Tech Note #010

Encrypted TLVs in Augusta

Rev. A

Revised 10/03/2016

International Technologies & Systems Corporation
10721 Walker Street, Cypress, CA 90630-4720; Tel: (714) 761-6368; Fax (714) 761-8880
www.idtechproducts.com

Purpose

This document explains how to control ID TECH's Augusta reader to produce MSR-comparable track data in the encrypted output of EMV transactions, through the use of tags DFEF4B, DFEF4C, and DFEF4D.

The information presented here augments, but does not supersede or replace, the more extensive information provided on this subject in document 80000502-001, *ID TECH Encrypted Data Output Formats*. Please continue to refer to that document for the most authoritative, up-to-the-minute information on encrypted data handling.

Applicability

The information described here is applicable to ID TECH Augusta readers with firmware version USB-HID V1.01.002 or above. Run command 78 31 to obtain the detailed firmware version number for your unit.

What You Need to Know

Your payment processor may require you to provide encrypted track data that, when decrypted, looks something like this:

3b343736313733393030313031303031303d31353132323031313134333837383038393f

Which, after rendering as ASCII, would look like:

;4761739001010010=15122011143878089?

Using terminal configuration tag DFEF4B, you can cause such data to appear in tag DFEF4D. What's more, you can control:

- Which tracks (1, 2, or 3) are allowed to appear in DFEF4D
- Whether sentinels appear (or not) in tracks 1, 2, or 3, in tag DFEF4D
- Whether PAN data appears, as a separate element, in DFEF4D
- Whether to include *all* eligible track data, or just the first element found

If you do not specify tag DFEF4B in your terminal configuration settings, tag DFEF4D will not appear in the transaction output; instead, you will see the track-data tags appropriate to your EMV transaction (e.g., tag 56 for contactless track 1, tag 57 for contact track 2, etc.), and those tags will be encrypted in the manner previously described in ID TECH document 80000502-001.

What Do the Flags in DFEF4B Do?

Tag DFEF4B is a configuration tag whose Value is three bytes long. The first byte contains flags that allow you to control data-reporting behavior. (The other two bytes are currently Reserved for Future Use.)

Byte 1:

8	7	6	5	4	3	2	1	NOTES
-	-	-	-	-	-	-	X	0 – Disable Track 3 Sentinels 1 – Enable Track 3 Sentinels
-	-	-	-	-	-	X	-	0 – Disable Track 2 Sentinels 1 – Enable Track 2 Sentinels
-	-	-	-	-	X	-	-	0 – Disable Track 1 Sentinels 1 – Enable Track 1 Sentinels
-	-	-	-	X	-	-	-	0 – Disable Track 3 1 – Enable Track 3
-	-	-	X	-	-	-	-	0 – Disable Track 2 1 – Enable Track 2
-	-	X	-	-	-	-	-	0 – Disable Track 1 1 – Enable Track 1
-	X	-	-	-	-	-	-	0 – Disable PAN 1 – Enable PAN
X	-	-	-	-	-	-	-	0 – All Data Elements Found 1 – Only First Element Found

Byte 2: RFU

Byte 3: RFU

You can use the top bit of the first byte of DFEF4B to control search behavior: If the bit is ON, all data elements requested will be provided (if they exist). If the bit is OFF, only the *first* element found will be retrieved and placed in DFEF4D.

If you request multiple data items, they will be concatenated. To know the original lengths of those items, you must retrieve and inspect Tag DFEF4C

The default value of tag DFEF4B is 0x12 (Track 2 enabled, with Sentinels).

Data Search Order

When **"Only First Element Found" (bit 8 = 1)** is set in DFEF4B, Tag DFEF4D will be populated with a *single* data element according to the following search order:

Track 2, Tag 57 (converted to alpha numeric format)

Track 2, Tag 9F6B
Track 2, Tag 5F22
Track 1, Tag 56
Track 1, Tag 5F21
PAN, Tag 5A (converted to alpha numeric format)
Track 3, Tag 58
Track 3, Tag 5F23

Regardless of the original format, the data will be placed in the DFEF4D tag in alpha-numeric format, such that after decryption (and with padding removed) the data will look similar to:

3b343736313733393030313031303031303d31353132323031313134333837383038393f

Thus, after rendering as ASCII, it would look like:

;4761739001010010=15122011143878089?

