Sahara Protocol

Specification

80-N1008-1 J

November 3, 2014

Submit technical questions at:
https://support.cdmatech.com/

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Contents

1 Introduction 6

1.1 Purpose 6
1.2 Conventions 6
1.3 References 6
1.4 Technical assistance 6
1.5 Acronyms 7

2 Overview 8

3 Interface 10

3.1 Overview 10
3.2 Commands 10
3.2.1 Hello packet 12
3.2.2 Hello Response packet 13
3.2.3 Read Data packet 14
3.2.4 End of Image Transfer packet 15
3.2.5 Done packet 15
3.2.6 Done Response packet 16
3.2.7 Reset packet 16
3.2.8 Reset Response packet 16
3.2.9 Memory Debug packet 17
3.2.10 Memory Read packet 17
3.2.11 Command Ready packet 18
3.2.12 Command Switch Mode packet 18
3.2.13 Command Execute packet 19
3.2.14 Command Execute Response packet 19
3.2.15 Command Execute Data packet 20
3.2.16 64bit Memory Debug packet 20
3.2.17 64bit Memory Read packet 21
3.2.18 64bit Read Data packet 22
3.3 Status codes 23

4 Operation 25

4.1 Overview 25
4.2 Successful Image Transfer sequence 25
4.3 Successful Memory Debug sequence 28
4.4 Successful Command sequence 30
4.5 Protocol implementation 32
4.5.1 Target state machine 32
4.5.2 Host state machine 38

4.6 Error handling 42
4.7 Parallel image transfers 42
    4.7.1 Single host, multiple targets 42
    4.7.2 Multiple hosts, single target 42
    4.7.3 Single host, single target, parallel images 42

Figures

Figure 3-1 Sahara packet structures 10
Figure 4-1 Successful Sahara Image Transfer sequence 27
Figure 4-2 Successful Sahara Memory Debug sequence 29
Figure 4-3 Successful Sahara Command sequence 31
Figure 4-4 Sahara state machine (target side) 34
Figure 4-5 Sahara state machine (target side) – Receive Image 35
Figure 4-6 Sahara state machine (target side) – Memory Debug mode 36
Figure 4-7 Sahara state machine (target side) – Command mode 37
Figure 4-8 Sahara state machine (host side) 39
Figure 4-9 Sahara state machine (host side) – Command mode 40
Figure 4-10 Sahara state machine (host side) – Memory Debug mode 41

Tables

Table 1-1 Reference documents and standards 6
Table 3-1 Commands 10
Table 3-2 Hello packet 12
Table 3-3 Modes supported 13
Table 3-4 Hello Response packet 13
Table 3-5 Read Data packet 14
Table 3-6 End of Image Transfer packet 15
Table 3-7 Done packet 15
Table 3-8 Done Response packet 16
Table 3-9 Reset packet 16
Table 3-10 Reset Response packet 16
Table 3-11 Memory Debug packet 17
Table 3-12 Memory Read packet 17
Table 3-13 Command Ready packet 18
Table 3-14 Command Switch Mode packet 18
Table 3-15 Command Execute packet 19
Table 3-16 Supported client commands 19
Table 3-17 Command Execute Response packet 19
Table 3-18 Command Execute Data packet 20
Table 3-19 Memory Debug packet 20
Table 3-20 Memory Read packet 21
Table 3-21 Read Data packet 22
Table 3-22 Status and error codes 23

Revision history

Revision	Date	Protocol revision	Description
A	Apr 2010	1.0	Initial release
B	May 2010	1.0	Defined additional status and error codes
C	May 2010	1.0	Fixed incorrect value in Done Response packet table
D	Jul 2010	2.0	Added Memory Debug support and updated Hello and Hello Response packets
E	Aug 2010	2.1	Added Command mode
F	Jan 2011	2.2	Added Command Execute commands to switch to DMSS download protocol and Streaming download protocol
G	Feb 2011	2.3	Added support for switching modes while in Memory Debug; fixed diagram for Command Mode – Host
H	Dec 2011	2.4	Removed check for image ID type; added command to read debug data, to get software version in SBL; returned hashes from APPS/MBA/MSS segments by OEM PK Hash; removed MSM reset; added image type check for SBL1/eDL type.
	Jan 2013	2.5	Added error code of SAHARA_NAK_IMAGE_AUTH_FAILURE
J	Nov 2014	2.8	Added command to support 64-bit read data packet

Note: There is no Rev. I, O, Q, S, X, or Z per Mil. standards.

1 Introduction

1.1 Purpose

This document provides information on the Sahara protocol, which is used to transfer data to and from memory. It describes the Sahara packet structures, packet flows, and intended use.

NOTE: Sahara does not provide a mechanism for authenticating/validating the data sent using the protocol. Such mechanisms are beyond the scope of the protocol and can be implemented in conjunction with this protocol as data is being transferred.

1.2 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Code variables appear in angle brackets, e.g., `<number>`.

Commands to be entered appear in a different font, e.g., **`copy a:*.* b:`**.

Shading indicates content that has been added or changed in this revision of the document.

1.3 References

Reference documents are listed in Table 1-1. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 1-1 Reference documents and standards

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Application Note: Enable Secure Boot on MSM8974 ASICs*	80-NA157-20
Standards		
S1	*Mobile Station-Base Station Compatibility Standard for Dual-mode Wideband Spread Spectrum Cellular System*	TIA/EIA Interim Standard IS-95-A May 1995

1.4 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://support.cdmatech.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

1.5 Acronyms

For definitions of terms and abbreviations, see [Q1].

2 Overview

The Sahara protocol is designed primarily for transferring software images from a host to a target. It provides a simple mechanism for requesting data to be transferred over any physical link.

The protocol supports two basic packet types: command packets and data packets. Command packets are sent between the host and the target to set up transfers of data packets.

Target-driven protocol

The protocol minimizes data transfer overhead by minimizing the number of command packets sent between the host and the target. This is accomplished by making the protocol completely target-driven and by having the target perform all data processing. The host simply waits for a data transfer request, which contains the following information:

- The data image to transfer
- The offset into the image to start reading from
- The data transfer length

The host does not need to process or extract any information from the actual image data – it simply sends the image data as "raw" data to the target, without any packet header attached to the packet. Because the target initiates the data transfer request, it knows exactly how much data to receive. This enables the host to send data without a packet header, and the target to directly receive and store the data.

The target requests data from the host as needed. The first data item it requests is the image header for a given image transfer. Once the target has processed the image header, it knows the location and size of each data segment in the image. The image header also specifies the destination address of the image in the target memory. With this information, the target can request data from the host for each segment and directly transfer the data to the appropriate location in the target memory.

Packet processing

The protocol minimizes packet processing by relying on the physical transport layer to provide reliable transfer of data. No framing, HDLC (High-level Data Link Control) encoding, or CRC (Cyclic Redundancy Check) is applied to the packets at the protocol level.

Each command packet type has a well-defined structure which minimally contains a command ID and packet length. Using this information, the length of each command packet can be validated by comparing the length of the command packet received to either of two values:

- The expected packet length for the given command ID
- The length field contained in the packet itself

NOTE: Sahara can easily be extended to support command packet validation by adding a CRC field to the end of each packet. Data packets can also be validated for data integrity using various authentication methods; however, this is beyond the scope of the protocol.

