AN3219
Application note

Setting manufacturing certificates and installation codes for the STM32W108 platform

1 Introduction

The ZigBee Smart Energy specification uses public/private key technology to authenticate a device joining a Smart Energy network and provides a means to securely establish encryption keys for future transactions. The Smart Energy specification uses Elliptical Curve Cryptography (ECC) for cryptographic authentication and key generation.

The Smart Energy profile uses ECC with the key establishment cluster to derive a link key. It also uses ECC for creating digital signatures for the event status messages of the Demand Response Load Control Cluster (DRLC).

Currently Certicom (www.certicom.com) provides both the certificates and the ECC technology for use in ZigBee Smart Energy networks.

Contents

2 Security components

When referring to Smart Energy Certificates, there are actually three components that make up the ECC security data contained within a node:

- Certificate
- Static Private Key
- Certificate Authority (CA) Public Key

2.1 Certificate

The first component of the ECC security data is the certificate. This is the device's IEEE address and static public key, which are signed using the CA's private key. It is 48 bytes in length. The certificate is associated with a particular IEEE address. As a result, the IEEE address of the device must match the value contained in the certificate.

Although this is part of the security data, it is not secret, and does not need to be handled in the same fashion as the private key. It is transmitted over the air during key establishment. *Table 1* summarizes the four different fields contained within the certificate body.

Table 1. Certificate body fields

Field name	Length (bytes)	Description
Public Key Reconstruction Data	22	Device's public key signed by the CA's private key.
Subject	8	Contains the IEEE address associated with the certificate, in big endian format.
Issuer	8	Identity of the CA that issued the certificate.
Attributes	10	An extra set of data associated with the device whose authenticity is guaranteed by the CA. It is not currently used by ZigBee Smart Energy.

2.2 Static private key

The second component of the ECC security data is the static private key. This is a secret piece of data that should be carefully protected. It is recommended that during the manufacturing process only those computers that need to know this data to program the chip have access to it.

2.3 CA public key

This is the CA's public key that corresponds to the secret private key held by Certicom. It is used to authenticate certificates received by remote devices and validate the keys derived using the Smart Energy Key Establishment Cluster. While it is not transmitted over the air, it is also public information, and does not need to be kept secret.

3 Certificate authorities

There are two Certificate Authorities (CA) provided by Certicom for use in Smart Energy networks:

- A Test CA that is intended only for use in internal development environments.
- The Production CA that is intended for use in real Smart Energy deployments.

Note: Certificates from the Test CA and the Production CA are not compatible.

There are two CAs because developers and installers have different security needs. Developers need to be able to test and debug devices where they will have access to the private keys. Their interests lie in having access to internal data in order to accomplish their goal of writing software. On the other hand, installers want reasonable assurances that a device will not have compromised security data. Their goal is to have a secure installation. Certicom can meet both needs by providing different security for each CA.

3.1 Test CA

Certicom provides a Test CA for quickly and easily generating certificates for use in development units wishing to use ECC. The following link can be used to obtain certificates from the Test CA: http://www.certicom.com/index.php/gencertregister

EmberZNet provides two test certificates in their Smart Energy sample application, bound to IEEE addresses 0000000000000001 and 0000000000000002. These may be used in combination with the Test CA to perform key establishment. They can be found in the source code file app/ha/ami-key-establishment-test-certs.c.

3.2 Production CA

The Production CA is the official CA that is used by all deployed devices. All manufactured devices should contain the Production CA's public key, and a certificate and private key associated with the Production CA.

Production CAs are obtained through a different mechanism from Test CAs. Since manufacturing involve hundreds or potentially thousands of devices, a different process is used to obtain bulk certificates for a block of IEEE addresses. In addition, security of the transmission of the private keys is much more important and thus is handled differently from the Test CA.

It is recommended that a manufacturer obtain their own block of IEEE addresses for production units instead of using the internal one programmed on the chip. Certificates identify a Smart Energy device and its vendor and STMicroelectronics's chip may be used in many Smart Energy deployments. In addition, it is easier to issue certificates for blocks of IEEE addresses. STMicroelectronics cannot guarantee a set of known IEEE addresses with a particular set of chips, so it is recommended that the manufacturer install their own to make it easier to manage them.

