Instructions for using Image-Builder

1 Overview

Image-builder is Ember's tool for creating ZigBee Over-the-air (OTA) bootloader files. It takes an existing file (or multiple files) and wraps them in the file format as declared in the Zigbee specification. The files it wraps are normally bootloader files (such as EBL).

There are two version of the tool, one with ECC and one without ECC. Due to U.S. Export regulations the version with ECC may not be distributed to certain countries. The version without ECC will be distributed with the normal stack releases while the version with ECC will only be distributed through the Ember Portal.

The non-ECC version of the tool can do everything as the ECC version, except for create and verify ECDSA signatures.

For more information about ZigBee Over-the-air please see the **Over-the-air Plugins** section in the **Application Framework v2 Developer's Guide** provided with the Ember stack release.

1.1 Compatibility

As of this writing, the image-builder tool only works on 32-bit operating systems built for an x86 processor. In addition, it has only been specifically tested on Windows XP, and on Linux 2.6 kernels.

If running on Windows 7 or Vista, it is advised to run the program in Windows XP compatibility mode.

2 Help

Running the tool without arguments will give you the program's CLI syntax.

```
bash$ ./image-builder-ecc
image-builder (C) 2010 by Ember Corporation.
Version: 1.0
ECC signature support present.

Unknown usage error.

Usage: image-builder <operation> [ <additional arguments> ]

Print operation: Arguments for printing OTA header information
  -p, --print=<filename|directory> print either (1) the OTA header of the specified file, or (2)
all OTA files found in the directory.

Create operation: Arguments for creating OTA files
  -c, --create=<filename>        create OTA file
  -v, --version=<4-bytes-hex>    firmware version
  -m, --manuf-id=<2-bytes-hex>   manufacturer ID
```

```
  -i, --image-type=<2-bytes-hex> image type ID
  -s, --stack-version=<2-bytes>  zigbee stack version (optional)
  --string=<text>                header string text (optional)
  --min-hw-ver=<2-bytes-hex>     Minimum hardware version (optional)
  --max-hw-ver=<2-bytes-hex>     Maximum hardware version (optional)
  --upgrade-dest=<8-bytes-hex>   The EUI64 of the device the file is intended for (optional)
  -t, --tag-id=<2-bytes-hex>     tag identifier
  -l, --tag-length=<32-bit length> length of dummy data for tag
  -f, --tag-file=<filepath>      file to include as data for associated tag
  --security-credentials=<1-byte-hex> The security credentials required for this upgrade.
  --test-sign                    Sign the image with the built-in security key
  --sign=<filename>              Sign using certificate and private key from file (or stdin)

Help operation
  -h, --help                     display this usage and exit
  --help-signing                 Print syntax for signing certificate files
```

3 Displaying OTA Files

You can see the contents of OTA files by using the **–p** command-line option. Ember provides a sample file in the App. Framework release. Its location is: **app/framework/plugin/ota-storage-simple-ram/ota-static-sample.ota**.

```
bash$ ./image-builder-ecc -p app/framework/plugin/ota-storage-simple-ram/ota-static-sample.ota
image-builder (C) 2010 by Ember Corporation.
Version: 1.0
ECC signature support present.

File: app/framework/plugin/ota-storage-simple-ram/ota-static-sample.ota
Magic Number:         0x0BEEF11E
Header Version:       0x0100
Header Length:        56 bytes
Field Control:        0x0000
Manufacturer ID:      0x1002
Image Type:           0x5678
Firmware Version:     0x00000005
Stack Version:        0x0002
Header String:        The latest and greatest upgrade.
Total Image Size:     172 bytes
Total Tags:           3
  ID:                 0x0000   (Upgrade Image)
  Length:             0 bytes
  ID:                 0x0002   (ECDSA Signing Certificate)
  Length:             48 bytes
    Subject: (>)000D6F0000198B36
    Issuer:  (>)5445535453454341 (Certicom TEST CA)
  ID:                 0x0001   (ECDSA Signature)
  Length:             50 bytes
    Signer:  (>)000D6F0000198B36
     Data:      03C82682E7F7FCF0  0033C6D44E7ACA78
                AAA7E11997034FA6  59C4EEC134ACF40F
                32C58994F4714082  493D

Using Certicom TEST CA issued certificate.

Message Digest: AC9DC23C77A1A7911EA8075292F7B53C

Signature is valid
```

4 Creating an OTA File

To create an OTA file you need to first decide the parameters of the firmware image that will be embedded in the OTA file.

The basic build parameters are **Manufacturer ID, Image Type Id,** and **Version**. Together these form an identifier to uniquely denote the image for all OTA servers that serve up the file to clients.