When **"All Data Elements Found" (BIT 8)**, is specified in DFEF4B, Tag DFEF4D will be populated with a single instance of each requested data element, according to the following order:

Track 1 requested (bit 6 = 1). Includes first instance of:

Tag 56 = Track 1 Equivalent
Tag 5F21 = Track 1, identical to the data coded

Track 2 requested (bit 5 = 1). Includes first instance of:

Tag 57 = Track 2 Equivalent (converted to alpha numeric format)
Tag 9F6B = Track 2 Data
Tag 5F22 = Track 2, identical to the data coded

Track 3 requested (bit 4 = 1). Includes first instance of:

Tag 58 = Track 3 Equivalent
Tag 5F23 = Track 3, identical to the data coded

PAN requested (bit 7 = 1). Includes:

Tag 5A = PAN (converted to alpha numeric format)

Sentinels

For any found data element of Track1, Track2 or Track3, sentinels will be included or not included according to the preferences set in bits 1, 2 and 3.

For any data element captured as compressed numeric, the following rules apply:

- Padding (0xf) will not be included
- Center separators: 0xd will be converted to 0x3d ("=")
- Data will be encoded as an ASCII representation of binary data
  example: 0x123f = 0x313233 = "123" (ignore padding)
  example: 0x1234 = 0x31323334 = "1234"
  example: 0x123d456f = 0x3132333d343536 = "123=456"

How to Use Tag DFEF4B to Control EMV Track Data Output

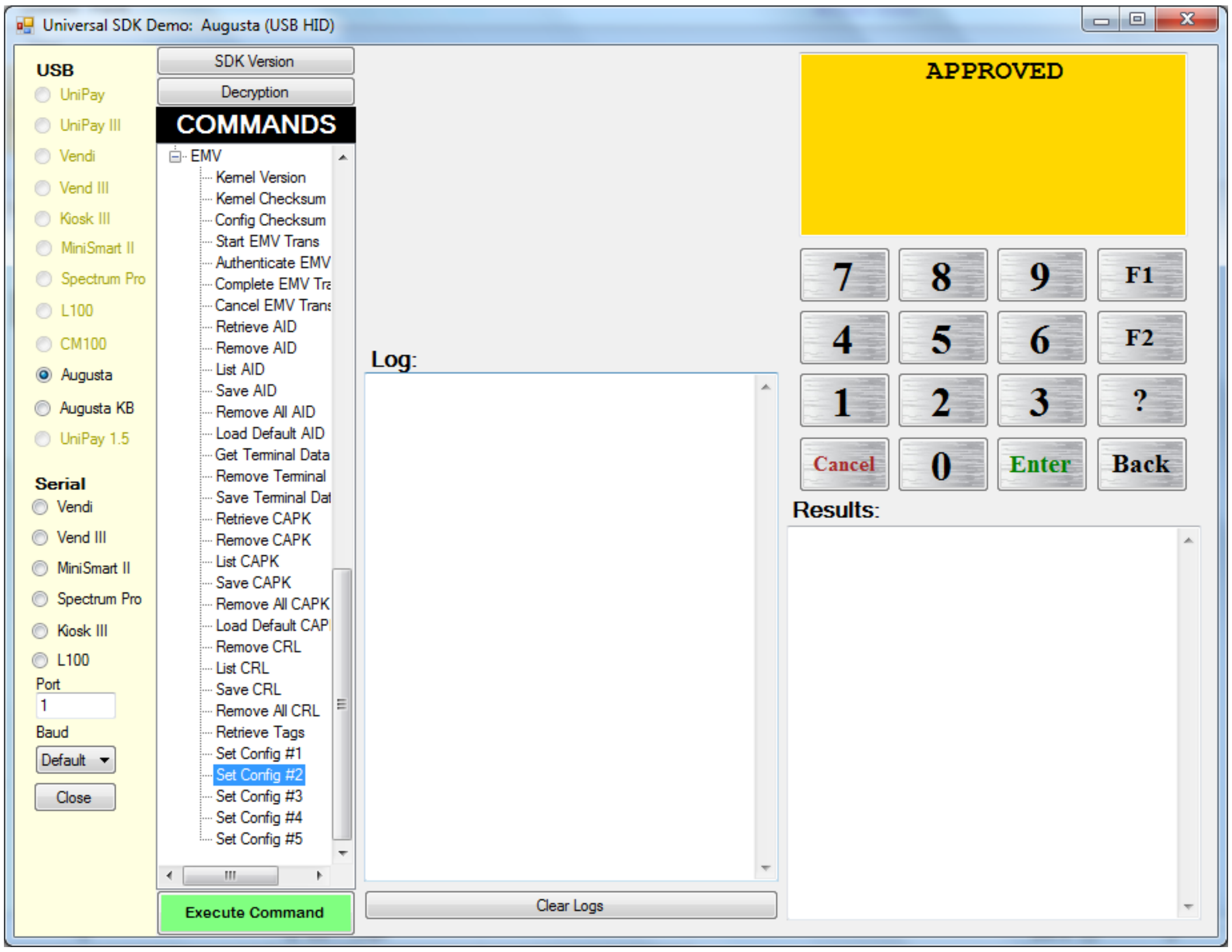
Tag DFEF4B is a configuration tag. Apply it using the Save Terminal Settings command. (In Augusta, this is command 72 46 02 03.)