Contact Certicom for more information about obtaining production certificates.

4 Certificate validation

It is important to note that ECC uses a technology called implicit certificates. Traditional SSL certificates used on the Internet contain the unencrypted public key of the device along with a separate signature field that ensures the authenticity of the data. However, with implicit certificates, the two pieces of data are combined together to shrink the size of the certificate for use in embedded devices. This is known as Public Key Reconstruction data.

As a result of the combination, it is not possible to verify an implicit certificate without using the certificate as part of the Key Establishment Cluster. Only by performing Key Establishment to derive a link key can a device determine if its certificate is valid.

5 Matching the components

It is important to note that the four pieces of security data installed for Smart Energy must all match up correctly. The installed certificate must be the correct one for the private key that is also installed. The IEEE address of the device must match the certificate that is installed. Lastly, the certificate must have been issued by the same CA whose public key is installed along with it.

If any of the components is mismatched, the device will never be able to successfully perform key establishment to authenticate itself on a Smart Energy network.

5.1 Manufacturing test

It is recommended that manufacturers validate their process by testing the installed certificate in their first batch of devices. A full test involves successfully joining a Smart Energy network and using the Key Establishment Cluster to derive a link key. The devices in the test should be using Production Certificates issued by the Production CA.

The STM32W108 platform can store application-specific manufacturing data into an area of Flash memory that can be used to store unique device information.

Note: Hard-coding a device's unique information in source code does not work when using the same application image on multiple devices. The preferred alternative is to have the software read in the device's unique information from the manufacturing area of Flash memory.

6 Working with certificates on the STM32W108

6.1 Overriding the default EUI64

By default, a STMicroelectronics EUI64 address is pre-programmed into all STM32W108 chips. Customers may override this by writing their own EUI64 into the Custom EUI64 area of the manufacturing flash memory.

When programming certificates and installation codes, the em3xx_load.exe tool allows you to override the STMicroelectronics EUI64. When a custom EUI64 is already present, the certificate or installation code must use an EUI64 that matches the custom one.

If you do not want to override the EUI64, then the certificate and installation codes must use STMicroelectronics EUI64s, and those EUI64s must match what is already programmed on the chip. You cannot override the EUI64 of the device with another STMicroelectronics EUI64.

The reason these requirements are in place is to prevent certificates or installation codes from being programmed on the wrong device. Both of these items are bound to the EUI64 of the device and therefore a mismatch will cause problems when deploying an end product.

To program a device with a certificate or installation code, use the em3xx_load.exe.

7 Programming the certificates into a device

7.1 Checking the node information

Prior to modifying the certificate it is best to verify there is connectivity with the device to be programmed, and what information is currently stored on the node. To do this, execute the following command:

```
$ em3xx_load.exe --cibtokensprint

em3xx_load (Version 1.0b27.1264522950)

Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108

'General' token group
TOKEN_MFG_CIB_OBS          [16 byte array ] : A55AFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION   [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64    [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_STRING           [16 byte string] : "" (0 of 16 chars)
                                              FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME       [16 byte string] : "" (0 of 16 chars)
                                              FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID         [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG       [16-bit integer] : 0xFFFF
```

```
TOKEN_MFG_BOOTLOAD_AES_KEY [16 byte array ] : FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
TOKEN_MFG_EZSP_STORAGE     [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM [16-bit integer] : 0xFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CA Public Key        [22 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFF
Device Private Key   [21 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFF
CBKE Flags           [ 1 byte array ] : FF

'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : FFFF
Install Code       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CRC                [16-bit integer] : 0xFFFF

DONE
```

The pre-programmed STMicroelectronics EUI64 is not shown in this output as it is part of the STMicroelectronics-programmed manufacturing data rather than the customer-programmed manufacturing data. To obtain the STMicroelectronics EUI64 of an STM32W108 device, use the following command:

```
em3xx_load --read @080407A2-080407A9
```

