

# Setting Manufacturing Certificates and Installation Codes

For the EM250 SoC Platform and EM260 Co-Processor

## Contents

|  |    |
|--|----|
| Background.....  | 2  |
| Overriding the Default EUI64.....                              | 2  |
| Verifying the software version .....                           | 2  |
| Obtaining Certificates.....                                    | 3  |
| Format of the certificate file.....                            | 3  |
| Programming the Certificates into a Device .....               | 3  |
| Checking the node information .....                            | 3  |
| Writing the certificate into the manufacturing area .....      | 5  |
| Verifying the stored certificate .....                         | 5  |
| Erasing the Certificate .....                                  | 6  |
| Running em2xx_patch .....                                      | 7  |
| Creating Installation Codes.....                               | 8  |
| Format of the installation code file .....                     | 8  |
| Calculating the installation code CRC.....                     | 8  |
| Programming the Installation Code on a Device.....             | 8  |
| Checking the installation code on a device .....               | 8  |
| Writing the installation code into the manufacturing area..... | 10 |
| Verifying the stored installation code.....                    | 10 |
| Erasing the installation code .....                            | 11 |



## Background

The EM250 SoC platform and EM260 co-processor can both store application-specific manufacturing data into an area of flash that can be used to store unique device information. This information is read-only at run-time and must be programmed via the SIF interface.

The Smart Energy Profile requires the use of ECC-implicit certificates and private keys to verify a device's identity through the Key Establishment Cluster, and enable it to sign Report Event Status messages for the Demand Response Load Control server cluster. Ember has provided a set of test certificates that are hard-coded in the source code file *ami-key-establishment-test-certs.c* that allow developers to perform Smart Energy security operations during development.

**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.

## Overriding the Default EUI64

By default, an Ember EUI64 address is pre-programmed into all EM2xx chips. Customers may override this by writing their own EUI64 into the Custom EUI64 area of the manufacturing flash.

When programming certificates and installation codes the `em2xx_patch` tool allows you to override the Ember EUI64. When this is done, the `-Override` option must be passed to the `em2xx_patch` tool. 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 Ember EUI64s, and those EUI64s *must* match what is already programmed on the chip. You cannot override the EUI64 of the device with another Ember 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, the device must be connected via the SIF debug cable to an Ember Insight Adapter. The SIF tools provided with the xIDE 2.0 installation for EM250 users, or the 3.3 stack install for EM260 users, provide the basic elements needed to do this. The tools are `em2xx_read.exe` and `em2xx_patch.exe`.

### Verifying the software version

To verify that you have the correct version of the tools, open a command prompt, and change to the directory where the tools are installed. The default for the xIDE 2.0 Installation this is `C:\Program Files\Ember\xIDE_EM250\SIF\bin\`. For the EM260, the default location is `C:\Ember\EmberZNet3.3.0-em260\tool\em2xx_load\`.

To verify the software version, run the tool without any arguments. You should see the following output:

```
C:\Program Files\Ember\xIDE_EM250\SIF\bin> em2xx_patch.exe
em2xx_patch.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
```

If the software version of the tool is at least 3.0 build 1, then it will support programming certificates and installation codes. If you have an older version, download the latest copy of xIDE from the Ember Support Site.

## Obtaining Certificates

To get started, you must obtain certificates from the Root Certificate Authority. Certicom manages this authority for the ZigBee Smart Energy profile. Contact Certicom directly or visit <http://ecommerce.certicom.com/index.php/gencertregister> to obtain certificates.

### Format of the certificate file

The format of the certificate file obtained from Certicom must be as follows:

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

If this is not the case, then you should create a copy of the certificate file and change it to match this format. The tool provided by Ember expects the data to be in this format for programming certificates.

## Programming the Certificates into a Device

### 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:

```
em2xx_read -sid <host | ip-address> -Mfg 0000-016f -Run
```

You should see something similar to the following. Note the **bold values**.

```
C:\>em2xx_read -sid 172.16.0.8 -Mfg 0000-016f -Run
em2xx_patch.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
Created Ethernet Pod 10 Slave 10 (TCP:172.16.0.8:0)
SIF Slave id change from 1 to 10 (pod 10) succeeded
XAP2= DbgEna performed
XAP2= Stop performed
XAP2= DbgEna performed
XAP2= Reset performed
Manuf:
###--> addrHi=16f addrLo=0
#0000: 55AA.55AA.5136.08B2.0006.1F2A.FFFF.FFFF U.U.Q6.....*....
```

Word 0x000A:  
Internal IEEE  
Address

Word 0x0041:  
Custom IEEE  
Address

Word 0x0094:  
Certificate,  
CA Public Key  
Private Key  
Metadata

```
#0008: 01FE.0000.796C.0600.006F.0D00.0003.FFFF ... .yl...o.....
#0010: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0018: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0020: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0028: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0030: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0038: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0040: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0048: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0050: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0058: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0060: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0068: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0070: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0078: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0080: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0088: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0090: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0098: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00a0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00a8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00b0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00b8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00c0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00c8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00d0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00d8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00e0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00e8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00f0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00f8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0100: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0108: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0110: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0118: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0120: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0128: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0130: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0138: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0140: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0148: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0150: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0158: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0160: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0168: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
XAP2= Stop performed
XAP2= Reset performed
XAP2= Issuing SW_RESET
```

The data stored at WORD 0x000A is the device’s internal IEEE address, in little-endian format. You should verify that the certificate you received is for that address.

