

# Second Level Boot

UNDER CONSTRUCTION

## Overview

For better comprehension, **Second Level Boot** is generic term. **Secure Flexible Loader** is one (SiFive) implementation of **SLB**.

The **SBR** is going to check for **SFL** (or equivalent) security integrity before launching it. Of course if this process fails, **SBR** does not launch **SFL** and goes into "*download mode*".

## Boot Policy

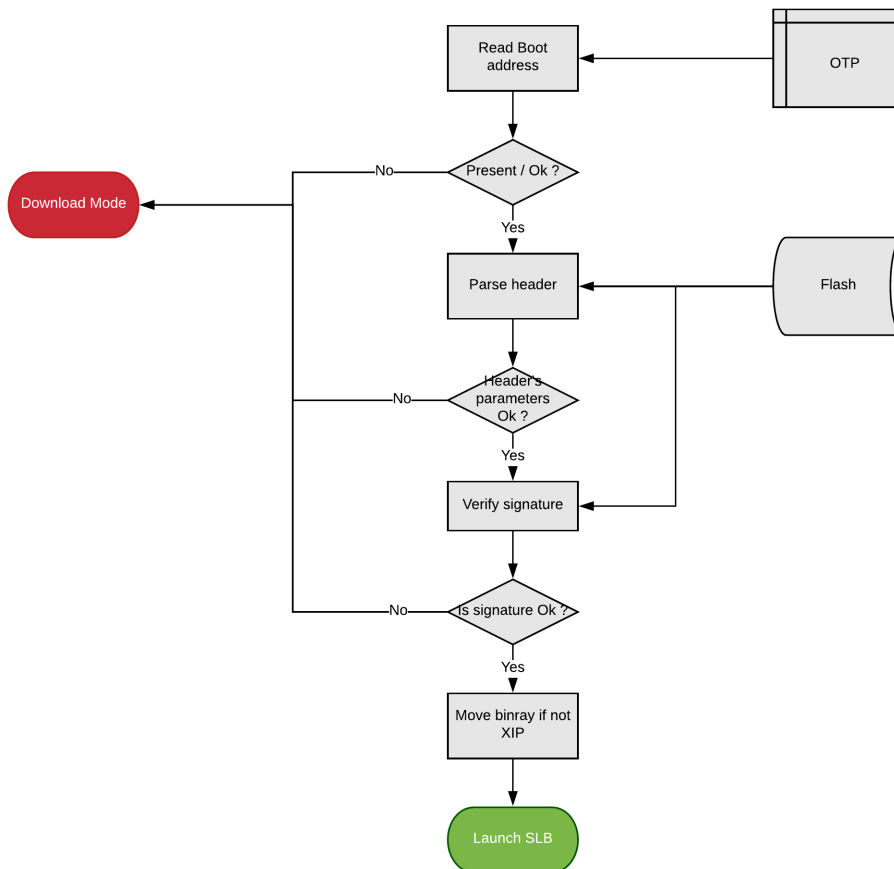
Boot policy determines how **SBR** access to binary image stored on platform and what it expects to find in **SLB** storage location.

Consequent access to binary is possible thanks to "*Boot\_Address*". This parameters is stored in platform's non-volatile internal memory (e.g. **OTP**). It has defined depth to allow user to chane in time this value. It is completely platform dependent.

## Raw Boot

The "Raw Boot" characteristic is to be independent from any file system or software storage partitioning management. It means it is not based upon **MBR** nor on **GPT**.

