType” – {Device type (string)}
      * ”deviceUser” – {Device user (string)}
      * ”deviceBootloader” – {Device bootloader data (string)}
      * ”hasRootAccess” – {Returns true/false, depending on whether or not device has Root access (string)}
      * ”hddSizeTotal” – {Total size of internal storage in bytes (double)}
      * ”hddSizeFree” – {Free space of internal storage in bytes (double)}
      * ”SDMounted” – {Returns true/false, depending on whether or not device has a mounted SD card (string)}
      * ”SDSizeFree” – {Free space of SD Card stored in bytes (double)}
      * ”SDSizeTotal” – {Total space of SD card in bytes (double)}
      * ”RAMTotal” – {Total space of RAM in bytes (double)}
      * ”RAMFree” – {Free space of RAM in bytes (double)}
      * ”CPUInfo” – {Information about CPU (string)}
* **Contacts information**: Obtain stored contacts and associated data
  + **Input:** 
    - Data header=“CONTACTS\_GET” (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”contacts\_get”
    - Data type: POST
    - Variable: **“v0”** - Contains JSON array -> Columns:
      * “name” – {Name (string)}
      * “phonenumber[1..n]” – {Phone number(s) (string)}
      * “email[1..n]” – {Email(s) (string)}
* **Sound recording**: Records from microphone for n given seconds and sends back binary of recording data
  + **Input:** 
    - Data header=“SOUND\_RECORD” (string)
    - Parameters:
      * DURATION\_SECONDS (long)
  + **Output:** 
    - Data header: mode=”sound\_record”
    - Data type: POST
    - Variables: Single variable “data” containing binary of a 3GPP media file
      * “data” – {Base64 encoded binary data. Note: See ***Output packet structure*, *section 2***, if decoding is needed - (string)}
* **Camera snapshot**: Captures a snapshot from front or back camera
  + **Input:** 
    - Data header=“CAM\_SHOT” (string)
    - Parameters:
      * CAMERA\_INDEX – {Index starting from the back-facing camera on the device from 0 to N of camera devices - (long)}
  + **Output:** 
    - Data header: mode=”cam\_shot”
    - Data type: POST
    - Variables: Single variable “data” containing binary of a JPG image.
      * “data” – {Base64 encoded binary data. Note: See ***Output packet structure*, *section 2***, if decoding is needed - (string)}
* **Gallery**: Obtains resized versions (480P) of all stored images in Gallery/Photos in chunks of 25 images per packet.
  + **Input:** 
    - Data header=“GALLERY\_GET” (string)
    - Parameters:
      * ARR\_MD5 [OPTIONAL] – {Array of MD5 checksums, split by vertical tab, “\t”. This parameter is an optional caching mechanism. Images having a matched checksum will be excluded – (string[])}
  + **Output:** 
    - Data header: mode=”gallery\_get”
    - Data type: POST
    - Variable: **“v0”** - Containing JSON array -> Columns:
      * “name” – {File name - (string)}
      * “f\_size” – {Full image size - (long)}
      * “t\_size” – {Thumbnail size - (long)}
      * “exif\_model” – {Exif model - (string)}
      * “exif\_make” – {Exit make - (string)}
      * “date” – {Date of creation - (string)}
      * “md5” – {MD5 checksum, used - (string)}
      * “data” – {Base64 encoded thumbnail binary data - (string)}
    - Variable: **“v1”** – {Status (“2” = buffering, “1” = final packet) – (long)}
    - Variable: **“v2”** – {Current chunk position – (long)}
    - Variable: **“v3”** – {Total number of images – (long)}
* **Installed applications**: Obtain information about installed apps
  + **Input:** 
    - Data header=“INSTALLED\_APPS\_GET” (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”installed\_apps\_get”
    - Data type: POST
    - Variable: **“v0”** - Contains JSON array -> Columns:
      * “appName” – {Name (string)}
      * “appPackageName” – {Package name (string)}
      * “appPermissions” – {Permissions (string)}
      * “appProcName” – {Process name (string)}
      * “appVersion” – {Version (string)}
      * “appInfo” – {Additional information (string)}
      * “appIconData” – {Base64 encoded icon binary data - (string)}
* **Wi-Fi networks**: Listing of all available Wi-Fi networks
  + **Input:** 
    - Data header=“ WIFI\_GET” (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”wifi\_get”
    - Data type: POST
    - Variable: **“v0”** - Contains JSON array -> Columns:
      * “ssid” – {SSID (string)}
      * “bssid” – {MAC address of the wireless access point (string)}
      * “frequency” – {Frequency in GHZ (string)}
      * “rssi” – {Received signal strength indicator in dBm (string)}
      * “capabilities” – {WiFi capabilities (string)}
* **Clipboard**: Obtain clipboard data
  + **Input:** 
    - Data header=“CLIPBOARD\_GET” (string)
    - No parameters required
  + **Output:** 
    - Data header: mode=”clipboard\_get”
    - Data type: POST
    - Variable: **“v0”** - Contains JSON array -> Columns:
      * “html” – {HTML formatted data (string)}
      * “text” – {Raw data (string)}
* **Files**: Obtains a list of files from a given directory
  + **Input:** 
    - Data header=“FILES\_GET” (string)
    - Parameters:
      * Path – {Base64 encoded path. Use Base64(“0”) in order to query root (string)}
  + **Output:** 
    - Data header: mode=”files\_get”
    - Data type: POST
    - Variable: **“v0”** - Containing JSON array -> Columns:
      * “name” – {File name - (string)}
      * “type” – {Item type: “d” or “f” - (string)}
      * “size” – {Item size (“-1” for dir - (long)}
      * “hidden” – {Item visibility, 1/0 - (long)}
      * “l\_mod” – {Last modified - (string)}