V.BTTN DEVELOPER QUICK START GUIDE

VERSION: 12

DATE: 1/19/2018

CONTACT: abonilla@hoyosintegrity.com

1. V.BTTN - Feature Summary:

   V.BTTN is a wearable Bluetooth button with limitless applications. By pairing it with a Bluetooth Smart Ready (Bluetooth 4.0) mobile phone or a PC, it can be programmed to do anything – from ordering dinner, to sending a text message, to remotely activating the camera shutter.

   Additional information:

   Wireless (Up to 75' Indoors, up to 300' Outside)

   Waterproof (Meets IP 67 (depth of 1 meter for 30 minutes)

   Up to One year battery life – no need to charge

   Accelerometer for gesture control

   Audible Tones and LED visible feedback

   Range of accessories including money clip, pendant, and wristband

   Tiny – size of a poker chip

2. V.BTTN – Available Bluetooth features.

   The V.BTTN can be programmed to:
   a. Detect a short press and release. When this feature is enabled, upon pressing the button, the app will be notified. On a button release, the app will be notified again with a different notification value.
   b. Detect a long press-release. Example, the V.BTTN can notify your application if the user pressed-and-released the button for 2-9 seconds. With this feature, only one indication is sent to the app, only if the user press and released the button between 2-9 seconds (Press-and-release 10-23 seconds places the V.BTTN in advertising mode)
   c. To notify the app when a fall is detected. If this feature is enabled, and the V.BTTN is dropped from about 4 feet, a notification will be sent to the application.

d. To disable all sounds and LED functions on the V.BTTN (Stealth Mode)
e. Disable just sound or disable just LED functions.
f. Enable the accelerometer, and report x, y, z accelerations to the application (note that enabling the accelerometer greatly reduces battery life).
g. To buzz and turn on LED using the Immediate Alert Service (Find Me feature)
h. Report battery level to the application.

See the Bluetooth 4.0 Service descriptions in this document for technical details.

3. Bluetooth BLE background

Bluetooth low energy (BLE) technology was introduced in 2011, through the Bluetooth Specification v4.0 (Bluetooth v4.0). With its extremely low power consumption, unique characteristics, and new features, Bluetooth low energy technology enables new applications that were not practical with Classic Bluetooth technology. Coin cell battery-operated sensors and actuators in medical, industrial, consumer, and fitness applications (also known as “Smart”) can now smoothly connect to Bluetooth low energy technology–enabled smart phones, tablets, or gateways (also known as “Smart Ready”). Bluetooth low energy technology is ideal for applications requiring periodic transfer of small amounts of data...Bluetooth low energy technology is easy to set up, robust, and reliable in tough environments.

The behavior of a Bluetooth connection—whether Classic or low energy—is determined by the Bluetooth profiles a device has implemented. Devices can only communicate if they both have the same Bluetooth profile implemented.

Application developer must implement the corresponding profiles on the mobile app side. These profiles must match the profiles on the BLE device (i.e. V.BTTN).

There are some important differences between what profiles are available for Classic Bluetooth technology and for Bluetooth low energy technology.

A good example of the differences is seen in serial port emulation. Classic Bluetooth technology provides the serial port profile (SPP) for emulation of serial data connections. Bluetooth low energy technology provides no such support in the standard Specification v4.0; Many other profiles are not offered for Bluetooth low energy technology because of the differences in the connection models. The Classic Bluetooth scenarios that are not part of Bluetooth low energy technology include headset (HSP), object exchange (OBEX), audio distribution (A2DP), video distribution (VDP), and file transfer (FTP).

Home automation is a good example where devices can be controlled using LE technology; Health and Fitness is another where wearable sensors can monitor things like heart rate. They might even be coupled with location awareness to allow correlations to be drawn between effort and heart rate. This list is endless and is probably only limited by imagination.

From a topology perspective Bluetooth low energy (BT 4.0 LE) devices can take on a number of roles:

I. Scanner
II. Advertiser
III. Master
IV. Slave

Devices such as phones or tablets would generally (but not exclusively) adopt the role of “Scanner” and would “discover” other devices that have adopted the “Advertiser” role by a process called discovery which can be active (“are there any devices out there?”) or passive (“I’ll listen whilst devices advertise their presence”). Devices that adopt the “Advertiser” role are generally (but not exclusively) smaller footprint devices such as heart rate monitors or temperature sensors or the V.BTTN device.

