Secure Boot Tool Components - User Guide

UG98S08

Maxim Integrated

2020-10-20

1 Introduction

Secure Boot Tool (SBT) is a software tool set containing command-line applications such as device supporting, SCP session building, signing binary file and managing key/program load with source codes that helps the customers to develop their own boot/load tools for production if needed. It also includes Secure Communication Protocol (SCP) commands with Host Security Module (HSM) parameters to support PCI-PTS-compliant CRK handling and keys protection.

SBT includes mainly there binaries, these are:

1. `sign_app`
2. `build_scp_session`
3. `send_scp`

this document explain usage of these binaries and explain parameters of them.
Below there is one section for each of these binaries.

2 Sign App

sign_app build binary file including signature and SLA header from customer application binary file.

Usage

```
sign_app [OPTION] [PARAMETERS] [APP [KEYFILE]]
```

2.1 General Options

Option	Parameter	Example	Explanation
`Select Chip`	-c CHIP_NAME	sign_app -c MAX32520 …	Define chip that be signed
`Help`	-h	sign_app -h	Print help output
`Version`	-v	sign_app -v	Print version of binary
`Debug`	-d	sign_app -d	Enable debug output

2.2 Parameters

Parameters are used by priority in the following order :

1. Command line
2. Configuration file " INIFILE " in the current folder
3. Chip default parameters selected by the -c option or the MAXIM_SBT_DEVICE env variable.
4. Software default values.

2.2.1 Signing algorithm

```
algo=algo
```

`algo` - Algorithm to be used to sign the application Please refers to CHIP documentation to select the corect one. Available algorithms are :

- `rsa`
- `rsa_paola`
- `none`
- `ecdsa`

- `crc32`
- `sha256`

2.2.2 Key file

```
key_file=file.key
```

UCL format private key file for SCP packet signing. For more information see UCL Key Format.

2.2.3 Signature Only

```
signonly=yes
signonly=no
```

`yes` - Only a sig file containing the signature will be generated `no` - A signed binary (binary + signature) file will be also generated.

2.2.4 Generate SLA header

```
header=yes
header=no
```

`yes` - SLA header will be generated according to parameters and added at the begining of binary `no` - No header will be egenrate, it is supposed that the header is already present in binary

2.2.5 Verbose

```
verbose=level
```

verbose level (0-5)

2.3 Header Parameter

2.3.1 App version

```
application_version=version
```

`version` - Version of the application - 4 bytes hexadecimal encoded : (ex : `012AC567 0xABCDEF01`)

2.3.2 Bootloader Version

```
rom_version=version
```

`version` - Version of the targeted Bootloader, Please refers to CHIP documentation to select the corect one. - 4 bytes hexadecimal encoded : (ex : `012AC567 0xABCDEF01`)

2.3.3 Load Address

```
load_address=address
```

`address` - Address of the location where the application will be copy before executed. - 4 bytes hexadecimal encoded : (ex : `012AC567 0xABCDEF01`)

2.3.4 Jump Address

```
jump_address=address
```

`address` - Address of the instruction where the bootloader will jump. - 4 bytes hexadecimal encoded : (ex : `012AC567 0xABCDEF01`)

2.3.5 Arguments

```
arguments=args
```

`args` - Argument for the application to include inside the header. The pointer arguments will be store in r0 and the length in r1 before jumping in the application. - String (ex : `argument1 argument2`)

2.3.6 Boot Method

```
boot_method=cmsis
boot_method=direct
```

`cmsis` - The Jump address points to the value ot the *Stack pointer* followed by the address of the *reset handler*. The boot loader will setup the Stack pointer and then jump to the "reset handler*. `direct` - The bootloader will directly jump to the Jump address and the application is responsible to setting up the stack.

2.4 HSM Parameter

This application can use a Thales(R) HSM for key storage and cryptographics operation. By default the application use it's builtin cryptographics functions.