This reads out the STMicroelectronics EUI64 in least-significant byte (LSB) notation. For example, the command above produces the following output (indicating an STMicroelectronics EUI64 value of `0x0008E102000001FE`):

```
$ em3xx_load --read @080407A2-080407A9
em3xx_load (Version 1.0b27.1264522950)

Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108
Reset Chip
Getting memory from 0x080407A2 through 0x080407A9

{address:  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F}
080407A0:  .  . FE 01 00 00 02 E1 80 00  .  .  .  .  .  .

Run (by toggling nRESET)
DONE
```

If the custom EUI64 (IEEE address) is set to all Fs, then the STMicrolectronics EUI64 address is used. If this value is not all Fs, then it used instead of the STMicroelectronics EUI64.

If the data within the Smart Energy token group (certificate, private key, root CA public key, metadata) is set to all Fs, then the data is not valid and will not be used by the software.

7.2 Format of the certificate file

To program a certificate into the tokens, it is necessary to create a simple text file with the security information in it. The following is the format of that file expected by the em3xx_load.exe tool.

```
CA Public Key: <hex-string>
Device Implicit Cert: <hex-string>
Device Private Key: <hex-string>
```

7.3 Writing the certificate into the manufacturing area

To write the certificate into the manufacturing area, execute the following command:

```
$ em3xx_load.exe --cibtokenspatch sample_cert.txt
em3xx_load (Version 1.0b27.1264522950)

Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108
Reset Chip

Writing to address 0x0804087E for token 'Device Implicit Cert'
Writing to address 0x080408AE for token 'CA Public Key'
Writing to address 0x080408C4 for token 'Device Private Key'
Writing to address 0x080408D9 for token 'CBKE Flags'

NOTE: Writing Custom EUI64 '0100000000000000' to match certificate.
Address 0x08040812.

Create image file
Install RAM image
Verify RAM image
Install Flash image
Verify Flash image
Run (by toggling nRESET)
DONE
```

In addition to writing the certificate, this will update the custom IEEE address so that it matches the data in the certificate. This ensures that there is parity between the data in the certificate and the IEEE address used by the local device for its identity on the network.

7.4 Verifying the stored certificate

After writing the certificate, it is best to verify the information by executing em3xx_load again as shown in the example below.

```
$ em3xx_load.exe  --cibtokensprint
em3xx_load (Version 1.0b27.1264522950)

Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108


'General' token group
TOKEN_MFG_CIB_OBS          [16 byte array ] : A55AFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION   [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64    [ 8 byte array ] : 0100000000000000
TOKEN_MFG_STRING           [16 byte string] : "" (0 of 16 chars)
                                              FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME       [16 byte string] : "" (0 of 16 chars)
                                              FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID         [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG       [16-bit integer] : 0xFFFF
TOKEN_MFG_BOOTLOAD_AES_KEY [16 byte array ] : FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
TOKEN_MFG_EZSP_STORAGE     [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM [16-bit integer] : 0xFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : 03045FDFC8D85FFB 8B3993CB72DDCAA5
                                        5F00B3E87D6D0000 0000000000015445
                                        5354534543410109 0006000000000000
CA Public Key        [22 byte array ] : 0200FDE8A7F3D108 4224962A4E7C54E6
                                        9AC3F04DA6B8
Device Private Key   [21 byte array ] : 00B8A900FCADEBAB BFA383B540FCE9ED
                                        438395EAA7
CBKE Flags           [ 1 byte array ] : 00

'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : FFFF
Install Code       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CRC                [16-bit integer] : 0xFFFF

DONE
```

8 Erasing the certificate

If the wrong device is programmed with a certificate, or it is necessary to remove this security data from the device, complete the following procedures.