Once devices have discovered one another one will act as an “Initiator” (typically the phone or tablet type device) and attempt to connect to one of the devices that it has discovered. If successful it will adopt the role of “Master” and the other will adopt the role of “Slave”. “Master” devices will initiate commands and requests to “Slave” devices which will respond.

4. After Connection:

   Once a device has been discovered the next task is to figure out what services does the device offer. So, what’s a service? Well, a service consists of: a collection of characteristics.

   It is recommended that the developers go through the information from Bluetooth SIG related to Low Energy: http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx

5. Development platforms:

   Windows 8 – Any PC that includes Bluetooth 4.0 LE (Bluetooth Smart Ready). Developers can use the standard Bluetooth Profile Driver functions and the new Bluetooth Low Energy functions, introduced in Windows 8, to create BLE client applications. Windows 8 BLE reference: http://msdn.microsoft.com/en-us/library/windows/hardware/jj159880(v=vs.85).aspx

   Android - Android phones with Bluetooth 4.0 hardware and Android version 4.3 or later, i.e. Samsung Galaxy S4, S5, HTC One (M8), Nexus 4 & 5, Moto X. The following resources will help when developing for Android: Android BLE Developer Reference: http://developer.android.com/guide/topics/connectivity/bluetooth-le.html

   Android has a very good sample BLE app, which it is recommended to start development with. Look for the sample project called “BluetoothLeGatt” in the Android SDK.

   iOS and Mac OS - iPhone 4S and newer

Mac OS X support: the 2011 MacBook Air or newer, the 2012 MacBook Pro or newer, the 2012 iMac or newer, the 2011 Mac Mini or newer and the 2013 Mac Pro.

iOS BLE Developer Reference:
https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html

6. List of Profiles (SIG Services) that V.BTTN implements:

   V.BTTN implements the following Bluetooth SIG GATT based services. Your application has to implement the matching services (profiles) on the phone/tablet side. Note that not all the services have to be implemented; Only the ones that contain the interested functionality or information. For detail information on Bluetooth SIG adopted specification, please go to:

   https://www.bluetooth.com/specifications/gatt

   a. Device Information Service (UUID: 0x0000180A-0000-1000-8000-00805F9B34FB)

      The Device Information Service exposes information about V.BTTN like Model Number, Firmware/Software version, Hardware version, etc. Using the official SIG link above, you may get UUID details for this service, and the UUID for the characteristic within this service.

   b. Immediate Alert Service (UUID: 0x00001802-0000-1000-8000-00805F9B34FB)

      The Immediate Alert Service cause an alert in the V.BTTN when it is written with a value other than "No Alert." The following values can be written to the Immediate Alert Service's characteristic UUID (0x00002A06-0000-1000-8000-00805F9B34FB)

      The V.BTTN alert behavior is as follow:

      00 - No Alert.

      01 - Chirp every 2 seconds for 20 seconds.

      02 - Chirp + blink red LED every 2 seconds for 20 seconds

   c. Battery Service (UUID: 0x0000180F-0000-1000-8000-00805F9B34FB)

      The Battery Service exposes the Battery Level of the coin cell in V.BTTN. Battery level are reported as a percentage, e.g. 5C is 92%. The battery service's characteristic UUID (0x00002A19-0000-1000-8000-00805F9B34FB) can be used to subscribe for battery update notification (i.e. whenever battery level changes) or immediately read the battery level. Please refer to the official Bluetooth SIG link above for additional details on the battery service.

7. VSN GATT Service (UUID: 0xFFFFFFF0-00F7-4000-B000-000000000000)

   VSN GATT Simple Service implements characteristics that allow applications to interact with V.BTTN's specific functions like short/long button press detection, fall detection, silent/stealth mode, etc. This is a custom VSN service (profile) so the UUID details for this service will not be found on the official Bluetooth SIG site. We provide you a table below with the VSN Service UUID and the characteristic UUIDs within the service.