4.1 Manufacturer ID

The manufacturer ID is the ZigBee assigned value that is unique to the manufacturer. For example, Ember's manufacturer ID is 0x1002. Manufacturers must contact the ZigBee Alliance to obtain their own manufacturer ID.

4.2 Image Type ID

The image type ID is an identifier to indicate the specific product group that the update applies to. For example, if your company manufactured light switches and thermostats which required different flavors of software you would use different image type IDs to indicate which product the upgrade file is for.

A few values have been specified by ZigBee, but the rest are open to each manufacturer's interpretation (0x0000-0xFFBF). The manufacturer specific values for the image type are unique to each manufacturer ID. Therefore different vendors may use the same image type ID within the manufacturer specific range without a conflict.

Image Type ID	Description
0x0000 – 0xffbf	Manufacturer Specific
0xffc0	Security credential
0xffc1	Configuration
0xffc2	Log
0xffc3 – 0xfffe	Reserved (unassigned)
0xffff	Reserved: wild card

4.3 Version

The Version is the software version that is associated with the file or files that are wrapped in the OTA package. This version is used by the OTA cluster to query when a new software update is ready. In general it is recommended that this be an increasing number for each newer software version. However a developer is free to use whatever scheme they would like. For example the Ember version number is encoded as hexadecimal digits, so version 4.2.0 build 1 would be 0x00004201.

4.4 Tags

Tags are blobs of data inside the OTA file that are interpreted by the device receiving and processing the update. A few tags are defined and global to all devices, such as the **ECDSA Signature Tag** and the **ECDSA Signer Certificate tag**. However tags can be manufacturer specific and contain information only pertinent to

certain devices that know how to handle them. Each tag is labeled with an identification number. The list of identification numbers is specified by the OTA cluster.

Tag Identifiers

0x0000	Upgrade Image
0x0001	ECDSA Signature
0x0002	ECDSA Signing Certificate
0x0003 – 0xefff	Reserved by ZigBee
0xf000 – 0xffff	Manufacturer Specific Use

An OTA file should have at least one tag containing data in it to be of use to the client receiving the file. There is no limit to the maximum number of tags in the file. The main upgrade file (such as the EBL file) is normally specified as tag 0x0000 (upgrade file), but this is not required by the specification. Ember's OTA cluster code will look for this tag when passing this data to the bootloader of the device.

For the ZigBee Over-the-air bootload cluster certification tests a NULL upgrade file is used for many of the tests. This is a small file that contains no upgrade code and will not be passed to the bootloader. However the OTA cluster code will download and process it. These files should be specified with a tag OTHER than 0x0000 (upgrade image) inside of them. They may in fact contain no tags and the Ember OTA client code will not process them.

4.5 String

The OTA header string is a human readable string that indicates what the upgrade file is and who it is for. This string is not used by the OTA cluster code and it is only for display purposes.

4.6 Signing

The ZigBee Smart Energy Profile requires that OTA files be signed by the manufacturer. Downloaded files must be validated by the OTA client prior to installation. When images are signed the signer's certificate is included automatically as a tag in the file, and a signature tag is added as the last tag in the file.

There are two ways to sign a file, using a built-in test certificate and using a user supplied certificate.

4.6.1 Built-in Test Certificate

The tool contains a built-in test certificate that can be used to sign images with the **--test-sign** option. This certificate is not intended for production use but is provided as a sample to allow signing and verification to be done in development environments. This built-in certificate used by the tool will automatically be accepted by the default App. Builder configuration for the plugin **OTA Client**.

For generation of production images to be shipped to deployed, production devices it is highly recommended that manufacturers do NOT use the built-in test certificate. See "User Supplied Certificates" below.

Handling of production signer certificates must be done differently than test certificates. For more information on this topic please see the **Over-the-air Plugins** section in the **Application Framework v2 Developer's Guide** provided with the Ember stack release.