Synchronous communication

The protocol assumes that all communication between the host and the target is completely synchronous. Each command packet sent from the target to the host is acknowledged with a command or data packet sent from the host back to the target. Similarly, each command packet sent from the host to the target is acknowledged with a command or data packet.

Although the link between the host and the target is expected to be reliable, if an error occurs during the transmission of a command packet from the host to the target, and as a result the target receives an erroneous packet, the target sends the host an error response and exits gracefully.

Timer mechanisms can be implemented on both the host and the target to support the retransmission of packets in case of transmission failures. However, the implementation of such mechanisms is outside the scope of the protocol – it specifies only what happens when unexpected or erroneous packets are received on the target side.

Extensibility

The protocol defines a fixed set of command structures and packet flows. However, it can easily be extended to support additional command structures and state transitions (as described later in this document).

3 Interface

3.1 Overview

The Sahara protocol defines two types of packets:

- Command packets
- Data packets

The structure of these packets is shown in Figure 3-1.

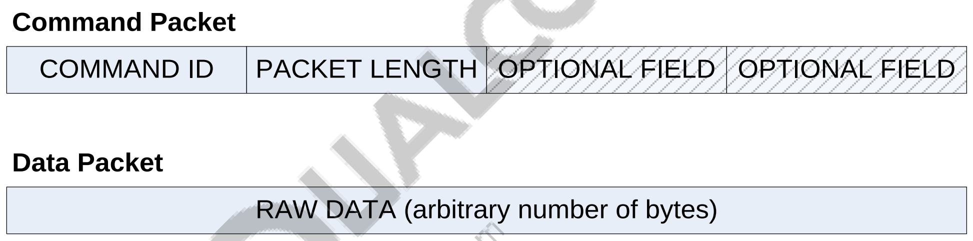

Figure 3-1 Sahara packet structures

Command packets contain at minimum a command ID and packet length. Depending on the command, the packet may contain additional command-specific fields.

NOTE: The command packet structure enables future revisions of the protocol to easily add fields to the end of a packet type, while preserving compatibility with the packet structure of previous protocol versions.

3.2 Commands

The commands used in the Sahara command packets are listed in Table 3-1.

Table 3-1 Commands

ID Value (HEX)	Command	Sent by	Minimum protocol revision	Description
0x00	—	—	—	Invalid
0x01	Hello	Target	1.0	Initialize connection and protocol
0x02	Hello Response	Host	1.0	Acknowledge connection and protocol sent by target; also used to set mode of operation for target to execute in
0x03	Read Data	Target	1.0	Read specified number of bytes from host for a given image

ID Value (HEX)	Command	Sent by	Minimum protocol revision	Description
0x04	End of Image Transfer	Target	1.0	Indicate to host that a single image transfer is complete; also used to indicate a target failure during an image transfer
0x05	Done	Host	1.0	Acknowledgement from host that a single image transfer is complete
0x06	Done Response	Target	1.0	Indicate to host: ▪ Target is exiting protocol ▪ Whether or not target expects to re-enter protocol to transfer another image
0x07	Reset	Host	1.0	Instruct target to perform a reset
0x08	Reset Response	Target	1.0	Indicate to host that target is about to reset
0x09	Memory Debug	Target	2.0	Indicate to host that target has entered a debug mode where it is ready to transfer its system memory contents
0x0A	Memory Read	Host	2.0	Read specified number of bytes from target's system memory, starting from a specified address
0x0B	Command Ready	Target	2.1	Indicate to host that target is ready to receive client commands
0x0C	Command Switch Mode	Host	2.1	Indicate to target to switch modes: ▪ Image Transfer Pending mode ▪ Image Transfer Complete mode ▪ Memory Debug mode ▪ Command mode
0x0D	Command Execute	Host	2.1	Indicate to target to execute a given client command
0x0E	Command Execute Response	Target	2.1	Indicate to host that target has executed client command; also used to indicate status of executed command
0x0F	Command Execute Data	Host	2.1	Indicate to target that host is ready to receive data resulting from executing previous client command
0x10	64bit Memory Debug	Target	2.5	Indicate to host that target has entered a debug mode where it is ready to transfer its 64-bit system memory contents
0x11	64bit Memory Read	Host	2.5	Read specified number of bytes from target's system memory, starting from a specified 64-bit address
0x12	64bit Read Data	Target	2.8	Read specified number of bytes from host for a given 64-bit image
All others	—	—	—	Invalid

NOTE: The minimum protocol version indicates the lowest protocol version that supports the given command.

3.2.1 Hello packet

When the target sends a Hello packet, it uses the format shown in Table 3-2.

Table 3-2 Hello packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x00000001.
Length	4	Length of packet (in bytes)	Target sets this field to 0x00000030.
Version Number	4	Version number of this protocol implementation	Target sets this field to indicate the current version of protocol that it is running. The value is 0x00000002.
Version Compatible	4	Lowest compatible version	Target sets this field to indicate the lowest version of the protocol that it supports.
Command Packet Length	4	Maximum command packet length (in bytes) the protocol supports	Target sets this based on buffer used in protocol implementation.
Mode	4	Expected mode of target operation	Target sets this based on the mode of operation to execute in.
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use

The Hello packet is the first packet that the target sends to the host. If the host receives any other packet, it sends a reset command to the target.

When the host receives a valid Hello packet, it first verifies that the protocol running on the target is compatible with the protocol running on the host. If the protocols are mismatched, the host sends a reset command to the target.

The target also sends the maximum length of the command packet that it supports – the host uses this information to avoid sending more bytes than the target can support in the receiving command buffer.

The target also sends the mode of operation it expects to enter based on the bootup sequence. Sahara currently supports the following modes shown in Table 3-3:

- Image Transfer Pending mode – Transfer image from the host; after completion, the host should expect another image transfer request.
- Image Transfer Complete mode – Transfer image from the host; after completion, the host should not expect another image transfer request
- Memory Debug mode – The host should prepare to receive a memory dump from the target

- Command mode – The host executes operations on the target by sending the appropriate client command to the Sahara client running on the target. The client command is interpreted by the Sahara client and the corresponding response sent upon execution of the given command.

Table 3-3 Modes supported

Mode	Mode ID (HEX)	Description
SAHARA_MODE_IMAGE_TX_PENDING	0x0	Image Transfer Pending mode
SAHARA_MODE_IMAGE_TX_COMPLETE	0x1	Image Transfer Complete mode
SAHARA_MODE_MEMORY_DEBUG	0x2	Memory Debug mode
SAHARA_MODE_COMMAND	0x3	Command mode

3.2.2 Hello Response packet

When the host sends a Hello Response packet, it uses the format shown in Table 3-4.

Table 3-4 Hello Response packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x00000002.
Length	4	Length of packet (in bytes)	Host sets this field to 0x00000030.
Version Number	4	Version number of this protocol implementation	Host sets this field to indicate maximum version of the protocol that host supports.
Version Compatible	4	Lowest compatible version	Host sets this field to indicate lowest version of the protocol that it supports.
Status	4	Success or error code	Host sets this field based on the Hello packet received; if target protocol matches host and no other errors, a success value is sent.
Mode	4	Mode of operation for target to execute	Host sets this based on the mode of operation it wants target to execute in. By default, host will copy the mode received in the Hello packet.
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use
RESERVED	4	Unused	Reserved for future use

Once the host validates the protocol running on the target, it sends the following information to the target:

- The protocol version that it is running
- The minimum protocol version it supports
- The mode of operation

The host sets the packet Status field to "success" if no errors occur on the host side. Once the target receives this packet, it can proceed with data transfer requests or memory debug.