UUID	Mnemoic	Value(default)	Permission	Notes
0x2800	<<Primary Service>>	0xFFFFFFF0-00F7-4000-B000-000000000000 (VSN SERVICE UUID)	READ	Start of VSN Service
0x2803	<<Characteristic>>	0A (properties: read/write) F1 FF (UUID: 0xFFFFFFF1-00F7-4000-B000-000000000000)	READ	Stealth Mode Configuration Characteristic Declaration
0xFFFFFFF1…	<<Stealth Mode Configuration>>	0 (1 byte)	READ | WRITE	Bit Mask field: 00 : Normal Mode 01 : Stealth Mode (Sound) 02 : Stealth Mode (LED) (e.g. 03 is Stealth Sound+LED)
0x2901	<<Characteristic User Description>>	"Stealth Config" (17 bytes)	READ	
0x2803	<<Characteristic>>	0A (properties: read/write) F2 FF (UUID: 0xFFFFFFF2-00F7-4000-B000-000000000000)	READ	Detection Configuration Characteristic Declaration
0xFFFFFFF2…	<<Detection Mode Configuration>>	0 (1 byte)	READ | WRITE	Bit mask to enable Alerts 0x01 - Enable button press detection 0x02 - Enable button alert detection 0x04 - Enable Fall alert detection 0x08 - Enable High-G alert detection 0x10 - Enable raw accelerometer reporting
0x2901	<<Characteristic User Description>>	"Detection Config" (17 bytes)	READ	
0x2803	<<Characteristic>>	08 (properties: write only) F3 FF (UUID: 0xFFFFFFF3-00F7-4000-B000-000000000000)	READ	Urgent Alert Characteristic Declaration
0xFFFFFFF3…	<<Urgent Alert>>	0 (1 byte)	WRITE	0 - No Alert 1 - Urgent Alert
0x2901	<<Characteristic User Description>>	"Urgent Alert" (17 bytes)	READ	
0x2803	<<Characteristic>>	10 (properties: notify) F4 FF (UUID: 0xFFFFFFF4-00F7-4000-B000-000000000000)	READ	Detection Notification Characteristics Declaration
0xFFFFFFF4…	<<Detection Notification>>	0 (1 byte)	(none)	Notifications 00 - button release detected 01 - button press detected 03 - Button press-release between 2-10 secs 04 - Fall event detected 05 - High-G event detected
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	0 = Disable Notification 1 = Enable Notification
0x2901	<<Characteristic User Description>>	"Detection Notify" (17 bytes)	READ	
0x2803	<<Characteristic>>	08 (properties: write only) F5 FF (UUID: 0xFFFFFFF5-00F7-4000-B000-000000000000)	READ	Verification Characteristic Declaration
0xFFFFFFF5…	<<Verification>>	00:00:00:00:00 (5 bytes)	WRITE	Verification key: 80:BE:F5:AC:FF
0x2901	<<Characteristic User Description>>	"" (17 bytes)	READ	

Table 1. VSN Service, Attribute Table

a. Stealth Mode Configuration (UUID: 0xFFFFFFF1-00F7-4000-B000-000000000000)

The first characteristic of the VSN GATT Service is the Stealth Mode Configuration characteristic (0xFFFFFFF1…). This characteristic is used to control the V.BTTN's LED and buzzer behavior. The following values can be written to this characteristic.

Default value: 00

00 : Normal Mode (Sound and LED are active on the V.BTTN)

01 : Stealth Sound Mode (Sound is disabled)

02 : Stealth LED Mode (LED is disabled)

(e.g. 03 is Stealth Sound+LED both are disabled)

b. Detection Configuration (UUID: 0xFFFFFFF2-00F7-4000-B000-000000000000)

The Detection Configuration Characteristic enables/disables V.BTTN's detection modes. The following values can be written to this characteristic to enable/disable the button detection, and accelerometer modes. Once the detection mode is configured, V.BTTN will notify the phone/PC using the Notification Characteristic described in section (d):

Default setting: 0x02

0x01 - Enable short button press/release detection. App will be notified via Notification Characteristic, when the event is detected.

0x02 - Enable long (urgent) button press detection. App will be notified via Notification Characteristic, when the button is pressed and released within 2-8 seconds. V.BTTN will also buzz and blink the red LED.

