astra telematics

Application Note: Using BLE with Astra Devices

Scope
AT240BLE, AT400BLE

Overview
This application note describes the processes and terminology involved in connecting with Astra devices via BLE, for the purposes of sending $ commands, including the associated authentication options using $RSDK.

Related Documents
The following documents are recommended reading to accompany this document:

- AT240 & AT400 User Guides
- Astra Telematics Command Reference.pdf

The above documents can be obtained from:

http://www.gps-telematics.co.uk/downloads.htm

Device Configuration for BLE
1. BLE options are configured using the $BLEC command:

   $BLEC,<ble-enable>,<ble-store-bonding>,<adv-min-int-ms>,<adv-max-int-ms>,<adv-channels>,<rsdk-enable>,<auth-time-min>,<security-level>,<ble-name-prefix>

<ble-enable>	1 = enable BLE 0 = disable BLE
<ble-bondings>	0 = don't store bondings 1 = store bondings & clear on reboot 2 = store bondings & restore on reboot
<adv-min-int-ms>	min advertising interval in milliseconds
<adv-max-int-ms>	max advertising interval in milliseconds
<adv-channels>	advertising channels (bitfields); 7 = all channels bit 0 = channel 37 bit 1 = channel 38 bit 2 = channel 39
<rsdk-enable>	1 = enable RSDK security 0 = disable RSDK security
<auth-time-min>	0 = disable timeout (indefinite) Time in minutes to allow an authorised device to send BLE commands, when not using RSDK. When RDSK is enabled, the authorisation timeout is defined with each RSDK expiry time (see below)

<security-level>[1]	0 = no security (no pairing required) 1 = unauthenticated pairing with encryption 2 = authenticated pairing with encryption 3 = authenticated LE Secure Connections pairing with encryption
<ble-name-prefix>[2]	prefix for BLE name, in ascii, up to 8 characters Default prefix will be the device model name, e.g. "AT240V8"

1 implemented from Astra firmware version 7.0.39.xx / BLE firmware 2.13 and later
2 implemented from Astra firmware version 7.0.40.xx

Default: $BLEC,1,1,100,200,7,0,0,1,(model name)

E.g. on AT240V8 the defaults will be:

Default: $BLEC,1,1,100,200,7,0,0,1,AT240V8

2. Authentication of connected BLE devices is achieved using a Revocable Secure Digital Key or RSDK, which is defined using the command:

   $RSDK,<mode>,<rsdk>,<expiry-min>

<mode>	action: ADD, REMOVE, DISPLAY, CLEAR (entire list)
<rsdk>	32 byte RSDK in ascii coded hex (RSDK < 32 bytes will be padded with leading zeroes)
<expiry-min>	expiry time of the RDSK in minutes 0 = disable expiry timeout (default)

   Max RSDK list size: 10

   RSDKs for authorised users are sent to the device from the host server. Once stored, a user may connect to the Astra device using BLE, as defined in the following section. Once authenticated and bonded with the Astra device, the user may send Astra $ commands to the device.

   example – add an RSDK to the device list:

   `$RSDK,ADD,101112131415161718191A1B1C1D1E1F000102030405060708090A0B0C0D0E0F`

   Note 1: since no expiry time has been specified in the command, the default value of 0 will apply, which means that the RSDK will never expire

   Note 2: each Astra device has a local list of up to 10 RSDKs. Expired RSDKs will be automatically deleted, as and when they expire. In case the list gets full, recently used RSDKs will take priority and the new RSDK will overwrite the one which has been unused for the longest time.

3. Astra device BLE name will be the device model followed by the last 7 digits of the device IMEI, for example:

   AT240V8-1234567

astra telematics

BLE Module Base Firmware

There are 2 versions of the BLE module firmware, which defines the GATT profiles etc:

2.10.1.409 (released 15/03/18) – Basic working release
2.13.0.149 (released 08/01/20) – Allow unencrypted insecure connections

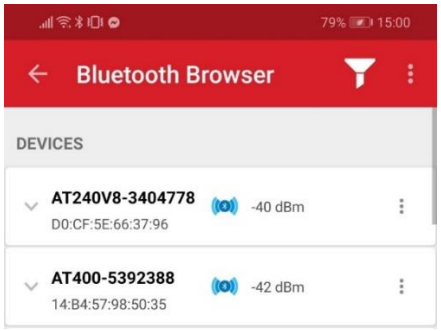
Authentication, Pairing and Bonding with BLE

The example below uses the Blue Gecko Android or iOS mobile phone application to go through the process of connection, pairing and bonding with the Astra device.

