USER'S GUIDE

Apollo4 Family Secure Update

Ultra-Low Power Apollo SoC Family

A-SOCAP4-UGGA02EN v1.4

Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT, AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES. “TYPICAL” PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING “TYPICALS” MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER’S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.

Revision History

Revision	Date	Description
1.0	October 31, 2020	Initial Release.
1.1	February 23, 2021	▪ Removed duplicate figure in section 5.1.2 Connection Establishment (HELLO and STATUS) Messages. ▪ Updated Table 3-1 Common Field Definitions
1.2	November 19, 2021	▪ Updated document cover/back page template. ▪ Retitled and updated content in section 2 Security Features of Image Update ▪ Updated content in section 3 Image Formats ▪ Updated content in section 4 Secure Image Upgrade Flow ▪ Updated content in section 6 OEM Secondary Bootloader
1.3	October 19, 2022	▪ Updated document template
1.4	July 6, 2023	▪ Updated document title

Reference Documents

Document ID	Description
A-SOCAP4-UGGA05EN	Apollo4 SoC Security User's Guide
A-SOCAP4-UGGA01EN	Apollo4 and Apollo4 Blue OEM Provisioning Tools User's Guide

Table of Contents

List of Tables

List of Figures

SECTION

1

Introduction

This document describes the methods, image formats, and protocols supported by the Apollo4 Secure Bootloader (SBL) for both secure image and wired updates.

1.1 Terminology

This section defines some of the terminologies used in this document.

Table 1-1: Terminology

Abbreviation	Definition
OTA	Over the Air
OTP	One Time Programmable
RSA	A public-key cryptosystem that is widely used for secure data transmission.
SBL	Secure Bootloader

SECTION

2

Security Features of Secure Image Update

The following sections describe the specific security features of the Image update flow using Secure Bootloader (SBL).

2.1 Integrity Check

Integrity Check using CRC32 (Ethernet FCS) function.

2.2 Authentication

Authentication uses RSA 3072 signature verification on the **Signature** field using the PK from the existing certificate chain.

The **Auth Key Index** field (0-2) is specified to determine which PK is used based on the 3-stage pre-installed Certificate chain.

Since the authentication relies on pre-existing certificate chain, this feature is only supported if Secure Boot is enabled.

2.3 Decryption

Decryption is performed using AES-128 CTR and the **Encryption Info** field. The Encryption Info comprises of the Nonce/IV and the Encrypted Key. The Encrypted key is decrypted using the provisioned Key Encryption Key (KEK) on the device, and is thereafter used to decrypt the ciphertext in conjunction with the supplied Nonce/IV.

KEK Index determines which of the preinstalled keys is used to decrypt the Encrypted key. Values 0 and 1 correspond to the hardware keys, KCP and KCE respectively. Indices 0x80 onwards signify keys in the Key bank area, with each 128b key corresponding to one index.

2.4 Anti-Rollback

When in Secure Mode, the SBL uses a combination of the OTP_HBK1MINVER* words and the swVer in the OEM Content Certificate to provide rollback prevention. This will prevent an OTA from upgrading to an OEM image using a certificate with a version that is lower than that, of the current certificate chain. All the Certificates in a chain should all have the same version. Hence, to implement the anti-rollback, the full certificate chain needs to be refreshed with an updated version number, using a ‘Certificate Chain Update’ operation.

2.5 Image Protection

SBL can be instructed to apply image protection features to assets in the MRAM. The MRAM blocks for protection are specified in 16K granularity.

The blocks can be marked as *Write-Protected* to prevent overwriting the images either intentionally using malicious programs, or unintentionally. Such Write-protected images can be upgraded only through SBL by maintaining the secure upgrade trust chain.

The blocks can be marked as *Copy-Protected* to prevent Read access to the pages. This can be used to avoid exposing sensitive algorithms or programs from prying eyes. Care must be taken when using this feature to protect executable code—so as to generate the code using appropriate tool options so that there are no data reads in the code memory.

These protection attributes are specified in the OTP (OTP_SBL_”PROT”), as part of provisioning, and hence are statically predefined.

SBL also supports dynamic MRAM protection through content certificate. On successful installation/verification through SBL, the OEM Content Certificate can also be used to direct the SBL to apply specified protections for each image it refers to.

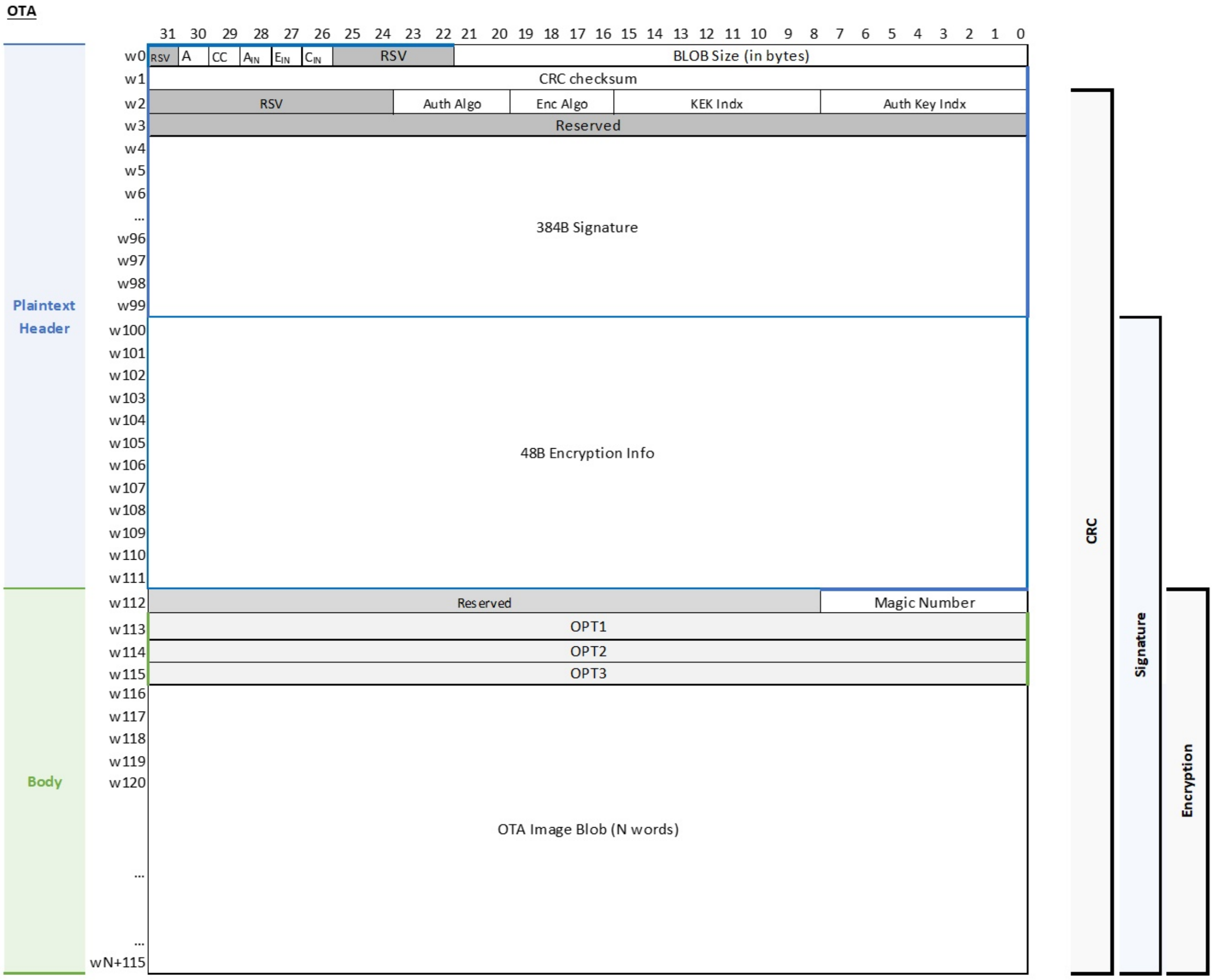
SECTION 3

Image Formats

This section describes the generic format for all the OTA images.