The data stored at WORD 0x0041 is the custom value for the IEEE address, in little-endian format. If this value is set to all F’s, then the value for the internal IEEE address is used. If this value is not all F’s, then it used instead of the internal IEEE address.

The data stored at WORD 0x0094 is the security data (certificate, private key, root CA public key, metadata). If this is set to all F's then the data is not valid and will not be used by the software.

### Writing the certificate into the manufacturing area

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

```
em2xx_patch -Mfg -sid <host | ip-address> -loadCert <certificate-file> -Run
```

For example, you should see something similar to the following:

```
C:\>em2xx_patch.exe -Mfg -sid 172.16.0.8 -loadCert 0000000000000001.txt -Run
em2xx_patch.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
Device Implicit Cert: 0304 5FDF C8D8 5FFB 8B39 93CB 72DD CAA5 5F00 B3E8 7D6D 000
0 0000 0000 0001 5445 5354 5345 4341 0109 0006 0000 0000 0000
CA Public key: 0200 FDE8 A7F3 D108 4224 962A 4E7C 54E6 9AC3 F04D A6B8
Device Private key: 00B8 A900 FCAD EBAB BFA3 83B5 40FC E9ED 4383 95EA A7
Setting Custom IEEE Address from Cert: (<)0100000000000000
SIF Slave id change from 1 to 10 (pod 10) succeeded
XAP2= DbgEna performed
XAP2= Stop performed
XAP2= DbgEna performed
XAP2= Reset performed
Programming FLASH...
XAP2= Run performed
MFGcu 0041-0044 (0004)
MFGcu 0094-00c1 (002e)
Total Manuf Used=0032 words
XAP2= Stop performed
Verifying FLASH...

****Application load: SUCCESS****
****Verification:      SUCCESS****

XAP2= Stop performed
XAP2= Reset performed
XAP2= Issuing SW_RESET
```

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.

### Verifying the stored certificate

After writing the certificate, it is best to verify the information by executing `em2xx_read` again. Note the **bold** values.

```
C:\>em2xx_read -sid 172.16.0.8 -Mfg 0000-016f -Run
em2xx_patch.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
SIF Slave id change from 1 to 10 (pod 10) succeeded
XAP2= DbgEna performed
XAP2= Stop performed
XAP2= DbgEna performed
XAP2= Reset performed
Manuf:
###--> addrHi=16f addrLo=0
```

```

#0000: 55AA.55AA.5136.08B2.0006.1F2A.FFFF.FFFF U.U.Q6.....*....
#0008: 01FE.0000.796C.0600.006F.0D00.0003.FFFF ...y1...o.....
#0010: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0018: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0020: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0028: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0030: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0038: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0040: FFFF.0100.0000.0000.0000.FFFF.FFFF.FFFF .....
#0048: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0050: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0058: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0060: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0068: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0070: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0078: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0080: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0088: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0090: FFFF.FFFF.FFFF.FFFF.0304.5FDF.C8D8.5FFB ....._..._.
#0098: 8B39.93CB.72DD.CAA5.5F00.B3E8.7D6D.0000 .9..r..._...}m..
#00a0: 0000.0000.0001.5445.5354.5345.4341.0109 .....TESTSECA..
#00a8: 0006.0000.0000.0000.0200.FDE8.A7F3.D108 .....
#00b0: 4224.962A.4E7C.54E6.9AC3.F04D.A6B8.00B8 B$.*N|T...M....
#00b8: A900.FCAD.EBAB.BFA3.83B5.40FC.E9ED.4383 .....@...C.
#00c0: 95EA.A700.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00c8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00d0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00d8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00e0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00e8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00f0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00f8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0100: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0108: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0110: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0118: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0120: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0128: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0130: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0138: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0140: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0148: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0150: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0158: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0160: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0168: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
XAP2= Stop performed
XAP2= Reset performed
XAP2= Issuing SW_RESET

```

In this case, the device has been set with the data in test certificate #1 (associated with IEEE address (->)0000000000000001).