a. Open the app and select “Bluetooth Browser”

b. Power up the Astra device

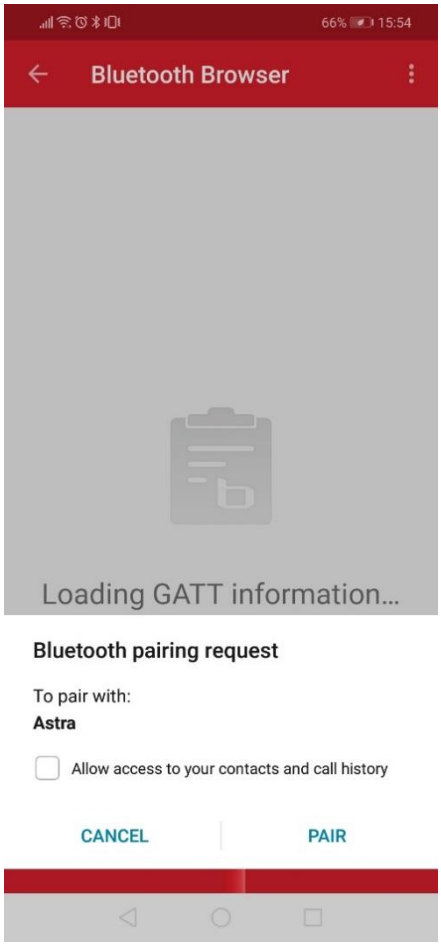
c. The Astra device will appear with the device model number (or alternative device name specific in the $BLEC command) followed by the last 7 digits of the IMEI, as below:



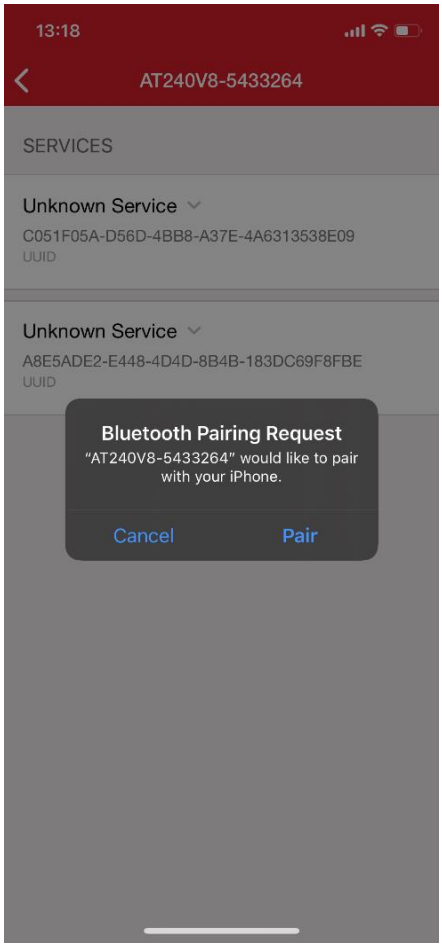
d. Select the Astra device.

e. If you have the $BLEC <security-mode> option set to any value but zero, you will see a pairing prompt on first connection. Press ‘Pair’ to pair:

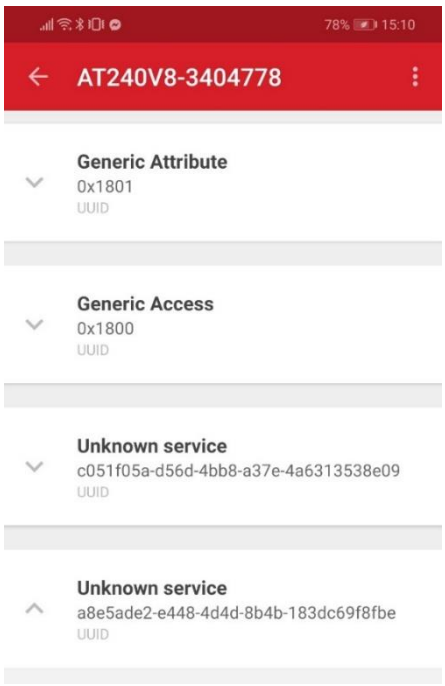
Android:



iOS:



f. You should now see the device screen:



This service is the RSDK service, solely for authenticating using a valid RSDK

This service is the Command service, for sending $ commands and reading the responses from the device

g. First, an RSDK must be stored in the Astra device using the $RSDK command. The RSDK itself is a maximum of 32 bytes, and it is defined in hex, hence a maximum of 64 characters for the RSDK itself. If RSDK length is less than 32 bytes, there is no need add leading zero-padding.