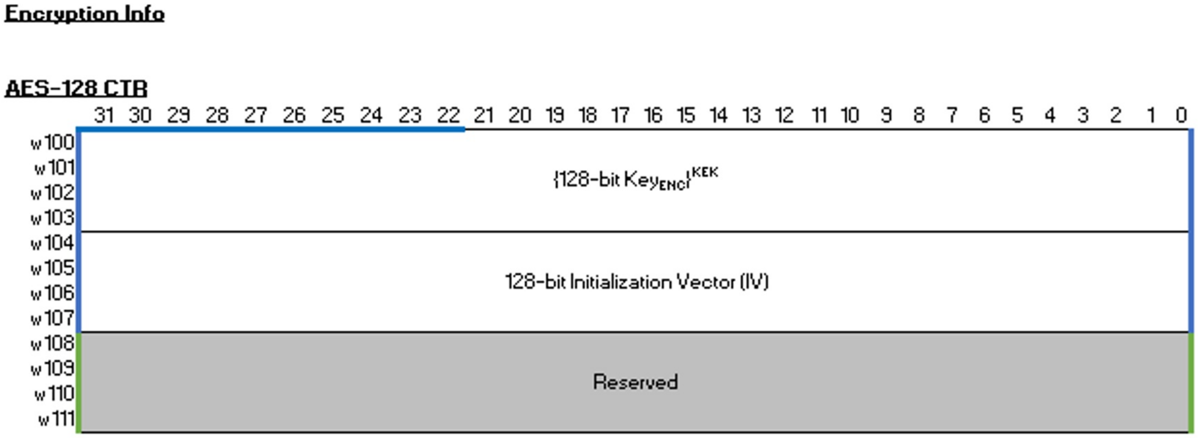
- This is the format for both Secure and Non-Secure OTA Updates, as well as non-firmware updates.
  - The image format supports error detection, authentication, and decryption.
- The **Signature** field is optional, based on value of $\mathbf{A_{IN}}$ (Value of 0 implies no Authentication, and hence no signature).
- The **Encryption Info** field is optionally present, if the $\mathbf{E_{IN}}$ is non-zero.
- The image format and the offset of the fields will change if the optional **Signature** and/or **Encryption Info** is not present. The generic format in Figure 3-1 on page 12 and Figure 3-2 on page 12 depicts the case when both are present.

Figure 3-1: OTA Image Generic Format with Signature and Encryption Info - A


OTA
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
w0 RSV A CC A_IN E_IN C_IN RSV BLOB Size (in bytes)
w1 CRC checksum
w2 RSV Auth Algo Enc Algo KEK Indx Auth Key Indx
w3 Reserved
w4 w5 w6 ... w96 w97 w98 w99
384B Signature
w100 w101 w102 w103 w104 w105 w106 w107 w108 w109 w110 w111
48B Encryption Info
w112 Reserved Magic Number
w113 OPT1
w114 OPT2
w115 OPT3
w116 w117 w118 w119 w120 ... ... wN+115
OTA Image Blob (N words)
Plaintext Header
Body
CRC
Signature
Encryption

If present, the **Encryption Info** field is formatted as shown in Figure 3-2.

Figure 3-2: OTA Image Blob with Encryption Info



3.1 Common Field Definitions

Table 3-1: Common Field Definitions

Field	Definitions
Magic Number	Unique 8-bit value used to reference a valid image type. 0xA3 = SBL, 0xAF = PATCH, 0xAE = AMBIQ KEYBANK UPDATE, 0xC0 = SECURE FIRMWAREMAIN, 0xCC = OEM CERT CHAINCHILD, 0xCB = NONSECURE FIRMWARE, 0xCE = OEM KEY REVOKE, 0xCD = DOWNLOAD, 0xCF = INFO0
BLOB Size	Size of the image blob body (in bytes)
A	Ambiq Owned (Determines the restrictions on the magic#, and the keys to use for Authentication/ Encryption)
CRC Checksum	CRC checksum signature for the image. CRC is computed over the **CRC** region marked on the decrypted blob.
CC	Contains embedded Content Cert
A_{IN}	Install-Authenticate enable bit - If '1', the image must be authenticated on installation.
C_{IN}	Install-CRC enable bit - If '1', the image must be CRC verified on installation.
E_{IN}	Install-Decryption enable bit - If '1', the image must be decrypted on installation.
KEK Indx	Key-Encryption-Key Index. This index specifies which KEK should be used within the KEK bank for all key unwrap functions MSB indicates if it is a predefined key, or from key bank. Key bank keys are indexed in 128b increment.
Enc Algo	Encryption Algorithm - specifies which algorithm is to be used for decrypting the image. 0: N/A 1: AES-128 CTR others: not supported
Auth Key Indx	Authentication Key Index. This index specifies which authentication key should be used within the Auth Key bank for authentication of this image. Only three values are supported right now, which will be the PK of the existing Cert chain.
Auth Algo	Authentication Algorithm. Specifies which algorithm is to be used for authenticating the image. 0: N/A 1: RSA3072 others: not supported
Signature	384B RSA3072 signature of the Install Blob following.
128-bit IV	128-bit Initialization Vector used for seeding the encryption/decryption of the image.
$\{128\text{-bit Key}_{ENC}\}^{KKK}$	128-bit KEK wrapped encryption key
OPT*	Optional parameters (depends on Magic number)

3.2 Optional Field Definitions

Optional fields are interpreted based on the type of update (magic#).

3.2.1 Firmware Update

Figure 3-3: Optional Field Definitions for Firmware Updates (Secure or Nonsecure)

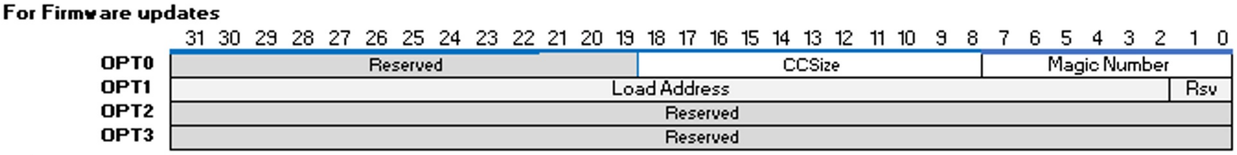


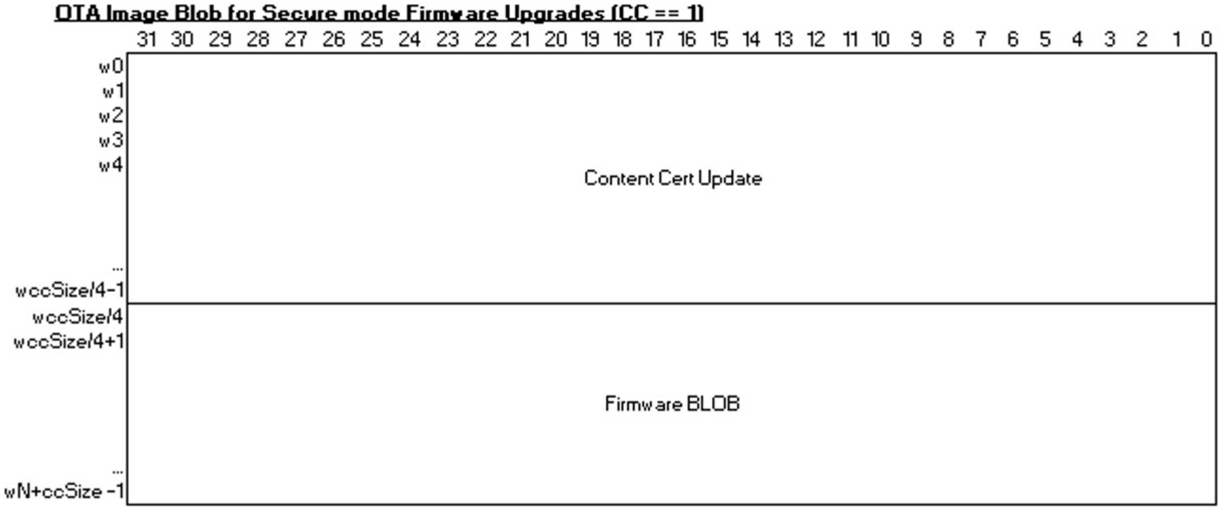
Table 3-2: Field Definitions for Firmware Updates

Field	Definitions
CCSize	Size of the bundled Content Certificate (only relevant if CC is set in W0 for SECURE FIRMWARE update)
Load Address	msb of Address to which the image should be installed.