3.2.3 Read Data packet

When the target sends a Read Data packet, it uses the format shown in Table 3-5.

Table 3-5 Read Data packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x00000003.
Length	4	Length of packet (in bytes)	Target sets this field to 0x00000014.
Image ID[1]	4	ID of the image to be transferred	Target sets this field to correspond to the image it wants host to transfer.
Data Offset	4	Offset into the image file to start transferring data from	Target sets this field to the offset (in bytes) into the image file that it wants to retrieve data from.
Data Length	4	Number of bytes target wants to transfer from the image	Target sets this field to the number of bytes it wants to read from the image file.

[1]From revision 2.4, image ID checking is removed for B-family chips.

To initiate an image transfer, the target fills this packet with the image ID corresponding to the image it wants to receive. The target also sends the offset into the image file and the length of the data (in bytes) it wants to read from the image.

This packet serves as a generic data transfer packet when any image data is to be transferred from the host to the target. It allows flexibility in the way the image is transferred from the host to the target. Because the target controls what data gets transferred, it can determine what parts of the image get transferred and in what order. The host does not need to know anything about the structure of the image; it only needs to open the file and start transferring the data to the target based on the parameters specified in the packet. This gives the target complete control over how the images are transferred and processed.

As soon as the host receives this packet, the host is expected to respond with a data packet. The data packet must contain just the image data and must be of the length specified in the Read Data packet.

Several error conditions can occur if the host receives any of the following in a Read Data packet:

- Invalid or unsupported image ID
- Invalid data offset
- Invalid data length

If any of the above fields are invalid, or if any other error occurs on the host, the host can send a data packet with a length that does not match what the target was expecting. The resulting error forces the target to send an End of Image Transfer packet with an error code in the Status field (see packet structure in Table 3-6). This transaction enables both the target and the host to enter an error handling state.

The current version of the protocol can be implemented by a state machine where any error that occurs results in the host sending a Reset packet (see Section 4.5).

3.2.4 End of Image Transfer packet

When the target sends an End of Image Transfer packet, it uses the format shown in Table 3-6.

Table 3-6 End of Image Transfer packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x00000004.
Length	4	Length of packet (in bytes)	Target sets this field to 0x00000010.
Image ID[1]	4	ID of an image that was being transferred	Target sets this field to correspond to the image it was transferring or processing.
Status	4	Success or error code	Target sets this field based on whether an image was transferred successfully or not.

[1]From revision 2.4, image ID checking is removed for B-family chips.

If an image transfer is successfully completed, the target sends the host an End of Image Transfer packet with a “success” status. The target then waits for the host to send a Done packet.

If any error occurs during the transfer or processing of the image data, the status is set to the corresponding error code, and the target waits for a different command packet.

The current version of the protocol can be implemented by a state machine where the target assumes the host is always going to send a Reset packet after an error is sent in the End of Image Transfer packet (see Section 4.5). However, the protocol allows the flexibility of other command packets to be sent from the host to the target in response to the End of Image Transfer error packet.

3.2.5 Done packet

When the host sends a Done packet, it uses the format shown in Table 3-7.

Table 3-7 Done packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x00000005.
Length	4	Length of packet (in bytes)	Host sets this field to 0x00000008.

If the host receives an End of Image Transfer packet with a “success” status, the host sends a Done packet to indicate to the target that it can exit the protocol and continue executing.

If the target wishes to transfer another image from the host, it must re-initiate the protocol by starting with another Hello packet.

3.2.6 Done Response packet

When the target sends a Done Response packet, it uses the format shown in Table 3-8.

Table 3-8 Done Response packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x00000006.
Length	4	Length of packet (in bytes)	Target sets this field to 0x0000000C.
Image Transfer Status	4	Indicates whether target is expecting to receive another image or not	Target sets this field to correspond to whether all image transfers are complete, or an image transfer is pending.

If the target receives a Done packet, it responds with a Done Response packet containing the image transfer status:

- If all the images have been transferred, the target sends a "complete" status to enable the host to exit the protocol.
- If all the images have not been transferred, the target sends a "pending" status. The target will assume the host will continue to execute the protocol and wait for another Hello packet to arrive.

3.2.7 Reset packet

When the host sends a Reset packet, it uses the format shown in Table 3-9.

Table 3-9 Reset packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x00000007.
Length	4	Length of packet (in bytes)	Host sets this field to 0x00000008.

The host sends a Reset packet whenever it wants to reset the target.

The target services a Reset request only if it is in a state where reset requests are valid. If the target receives an invalid reset request, the target sends an error in an End of Image Transfer packet.

From revision 2.4, target reset is no longer supported for B family chips.

3.2.8 Reset Response packet

When the target sends a Reset Response packet, it uses the format shown in Table 3-10.

Table 3-10 Reset Response packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x00000008.
Length	4	Length of packet (in bytes)	Target sets this field to 0x00000008.

If the target receives a valid reset request, it sends a Reset Response packet just before it resets.

The purpose of this response is for the target to acknowledge to the host that the target received the reset request. If the host does not receive the Reset Response command from the target (or receives a different command), it can attempt to resend the request.

3.2.9 Memory Debug packet

When the target sends a Memory Debug packet, it uses the format shown in Table 3-11.

Table 3-11 Memory Debug packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x00000009.
Length	4	Length of packet (in bytes)	Target sets this field to 0x00000010.
Memory Table Address	4	Address of memory debug table	Target sets this field to the address in memory that stores the memory debug table.
Memory Table Length	4	Length in bytes of memory debug table	Target sets this field to the length of the memory debug table.

The target initiates a memory dump by sending the host a Memory Debug packet. This packet contains the address and length of the memory debug table. The memory debug table is a listing of memory locations that can be accessed and dumped to the host. Each entry in the table is a data structure with the following type:

```
struct sahara_packet_memory_debug
{
  uint32 command;                /* command ID */
  uint32 length;                 /* packet length incl command and
length */
  uint32 memory_table_addr;      /* location of memory region table */
  uint32 memory_table_length;    /* length of memory table */
};
```

The length of the memory table is the size of the structure multiplied by the number of entries in the table.

Given the memory table address and length, the host can issue a Memory Read to retrieve the table. Once the host receives the memory table information, it can decode each entry and issue Memory Read requests to dump each memory location.

3.2.10 Memory Read packet

When the host sends a Memory Read packet, it uses the format shown in Table 3-12.

Table 3-12 Memory Read packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x0000000A.
Length	4	Length of packet (in bytes)	Host sets this field to 0x00000010.

Field	Length (bytes)	Description	Value release 2.1
Memory Address	4	Memory location to read from	Host sets this field to the address in memory that it wants to read from.
Memory Length	4	Length in bytes of memory to read	Host sets this field to the memory length it wishes to read.

The host repeatedly issues Memory Read commands for each section of memory it wishes to dump. The accessible regions are defined in the memory debug table. For each Memory Read command received, the target verifies that the specified memory (address and length) is accessible and responds with a Raw Data packet. The content and length of the Raw Data packet are the memory dump starting from the memory address and length specified in the Memory Read packet. The memory debug table can also be read using a Memory Read command by setting the address and length to the values specified in the Memory Debug packet.