0x04 - Enable Fall detection. App will be notified via Notification Characteristic, when a fall event is detected. V.BTTN will also buzz and blink the red LED.

0x08 – Enable High-G alert detection. App will be notified via Notification Characteristic, when high-G is detected.

0x10 – Enable raw accelerometer reporting. The individual x, y, z axis data will be reported thru the accelerometer service described in section 8.

The accelerometer modes (0x04, 0x08 and 0x10 are mutually exclusive).

CAUTION: Enabling accelerometer service greatly reduces the V.BTTN's battery life.

c. Long button press ACK (UUID: 0xFFFFFFF3-00F7-4000-B000-000000000000)

The V.BTTN will preserve the long button press event until the event is acknowledged by the App. If no acknowledgement is sent to the V.BTTN, it will send a notification every time it reconnects to the phone/PC and after Notification Characteristic has been enabled.

The long button press acknowledgement Characteristic (0xFFFFFFF3…) is used by the App to acknowledge receipt of long button press notification. To acknowledge the event, the App must write value 01 to this characteristic.

01 – Long button press ACK. V.BTTN will buzz and flash red-green LEDs to indicate that the ACK was received.

V.BTTN will flash red-green LED until a disconnect event or until it is cleared by the App. To clear the flashing LED state, the App must write value 00 to this characteristic.

00 – Clear flashing LED state.

d. Notification Characteristic (UUID: 0xFFFFFFF4-00F7-4000-B000-000000000000)

The Notification Characteristic uses the Client Characteristic Configuration (0x2902) to enable/disable notifications. Once enabled, V.BTTN notifies the application (using characteristic 0xFFFFFFF4-00F7-4000-B000-000000000000) when a preconfigured event (like button press or fall, see section e) is detected by the V.BTTN.

Notifications values:

00 - Button release detected

01 - Button press detected

03 - Button press-release between 2-10 seconds

04 - Fall event detected

05 – High-G event detected

e. V.BTTN Verification (UUID: 0xFFFFFFF5-00F7-4000-B000-000000000000)

After scanning for and connecting with V.BTTN, the App must write the verification key (80:BE:F5:AC:FF) to the Verification Characteristic. V.BTTN will disconnect from the phone/PC if it does not receive the verification key within 30 seconds from the connection. Refer to section 5, Connecting to V.BTTN for basic steps needed for your application to connect to V.BTTN.

8. Accelerometer Service

The V.BTTN includes a 3-axis accelerometer. The Accelerometer Service exposes raw accelerometer data to the application.

CAUTION: Enabling accelerometer service greatly reduces the V.BTTN's battery life.

Once raw reporting is enabled (see UUID 0xFFFFFFF2-00F7-4000-B000-000000000000), individual (X, Y, Z) axis notification can be configured by writing "01 00" to the Client Characteristic Configuration for the corresponding axis.

Once notification is enabled, changes to X, Y, Z axis will be received thru characteristics 0xFFFFFFA3-00F7-4000-B000-000000000000, 0xFFFFFFA4-00F7-4000-B000-000000000000 and 0xFFFFFFA5-00F7-4000-B000-000000000000, respectively.

NOTE: V.BTTN default Bluetooth wake-up interval is 1 second. For applications that require shorter wake-up interval, see Connection Control Service to update the connection interval.

UUID	Mnemoic	Value(default)	Permission	Notes
0x2800	<<Primary Service>>	0xFFFFFFA0-00F7-4000-B000-000000000000 (ACCEL SERVICE UUID)	READ	Start of Accelerometer Service
0x2803	<<Characteristic>>	10 (properties: notify) A3 FF (UUID: 0xFFFFFFA3-00F7-4000-B000-000000000000)	READ	X-Axis Characteristic declaration
0xFFFFFFA3…	<<X-Axis Value>>	00:00 (2 bytes)	(none)	X-Axis value
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	X-Axis notification configuration
0x2901	<<Characteristic User Description>>	"X-Axis" (17 bytes)	READ	
0x2803	<<Characteristic>>	10 (properties: notify) A4 FF (UUID:0xFFFFFFA4-00F7-4000-B000-000000000000)	READ	Y-Axis Characteristic declaration
0xFFFFFFA4…	<<Y-Axis Value>>	00:00 (2 bytes)	(none)	Y-Axis value
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	Y-Axis notification configuration
0x2901	<<Characteristic User Description>>	"Y-Axis" (17 bytes)	READ	
0x2803	<<Characteristic>>	10 (properties: notify) A5 FF (UUID: 0xFFFFFFA5-00F7-4000-B000-000000000000)	READ	Z-Axis Characteristic declaration
0xFFFFFFA5…	<<Z-Axis Value>>	00:00 (2 bytes)	(none)	Z-Axis value
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	Z-Axis notification configuration
0x2901	<<Characteristic User Description>>	"Z-Axis" (17 bytes)	READ	