In the non-secure mode, the embedded OTA Blob inside contains the raw image.

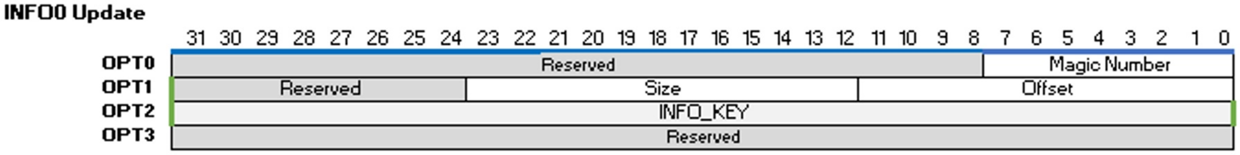
For Secure Mode, the OTA Blob is shown in Figure 3-4 as the updated content certificate is also bundled along:

Figure 3-4: OTA Image Blob with a Content Certificate



3.2.2 INFO0 Update

Figure 3-5: INFO0 Update



INFO0 Update

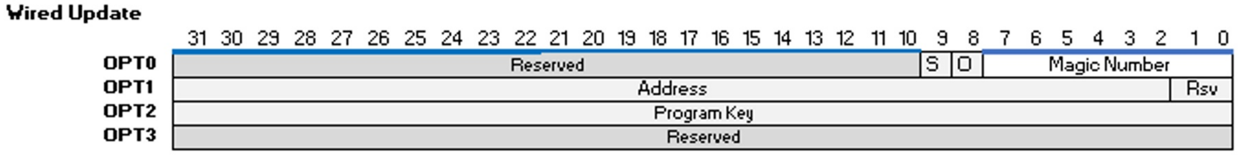
	31 … 0		
OPT0	Reserved		Magic Number
OPT1	Reserved	Size	Offset
OPT2	INFO_KEY		
OPT3	Reserved		

Table 3-3: Field Definitions for INFO0 Update

Field	Definitions
INFO_KEY	32-b key required for programming the INFO0
Size	The size of the update
Offfset	The offset in the Infospace where the update needs to happen.

3.2.3 Wired Update (Download)

Figure 3-6: Wired Update



Wired Update

	31 … 0			
OPT0	Reserved	S	O	Magic Number
OPT1	Address			Rsv
OPT2	Program Key			
OPT3	Reserved			

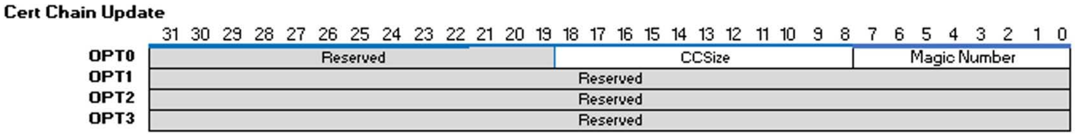
Table 3-4: Field Definitions for Wired Update

Field	Definitions
O	Implying an OTA process request
S	Implying an SBL OTA
Program Key	32-bit key required for programming the MRAM (PROG_KEY)
Address	msb of Address in flash to load the image to

The embedded OTA blob contains the raw download image.

3.2.4 Certificate Chain Update

Figure 3-7: Certificate Chain Update



Cert Chain Update

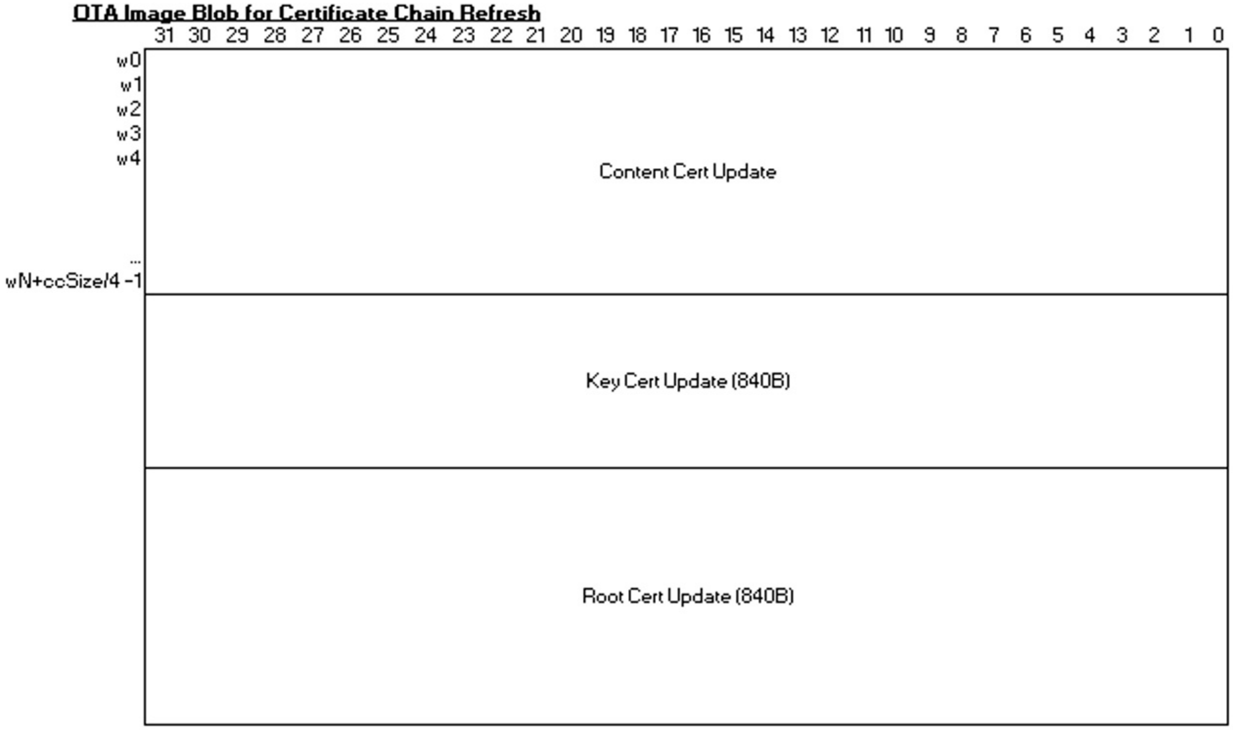
	31 … 0		
OPT0	Reserved	CCSize	Magic Number
OPT1	Reserved		
OPT2	Reserved		
OPT3	Reserved		

Table 3-5: Field Definitions for Certificate Chain Update

Field	Definitions
CCSize	Size of the bundled Content Certificate

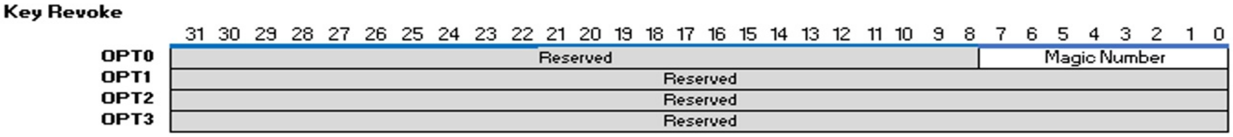
The embedded OTA blob contains the full certificate chain, as shown below.

Figure 3-8: OTA Image Blob for Certificate Chain Refresh



3.2.5 Key Revocation Update

Figure 3-9: Key Revocation Update



The embedded OTA blob contains raw 64b bitmask, with each bit corresponding to 128b key in the keybank. A value of 1 indicates a request to revoke the corresponding key.

SECTION 4

Secure Image Upgrade Flow

The Apollo4 SoC supports a secure in-field image upgrade flow using the Secure Bootloader (SBL). The Secure Bootloader has access to the key storage in the OTP memory and can be used to optionally decrypt, authenticate and validate upgrade images before installing them. If configured so (based on the security policy configurations in OTP), only properly signed and protected images will be allowed in an update.