## 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.

**Warning:** Incorrectly executing these steps can damage the device and prevent it from functioning.

### Running em2xx\_patch

In order to wipe the existing certificate data it is necessary to specify an additional option of **-Override** to `em2xx_patch`. Instead of specifying a real filename for a certificate, the special keyword `null` is used. The following is the command-line:

```
em2xx_patch -Mfg -Override -sid <host | ip-address> -loadCert null -Run
```

```
C:\>em2xx_patch.exe -Mfg -Override -sid 172.16.0.8 -loadCert null -Run
em2xx_patch.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
Device Implicit Cert: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFF
F FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
CA Public key: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
Device Private key: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FF
Setting Custom IEEE Address from Cert: (<)ffffffffffffffff
SIF Slave id change from 1 to 10 (pod 10) succeeded
XAP2= DbgEna performed
XAP2= Stop performed
XAP2= DbgEna performed
XAP2= Reset performed
Programming FLASH...
XAP2= Run performed
MFGcu 0041-0044 (0004)
MFGcu 0094-00c1 (002e)
Total Manuf Used=0032 words
XAP2= Stop performed
Verifying FLASH...

****Application load: SUCCESS****
****Verification:      SUCCESS****

XAP2= Stop performed
XAP2= Reset performed
XAP2= Issuing SW_RESET
```

## Creating Installation Codes

The Smart Energy group has no formal requirements for installation code. It has only a set of best practices regarding how to use them.

Installation codes do not have to be unique across all devices made by all manufacturers. However, it is recommended that they be sufficiently random to prevent attackers from trying to guess the link key based on known patterns in installation codes.

### Format of the installation code file

The installation code must be specified in a file whose name will be passed into the em2xx\_patch program. The format of the file is as follows:

```
EUI: <big-endian-ascii-hex>
Install Code: <ascii-hex>
CRC: <big-endian-ascii-hex>
```

Here is a sample installation code file:

```
EUI: 0000000000000001
Install Code: 000102030405060708090a0b0c0d0e0f
CRC: e913
```

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

### Calculating the installation code CRC

The Smart Energy Installation code uses a CCITT CRC standard polynomial  $X^{16} + X^{12} + X^5 + 1$ . Sample code that calculates the CRC may be found in Zigbee Alliance Document **SE Profile R14 Errata**, document number 084914r04.

## Programming the Installation Code on a Device

### 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:

```
em2xx_read -sid <host | ip-address> -Mfg 0000-016f -Run
```

You should see something similar to the following. Note the **bold** values.

```
C:\>em2xx_read.exe -sid 172.16.0.6 -Mfg 0000-016f -Run
em2xx_read.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
SIF Slave id change from 1 to 0 (pod 1) succeeded
XAP2= DbgEna performed
XAP2= Stop performed
XAP2= DbgEna performed
XAP2= Reset performed
Manuf:
###--> addrHi=16f addrLo=0
#0000: 55AA.55AA.5136.08B2.0003.0129.FFFF.FFFF U.U.Q6.....)....
#0008: 01FE.0000.15B0.0500.006F.0D00.0003.FFFF .....o.....
#0010: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0018: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....

```

Word 0x000A:  
Internal IEEE  
Address

Word 0x0041:  
Custom IEEE  
Address

```

#0020: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0028: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0030: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0038: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0040: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0048: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0050: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0058: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0060: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0068: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0070: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0078: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0080: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0088: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0090: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0098: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00a0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00a8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00b0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00b8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00c0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00c8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00d0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00d8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00e0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00e8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00f0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00f8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0100: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0108: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0110: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0118: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0120: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0128: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0130: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0138: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0140: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0148: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0150: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0158: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0160: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0168: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
XAP2= Stop performed
XAP2= Reset performed
XAP2= Issuing SW_RESET

```

Word 0x00C2:  
Install Code  
Metadata:  
(2 bytes),  
install code  
(16 bytes),  
and CRC  
(2 bytes)

The data at word 0x00C2 is the installation code metadata, the install code, and the CRC. The install code metadata is automatically programmed by the em2xx\_patch tool. You calculate the CRC and specify it using the installation code file.

## Writing the installation code into the manufacturing area

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

```
em2xx_patch -Mfg -sid <host | ip-address> -loadInst <install-code-file> -Run
```