To erase token data create an erase file listing the token name using `!ERASE!` as the token data. For example:

```
CA Public Key: !ERASE!
Device Implicit Cert: !ERASE!
Device Private Key: !ERASE!
$ em3xx_load.exe --cibtokenspatch sample_erase_cert.txt
em3xx_load (Version 1.0b27.1264522950)

Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108
Reset Chip

Writing to address 0x0804087E for token 'Device Implicit Cert'
Writing to address 0x080408AE for token 'CA Public Key'
Writing to address 0x080408C4 for token 'Device Private Key'
Writing to address 0x080408D9 for token 'CBKE Flags'

NOTE: Writing Custom EUI64 'FFFFFFFFFFFFFFFF' to match certificate.
Address 0x08040812.

Create image file
Install RAM image
Verify RAM image
Install Flash image
Verify Flash image
Run (by toggling nRESET)
DONE
```

9 Smart energy installation code overview

Smart Energy Installation codes are provided as a means for a device to join a ZigBee network in a reasonably secure fashion. The installation code itself is a random value printed on the joining device, which is used to encrypt the initial message exchange between it and the Energy Services Portal (ESP).

The installation code can be thought of as similar to the PIN code on Bluetooth devices when two devices are paired. The PIN code is provided as an authorization code for the parent device so that the joining device knows it is receiving information securely, such as when a hands-free headset is paired to a cellular phone.

The installation code for a Smart Energy device is printed on the outside of the device, and provided via an out-of-band mechanism to the utility along with the IEEE of the device. The utility then securely transports that information to the ESP.

The ESP and the joining device use the installation code as a shared key to establish an initial bond of trust allowing the new device to join the Smart Energy network.

10 Security use

An installation code is used to create a preconfigured link key. The installation code is transformed into a link key by using an AES hash. For more information and sample code, consult the ZigBee Smart Energy Profile Specification (ZigBee Document Number 075356r15). You can download the specification from the ZigBee Technical Documents web page at http://zigbee.org/ZigBeeSmartEnergyPublicApplicationProfile/tabid/312/Default.aspx.

The derived ZigBee link will be known only by the ESP and the joining device. The ESP uses that key to securely transport the ZigBee network key to the device. Once the device has the network key, it can communicate at the network layer with the Smart Energy network. It has the ability to perform service discovery and begin the application's initialization process.

The installation code, while not exactly a secret, cannot be easily guessed by a malicious device that hears the initial exchange between the joining device and ESP. Without knowledge of the installation code and thus the key, it cannot decrypt the messages.

The initial link key derived from the installation code does not have full access privileges on a Smart Energy network. Attempts to use it for Smart Energy messaging are not allowed and will be ignored by other Smart Energy devices. Shortly after joining a network, a device must use the Key Establishment cluster to establish a new link key with the ESP. Only when key establishment completes successfully will a device have full privileges on the network and be able send and receive certain Smart Energy messages.

11 Installation code format

The installation code is made up of a 6-, 8-, 12-, or 16-byte random, hexadecimal number with a 2-byte CRC appended to the end. As far as the user is concerned, the CRC is part of the installation code and he do not need to know that it is there or why. Therefore, from the user's point of view, the length of the install code is 8, 10, 14, or 18 bytes.

The choice of 6-, 8-, 12-, or 16-byte length for the installation code is up to the device vendor. Manufacturing and managing the list of installation codes will play a part in choosing the size, security, and user experience in installing the device. A larger installation code size will mean less of a chance of an attacker “guessing” the installation code and eavesdropping on the initial join. However smaller installation code is much easier for a user to read off the device during installation.

12 Installation code CRC

The installation code CRC is mechanism used to verify the integrity of an installation code when it is transmitted via an out-of-band mechanism to the utility. This transport mechanism involves human interaction in some way. As a result, the CRC was designed as a way to verify that an installation code is valid and was not mistakenly changed during transport.

The Smart Energy installation model enables users to install a device themselves. Users simply read the installation code on the back of the device and enter it into a webpage or provide it over the phone to a utility service. Because the number is a hexadecimal value, it is easy to transpose digits or read the incorrect value.

12.1 Validation

The ZigBee Smart Energy Profile Specification expects that the utility will perform basic checking of the installation code for validity. The utility calculates the CRC over all bytes in the installation code except the final two. It then compares the final two bytes of the installation code with the calculated CRC to see if they match. If they do not match, the user entering the installation code can be informed immediately that it does not look valid. The user should then double check the value.