The Secure Bootloader can be used to securely update itself, the customer application image, a customer secondary bootloader, or other assets. The following sections lists out steps in the secure in-field image update flow. The same update flow is also used for accomplishing updates for INFO0, Certificate Chain, as well as to revoke the keybank keys.

4.1 Image Download

The In-Field upgrade process is initiated by a user application downloading an image blob to the MRAM. This part of the upgrade process is specific to individual deployment scenarios and the implementation is left to the customers. The Secure Upgrade framework does not mandate any specifics for this process. Depending on the deployment model, the image download could happen over traditional wired interfaces (e.g., I^2C, SPI, or UART), or wirelessly OTA (Over the Air) using BLE, or other wireless protocols. The Upgrade application running on the Apollo4 SoC and its counterpart on the host/cloud side could implement their own protocol to ensure integrity, secrecy, and authenticity of the image blob itself.

4.2 OTA Descriptor and OTA Pointer

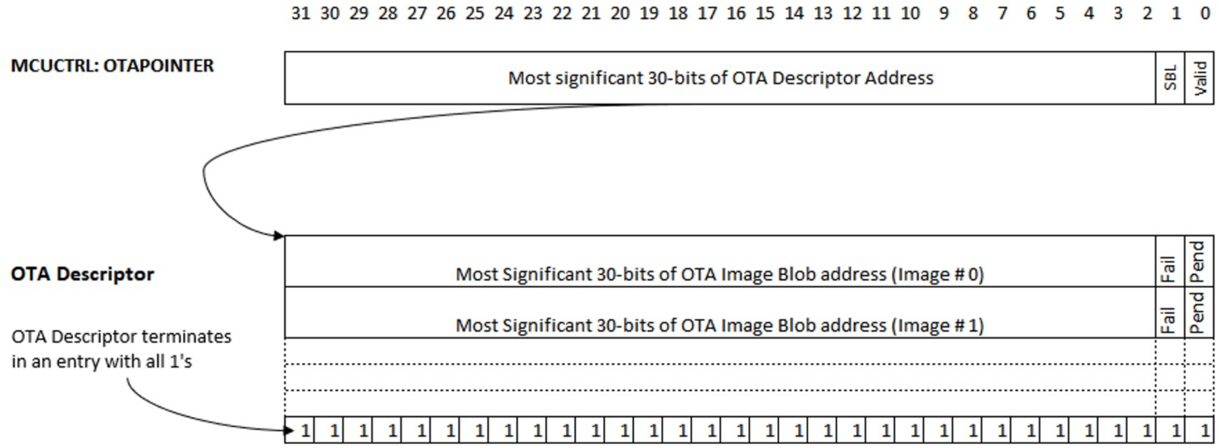
The customer application builds an OTA Descriptor containing the information about the upgrade image blob(s) corresponding to the update requests.

OTA Descriptor needs to be built in MRAM. The framework supports multiple updates in same operation. The OTA Descriptor consists of a list of upgrade image blobs, with a special End of

List Marker. This allows the upgrade application to update more than one images in one boot cycle. For example, the upgrade application could construct a single OTA Descriptor to upgrade main firmware as well as certain third-party library images located separately in the MRAM. OTA Descriptor supports up to eight images[1].

The download application communicates the OTA Descriptor information with the Secure Bootloader by initializing a special register, MCUCTRL:OTAPOINTER.

Figure 4-1: OTA Descriptor and OTA Pointer


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
MCUCTRL: OTAPOINTER
Most significant 30-bits of OTA Descriptor Address
SBL
Valid
OTA Descriptor
Most Significant 30-bits of OTA Image Blob address (Image # 0)
Fail
Pend
Most Significant 30-bits of OTA Image Blob address (Image # 1)
Fail
Pend
OTA Descriptor terminates in an entry with all 1's
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

OTAPOINTER will be initialized as follows:

- Most significant 30-bits will correspond to most significant 30-bits of OTA Descriptor.
- Least Significant bit (bit 0) should be initialized to 1 to indicate a valid OTA Descriptor.
- Bit 1 should be initialized to 1 if the list contains an SBL OTA.

The **OTA Descriptor** contains a list of entries (up to 8), each corresponding to an OTA blob, list terminating in 0xFFFFFFFF. Each list entry word are comprised of following:

- Most significant 30-bits will correspond to most significant 30-bits of OTA blob pointer.
- Least Significant 2-bits should be initialized to '11' to indicate a valid OTA Pending.

Note that the OTA Image Blobs needs to be aligned at a 16-byte boundary to ensure that the encryption portion falls at 128b alignment (block size).

4.3 Reset

After accepting the required updates, the download application constructs the OTA Descriptor and initializes the **MCUCTRL:OTAPOINTER** register accordingly. The actual Update is only initiated on the next Reset, which kicks in the bootloader.

[1] Note that when using multiple SECURE FIRMWARE updates together, the embedded content certificate of each image should reflect the updated content at respective stage

4.4 Upgrade Verification

As part of boot process, the SBL inspects **MCUCTRL:OTAPOINTER** for any updates to be processed. If present, each update blob is processed as per the configured security policy.

- The security policy can be configured (via OTP memory) to mandate Authentication to ensure only properly signed images will be accepted.
- The Secure Bootloader also supports encrypted image blobs, and the same can also be mandated by the security policy.

The SBL enforces the configured security policy and validates the image blobs against the security assets in OTP memory, and/or pre-installed certificate chain in the MRAM. After optional decryption and authentication, if the image is found to be good, the SBL then proceeds with installation of the image. In the case of secure firmware upgrade, SBL also verifies the bundled content certificate update, and installs the new content certificate along with the new image, only if the content certificate checks out (which implies all the images in the content certificate need to match up, if there are more than one).

4.5 Installation

A validated OTA image is installed to its designated place by the Secure Bootloader.

4.6 Feedback

The OTA flow also allows for feedback to the user application using the same OTA Descriptor to communicate the OTA status of individual images, back to the initiating application. This is accomplished using the same OTA Descriptor.

- General OTA descriptor errors are communicated back to the application using the high order bits in the **MCUCTRL:OTAPOINTER** register. Details can be found in **am_hal_secure_ota.h**.
- After the Secure Bootloader processes an OTA, it clears the least significant bit (bit 0).
  - Bit 1 indicates the status of the OTA: 0 for Success, 1 for Failure
    - Some of the higher order bits of the Upgrade Descriptor are used to return failure cause. Details can be found in **am_hal_secure_ota.h**.
  - If Bit 0 is still set to 1 (Pending) after the OTA, this implies some other error (e.g., invalid or improperly formed OTA descriptor list) caused SBL to not process this OTA. The OTA can be retried after clearing the issue.

SECTION

5

Wired Update Flow

The Apollo4 SBL provides support for an external host to connect during bootup to upgrade images, or to recover a failing device. Essentially, the SBL proxies as the “download application” for the Update process, if required. SBL initiates Recovery if any of the boot time validations fail, or if there is no valid image to boot to. In addition, there is an Override feature (configurable through OTP), which allows a forced image upgrade to be initiated using specific GPIO settings during the boot up.

The SBL uses the OTP configuration to determine the interfaces to examine (see **OTP_SEC_WIRED:IFC**). If enabled, it first looks for UART connection, then the IOS connection of either SPI then I^{2}C (these are mutually exclusive). The UART connection is associated with the timeout (see **OTP_SEC_WIRED_UART: TIMEOUT**), while the SBL waits for the initial HELLO packet from the Host. The IOS interface is polled and the SBL uses the Slave Interrupt pin (see **SEC_WIRED: SLVINTPIN**), to indicate to the host that it is ready to receive packets. In this case, it only waits for a fixed timeout of 500 msec before exiting the update process.

In all cases, an external host needs to follow a predefined messaging protocol to instruct SBL to upgrade assets on the device. Figure 5-1 on page 21 illustrates a high-level message exchange to initiate an “upgrade” using prevalent security policies of the device when using the UART. In this case, the **DATA** packets are sent as up to 8KB packets. The process starts with a **HELLO/ STATUS** exchange within the timeout period. It ends with the **RESET/ACK** exchange which sends the SBL into SWPOR or SWPOI reset.