Below, we describe how to use this tag to control encrypted data output using ID TECH's Universal SDK Demo App.

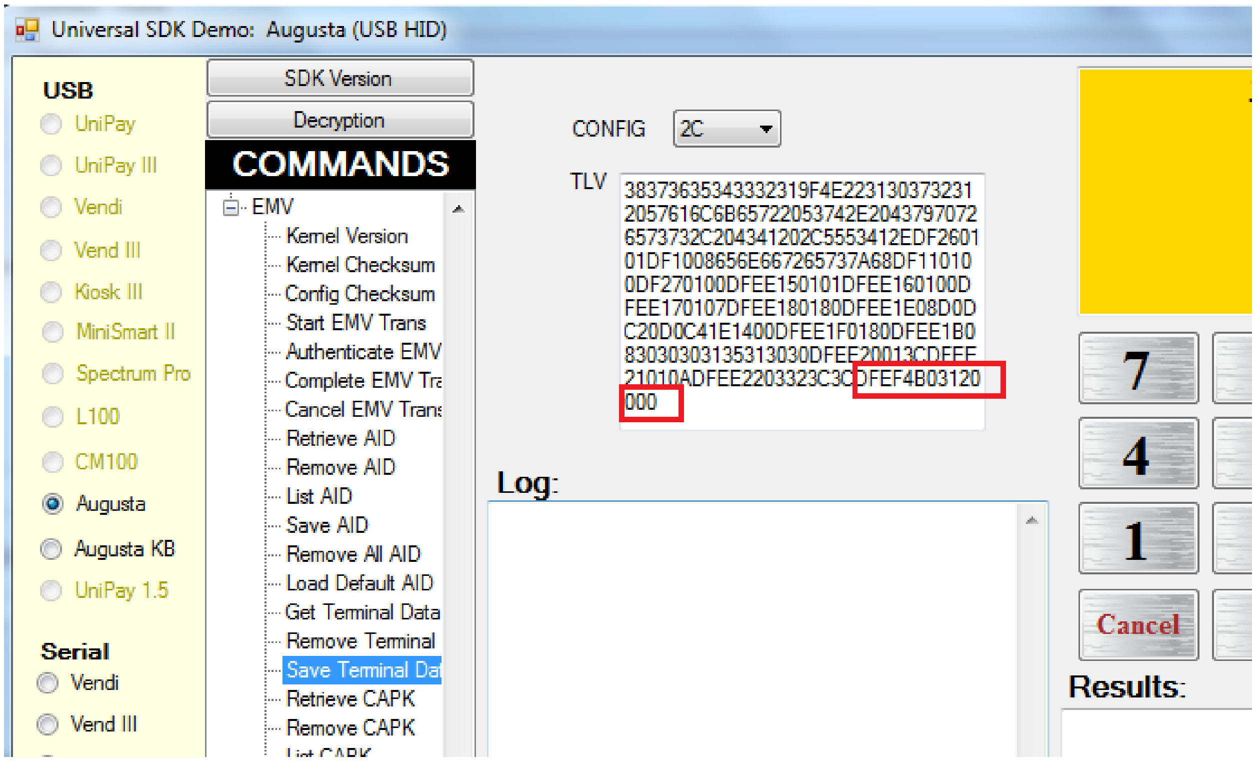
Example 1: Obtain Track 2 Data, with Sentinels
To obtain track 2 data, with sentinels, you would supply the value 0x12 for the first byte of tag DFEF4B.