If any error occurs on the target, an End of Image Transfer packet is sent with the corresponding error code. The host must distinguish the data sent from the target to recognize whether it is actual memory data or an End of Image Transfer packet. One way is to always request a memory length that does not equal the size of the End of Image Transfer packet.

On completion of a successful memory dump, the expected behavior is for the host to issue a reset command. However, the protocol does not force this implementation.

3.2.11 Command Ready packet

When the target sends a Command Ready packet, it uses the format shown in Table 3-13.

Table 3-13 Command Ready packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x0000000B.
Length	4	Length of packet (in bytes)	Host sets this field to 0x00000008.

This packet is sent from the target to the host to indicate the target is ready to execute client commands via the Command Execute packet.

3.2.12 Command Switch Mode packet

When the host sends a Command Switch Mode packet, it uses the format shown in Table 3-14.

Table 3-14 Command Switch Mode packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x0000000C.
Length	4	Length of packet (in bytes)	Host sets this field to 0x0000000C.
Mode	4	Mode of operation for target to execute	Host sets this based on the mode of operation it wants target to execute in.

The host sends this packet when it wishes the target to switch to another mode.

3.2.13 Command Execute packet

When the host sends a Command Execute packet, it uses the format shown in Table 3-15.

Table 3-15 Command Execute packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x0000000D.
Length	4	Length of packet (in bytes)	Host sets this field to 0x0000000C.
Client Command	4	Client command to execute	Host sets this based on the client command it wants target to execute.

The host sends this packet to execute the given client command on the target. The supported client commands are shown in Table 3-16. If the client command successfully executes, the target sends a Command Execute Response packet. If an error occurs, the target sends an End of Image Transfer packet with the corresponding error code.

Table 3-16 Supported client commands

Client Command ID Value (HEX)	Command	Minimum protocol revision	Description
0x00	No Operation	2.1	No operation is performed on target.
0x01	Serial Number Read	2.1	Retrieve serial number from target.
0x02	MSM H/W ID Read	2.1	Retrieve chip hardware ID from target.
0x03	Public Key Hash Read[1]	2.1	Retrieve hash of the root of trust certificate from target.
0x06	Read debug data	2.4	Retrieve error log from the supported target.
0x07	Get software version SBL	2.4	Provide the anti-roll back version supported for SBL segment.

[1]From revision 2.4, PK Hash returns three hashes for APPS, MBA, and MSS code segments for B-family chips.

The execution of the client commands is not defined by the protocol. That is outside the scope of the protocol. The handling of the commands and the response data is provided by the Sahara client on the target side. The Sahara protocol provides a unified set of client command IDs that can be extended in the future.

3.2.14 Command Execute Response packet

When the target sends a Command Execute Response packet, it uses the format shown in Table 3-17.

Table 3-17 Command Execute Response packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x0000000E.
Length	4	Length of packet (in bytes)	Target sets this field to 0x00000010.
Client Command	4	Client command to execute	Target sets this based on the client command it executed.

Field	Length (bytes)	Description	Value release 2.1
Response Length	4	Number of bytes for response data	Target sets this based on the length of the response data for the last executed client command.

The target sends this packet if it successfully executed the client command. The length of the data response is sent to allow the host to prepare to receive the data.

3.2.15 Command Execute Data packet

When the host sends a Command Execute Data packet, it uses the format shown in Table 3-18.

Table 3-18 Command Execute Data packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x0000000F.
Length	4	Length of packet (in bytes)	Host sets this field to 0x0000000C.
Client Command	4	Client command executed	Host sets this based on the last client command executed.

The host sends this packet if the response length received in the Command Execute Response packet was greater than 0. This packet indicates to the target to send the response data in a Raw Data packet. Upon receiving this packet, the target sends the response data.

3.2.16 64bit Memory Debug packet

NOTE: This section was added to this document revision.

When the target sends a 64bit Memory Debug packet, it uses the format shown in Table 3-19.

Table 3-19 Memory Debug packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x00000010.
Length	4	Length of packet (in bytes)	Target sets this field to 0x00000010.
Memory Table Address	8	Address of memory debug table	Target sets this field to the 64-bit address in memory that stores the memory debug table.
Memory Table Length	8	Length in bytes of memory debug table	Target sets this field to the 64-bit length of the memory debug table.

The target initiates a memory dump by sending the host a 64bit Memory Debug packet. This packet contains the 64-bit address and length of the memory debug table. The memory debug table is a listing of memory locations that can be accessed and dumped to the host. Each entry in the table is a data structure with the following type:

```
struct sahara_packet_memory_64bits_debug
{
  uint32 command;                 /* command ID */
  uint32 length;                /* packet length incl command and
length */
  uint64 memory_table_addr;     /* location of memory region table */
  uint64 memory_table_length;   /* length of memory table */
};
```

The memory table length is the structure size multiplied by the number of entries in the table.

Given the memory table address and length, the host can issue a Memory Read to retrieve the table. Once the host receives the memory table information, it can decode each entry and issue Memory Read requests to dump each memory location.

3.2.17 64bit Memory Read packet

NOTE: This section was added to this document revision.

When the host sends a Memory Read packet, it uses the format shown in Table 3-20.

Table 3-20 Memory Read packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Host sets this field to 0x00000011.
Length	4	Length of packet (in bytes)	Host sets this field to 0x00000010.
Memory Address	8	Memory location to read from	Host sets this field to the 64-bit address in memory that it wants to read from.
Memory Length	8	Length in bytes of memory to read	Host sets this field to the 64bit memory length it wishes to read.

The host repeatedly issues 64bit Memory Read commands for each section of memory it wishes to dump. The accessible regions are defined in the memory debug table. For each 64bit Memory Read command received, the target verifies that the specified memory (address and length) is accessible and responds with a Raw Data packet.

The content and length of the Raw Data packet are the memory dump starting from the memory address and length specified in the 64bit Memory Read packet. The memory debug table can also be read using a 64bit Memory Read command by setting the address and length to the values specified in the 64bit Memory Debug packet.

If any error occurs on the target, an End of Image Transfer packet is sent with the corresponding error code. The host must distinguish the data sent from the target to recognize whether it is actual memory data or an End of Image Transfer packet. One way is to always request a memory length that does not equal the size of the End of Image Transfer packet.

On completion of a successful memory dump, the expected behavior is for the host to issue a reset command. However, the protocol does not force this implementation.

3.2.18 64bit Read Data packet

NOTE: This section was added to this document revision.

When the target sends a 64bit Read Data packet, it uses the format shown in Table 3-21.

Table 3-21 Read Data packet

Field	Length (bytes)	Description	Value release 2.1
Command	4	Command identifier code	Target sets this field to 0x00000012.
Length	4	Length of packet (in bytes)	Target sets this field to 0x00000020.
Image ID[1]	8	ID of the image to be transferred	Target sets this field to correspond to the image it wants host to transfer.
Data Offset	8	Offset into the image file to start transferring data from	Target sets this field to the offset (in bytes) into the image file that it wants to retrieve data from.
Data Length	8	Number of bytes target wants to transfer from the image	Target sets this field to the number of bytes it wants to read from the image file.

[1]From revision 2.4, image ID checking is removed for B-family chips.