Figure 5-1: Message Exchange Using Prevalent Security Polices of Device

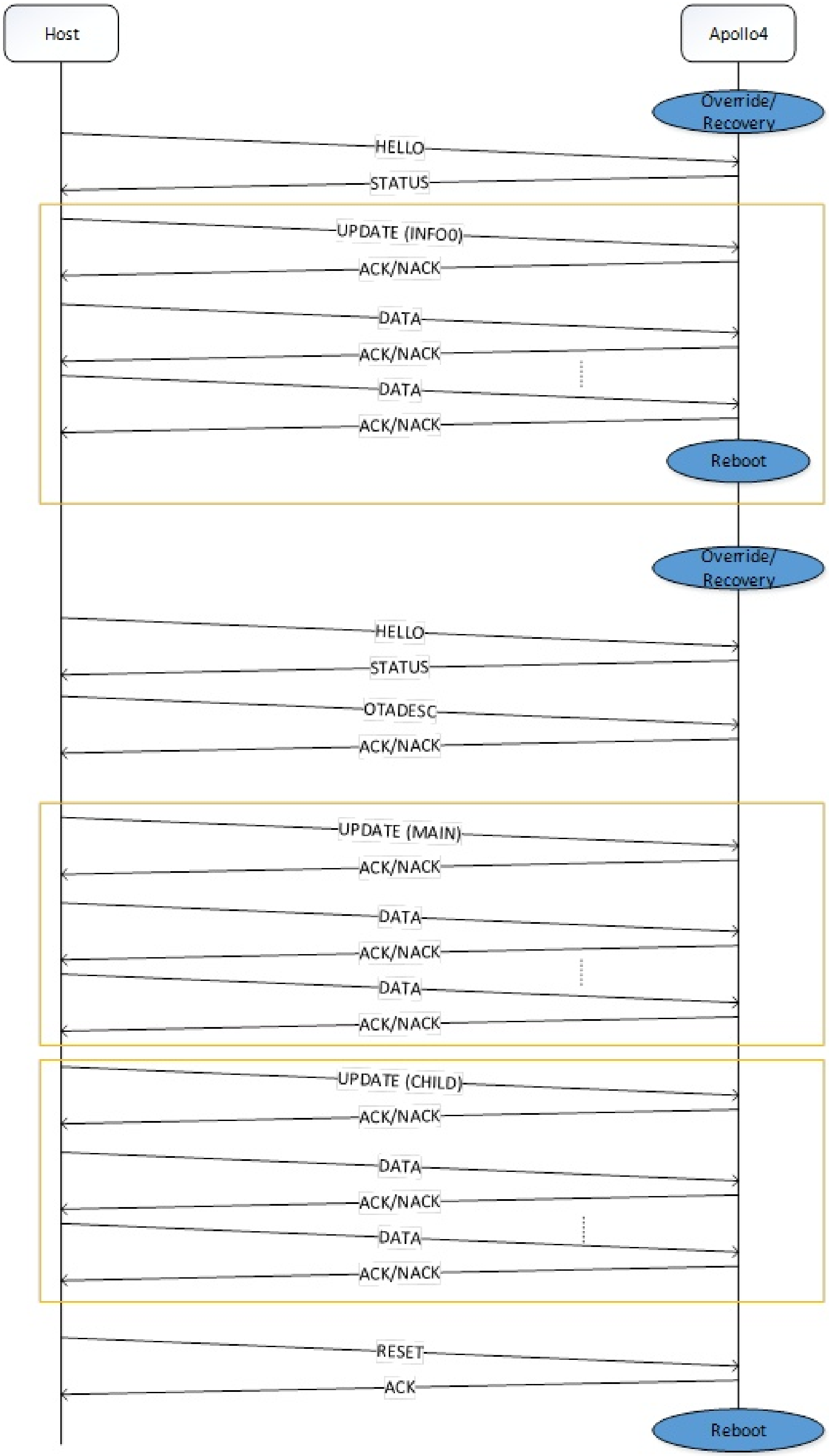


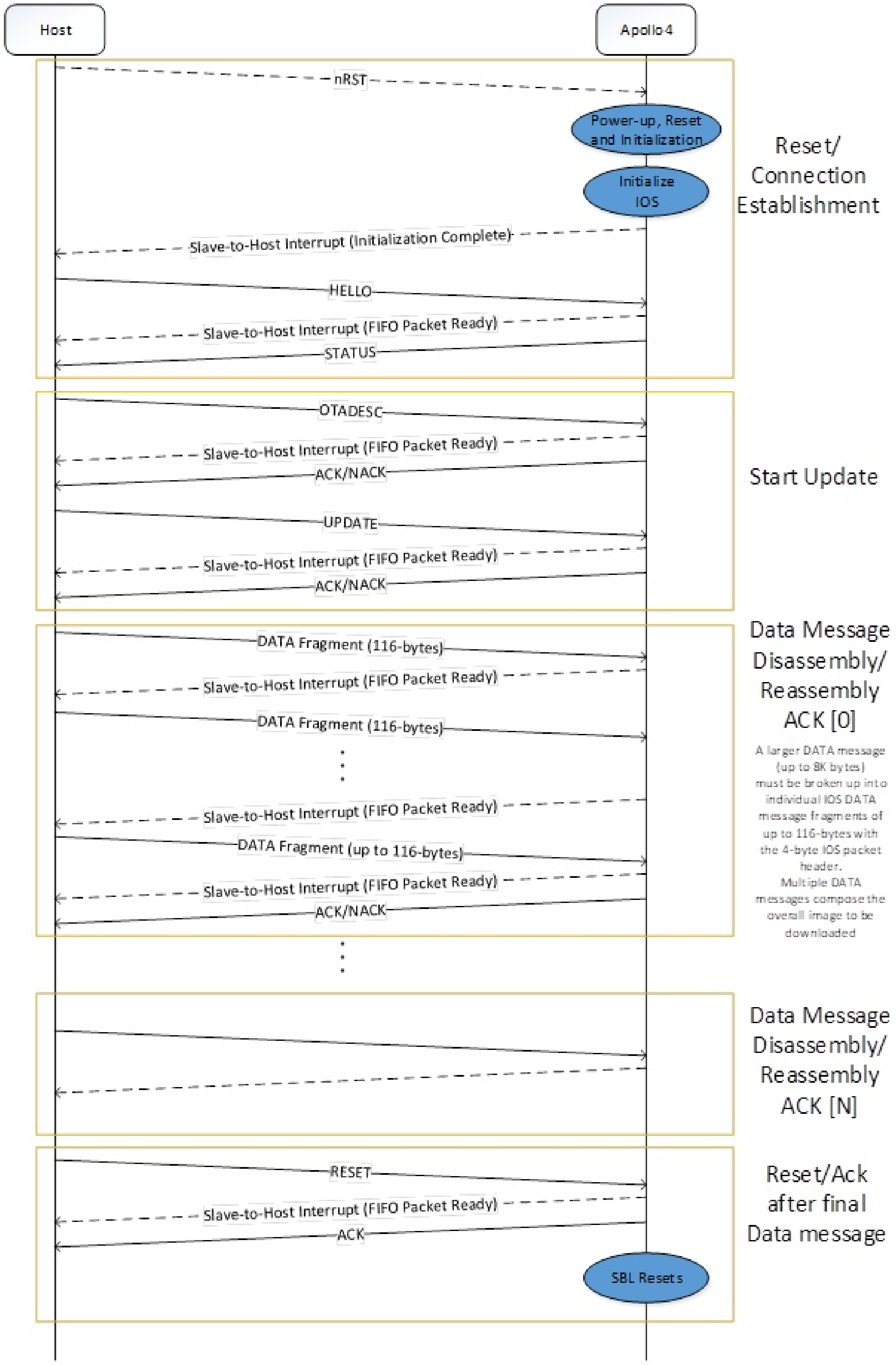
Figure 5-2 on page 23 shows the process for update when using the IOS as SPI or I^2C interface. The process starts with the Host resetting the Apollo4 Blue using nRST signal. Once initialization of the configured interface is complete, the SBL raises the Slave Interrupt pin to indicate it is ready to accept packets over the IOS Direct memory interface. All packets from the Host to the SBL are limited to 120 bytes due to the maximum LRAM memory size. This means the packets from the Host must be disassembled, and then reassembled by the SBL. Disassembly is done by adding a 32-bit header to each packet as follows:

```
typedef struct
{
      uint32_t length : 16;
      uint32_t resv : 14;
      uint32_t bEnd : 1;
      uint32_t bStart : 1;
} am_secboot_ios_pkthdr_t;
```

The start and end flags indicate the packets that represent the first and last of an original SBL wired protocol packet. The original packet data follows up to 116 bytes. The 116-byte Data Fragments compose an entire original DATA message and are stored by SBL up to 8KB. There is limited error checking except that the total bytes received must equal the length in the original message, or it will be rejected.