Option	Parameter	Explanation
HSM	`hsm=yes/no`	Define sing with HSM or not, default no
HSM Key Name	`hsm_key_name=name`	Define name of the key to use stored inside the HSM.
Thales DLL Location	`hsm_thales_dll=path`	Provide Thales cknfast DLL path
HSM SLot Number	`hsm_slot_nb=nb`	Define HSM slot number to use (usually : 1)

2.5 MAX3259x Parameter

2.5.1 SDRAM Power Down

```
sr_papd=value
```

`value` - DMC Primary SDRAM Power down register value - 1 bytes hexadecimal encoded : (ex : `0A` `0xA1`)

2.5.2 LPDDR Mode

```
sr_pext=value
```

`value` - DMC Primary LPDDR Mode register value - 1 bytes hexadecimal encoded : (ex : `0A` `0xA1`)

2.5.3 SDRAM Refresh

```
sr_prfsh=value
```

`value` - DMC Primary SDRAM Refresh register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8` `0x0123ACE8`)

2.5.4 SDRAM Configuration

```
sr_pcfg=value
```

`value` - DMC Primary SDRAM Configuration register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8` `0x0123ACE8`)

2.5.5 DMC Global configuration

```
dmc_gcfg=value
```

`value` - DMC Global config register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8` `0x0123ACE8`)

2.5.6 DMC Clock

```
dmc_clk=value
```

`value` - DMC Clock config register value - 1 bytes hexadecimal encoded : (ex : `0A 0xA1`)

2.5.7 UCI AES Key

```
uci0_ksrc_configencint=value
```

`value` - UCI AES Encryption key 0 register value - 1 bytes hexadecimal encoded : (ex : `0A 0xA1`)

2.5.8 UCI Area Config 0

```
uci0_ac1r_so=value
```

`value` - UCI Area Config 0 Start Offset register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8 0x0123ACE8`)

```
uci0_ac1r_eo=value
```

`value` - UCI Area Config 0 End Offset register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8 0x0123ACE8`)

2.5.9 UCI DDR Region 0 Config

```
uci0_ddr_r0=value
```

`value` - UCI DDR Region 0 Config register value - 4 bytes hexadecimal encoded : (ex : `0123ACE8 0x0123ACE8`)

3 Build SCP Session

The build_scp_session tool computes offline the SCP frames corresponding to the provided parameters.

Usage

```
build_scp_session [OPTION] [PARAMETERS]... [ FOLDER [EXTRA_PARAM]... ]
```

`OPTION`	See General Options
`PARAMETERS`	See Parameters
`FOLDER`	Output folder to store SCP Packet
`EXTRA_PARAM`	See Extra Parameters

3.1 General Options

Option	Parameter	Example	Explanation
`Select Chip`	-c CHIP_NAME	build_scp_session -c MAX32520 …	Define chip that be used
`Help`	-h	build_scp_session -h	Print help output
`Version`	-v	build_scp_session -v	Print version of binary
`Debug`	-d	build_scp_session -d	Enable debug output

3.2 Parameters

Parameters are used by priority in the following order :

1. Command line
2. Configuration file " INIFILE " in the current folder
3. Chip default parameters selected by the -c option or the MAXIM_SBT_DEVICE env variable.
4. Software default values.

3.2.1 SCP Script file

```
script_file=script.txt
```

`script.txt` - text file containing SCP operation to perform. For more information see SCP Script Commands

3.2.2 Output filename prefix

```
output_file=nameprefix
```

`nameprefix` - filename prefix of SCP generated file (packets and logfile). Default value : `session.txt`

3.2.3 Output folder

```
output_dir=dir
```

`dir` - folder where SCP files will be saved. If this folder does not exist it will be created.

3.2.4 Key file

```
key_file=file.key
```

UCL format private key file for SCP packet signing. For more information see *UCL Key Format* documentation.