To initiate a 64-bit image transfer, the target fills this packet with the image ID corresponding to the 64-bit image it wants to receive. The target also sends the 64-bit offset into the image file and the length of the data (in bytes) it wants to read from the image.

This packet serves as a generic data transfer packet when any 64-bit image data is to be transferred from the host to the target. It allows flexibility in the way the image is transferred from the host to the target.

Because the target controls what data gets transferred, it can determine what parts of the image get transferred and in what order. The host does not need to know anything about the structure of the image; it only needs to open the file and start transferring the data to the target based on the parameters specified in the packet. This gives the target complete control over how the images are transferred and processed.

As soon as the host receives this packet, the host is expected to respond with a data packet. The data packet must contain just the image data and must be of the length specified in the Read Data packet.

Several error conditions can occur if the host receives any of the following in a 64bit Read Data packet:

- Invalid or unsupported image ID
- Invalid data offset
- Invalid data length

If any of the above fields are invalid, or if any other error occurs on the host, the host can send a data packet with a length that does not match what the target was expecting. The resulting error forces the target to send an End of Image Transfer packet with an error code in the Status field (see packet structure in Table 3-6). This transaction enables both the target and the host to enter an error handling state.

The current version of the protocol can be implemented by a state machine where any error that occurs results in the host sending a Reset packet (see Section 4.5).

3.3 Status codes

Table 3-22 lists the status and error codes supported by the protocol.

Table 3-22 Status and error codes

Status/error code (hexadecimal)	Minimum protocol version	Description
0x00	1.0	Success
0x01	1.0	Invalid command received in current state
0x02	1.0	Protocol mismatch between host and target
0x03	1.0	Invalid target protocol version
0x04	1.0	Invalid host protocol version
0x05	1.0	Invalid packet size received
0x06	1.0	Unexpected image ID received[1]
0x07	1.0	Invalid image header size received
0x08	1.0	Invalid image data size received
0x09	1.0	Invalid image type received
0x0A	1.0	Invalid transmission length
0x0B	1.0	Invalid reception length
0x0C	1.0	General transmission or reception error
0x0D	1.0	Error while transmitting READ_DATA packet
0x0E	1.0	Cannot receive specified number of program headers
0x0F	1.0	Invalid data length received for program headers
0x10	1.0	Multiple shared segments found in ELF image
0x11	1.0	Uninitialized program header location
0x12	1.0	Invalid destination address
0x13	1.0	Invalid data size received in image header
0x14	1.0	Invalid ELF header received
0x15	1.0	Unknown host error received in HELLO_RESP
0x16	1.0	Timeout while receiving data
0x17	1.0	Timeout while transmitting data
0x18	2.0	Invalid mode received from host
0x19	2.0	Invalid memory read access
0x1A	2.0	Host cannot handle read data size requested
0x1B	2.0	Memory debug not supported
0x1C	2.1	Invalid mode switch
0x1D	2.1	Failed to execute command
0x1E	2.1	Invalid parameter passed to command execution
0x1F	2.1	Unsupported client command received
0x20	2.1	Invalid client command received for data response
0x21	2.4	Failed to authenticate hash table
0x22	2.4	Failed to verify hash for a given segment of ELF image
0x23	2.4	Failed to find hash table in ELF image

Status/error code (hexadecimal)	**Minimum protocol version**	**Description**
0x24	2.4	Target failed to initialize
0x25	2.5	Failed to authenticate generic image
All others	—	Invalid

[1]In 2.4 version, image ID checking is removed for B-family chips.

4 Operation

4.1 Overview

This chapter covers the following topics:

- How the Sahara protocol can be used to transfer an image from the host to the target, dump memory from the target to the host, or execute commands on the target and retrieve data
- An example state machine which implements the protocol and illustrates how to deal with errors that can occur in each mode (Image Transfer Pending, Image Transfer Complete, Memory Debug, Command)
- How a system can use the protocol to parallelize image transfer requests

4.2 Successful Image Transfer sequence

Figure 4-1 shows the packet flow for a successful Image Transfer sequence. The packet flow sequence is described below:

1. A Hello packet is sent from the target to the host to initiate the protocol with the mode set to either Image Transfer Pending or Image Transfer Complete (depending on the target's boot sequence).
2. Upon receiving the Hello packet and validating the protocol version running on the target, the host sends a Hello Response packet with a "success" status and the mode set to the mode received in the Hello packet.
3. Once the target receives the Hello Response, the target initiates the image transfer requests by sending Read Data packets. Each Read Data packet specifies which image the target wishes to receive and what part of the image to transfer.
4. During normal operation, the target first requests image header information which specifies the rest of the image, i.e., image size and where in system memory the image data is to be transferred). This request for the image header – which is sent as a Read Data request – requires the target to know the format of the image to be transferred. The protocol does not require the host to know anything about the image formats, allowing the host to simply read and transfer data from the image as requested by the target.

   If the image is a standalone binary image without any data segmentation, i.e., the data is entirely contiguous when stored as well as transferred to the target system memory, the target can request that the entire image data be sent in one transfer. If the image data is segmented and requires scattering of the data segments to noncontiguous system memory locations, the target can issue multiple Read Data requests to enable each data segment to be transferred directly to the respective destination address. This scattering information resides in the image header and is parsed by the target before issuing the Read Data requests.

5. Upon receiving a Read Data request, the host parses the image ID, data offset, and data length to transfer data from the corresponding image file. The host sends the data requested without any packet header. The target directly transfers this data to the destination address without any software processing or temporary buffering of the data in system memory. This is made possible by transferring the image header to the target and setting the receive buffer for the data to be just the destination address in system memory.
6. Once the target successfully receives all segments for an image, the target sends an End of Image Transfer packet with the image ID of the corresponding image, and a “success” status. This enables the host to stop reading and close the image file.
7. Upon receiving a successful End of Image Transfer, the host sends a Done packet to allow the target to exit the protocol.
8. Once the target receives the Done packet, the target sends a Done Response packet to the host. Within this packet, the target indicates whether it expects another image to be transferred. If another image is to be transferred to the target, the host can continue to run the protocol.