Note that because the FIFO mode is used for Slave to Host transfers, the message size limit is 1023 bytes so disassembly/reassembly is not required.

Figure 5-2: Process for Update/Recovery Using IOS as SPIC or I^2C Interface


Host
Apollo4
nRST
Power-up, Reset and Initialization
Initialize IOS
Slave-to-Host Interrupt (Initialization Complete)
HELLO
Slave-to-Host Interrupt (FIFO Packet Ready)
STATUS
Reset/ Connection Establishment
OTADESC
Slave-to-Host Interrupt (FIFO Packet Ready)
ACK/NACK
UPDATE
Slave-to-Host Interrupt (FIFO Packet Ready)
ACK/NACK
Start Update
DATA Fragment (116-bytes)
Slave-to-Host Interrupt (FIFO Packet Ready)
DATA Fragment (116-bytes)
Slave-to-Host Interrupt (FIFO Packet Ready)
DATA Fragment (up to 116-bytes)
Slave-to-Host Interrupt (FIFO Packet Ready)
ACK/NACK
Data Message Disassembly/ Reassembly ACK [0]
Data Message Disassembly/ Reassembly ACK [N]
RESET
Slave-to-Host Interrupt (FIFO Packet Ready)
ACK
SBL Resets
Reset/Ack after final Data message

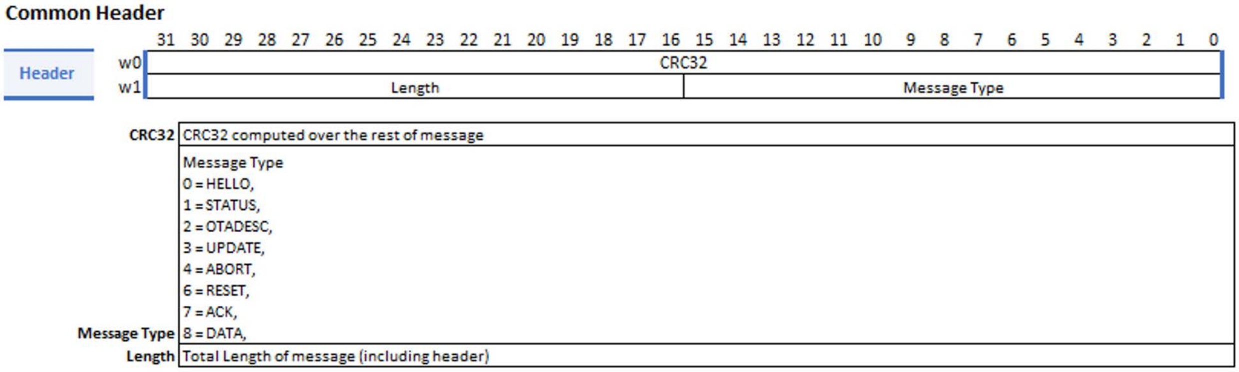
A larger DATA message (up to 8K bytes) must be broken up into individual IOS DATA message fragments of up to 116-bytes with the 4-byte IOS packet header.
Multiple DATA messages compose the overall image to be downloaded

5.1 Wired Update Messages

All the message formats below assume little-endian byte order.

Each message starts with a common header which defines the following message type and length, along with a CRC for error checking.

Figure 5-3: Wired Update Messages



Common Header

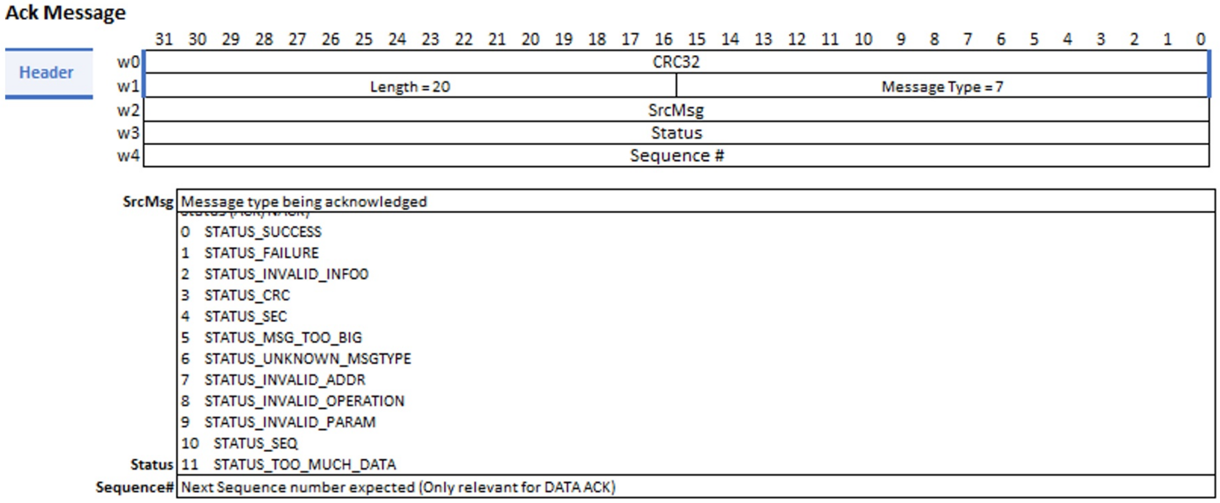
		31 … 16	15 … 0
Header	w0	CRC32	
	w1	Length	Message Type

CRC32	CRC32 computed over the rest of message
Message Type	Message Type 0 = HELLO, 1 = STATUS, 2 = OTADESC, 3 = UPDATE, 4 = ABORT, 6 = RESET, 7 = ACK, 8 = DATA,
Length	Total Length of message (including header)

5.1.1 Acknowledgment (ACK) Message

The SBL acknowledges most of the messages received using an ACK message (except for HELLO, which is acknowledged using STATUS message). Acknowledgments for DATA messages (described later), also include a sequence number, which can be used to implement a retransmission mechanism at host side if the connection medium is lossy.