We are going to use this RSDK in these examples:

101112131415161718191A1B1C1D1E1F
000102030405060708090A0B0C0D0E0F

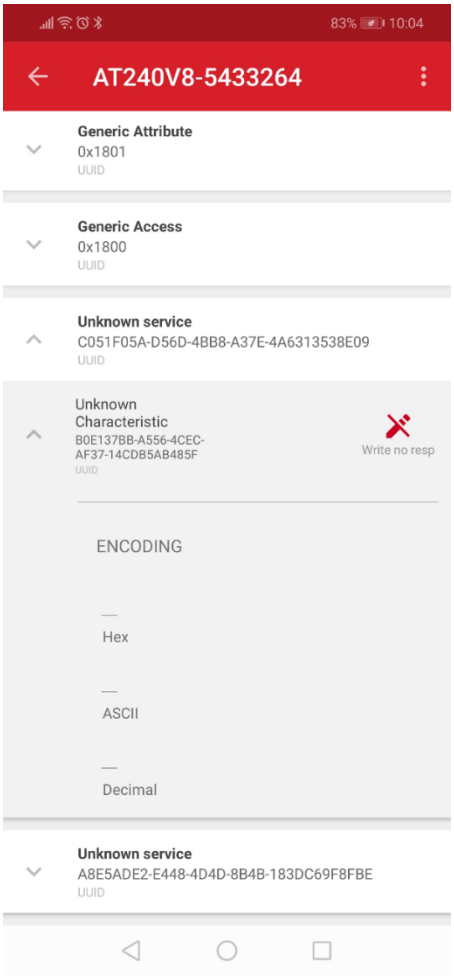
Add the RSDK to the whitelist:

$RSDK,ADD,101112131415161718191A1B1C1D1E1F000102030405060708090A0B0C0D0E0F

In this case, we haven't given an expiry time, so the RSDK will never expire.

h. Now that the Astra device has the RSDK stored, we can connect to the BLE using the same RSDK.

Select the RSDK service (c051f05a-d56d-4bb8-a37e-4a6313538e09):



i. Send the RSDK in 2 packets, as follows:

Vendor ID = 0xA5 B6
Auth version = 0x01
Sequence number = 0x00 in 1st packet, followed by 0x01 in 2nd packet

1st packet: A5 B6 01 00 + [most significant 16 bytes of RSDK]
2nd packet: A5 B6 01 01 + [least significant 16 bytes of RSDK]

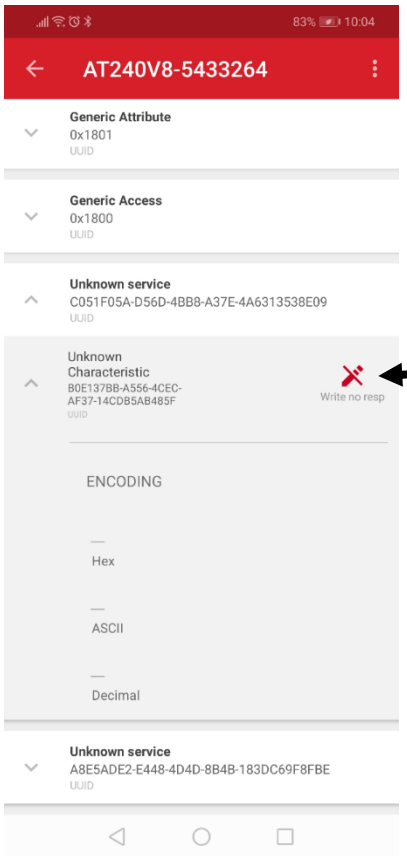
If the RSDK length is less than 32 bytes, leading zeroes must be added at this stage.

For example, the RSDK (spaces between bytes added here for readability):