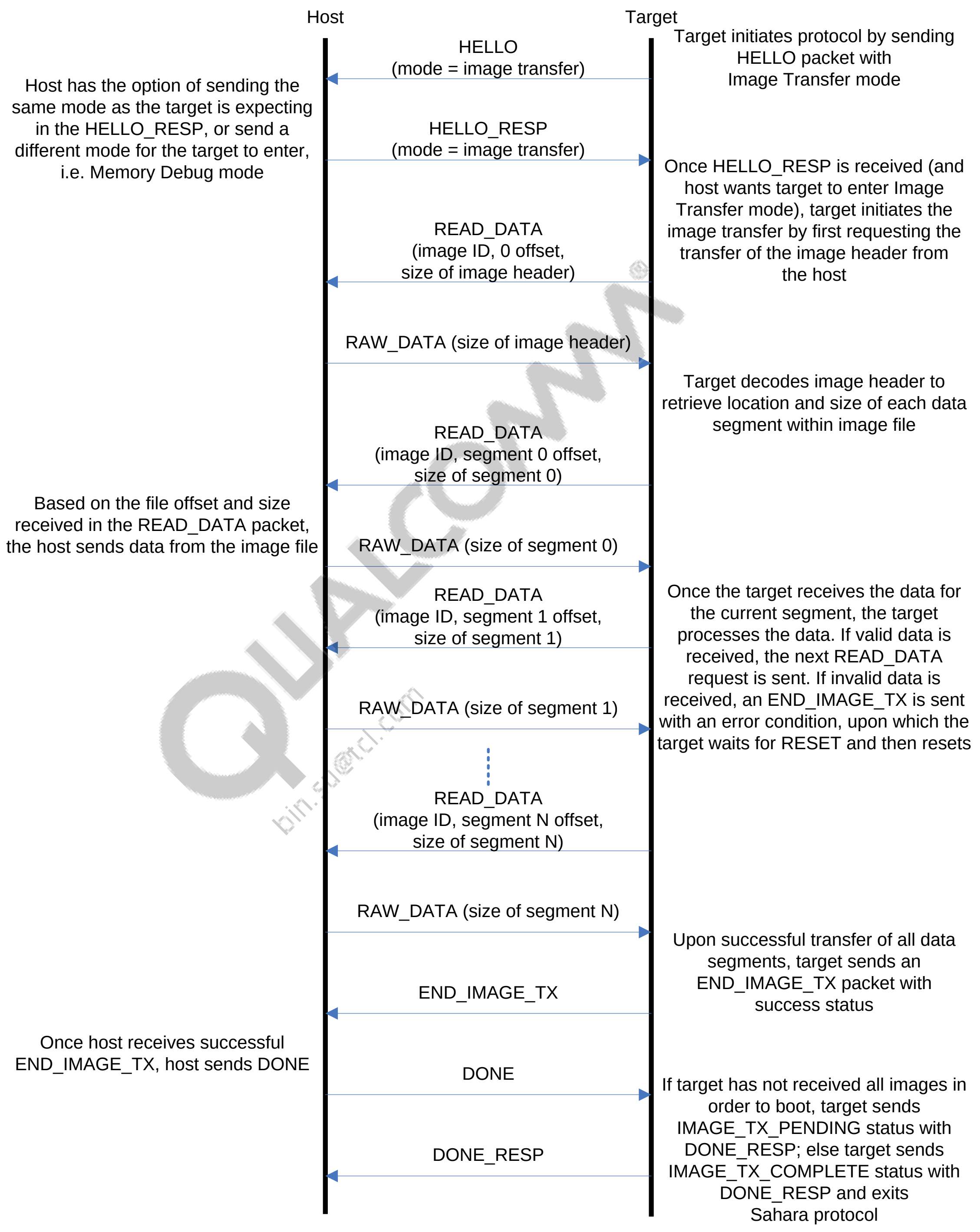
Host
Target
HELLO (mode = image transfer)
Target initiates protocol by sending HELLO packet with Image Transfer mode
Host has the option of sending the same mode as the target is expecting in the HELLO_RESP, or send a different mode for the target to enter, i.e. Memory Debug mode
HELLO_RESP (mode = image transfer)
Once HELLO_RESP is received (and host wants target to enter Image Transfer mode), target initiates the image transfer by first requesting the transfer of the image header from the host
READ_DATA (image ID, 0 offset, size of image header)
RAW_DATA (size of image header)
Target decodes image header to retrieve location and size of each data segment within image file
READ_DATA (image ID, segment 0 offset, size of segment 0)
Based on the file offset and size received in the READ_DATA packet, the host sends data from the image file
RAW_DATA (size of segment 0)
READ_DATA (image ID, segment 1 offset, size of segment 1)
Once the target receives the data for the current segment, the target processes the data. If valid data is received, the next READ_DATA request is sent. If invalid data is received, an END_IMAGE_TX is sent with an error condition, upon which the target waits for RESET and then resets
RAW_DATA (size of segment 1)
READ_DATA (image ID, segment N offset, size of segment N)
RAW_DATA (size of segment N)
Upon successful transfer of all data segments, target sends an END_IMAGE_TX packet with success status
END_IMAGE_TX
Once host receives successful END_IMAGE_TX, host sends DONE
DONE
If target has not received all images in order to boot, target sends IMAGE_TX_PENDING status with DONE_RESP; else target sends IMAGE_TX_COMPLETE status with DONE_RESP and exits Sahara protocol
DONE_RESP

Figure 4-1 Successful Sahara Image Transfer sequence

4.3 Successful Memory Debug sequence

Figure 4-2 shows the packet flow for a Memory Debug sequence. The packet flow sequence is described below:

1. A Hello packet is sent from the target to the host to initiate the protocol with mode set to Memory Debug.
2. Upon receiving the Hello packet and validating the protocol version running on the target, the host sends a Hello Response packet with a “success” status and mode set to Memory Debug.
3. Once the target receives the Hello Response, the target initiates the memory dump by sending a Memory Debug packet with the location and size of the memory debug table. This memory debug table specifies the memory regions that are accessible.
4. After receiving the Memory Debug packet, the host initiates a Memory Read packet to read the memory debug table and receives the table in a Raw Data packet.
5. The host proceeds to decode the table and issues Memory Reads for each accessible region. The data for each region is sent in a Raw Data packet.
6. Upon completion, the host issues a Reset to the target, upon which the target sends a Reset Response and proceeds to reset the target.

NOTE: In step 6, the host can alternatively send a Command Switch Mode packet to allow the target to switch modes and avoid a reset (this is not shown in Figure 4-2).