9. Connection Control Service

The Connection Control Service exposes the V.BTTN's connection parameters, which control how often the V.BTTN and the connected phone/PC should 'sync' in order to maintain the connection. The connection interval, slave latency and supervision timeout can be changed by writing 8-byte array to UUID 0xFFFFCCC2-00F7-4000-B000-000000000000

a. Connection Interval (2-byte) – time between 2 connection sync event. The advantage of long connection interval is that it will reduce V.BTTN's power consumption. The disadvantage is that if V.BTTN has data to send it has to wait until the next connection event.
Default: 0x1003 (min) / 0x2003 (max)
Range: 7.5ms to 3980ms

How to convert connection interval to HEX parameter

1. Connect interval unit (decimal) = connection interval time (ms) / 1.25 ms
2. Convert connection interval decimal unit to HEX
3. Reverse 2 byte HEX value

For example: Connection interval 1000ms.
Connection interval unit (dec) = 1000 ms / 1.25 ms = 800
Convert 800 to Hex. = 0x0320

Reverse 2-byte hex value: 0x2003

Use 0x20 0x03 as (max) connection interval in the byte array

b. Slave Latency (2-byte) – the number of connection sync events that V.BTTN can choose to skip if it does not have data to send.
Default: 0x0000 (do not skip connection sync events)
Range: 0x0000 to 0xF301 (499)

NOTE: Max value must not make effective connection interval > 16.0s
*Effective interval = Max interval * (1 + Slave latency) <= 16s*

c. Supervision Timeout (2-byte) – maximum amount of time between two connection sync events.
Default: 0x5802 (6s)
Range: 100ms to 32s

How to convert Supervision Timeout to HEX parameter

1. Supervision Timeout unit (decimal) = Supervision Timeout (ms) / 10 ms
2. Convert Supervision Timeout decimal unit to HEX
3. Reverse 2 byte HEX value

For example: Supervision Timeout 6s (6000ms)
Timeout unit (dec) = 6000ms / 10ms = 600
Convert 600 to Hex = 0x0258
Reverse 2-byte Hex value = 0x5802

Use 0x58, 0x02 as Supervision Timeout in the byte array

Note: Connection parameter settings are dependent on the phone's Bluetooth stack implementation. Please use the following guidelines for the connection parameters.

- Connection Interval Max * (Slave Latency + 1) <= 2s
- Connection Interval Min >= 20ms
- Connection Interval Min + 20ms <= Connection Interval Max
- Slave Latency <= 4
- Supervision Timeout <= 6s
- Interval Max * (Slave Latency + 1) * 3 < Supervision Timeout
- More information regarding the interval calculations can be found at the following link:
https://developer.apple.com/library/content/qa/qa1931/_index.html

UUID	Mnemoic	Value(default)	Permission	Notes
0x2800	<<Primary Service>>	0xFFFFCCC000F74000B0000000 00000000	READ	Start of Connection Control Service
0x2803	<<Characteristic>>	10 (properties: notify) C1 CC (UUID: 0xFFFFCCC1-00F7-4000-B000-000000000000)	READ	Connection Parameter Characteristic Declaration
0xFFFFCCC100F74000-B000000000000000	<<Connection Parameter>>	00:00:00:00:00:00 (6 bytes)	(none)	Connection Interval, Slave Latency and Supervision Timeout (2-bytes each)
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	Connection Parameter Notification configuration
0x2901	<<Characteristic User Description>>	"Connect Parameter"	READ	
0x2803	<<Characteristic>>	08 (properties: write) C2 CC (UUID: 0xFFFFCCC2-00F7-4000-B000-000000000000)	READ	Request Connection Parameter Characteristic Declaration
0xFFFFCCC200F74000-B000000000000000	<<Request Connect Parameter>>	00:00:00:00:00:00:00:00 (8 bytes)	READ | WRITE	Min/Max Connection Interval, Slave Latency and Supervision Timeout (2-bytes each)
0x2901	<<Characteristic User Description>>	"Request Parameter" (17 bytes)	READ	