3.2.5 Session Mode

```
session_mode=mode
```

`mode` - SCP session mode to be used for the communication with SBL. Please refers to CHIP documentation to select the corect one. Available mode are :

- `SCP_FLORA_RSA`
- `MSP_MAXQ1852_ECDSA`
- `SCP_ECDSA`
- `SCP_LITE_ECDSA`
- `SCP_PAOLA`

3.2.6 Protection Profile

```
pp=PP
```

PP - SCP Protection profile to be used for the communication with SBL. Please refers to CHIP documentation to select the corect one. Available protection profile are :

- RSA_2048
- RSA_4096
- ECDSA

3.2.7 Verbose

```
verbose=level
```

verbose level (0-5)

3.2.8 Chunk Size

```
chunk_size=size
```

size - maximum data size for one SCP packet (in bytes), this value have to be set according the CHIP used.

3.2.9 Maximum Flash Size

```
flash_size_mb=size
```

size - maximum flash size in Mo, this define the memory allocated when reading a data file (S19, HEX or SBIN)

3.2.10 USN - Unique serial Number

```
usn=USN
```

USN - Unique Serial Number of the device you want to personnalized the SCP session for (i.e. kill-chip command).

3.2.11 Transaction ID (MAXQ1852 only)

```
transaction_id=trid
```

`trid` - User Selected transaction ID when using MSP_MAXQ1852_ECDSA.

3.2.12 Transaction ID (MAXQ1852 only)

```
addr_offset=address
```

`address` - address offset added when reading S19 files and base address when reading SBIN files.

3.3 HSM Parameter

This application can use a Thales(R) HSM for key storage and cryptographics operation. By default the application use it's builtin cryptographics functions.

Option	Parameter	Explanation
HSM	`hsm=yes/no`	Define sing with HSM or not, default no
HSM Key Name	`hsm_key_name=name`	Define name of the key to use stored inside the HSM.
Thales DLL Location	`hsm_thales_dll=path`	Provide Thales cknfast DLL path
HSM SLot Number	`hsm_slot_nb=nb`	Define HSM slot number to use (usually : 1)

3.4 Extra Parameters

In order to make *SCP scripts* more modular extra parameters can be used. There are **TAGS** used in the script that will be replace at the execution with parameters provided in the command line.

The format of the at is the following **%PARAM_N%**. With N from 0 to 9.

3.4.1 Example

Let the following script :

```
wrtie-otp 08bc %PARM_0%
write-otp 09bc %PARM_1%
write-file %PARM_2%
```

With the following command line call

```
build_scp_session scp_folder 02654212 ED45830A firmware.sbin
```

will become :

```
wrtie-otp 08bc 02654212
write-otp 09bc ED45830A
write-file firmware.sbin
```

> Note: When Using Extra parameters the folder **MUST** be specified.

3.5 SCP Script Commands

3.5.1 Write File

```
write-file filename [address]
```

This command send the binary data contains in the provided file to the SBL for writing using the `WRITE DATA` SCP Command. It also erase the target memory are using the `ERASE DATA` SCP Command.

`filename`	S19 or sbin file containing the data to be send for writing to the SBL.
`address`	Start address to where writing data from sbin file or offset adddress to add to S19 addresses. Optionnal whith S19 files but mandatory with sbin files.

3.5.2 Write Only

```
write-only filename [address]
```

This command send the binary data contains in the provided file to the SBL for writing using the `WRITE DATA` SCP Command.

`filename`	S19 or sbin file containing the data to be send for writing to the SBL.
`address`	Start address to where writing data from sbin file or offset adddress to add to S19 addresses. Optionnal whith S19 files but mandatory with sbin files.

3.5.3 Verify file

```
verify-file filename [address [dump]]
```

This command send the binary data contains in the provided file to the SBL for verification against the content of the memory writing using the `COMPARE DATA` SCP Command.

`filename`	S19 or sbin file containing the data to be send for writing to the SBL.
`address`	Start address to where start data for verification from sbin file or offset adddress to add to S19 addresses. Optionnal whith S19 files but mandatory with sbin files.