4.6.1.1 Example using Test certificates

The following is an example of creating an OTA file from an Ember 260 NCP image and signing using the built-in test certificate.

```
bash$  ./image-builder-ecc --create test.ota --manuf-id 0x1002 --image-type 0x5678 --version
0x00000005 --string "em260 uart fifo ecc tokens" --tag-id 0x0000 --tag-file em260-uart-fifo-ecc-
tokens.ebl --test-sign
image-builder (C) 2010 by Ember Corporation.
Version: 1.0
ECC signature support present.

WARNING: Using internal test key and certificate.
WARNING: Using weak random number generator.  Use '--sign' for more secure generator.

File: test.ota
Magic Number:         0x0BEEF11E
Header Version:       0x0100
Header Length:        56 bytes
Field Control:        0x0000
Manufacturer ID:      0x1002
Image Type:           0x5678
Firmware Version:     0x00000005
Stack Version:        0x0002
Header String:        em260 uart fifo ecc tokens
Total Image Size:     112044 bytes
Total Tags:           3
  ID:                 0x0000   (Upgrade Image)
  Length:             111872 bytes
  ID:                 0x0002   (ECDSA Signing Certificate)
  Length:             48 bytes
    Subject: (>)000D6F0000198B36
    Issuer:  (>)5445535453454341 (Certicom TEST CA)
  ID:                 0x0001   (ECDSA Signature)
  Length:             50 bytes
    Signer:  (>)000D6F0000198B36
     Data:      03ECE34CF26E86E3  7EE3A313A7D9D2ED
                3C01FA1230029AF9  8523BD2759CB1E7F
                41D045109AAABDDB  A338

Using Certicom TEST CA issued certificate.

Message Digest: 77CA8B210A9AAE225CCA7736AFABE0A8

Signature is valid
```

4.6.2 User Supplied Certificates

The tool also allows users to supply their own certificates to sign their OTA files with the **--sign <filename>** option. The certificates may either be test certificates issued from the Certicom Test CA, or production certificates issued from the Certicom Production CA.

For generation of production images to be shipped to deployed, production devices it is highly recommended that manufacturers use their own certificates issued from the Certicom Production CA to sign images.

4.6.2.1 Format

The user supplied certificates may either be supplied from a text file on disk, or via the command-line STDIN. To provide the data using STDIN, use the **--sign** option with the special filename **stdin**. The program will read data from STDIN, expecting the same format as a file on disk. The format for both the file on disk and STDIN is as follows:

```
# Comment lines begin with a '#' and are ignored
Certificate: <48-byte hexadecimal array>
Private Key: <21-byte hexadecimal array>
```

Alternatively, the certificate files supported by **em2xx_patch** and **em3xx_load** are also accepted.

```
Device Implicit Cert: <48-byte hex array>
Device Private Key: <21-byte hex array>
# These two parameters are ignored, as they are not used by the image-builder tool.
CA Public Key: <22-byte hex array>
Device Public key: <22-byte hex array>
```

Note: Array data must be specified on the same continuous line, with a carriage return only at the end.

The tool will read in the data from the text file (or stdin), parse the values to make sure they are formatted correctly, and then use the data to sign the image. Below is an example of a text file with a user supplied certificate.

```
Certificate:0307834a5cfa185ee9c9550a6561212af7082ef6d319000d6f000092e04e544553545345434100000000
000000000000
Private Key:02032ed11b3ceeddae99ce00e23bc54564d16b18ea
```