To do this using the Universal SDK Demo App:

1. Launch the Demo App.
2. Connect your Augusta reader to the host. Be sure the Demo App recognizes it.
3. Confirm that you are using the latest firmware (V1.01.002 or above) by running command 78 31 if necessary.
4. In the command tree, open the EMV node and scroll down to **Set Config #2**. (See screen shot.)



5. Double-click the **Set Config #2** command, or use the **Execute Command** button to execute it.
6. Select (but don't execute) the **Save Terminal Data** command in the EMV command tree. This will cause a data pane to appear, pre-populated, in the upper middle portion of the window:



CONFIG 2C

TLV

```
383736353433332319F4E223130373231
2057616C6B65722053742E2043797072
6573732C204341202C5553412EDF2601
01DF1008656E667265737A68DF11010
0DF270100DFEE150101DFEE160100D
FEE170107DFEE180180DFEE1E08D0D
C20D0C41E1400DFEE1F0180DFEE1B0
83030303135313030DFEE20013CDFEE
21010ADFEE2203323C3CDFEF4B03120
000
```

7. Select **2C** from the CONFIG dropdown menu at the top (see above).
8. Manually add the following TLV to the TLV list, at the end (as shown in the red rectangles above): **DFEF4B03120000**.
9. Use the **Execute Command** button to execute the terminal configuration change. The Results pane should show a message of " Set Terminal Successful."
10. Insert a chip card into Augusta and execute **Start EMV Transaction**. (For this example, we used B2 test card 04, Visa: French.)
11. After the transaction completes, select the **Retrieve Tags** command in the command tree (but do not run it).
12. Type **DFEF4CDFEF4D** in the Retrieve Additional Tags pane at the top of the window. (This tells Augusta you want to retrieve tags DFEF4C and DFEF4D.)
13. Now use the **Execute Command** button to run the Retrieve Additional Tags command.
14. Check the Results pane. You should see something like:

```
DFEE12: 62994900750002a0015e
DFEF4C: 00240000
DFEF4D:
271795359221cc2d09fcf6c6421aa7e21c5d039d58e38091b0c8b9c37d420fba9ec5c303b745f527
```

The first line shows the transaction KSN (in tag DFEF12). The second line shows the lengths of the data elements retrieved in tag DFEF4D. (In this case, there is only one element, track 2, and it is 0x24 or 36 bytes long.) Finally, the encrypted track data you requested is in DFEF4D. (If the original, unencrypted data was not a multiple of 8 or 16 in length, it will have been zero-padded to such a multiple prior to TDES or AES encryption, respectively.)

Example 2: Obtain All Available Track Data, Plus PAN
To obtain all available track data, with sentinels, you would supply the value 0x7F for the first byte of tag DFEF4B.

1. Perform the first 7 steps of the previous example without change.
2. In Step 8, manually add the following TLV to the TLV list: **DFEF4B037F0000**.
3. Use the **Execute Command** button to execute the terminal configuration change. The Results pane should show a message of " Set Terminal Successful."
4. Insert a chip card into Augusta and execute **Start EMV Transaction**. For this example, we used Visa ADVT Test Card #43.
5. After the transaction completes, select the **Retrieve Tags** command in the command tree (but do not run it).
6. Type **DFEF4CDFEF4D** in the Retrieve Additional Tags pane at the top of the window. (This tells Augusta you want to retrieve tags DFEF4C and DFEF4D.)
7. Now use the **Execute Command** button to run the Retrieve Additional Tags command.
8. Check the Results pane. You should see something like:

```
DFEE12: 62994900750002a00160
DFEF4C: 00240010
DFEF4D:
3d42797f7f5d8caab8ff9ebb0d45c2f6a3089e64b6e55b75c4cd609559977812512ce24a7e910ad3ba
57f38dd2b35afeb4fb66b3d9234063
```

Notice that, this time, the value of DFEF4C shows one data element with length 0x24 and another with length 0x10. The first data element is, of course, track 2. The second is the 16-digit PAN.

If you use the Decryption feature of the latest version of the ID TECH Universal Demo App (see the button at the top of the COMMANDS pane), you can decrypt the payload of tag DFEF4D. In the foregoing example, we get:

```
3b343736313733393030313031303433323d313031323230313136333131373835383933f343736
31373339303031303130343332000000000

;4761739001010432=10122011631178589?4761739001010432
```

The raw, zero-padded alpha-numeric value of the payload is shown above the ASCII version. The track data (36 bytes) ends with a question mark sentinel. The PAN has an ASCII value of 4761739001010432.