Figure 5-4: Acknowledgment (ACK) Message



Ack Message

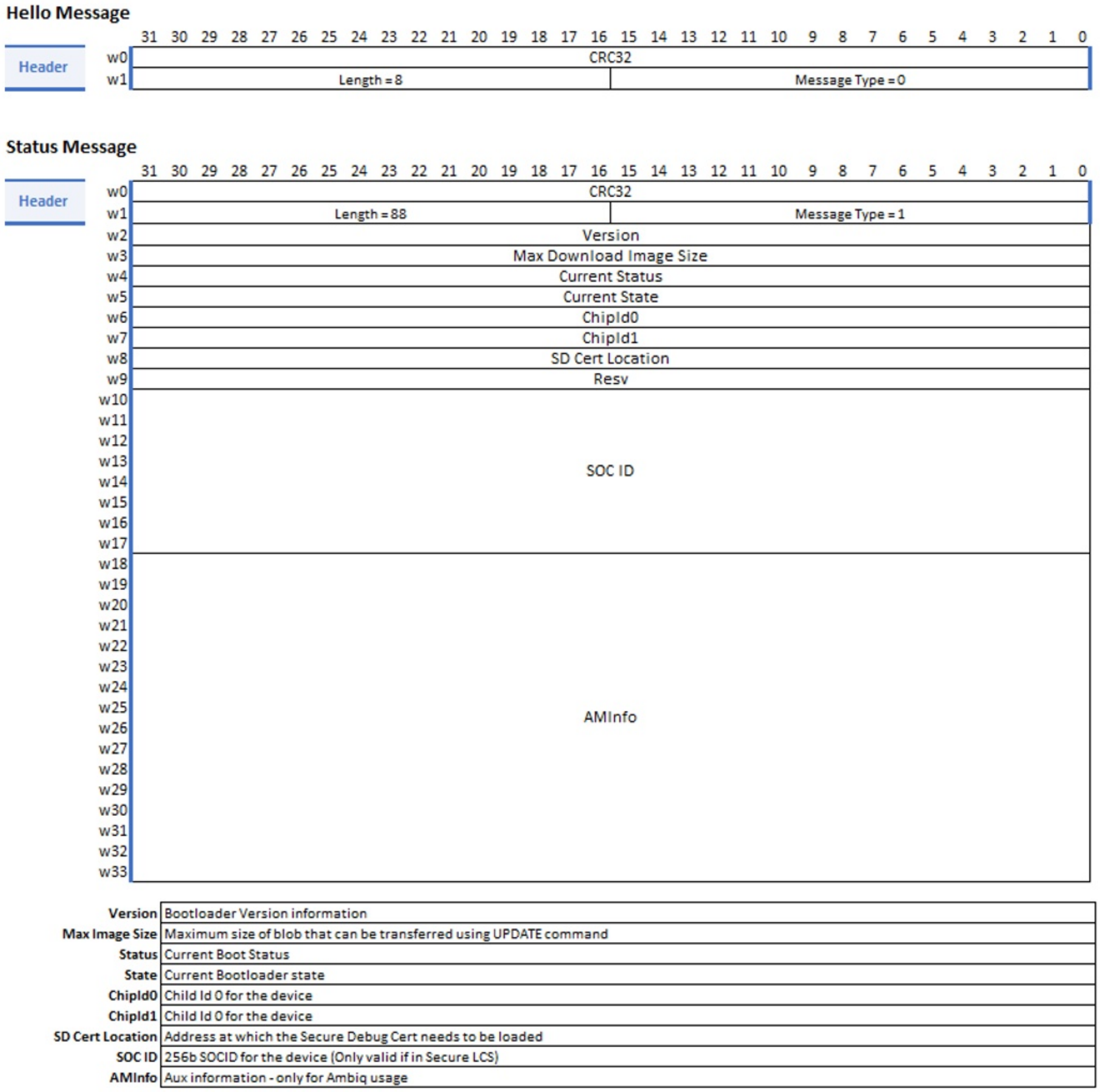
		31 … 16	15 … 0
Header	w0	CRC32	
	w1	Length = 20	Message Type = 7
	w2	SrcMsg	
	w3	Status	
	w4	Sequence #	

SrcMsg	Message type being acknowledged
Status	Status (ACK/NACK) 0 STATUS_SUCCESS 1 STATUS_FAILURE 2 STATUS_INVALID_INFO0 3 STATUS_CRC 4 STATUS_SEC 5 STATUS_MSG_TOO_BIG 6 STATUS_UNKNOWN_MSGTYPE 7 STATUS_INVALID_ADDR 8 STATUS_INVALID_OPERATION 9 STATUS_INVALID_PARAM 10 STATUS_SEQ 11 STATUS_TOO_MUCH_DATA
Sequence#	Next Sequence number expected (Only relevant for DATA ACK)

5.1.2 Connection Establishment (HELLO and STATUS) Messages

There is an initial handshake between the host and SBL, which can assist in determining the reason why SBL got into the wired update mode. A HELLO message is sent from host, to which the SBL responds with a STATUS message. The STATUS message also provides information about the largest size of the image blob that can be downloaded using SBL, along with other useful information about the device.

Figure 5-5: Connection Establishment (HELLO and STATUS) Messages



Hello Message

		31 – 16	15 – 0
Header	w0	CRC32	
	w1	Length = 8	Message Type = 0

Status Message

		31 – 16	15 – 0
Header	w0	CRC32	
	w1	Length = 88	Message Type = 1
	w2	Version	
	w3	Max Download Image Size	
	w4	Current Status	
	w5	Current State	
	w6	ChipId0	
	w7	ChipId1	
	w8	SD Cert Location	
	w9	Resv	
	w10–w17	SOC ID	
	w18–w33	AMInfo	

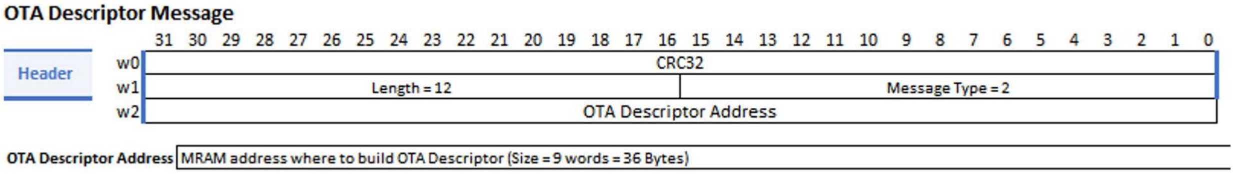
Version	Bootloader Version information
Max Image Size	Maximum size of blob that can be transferred using UPDATE command
Status	Current Boot Status
State	Current Bootloader state
ChipId0	Child Id 0 for the device
ChipId1	Child Id 0 for the device
SD Cert Location	Address at which the Secure Debug Cert needs to be loaded
SOC ID	256b SOCID for the device (Only valid if in Secure LCS)
AMInfo	Aux information - only for Ambiq usage

5.1.3 Secure Upgrade (OTADESC) Message

Upgrade for the firmware images using wired update undergoes the same secure OTA flow as applicable to wireless OTA. SBL just provides means to do wired download for the image blobs. So, in accordance, there are messages which instruct SBL to create an OTA Descriptor, followed by download of one or more image blobs (using UPDATE and DATA messages).

SBL reserves a fixed size in the MRAM starting at the specified address for the purpose of building the OTA Descriptor (to allow for up to 8 update images).

Figure 5-6: Secure Upgrade (OTADESC) Message



OTA Descriptor Message

<table>
<tr><th></th><th></th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td rowspan="3">Header</td><td>w0</td><td colspan="32">CRC32</td></tr>
<tr><td>w1</td><td colspan="16">Length = 12</td><td colspan="16">Message Type = 2</td></tr>
<tr><td>w2</td><td colspan="32">OTA Descriptor Address</td></tr>
</table>

OTA Descriptor Address	MRAM address where to build OTA Descriptor (Size = 9 words = 36 Bytes)