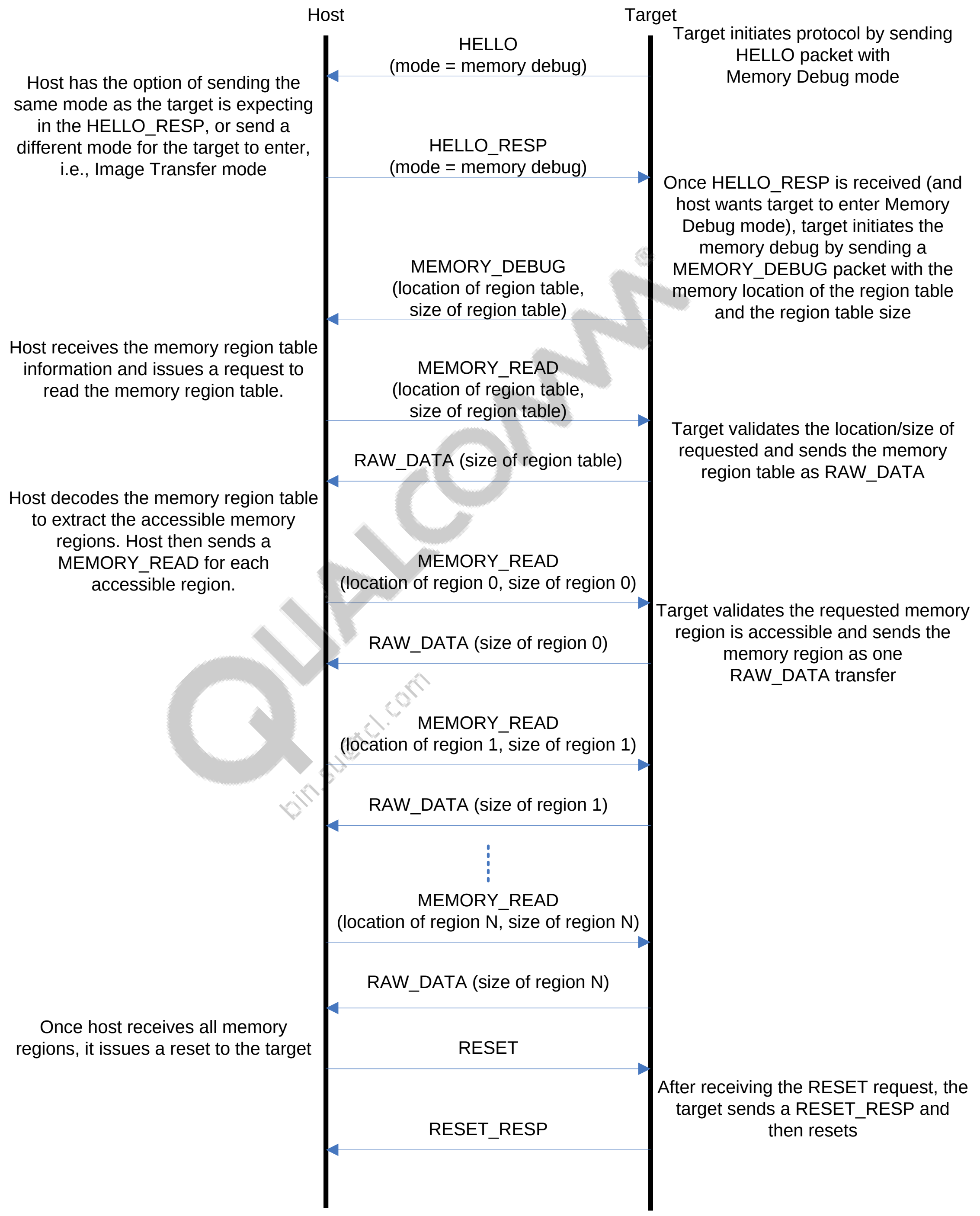
Host
Target
HELLO
(mode = memory debug)
Target initiates protocol by sending HELLO packet with Memory Debug mode
Host has the option of sending the same mode as the target is expecting in the HELLO_RESP, or send a different mode for the target to enter, i.e., Image Transfer mode
HELLO_RESP
(mode = memory debug)
Once HELLO_RESP is received (and host wants target to enter Memory Debug mode), target initiates the memory debug by sending a MEMORY_DEBUG packet with the memory location of the region table and the region table size
MEMORY_DEBUG
(location of region table, size of region table)
Host receives the memory region table information and issues a request to read the memory region table.
MEMORY_READ
(location of region table, size of region table)
Target validates the location/size of requested and sends the memory region table as RAW_DATA
RAW_DATA (size of region table)
Host decodes the memory region table to extract the accessible memory regions. Host then sends a MEMORY_READ for each accessible region.
MEMORY_READ
(location of region 0, size of region 0)
Target validates the requested memory region is accessible and sends the memory region as one RAW_DATA transfer
RAW_DATA (size of region 0)
MEMORY_READ
(location of region 1, size of region 1)
RAW_DATA (size of region 1)
MEMORY_READ
(location of region N, size of region N)
RAW_DATA (size of region N)
Once host receives all memory regions, it issues a reset to the target
RESET
After receiving the RESET request, the target sends a RESET_RESP and then resets
RESET_RESP

Figure 4-2 Successful Sahara Memory Debug sequence

4.4 Successful Command sequence

Figure 4-3 shows the packet flow for a Command sequence. The packet flow sequence is described below:

1. A Hello packet is sent from the target to the host to initiate the protocol. The target may or may not be in Command mode by default.
2. Upon receiving the Hello packet and validating the protocol version running on the target, the host sends a Hello Response packet with a “success” status and mode set to Command mode.
3. Once the target receives the Hello Response, the target initiates the Command sequence by sending a Command Ready packet.
4. After receiving the Command Ready packet, the host initiates a command by sending a Command Execute packet with a specific client command to execute.
5. Upon receiving this packet, the target executes the given client command and responds with a Command Execute Response packet. This packet indicates the response data length for the given executed client command.
6. Upon receiving the Command Execute Response, if the response data length is greater than 0, then the host proceeds to send a Command Execute Data packet to indicate it is ready to receive the data. Otherwise, the host proceeds to execute the next command.
7. Upon receiving the Command Execute Data packet, the target proceeds to send the response data in a Raw Data packet.
8. Once the host has sent all client commands to the target and received the resulting responses, it sends a Command Switch Mode packet to direct the target to switch out of Command mode and restart the protocol with a Hello packet.

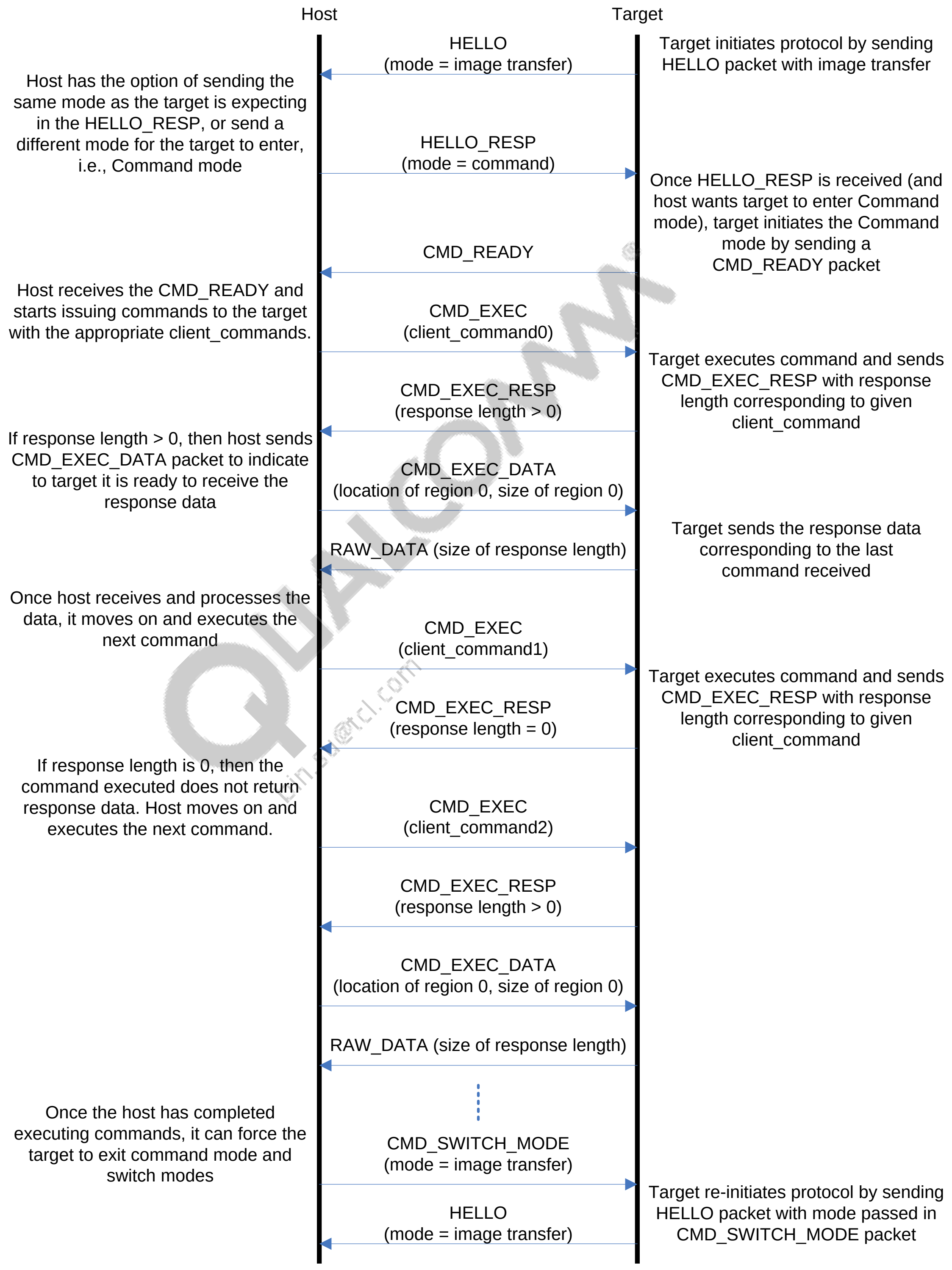
Host
Target
HELLO
(mode = image transfer)
Target initiates protocol by sending HELLO packet with image transfer
Host has the option of sending the same mode as the target is expecting in the HELLO_RESP, or send a different mode for the target to enter, i.e., Command mode
HELLO_RESP
(mode = command)
Once HELLO_RESP is received (and host wants target to enter Command mode), target initiates the Command mode by sending a CMD_READY packet
CMD_READY
Host receives the CMD_READY and starts issuing commands to the target with the appropriate client_commands.
CMD_EXEC
(client_command0)
Target executes command and sends CMD_EXEC_RESP with response length corresponding to given client_command
CMD_EXEC_RESP
(response length > 0)
If response length > 0, then host sends CMD_EXEC_DATA packet to indicate to target it is ready to receive the response data
CMD_EXEC_DATA
(location of region 0, size of region 0)
Target sends the response data corresponding to the last command received
RAW_DATA (size of response length)
Once host receives and processes the data, it moves on and executes the next command
CMD_EXEC
(client_command1)
Target executes command and sends CMD_EXEC_RESP with response length corresponding to given client_command
CMD_EXEC_RESP
(response length = 0)
If response length is 0, then the command executed does not return response data. Host moves on and executes the next command.
CMD_EXEC
(client_command2)
CMD_EXEC_RESP
(response length > 0)
CMD_EXEC_DATA
(location of region 0, size of region 0)
RAW_DATA (size of response length)
Once the host has completed executing commands, it can force the target to exit command mode and switch modes
CMD_SWITCH_MODE
(mode = image transfer)
Target re-initiates protocol by sending HELLO packet with mode passed in CMD_SWITCH_MODE packet
HELLO
(mode = image transfer)