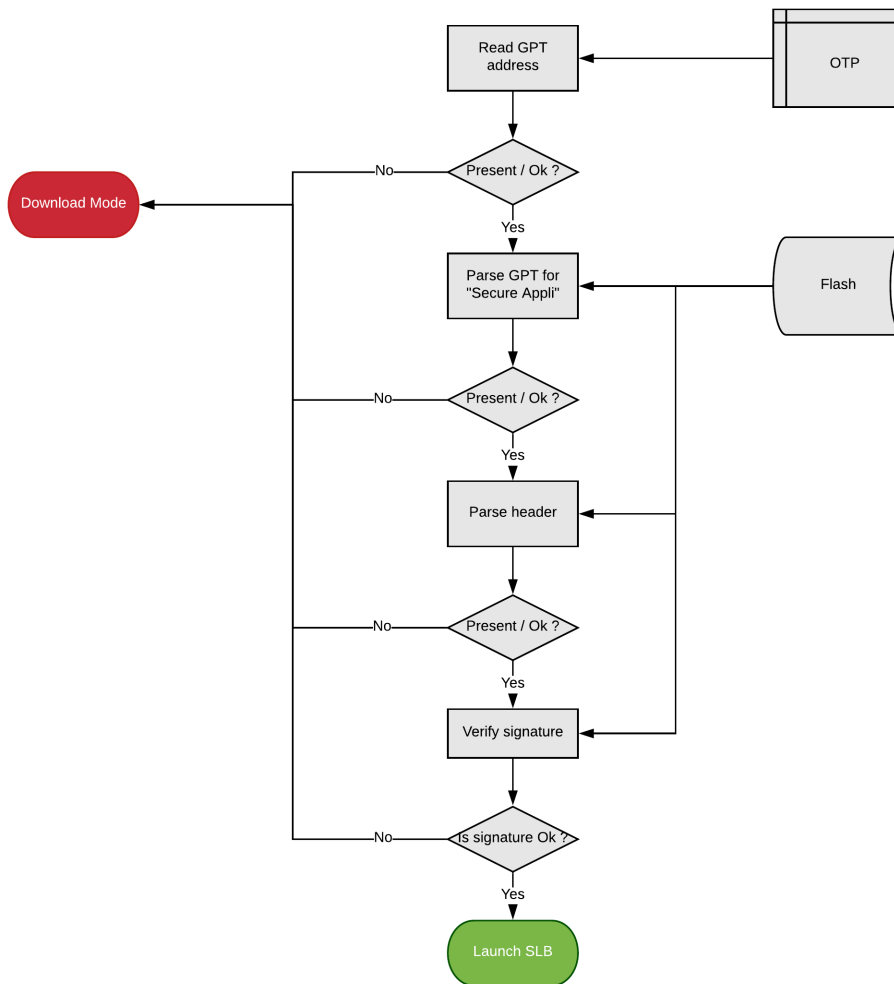
The "*Boot\_Address*" gives the address where to to directly find header of the binary to be checked and launched.



## GPT Boot

- Overview
- Boot Policy
  - Raw Boot
  - GPT Boot
- Format
  - Header
- Exit SBR

Boot is based on [GPT](#) stored in external/internal memory (e.g. [SPI flash](#)). Note that specific partition identifier



## Format

[ ... ]

This "secure" format applies to [SFL](#), launched by [SBR](#), and customer application, launched by [SFL](#).

## Header



## application secure header

- the secure header total size is **160 Bytes**
- the secure header fields are the following:

4 Bytes magic word (0xF17EA991)	4 Bytes magic word (0xF17EA992)	4 Bytes Secure Boot ROM version	4 Bytes firmware version	2 Bytes secure application type	2 Bytes address size	4 Bytes secure application image size	4 Bytes firmware start offset	16 Bytes copy address	16 Bytes execution address	2 Bytes signature algorithm	2 Bytes signature key identifier	96 Bytes ECDSA signature
---------------------------------------	---------------------------------------	---------------------------------------	--------------------------------	--	----------------------------	--	-------------------------------------	--------------------------	-------------------------------	-----------------------------------	--	--------------------------------

- magic word:** 0xF17EA991 or 91A97EF1<sub>16</sub>
- magic word:** 0xF17EA992 or 92A97EF1<sub>16</sub>
- secure boot ROM version:** 4 Bytes representing from which Secure Boot ROM base version this application is compliant with (e.g. version 2.7.3 is coded 0x02070003 or 03000702<sub>16</sub>).
- firmware version:** 4 Bytes representing the firmware version (e.g. version 1.6.2 is coded 0x01060002 or 02000601<sub>16</sub>).
- application type:** 2 Bytes
  - 0x0001 or 0101<sub>16</sub> for regular secure application image
  - 0x0FD4 or D40F<sub>16</sub> for encrypted secure application
- address size:** 2 Bytes representing address width
  - 0x0101 or 0101<sub>16</sub> for 32bits large
  - 0x4E4E or 4E4E<sub>16</sub> for 64bits large
  - 0xB2B2 or B2B2<sub>16</sub> for 128bits large addresses
- secure application image size:** 4 Bytes representing the full size in bytes of the secure application image, i.e. the secure header + the binary image; the binary image size is then this value - 160
- firmware start offset:** 4 Bytes representing the firmware (binary image) start offset after the header.
- copy address:** 16 Bytes representing where the binary image has to be copied before being executed
- execution address:** 16 Bytes representing where to jump for running the binary image
- signature information:** 2 Bytes representing the signature information:
  - Algorithm: 1 Byte, ECDSA id (0xA7). Others values are not supported for now.
  - Signing Key Identifier: 1 Byte, CSK id (0x84). Others values are not supported for now.
- signature size:** 2 Bytes representing signature length in bits (i.e. 384bits - 0x0180 or 8001<sub>16</sub> for the ECDSA p384r1)

## Exit SBR

[ ... ]

SBR has now successfully checked **SLB**<sub>(SFL)</sub>, it restores used IP(s) to its/their "start state".

When **SLB** has been checked successfully and is now ready to jump at "jump\_address" found in header.