4.6.2.2 Example using User Supplied Certificates

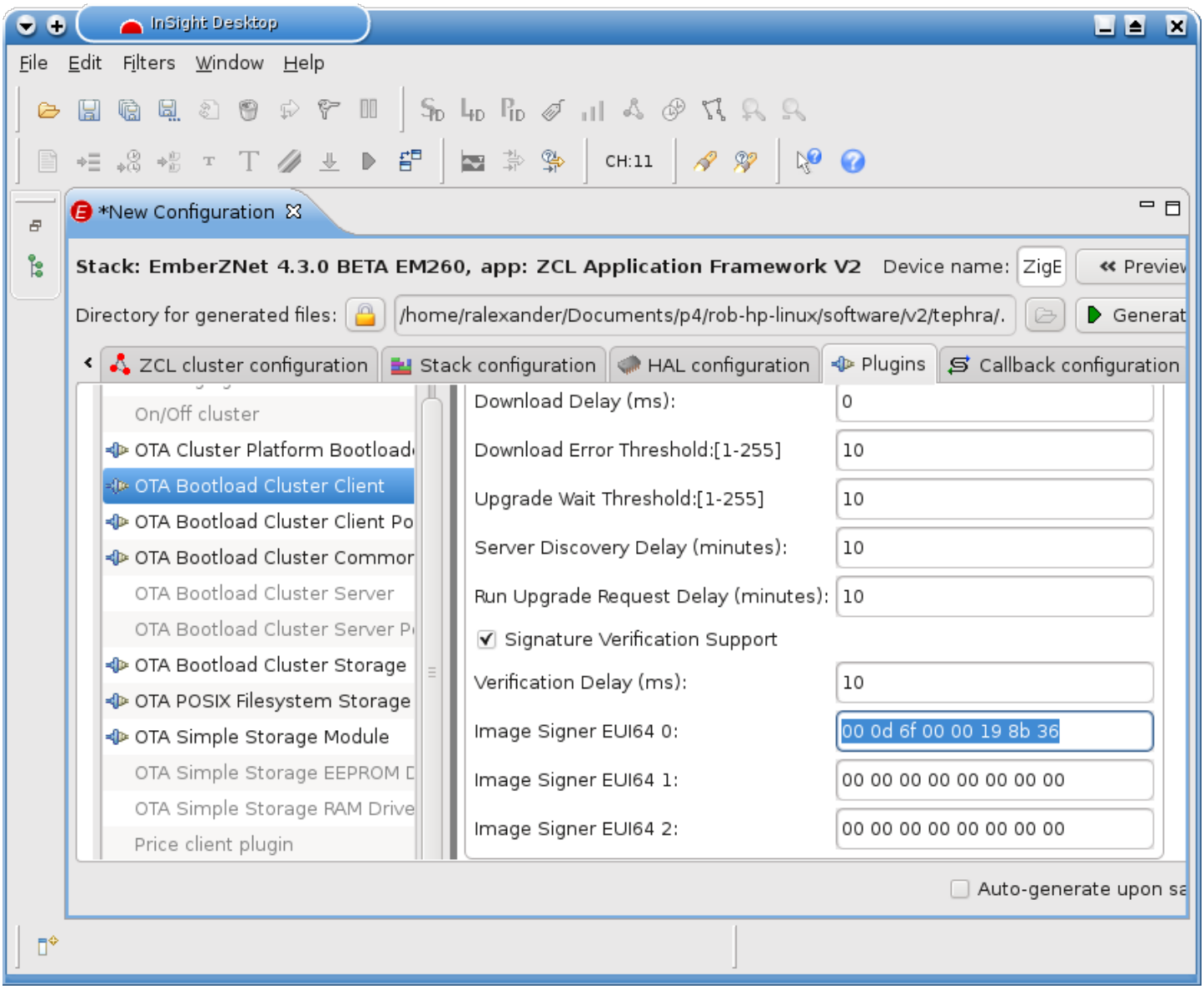
The following is an example of creating an OTA file from an Ember 260 NCP image and signing using a user supplied certificate.

```
$ ./image-builder-ecc --create test.ota --manuf-id 0x1002 --image-type 0x5678 --version 0x00000005
--string "em260 uart fifo ecc tokens" --tag-id 0x0000 --tag-file em260-uart-fifo-ecc-tokens.ebl
--sign user-supplied-cert.txt
image-builder (C) 2010 by Ember Corporation.
Version: 1.0
ECC signature support present.

Using user supplied key and certificate.

Using Certicom TEST CA issued certificate.
Using /dev/random for random number generation
Gathing sufficient entropy... (may take up to a minute)...

File: test.ota
Magic Number:         0x0BEEF11E
```

```
Header Version:       0x0100
Header Length:        56 bytes
Field Control:        0x0000
Manufacturer ID:      0x1002
Image Type:           0x5678
Firmware Version:     0x00000005
Stack Version:        0x0002
Header String:        em260 uart fifo ecc tokens
Total Image Size:     112044 bytes
Total Tags:           3
  ID:                 0x0000   (Upgrade Image)
  Length:             111872 bytes
  ID:                 0x0002   (ECDSA Signing Certificate)
  Length:             48 bytes
    Subject: (>)000D6F000092E04E
    Issuer:  (>)5445535453454341 (Certicom TEST CA)
  ID:                 0x0001   (ECDSA Signature)
  Length:             50 bytes
    Signer:  (>)000D6F000092E04E
     Data:      03E9F2CE826C4CB9  B734BBA2933F1F27
                3485DDE831030C78  F214589E1C46EB9D
                7C50C3FFA300D4C1  8C40

Using Certicom TEST CA issued certificate.

Message Digest: 040797A3C2BC13BC04D1B5C661E4D877

Signature is valid
```

4.6.3 Appbuilder Integration

In order for the device to accept an image signed using a User Supplied certificate, the EUI64 of that signer must be specified to the OTA Client Plugin. On the App Builder Plugin Tab, select the OTA Client plugin. Under the Options, change the **Image Signer EUI64 0** value to the EUI64 of the primary signer. If other EUI64s are used for signing, then change **Image Signer EUI64 1** and **Image Signer EUI64 2** to reflect the values of the other two signers' EUI64s.



By default, App. Builder uses the EUI64 of the built-in test certificate for Image Signer EUI64 0.