The ZigBee Smart Energy Profile Specification does not require the ESP to validate the installation code. The ESP expects to receive a pre-configured link key along with the IEEE address of the new joining device. It does not need to have any knowledge about how that key was derived. It is up to the particular utility how they wish to manage and transport the link key to the ESP.

For details on how the CRC is calculated, including sample code, consult the ZigBee Smart Energy Profile Specification (ZigBee Document Number 075356r15). You can download the specification from the ZigBee Technical Documents web page at http://zigbee.org/ZigBeeSmartEnergyPublicApplicationProfile/tabid/312/Default.aspx.

12.2 Generation

It is recommended that the installation code be a random number. This reduces the chances of an attacker guessing the installation code and compromising the initial join procedure. The installation code should not be based on the manufacturing process, such as tied to the IEEE address or sequential numbering based on the manufacturing lot.

If that were the case, an attacker with knowledge about the type of device being joined would have a known range of installation codes it could try to compromise the network and clone the device's identity. An installation code does not have to be unique across all Smart Energy devices for all manufacturers.

12.3 Labels

The device's installation code should be printed on a label on the outside of the device along with its IEEE address. Both elements should be identified with text indicating what they are. The installation code should not be printed on the outside of the box because that makes it easier for an attacker to gain knowledge of the installation code and potentially compromise the device. It is recommended that the installation code be printed in 2-byte blocks (for example, 83FE D340 7A93 9738 C552).

Note: The CRC should be appended to the installation code in little endian format on the label.

Example

The following is a 10-byte installation code label (8-byte random code with a 2-byte CRC):

```
83FE D340 7A93 9738 C552
```

The random number portion of the code is the sequential bytes: 0x83, 0xFE, 0xD3, 0x40, 0x7A, 0x93, 0x97, 0x38. The calculated CRC value is 0x52C5, but it is appended in little-endian format.

13 Programming the installation code on an STM32W108

13.1 Checking the installation code on a device

Prior to modifying the certificate, it is best to verify there is connectivity with the device to be programmed, and what information is currently stored on the node. To do this, execute the following command:

```
$ em3xx_load.exe  --cibtokensprint
em3xx_load (Version 1.0b27.1264522950)
Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108

'General' token group
TOKEN_MFG_CIB_OBS          [16 byte array ] : A55AFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION   [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64    [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_STRING           [16 byte string] : "" (0 of 16 chars)
                                              FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME       [16 byte string] : "" (0 of 16 chars)
                                              FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID         [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG       [16-bit integer] : 0xFFFF
TOKEN_MFG_BOOTLOAD_AES_KEY [16 byte array ] : FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
TOKEN_MFG_EZSP_STORAGE     [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM [16-bit integer] : 0xFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CA Public Key        [22 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFF
Device Private Key   [21 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFF
CBKE Flags           [ 1 byte array ] : FF

'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : FFFF
Install Code       [16 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CRC                [16-bit integer] : 0xFFFF

DONE
```

The data stored at TOKEN_MFG_CUSTOM_EUI_64 is the custom value for the IEEE address in little-endian format. If this value is set to all Fs, then the value for the internal IEEE address is used. If this value is not all Fs, then it used instead of the internal IEEE address.

The data in the last block is the installation code metadata, the installation code, and the CRC. The installation code metadata (Flags and CRC) is automatically programmed by the em3xx_load.exe tool.

13.2 Format of the installation code file

To program the installation code, create a simple text file with the value of the installation code (without the CRC) and the EUI of the device. This file is passed into the em3xx_load.exe program.

The format of the file is as follows:

```
Install Code: <ascii-hex>
```

Here is a sample installation code file. The CRC for that code is 0x52C5.

```
Install Code: 83FED3407A939738
TOKEN_MFG_CUSTOM_EUI_64: 08090A0B0C0D0E0F
```

Per the Smart Energy specification, the installation code may be 6, 8, 12, or 16 bytes in length (not including the two-byte CRC).