10. Connecting to V.BTTN

Following are steps to connect the V.BTTN to the application:

a. Enable Bluetooth on the phone.
b. Press-release V.BTTN button between 10-24 seconds to make visible to the app (discoverable)
c. Application performs scan for BLE devices and connects to it.
d. Application performs service discovery
e. Application sends verification key. This must be done within 30 seconds of connecting to V.BTTN.
f. Application sets detection configuration (mode), enables notification, etc.

11. V.BTTN LED and Buzzer

Following table describes V.BTTN LED and buzzer behavior in response to various user and/or application actions.

User/Application Interaction with Device	V.BTTN Device	
	LED	Buzzer
Insert or Replace the Battery	Buzzer/LED sequence as follows: Short chirp, blink RED LED, blink Green LED, short chirp	
Press and release button between 10 seconds and 24 seconds	1. Green LED blinks 5 times at 10 second press-hold mark 2. Then Green LED blinks every 10 seconds for 15 minutes or until connected	One short chirp once at 10 second press-hold mark
Application sends BLE connect to V.BTTN	Solid Green for 2 sec	One short chirp once
V.BTTN configured for long (urgent) button press detection and user press and release button between 2 to 9 seconds	1. Red LED blinks continuously until application acknowledge receipt of alert 2. Once application acknowledges, Red and Green blinks until application sends clear alert	One long chirp once
V.BTTN configured for fall detection and V.BTTN detects a fall event.	Same as button press activation alert use case (Same as above)	Same as button press activation alert use case (Same as above)
Phone and Device out of range (Bluetooh connection lost)	Red LED blinks every 10 seconds for 15 minutes or until back in range	Two short chirp once
Application write 02 to the Immediate Alert Service	Red LED blinks every two second upto 20 sec	Long chirp every two second up to 20 seconds

12. IOS Specific Tips and Recommendations

a. In iOS since iOS 7 core Bluetooth allows the saving of bluetooth connection status using CBCentralManager this allows your application to be notified of the bluetooth events you register for even if it's on the background in addition

if your app gets reopened after being force closed you will be notified on opening of registered events. More information about Core Bluetooth "bluetooth-central" can be found at the following link iOS Core Bluetooth Programming Guide. The following link provides more detail on the conditions which describe when your application will be reopened and the Bluetooth state restored. Conditions Under Which Bluetooth State Restoration Will Relaunch An App

The only time iOS will **NOT** automatically restore your Bluetooth state will be when the user Force quits the application by swiping it up after pressing the home button twice.
Apple recommends educating the user on how this will affect the application. We have found an alternative way to assist in making this as effortless as possible for them.

Registering your iOS application delegate to receive location notifications
Using CLLocationManager startMonitoringSignificantLocationChanges()
This will send a notification/wake up a force closed application when the user moves on our testing the application would reopen with as little as 25 feet of movement.

Additional links:
https://stackoverflow.com/questions/33130124/how-to-trigger-core-bluetooth-state-preservation-and-restoration
https://stackoverflow.com/questions/43578813/corebluetooth-and-ios-states

13. General recommendations on managing Bluetooth events.
    a. Initial searching of the V.ALERT
        i. Please instruct the user to press the V.ALERT for at least 10 second to put it in discovery mode.
    b. Bluetooth event for V. ALERT disconnected/out of range.
        i. We recommend displaying a notification and playing a sound to the user letting them know that they have gone out of range of the V.ALERT button.
        ii. In addition, this notification should be sticky/persistent until the device gets reconnected.
    c. Bluetooth event for V.ALERT reconnected
        i. Close the notification optionally let the user know the V.ALERT has been reconnected.