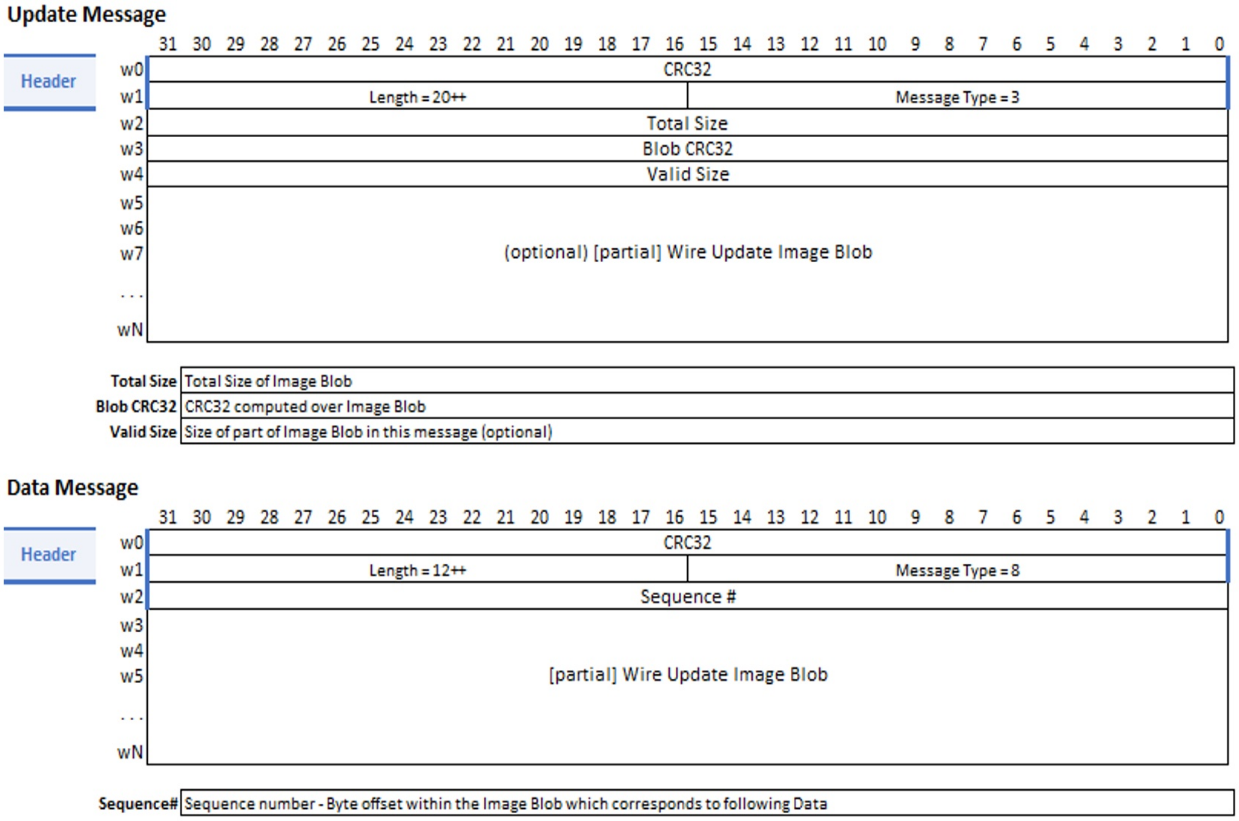
5.1.4 Wired Download (UPDATE and DATA) Messages

Any upgrade on the device comprises of an UPDATE message, which describes the nature and size of an upgrade. The UPDATE message is then followed by zero or more DATA messages to send the actual image blob. After all the data is received, SBL verifies the integrity and validates per prevalent security policy.

To avoid corrupting existing MRAM space with corrupted/unverified downloads, the image blobs are downloaded to SRAM, and written to MRAM only after verification for integrity and authentication. This implies that individual wired downloads are limited in size based on the available SRAM. SBL only uses DTCM for this purpose, and the amount of available memory for this could be further reduced, if some part of the same is being reserved by the application (using INFO0). Bigger size blobs can still be transferred by splitting them accordingly[1].

[1] Tools bundled with AmbiqSuite SDK provide means to handle disassembly/reassembly of large size images transparently.

Figure 5-7: Wired Download (UPDATE and DATA) Messages



Update Message

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Header	w0	CRC32	
	w1	Length = 20++	Message Type = 3
	w2	Total Size	
	w3	Blob CRC32	
	w4	Valid Size	
	w5 w6 w7 . . . wN	(optional) [partial] Wire Update Image Blob	

Total Size	Total Size of Image Blob
Blob CRC32	CRC32 computed over Image Blob
Valid Size	Size of part of Image Blob in this message (optional)

Data Message

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

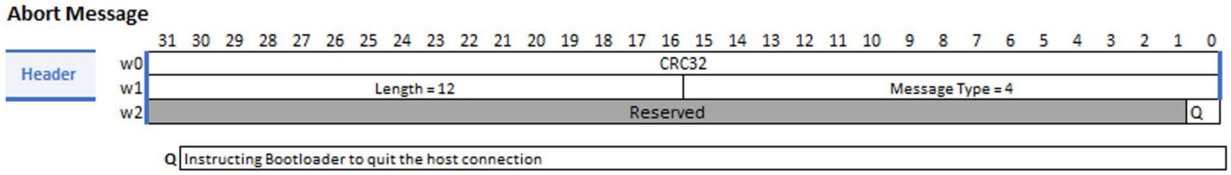
Header	w0	CRC32	
	w1	Length = 12++	Message Type = 8
	w2	Sequence #	
	w3 w4 w5 . . . wN	[partial] Wire Update Image Blob	

Sequence#	Sequence number - Byte offset within the Image Blob which corresponds to following Data

5.1.5 Termination (ABORT) Message

A Download in progress can be aborted using ABORT message. Host has a choice to continue the connection, or instruct SBL to quit the connection.

Figure 5-8: Termination (ABORT) Message



Abort Message

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

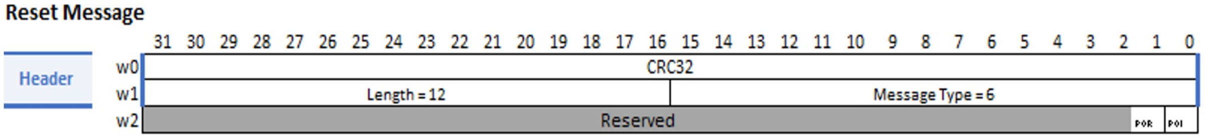
Header	w0	CRC32	
	w1	Length = 12	Message Type = 4
	w2	Reserved	Q

Q	Instructing Bootloader to quit the host connection

5.1.6 Reboot (RESET) Message

The Image blobs downloaded through UPDATE/DATA messages will have ‘O’ bit set to instruct SBL to schedule an OTA using the downloaded image. Multiple images can be combined together for the download step using this process. Actual Image upgrade is only initiated when a RESET message is received, as part of regular OTA processing by SBL.

Figure 5-9: Reboot (RESET) Message



POI	POI Reset mode
POR	POR Reset mode

SECTION 6 OEM Secondary Bootloader

The Apollo4 Secure Boot/Update flow allows the provision to incorporate an OEM secondary bootloader. A secondary bootloader can be used to supplement the core capabilities of the native Apollo4 SBL. Potential reasoning for implementing one can be:

- Need to support different authentication/encryption algorithms
- Need to support external memory device
- Other vendor specific enhancements

For designs incorporating a secondary bootloader, the latter replaces the OEM's main image. SBL treats the secondary bootloader as the main image and verifies/updates the same using native boot/update flow. The secondary bootloader can then implement the additional features before passing control to the OEM's main application image.

6.1 Device Programming Considerations for Secondary Bootloader

There are certain considerations while OEM programming a design with a secondary bootloader.

- Ensure **OTP_SECURITY:PLONEXIT** field is set to 1 during OEM provisioning in DM LCS.
  - Ensures OTP key bank will remain accessible to be used for image verification purposes.
  - It also allows Secondary bootloader to implement page lockouts for Read/Write protection by clearing additional bits in the **MCUCTRL:FLASHWPROT*** and **MCUCTRL:FLASHRPROT*** words.

6.2 Secondary Bootloader Implementation Considerations

6.2.1 Asset Protections

It is the responsibility of the secondary bootloader code to do the following before transferring to the main image to ensure proper security. It needs to "lock" the OTP keybank and restrict further access to flash protection registers, by asserting the protection lock by writing 1 to PROTLOCK bit in register REG_MCU_CTRL_BOOTLOADER.

6.2.2 Debugger Support

If Debugger Support is enabled for main image, but disabled during the Secondary Bootloader phase (Controlled by OTP_DCU_DISABLEOVERRIDE), Secondary Bootloader needs to implement additional logic to check if a halt is requested by the debugger after the bootloader, and if so, halt the processor to allow a debugger to connect, after re-enabling the debugger using REG_MCU_CTRL_BOOTLOADER_DEBUGGER.

- Checking for a Halt request from Debugger – Check if Least Significant Bit of register REG_MCU_CTRL_SCRATCH0 is set
- If set, clear the bit and halt the processor using DHCSR register.

6500 River Place Boulevard, Building 7, Suite 200, Austin, TX 78730
www.ambiq.com
sales@ambiq.com
+1 (512) 879-2850