13.3 Writing the installation code into the manufacturing area

To write the installation code into the manufacturing area, execute the following command:

```
$ em3xx_load.exe --cibtokenspatch install-code-file.txt
em3xx_load (Version 1.0b27.1264522950)

Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108
Reset Chip

NOTE:  Calculated install code CRC is 0x52C5.
Writing to address 0x080408DA for token 'Install Code Flags'
Writing to address 0x080408DC for token 'Install Code'
Writing to address 0x080408EC for token 'CRC'

NOTE: Writing Custom EUI64 '08090A0B0C0D0E0F'.  Address 0x08040812.

Create image file
Install RAM image
Verify RAM image
Install Flash image
Verify Flash image
Run (by toggling nRESET)
DONE
```

13.4 Verifying the stored installation code

After writing the installation code, it is best to verify the information by executing em3xx_load.exe again. Note the bold values.

```
$ em3xx_load.exe  --cibtokensprint
em3xx_load (Version 1.0b27.1264522950)

Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108

'General' token group
TOKEN_MFG_CIB_OBS          [16 byte array ] : A55AFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
TOKEN_MFG_CUSTOM_VERSION   [16-bit integer] : 0xFFFF
TOKEN_MFG_CUSTOM_EUI_64    [ 8 byte array ] : 08090A0B0C0D0E0F
TOKEN_MFG_STRING           [16 byte string] : "" (0 of 16 chars)
                                              FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_BOARD_NAME       [16 byte string] : "" (0 of 16 chars)
                                              FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
TOKEN_MFG_MANUF_ID         [16-bit integer] : 0xFFFF
TOKEN_MFG_PHY_CONFIG       [16-bit integer] : 0xFFFF
TOKEN_MFG_BOOTLOAD_AES_KEY [16 byte array ] : FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
TOKEN_MFG_EZSP_STORAGE     [ 8 byte array ] : FFFFFFFFFFFFFFFF
TOKEN_MFG_OSC24M_BIAS_TRIM [16-bit integer] : 0xFFFF

'Smart Energy CBKE (TOKEN_MFG_CBKE_DATA)' token group
Device Implicit Cert [48 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
CA Public Key        [22 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFFFF
Device Private Key   [21 byte array ] : FFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF
                                        FFFFFFFFFF
CBKE Flags           [ 1 byte array ] : FF

'Smart Energy Install Code (TOKEN_MFG_INSTALLATION_CODE)' token group
Install Code Flags [ 2 byte array ] : 0200
Install Code       [16 byte array ] : 83FED3407A939738 FFFFFFFFFFFFFFFF
CRC                [16-bit integer] : 0x52C5

DONE
```

When the installation code is less than 16 bytes (not including the CRC), the rest of the words will be all Fs. The final two bytes are the CRC displayed in big endian format. On the chip, it will be stored in the processor's native endianness (little-endian on the STM32W108).

14 Erasing the installation code

If the wrong device is programmed with a certificate, or you want to remove this security data from the device, complete the following steps.

14.1 Erase process

To erase token data create an erase file listing the token name using `!ERASE!` as the token data. For example:

```
Install Code: !ERASE!
TOKEN_MFG_CUSTOM_EUI_64: !ERASE!
```

This is the command-line:

```
$ em3xx_load.exe --cibtokenspatch install-code-file-erase.txt
em3xx_load (Version 1.0b27.1264522950)

Connecting to ISA via USB Device 0
DLL version 1.1.9, compiled Jan 29 2010 18:36:00
SerialWire interface selected
SWJCLK speed is 500kHz
Targeting STM32W108
Reset Chip

Writing to address 0x08040812 for token 'TOKEN_MFG_CUSTOM_EUI_64'
Writing to address 0x080408DA for token 'Install Code Flags'
Writing to address 0x080408DC for token 'Install Code'
Writing to address 0x080408EC for token 'CRC'

Create image file
Install RAM image
Verify RAM image
Install Flash image
Verify Flash image
Run (by toggling nRESET)
DONE
```

15 After reading this document

If you have questions or require assistance with the procedures described in this document, go to the STMicroelectronics web site at http://www.st.com/mcu, STM32W section

16 Revision history

Table 2. Document revision history

Date	Revision	Changes
28-May-2010	1	Initial release.