**Note:** If the certificate contains a non-Ember EUI, then the **-Override** option must be passed to the command.

For example, you should see something similar to the following:

```
C:>\em2xx_patch.exe -sid 172.16.0.6 -Mfg -loadInst test-install-code.txt -
Run -Override
em2xx_patch.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
Non-ember EUI64 found in file
Install Code: 0001 0203 0405 0607 0809 0A0B 0C0D 0E0F
CRC: E913
EUI: 0000 0000 0000 0001
Writing Custom EUI64 Address: 0000000000000001
SIF Slave id change from 1 to 0 (pod 1) succeeded
XAP2= DbgEna performed
XAP2= Stop performed
XAP2= DbgEna performed
XAP2= Reset performed
EUI was changed, checking for the current value...
Current custom EUI on the chip: FFFFFFFFFFFFFFFF
New custom EUI to be written: 0000000000000001
Programming FLASH...
XAP2= Run performed
MFGcu 0041-0044 (0004)
MFGcu 00c2-00cb (000a)
Total Manuf Used=000e words
XAP2= Stop performed
Verifying FLASH...

***Application load: SUCCESS***
***Verification:      SUCCESS***

XAP2= Stop performed
XAP2= Reset performed
XAP2= Issuing SW_RESET
```

## Verifying the stored installation code

After writing the installation code, it is best to verify the information by executing `em2xx_read` again. Note the **bold** values.

```
C:\>em2xx_read.exe -sid 172.16.0.6 -Mfg 0000-016f -Run
em2xx_read.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
SIF Slave id change from 1 to 0 (pod 1) succeeded
XAP2= DbgEna performed
XAP2= Stop performed
XAP2= DbgEna performed
XAP2= Reset performed
Manuf:
```

```

###--> addrHi=16f addrLo=0
#0000: 55AA.55AA.5136.08B2.0003.0129.FFFF.FFFF U.U.Q6.....)....
#0008: 01FE.0000.15B0.0500.006F.0D00.0003.FFFF .....o.....
#0010: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0018: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0020: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0028: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0030: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0038: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0040: FFFF.0100.0000.0000.0000.FFFF.FFFF.FFFF .....
#0048: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0050: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0058: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0060: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0068: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0070: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0078: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0080: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0088: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0090: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0098: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00a0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00a8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00b0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00b8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00c0: FFFF.FFFF.0006.0001.0203.0405.0607.0809 .....
#00c8: 0A0B.0C0D.0E0F.E913.FFFF.FFFF.FFFF.FFFF .....
#00d0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00d8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00e0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00e8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00f0: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#00f8: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0100: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0108: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0110: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0118: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0120: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0128: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0130: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0138: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0140: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0148: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0150: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0158: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0160: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
#0168: FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF.FFFF .....
XAP2= Stop performed
XAP2= Reset performed
XAP2= Issuing SW_RESET

```

### 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.

**Warning:** Incorrectly executing these steps can damage the device and prevent it from functioning.

### Running the em2xx\_patch

To wipe the existing certificate data, execute the `-Override` option with `em2xx_patch`. Instead of specifying a real filename for an installation code, use the special keyword `null`. This is the command-line:

```
em2xx_patch -Mfg -Override -sid <host|ip-address> -loadInst null -Run
```

```
C:\>em2xx_patch.exe -sid 172.16.0.6 -Mfg -loadInst null -Run -Override
em2xx_patch.exe Version 3.0 Build 1
Emsif Library Version 1.0.6
Writing Custom EUI64 Address: FFFFFFFFFFFFFFFF
SIF Slave id change from 1 to 0 (pod 1) succeeded
  XAP2= DbgEna performed
  XAP2= Stop performed
  XAP2= DbgEna performed
  XAP2= Reset performed
Programming FLASH...
  XAP2= Run performed
MFGcu 0041-0044 (0004)
MFGcu 00c2-00cb (000a)
Total Manuf Used=000e words
  XAP2= Stop performed
Verifying FLASH...

****Application load: SUCCESS****
****Verification:      SUCCESS****

  XAP2= Stop performed
  XAP2= Reset performed
  XAP2= Issuing SW_RESET
```

### After reading this document

If you have questions or require assistance with the procedures described in this document, please contact an Ember support representative at [support@ember.com](mailto:support@ember.com).

Copyright © 2008 by Ember Corporation

All rights reserved.

The information in this document is subject to change without notice. The statements, configurations, technical data, and recommendations in this document are believed to be accurate and reliable but are presented without express or implied warranty. Users must take full responsibility for their applications of any products specified in this document. The information in this document is the property of Ember Corporation.

Title, ownership, and all rights in copyrights, patents, trademarks, trade secrets, and other intellectual property rights in the Ember Proprietary Products and any copy, portion, or modification thereof, shall not transfer to Purchaser or its customers and shall remain in Ember and its licensors.

No source code rights are granted to Purchaser or its customers with respect to all Ember Application Software. Purchaser agrees not to copy, modify, alter, translate, decompile, disassemble, or reverse engineer the Ember Hardware (including without limitation any embedded software) or attempt to disable any security devices or codes incorporated in the Ember Hardware. Purchaser shall not alter, remove, or obscure any printed or displayed legal notices contained on or in the Ember Hardware.

Ember, Ember Enabled, EmberZNet, InSight, and the Ember logo are trademarks of Ember Corporation.

All other trademarks are the property of their respective holders.