10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

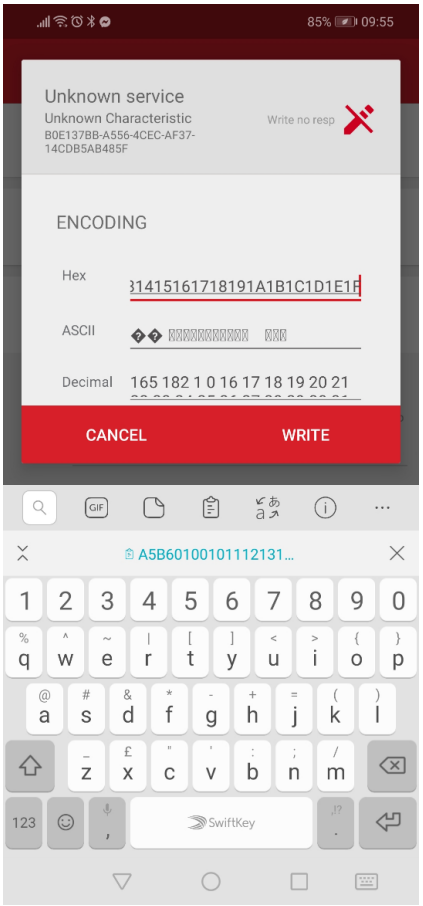
Is sent as two packets, first:
A5B60100101112131415161718191A1B1C1D1E1F
And then:
A5B60101000102030405060708090A0B0C0D0E0F

j. Select the ***Write*** or ***Write no resp*** option on the RSDK characteristic under the RSDK service highlighted before:

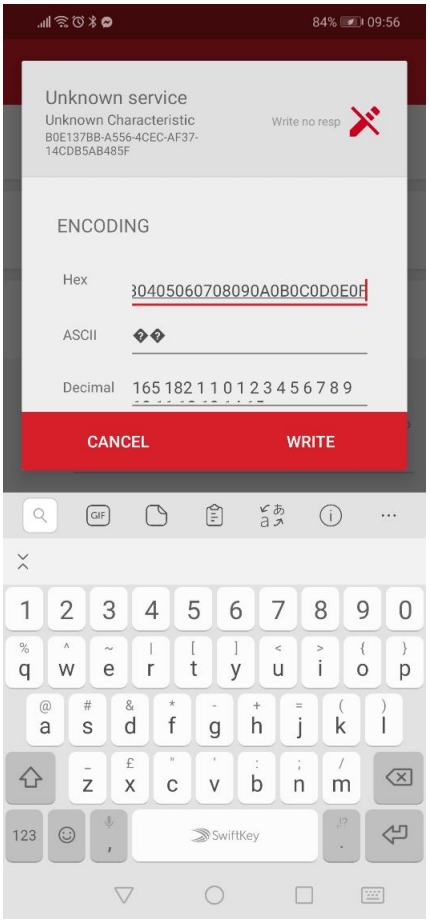


Select the ***Write*** or ***Write no resp*** option

k. Enter RSDK packet 1 bytes into the ***Hex*** field in the pop-up and select ***WRITE***:



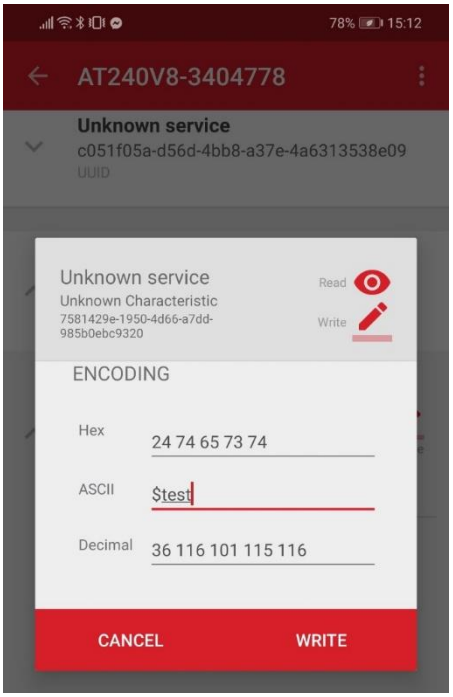
l. Do the same for packet 2, select ***Write*** or ***Write no resp*** option and enter 2nd RSDK packet into the ***Hex*** field and then ***WRITE***:



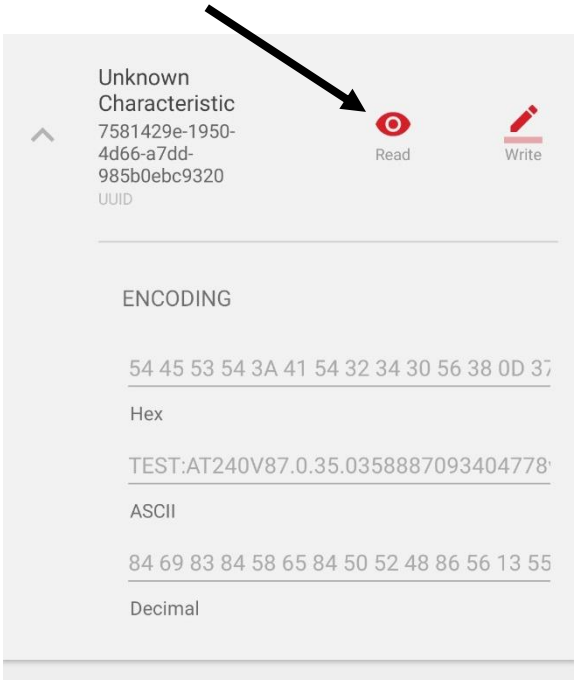
m. Now that we have authenticated using the RSDK, we can proceed to send commands