`dump`	yes/no Add a dummy dump packet for the SCP sender

3.5.4 Write CRK

```
write-crk filename
```

This command send `WRITE-CRK` SCP Command. It send the CRK with it's signature by the MRK.

`filename`	File containing the CRK sign by the MRK

3.5.5 Rewrite CRK

```
rewrite-crk old_crk_filename new_crk_filename
```

This command send `REWRITE-CRK` SCP Command. It send the *old* CRK and the *new* CRK with it's signature by the MRK.

`old_crk_filename`	File containing the old CRK sign by the MRK
`new_crk_filename`	File containing the new CRK sign by the MRK

3.5.6 Echo

```
echo
```

This command check the communication with the SBL by sending an `ECHO` SCP command.

3.5.7 Write OTP

```
write-otp offset data
```

This command write data inside the CHIP OTP using the `WRITE-OTP` SCP Command.

`offset`	Address offset inside the OTP memory.
`data` Da	ta to write at the offset specified.

3.5.8 Write Time-out

```
write-timeout target value
```

This command write the timeout configuration for the different SCP bus using the `WRITE-TIMEOUT` SCP Command.

`target`	Bus for which the timeout will be written. Possible value are : `0` - for UART `V` - for VBUS `U` - for USB `E` - for Ethernet `S` - for SPI
`Value`	Value of the Timeout in ms.

3.5.9 Write Parameter

```
write-param target value
```

This command write the parameter configuration for the different SCP bus using the `WRITE-PARAM` SCP Command.

`target`	Bus for which the parameter will be written. Possible value are : `0` - for UART `V` - for VBUS `U` - for USB `E` - for Ethernet `S` - for SPI
`Value`	Value of the parameter.

3.5.10 Write Stimulus

```
write-stim target value
```

This command write the stimulus configuration for the different SCP bus using the `WRITE-STIM` SCP Command.

`target`	Bus for which the stimulus will be written. Possible value are : `0` - for UART `V` - for VBUS `U` - for USB `E` - for Ethernet `S` - for SPI
`Value`	Value of the stimulus.

3.5.11 Write Deactivation

```
write-deact target
```

This command deactivate the different SCP bus using the `WRITE-DEACT` SCP Command.

`target`	Bus for which the stimulus will be written. Possible value are : `0` - for UART `V` - for VBUS `U` - for USB `E` - for Ethernet `S` - for SPI

3.5.12 Kill Chip

```
kill-chip
```

This command send the `KILL-CHIP` SCP command to SBL.

3.5.13 Kill Chip USN

```
kill-chip2
```

This command send the `KILL-CHIP2` SCP command to SBL with the Chip Unique Serial Number (USN) provided with the corresponding option.

3.5.14 Execute Code / Register Applet

```
execute-code address
```

This command send the `EXECUTE-CODE` SCP command to SBL. This will register an applet if the adress point to an applet header or will launch an application if the address point to an application header.

`address`	Address of the previously loaded SCP applet or Application.

3.5.15 Write Minimum Application Version

```
write-app-ver version
```

This command send the `WRITE_APP_VER` SCP command to SBL. This will setup the minimum requiered version for the customer application.

`version`	Minimum version required for the application 0xMMmmrrrr(i.e 0x01021240 for version 1.2.4672)

4 SCP Sender

The send_scp tool is used to communicate with device and send SCP package to device. To use it first open a command line like cmd or powershell on windows in any folder you want.

```
send_scp -c <CHIPNAME> -s <port COMx> <scp_folder>
```

> Note:
> When using USB interface **–serial-dsrdtr** parameters need to be specified. As below
> send_scp -c <CHIPNAME> -s <port COMx> **–serial-dsrdtr** <scp_folder>

5 Glossary

SBL	Secure Boot Loader
SCP	Secure Communication Protocol
USN	Unique Serial Number
OTP	One Time Programmable Memory
CRK	Customer Root Key
MRK	Maxim Root Key