n. Select the Command service b0e137bb-a556-4cecaf37-14cdb5ab485f.

o. Select ***Write***, enter the Astra $ command into the ***ASCII*** format field and then ***WRITE***



p. Now select ***Read*** to see the response from the Astra device:



Implementation Notes:

4. BLE Bonding error code descriptions can be found on page 154 of the following document: https://www.silabs.com/documents/public/reference-manuals/bluetooth-api-reference.pdf

5. Astra device will store BLE bonding information and keys for all paired devices, represented by the BT MAC Address.

   Note the difference between “pairing” & “bonding” in BLE:

   Bonding is the exchange of long-term keys after pairing occurs, and storing those keys for later use, creating permanent security between devices. Pairing is the mechanism that allows bonding to occur.

   https://www.silabs.com/community/wireless/bluetooth/knowledge-base.entry.html/2018/11/07/using_bluetooth_secu-VDcD

6. On the link layer, the Astra device will store LTKs (BLE Long Term Keys) for all bonded devices, represented by the BT MAC Address.

7. Once bonded, the Astra device and mobile device can start to communicate with commands. Please refer to the document below for details of the command protocol and supported commands:

   https://gps-telematics.co.uk/wp-content/uploads/2017/09/astra_device_configuration_guide.pdf

8. Commands should be sent to the Astra device using a WRITE REQUEST

9. Astra device will respond using a WRITE RESPONSE

10. Where a WRITE REQUEST is received from an unauthenticated user device (or where authentication has expired), the Astra device will respond with an UNAUTHENTICATED REQUEST error code

11. Commands and responses will be limited to a max length of 160 characters.

12. RSDKs are not allowed to expire whilst in a journey, so will only expire whilst ignition is OFF. In fact, any active connection (bonding) will remain after the RSDK expires, since a valid RSDK is required only to initiate the connection, not to maintain it.

Both authentication packets should be sent by the phone in a BLE WRITE REQUEST (and handled in the Astra device in the gecko_evt_gatt_server_user_write_request_id event).

If the auth header validation fails for any reason, the Astra device will respond with a WRITE RESPONSE containing a custom error code; the *att_errorcode* field of the response should be populated with an error code (if applicable, if not 0) in the range of 0x80 - 0x9F (this is the range reserved for application level errors in the BT stack):

- 0x80 Unauthenticated request . The authorization period has expired and the user needs to auth again (or the auth process wasn't even undertaken to begin with)
- 0x81 Authorization denied. RSDK is not on the Astra device memory
- 0x82 Unsupported authentication protocol vendor. The authentication protocol vendor is not supported by this Astra device
- 0x83 Unsupported authentication protocol version. The authentication protocol version specified in the Auth header is not supported by this ASTRA device (the ASTRA has a lower version).
- 0x84 Unsupported authentication protocol version. Same as before, but the ASTRA has a higher version.
- 0x85 Invalid sequence number or number out of expected sequence
- 0x87 Invalid or unsupported characteristic
- 0x88 Already busy with a command
- 0x89 Invalid operation (e.g. reading RSDK property instead of writing it)

If both WRITE REQUESTs of the auth process return an error code of 0, we can say that the auth process ended successfully and the user's phone is already authenticated and authorised. In practice, at the firmware level, this means that the Bonding handle (check Bluetooth API reference) of the phone making the requests is added to a whitelist and requests with this Bonding handle will be trusted for the next 60 minutes (if the Astra device is reset, the whitelist could be lost). After this period of time, the phone should receive an Unauthenticated request error code when issuing a command, and this should trigger at the phone another auth process, with a subsequent issuing of the same command again. This whole auth process should be transparent to the user and phone should be able to react gracefully to loss of authentication on these oblivious devices. If the connection is closed at any time, the auth process will be restarted and will expect the first packet with sequence number 0 to be sent first. This holds true even if the connection is closed immediately following the reception of the first packet i.e. the first packet must be sent again.