Figure 4-3 Successful Sahara Command sequence

4.5 Protocol implementation

The Sahara protocol can be implemented using state machines for both the target and the host. These state machines are shown in Figure 4-4 through Figure 4-10.

4.5.1 Target state machine

Figure 4-4 and Figure 4-5 show state machines which implement the target side of the packet flow shown in Figure 4-1. It illustrates how the actual data can be transferred for two types of image formats:

- Standalone binary
- Executable and Linkable Format (ELF)

The standalone binary format uses a simple image header which describes the size and destination address for the image data. ELF allows data to be segmented and scattered into noncontiguous sections of system memory.

Figure 4-4 and Figure 4-6 show state machines which implement the target side of the packet flow shown in Figure 4-2. They illustrate how a Memory Debug sequence would be executed.

Figure 4-4 and Figure 4-7 show state machines which implement the target side of the packet flow shown in Figure 4-3. They illustrate how a Command sequence would be executed.

The following list describes each state in the target state machine and how the target reacts to incoming packets when in these states:

- WAIT_HELLO_RESP – After the target sends a Hello packet, it waits until a Hello Response packet is received from the host. If an invalid packet or erroneous packet is received, the target sends an End of Image Transfer packet to the host with the corresponding error code. If a Reset packet is received, the target sends a Reset Response and then resets.
- DATA_BINARY_HDR – The target has received the standalone binary image header. If anything is wrong with the image header, the target sends an End of Image Transfer packet with the corresponding error code. If a valid image header is received, the target sends a single Read Data request to transfer the image data.
- DATA_BINARY – Once the image data is received, if the data is valid, the target sends an End of Image Transfer with “success” status. If any error occurs during the image transfer, the target sends an End of Image Transfer with the corresponding error code.
- DATA_ELF_HDR – The target has received the ELF header for an ELF image. If anything is wrong with the ELF header, the target sends an End of Image Transfer packet with the corresponding error code. If a valid ELF header is received, the target sends a single Read Data request to the program headers from the ELF image. The size and location of the program headers in the ELF image are embedded in the ELF header.

- DATA_ELF_PROG_HDR – The target has received the ELF program headers for an ELF image. If anything is wrong with the program headers, the target sends an End of Image Transfer packet with the corresponding error code. If valid program headers are received, the target processes them to determine the location of a hash table segment. A hash table can be used to validate the integrity of each data segment by applying a hash function to each data segment and storing the hash value in the hash table. Upon loading each ELF data segment, the hash function can be applied to each segment and the hash value compared to the one that was stored in the hash table.

  Specific hash algorithms and authentication routines are not described here, as they are outside the scope of this protocol.
- DATA_ELF_SEGMENTS – Once the location and size of each data segment is determined from the program headers, the target repeatedly sends Read Data requests until each data segment is transferred. Once all ELF segments are received, the target sends an End of Image Transfer with “success” status.
- WAIT_DONE – Once a single image transfer is complete, the target waits for a Done packet to be sent. If an invalid or any other packet is received, the target sends an End of Image Transfer packet with the corresponding error code. If a valid Done packet is received, the target sends a Done Response to the host, with the Image Transfer Status field set to “complete” or “pending” based on whether another image is to be transferred or not.
- WAIT_RESET – Any time an error occurs on the target, the target expects the host to send a Reset command. If the target receives a Reset command, it sends a Reset Response to the host and then resets. If an invalid or any other command is received, the target sends an End of Image Transfer packet with the corresponding error code.
- WAIT_MEMORY_READ – Once a Memory Debug packet has been sent to the host, the target continues to wait for a Memory Read packet, validates the incoming address and length, and then sends the corresponding data in a Raw Data packet to the host. If the target receives a Reset command, it sends a Reset Response to the host and then resets. If the target receives a Command Switch Mode command, it switches to the received mode and sends a Hello command. If an invalid or any other command is received, the target sends an End of Image Transfer packet with the corresponding error code.
- WAIT_CMD_EXEC – Once a Command Ready packet has been sent to the host, the target continues to wait for a Command Execute packet. Upon receiving the Command Execute packet, the target executes the given client command and sends the Command Execute Response with the corresponding response data length. If the response length is greater than 0, the target waits for a Command Execute Data packet. Otherwise, the target waits for another command. If the target receives a Reset command, it sends a Reset Response to the host and then resets. If the target receives a Command Switch Mode command, it switches to the received mode and sends a Hello command. If an invalid or any other command is received, the target sends an End of Image Transfer packet with the corresponding error code.
- WAIT_CMD_EXEC_DATA – If the response length is greater than 0 for an executed client command, the target waits for the Command Execute Data packet and sends the corresponding response data in a Raw Data packet to the host. Upon completion, the target waits for another command. If an invalid or any other command is received, the target sends an End of Image Transfer packet with the corresponding error code.

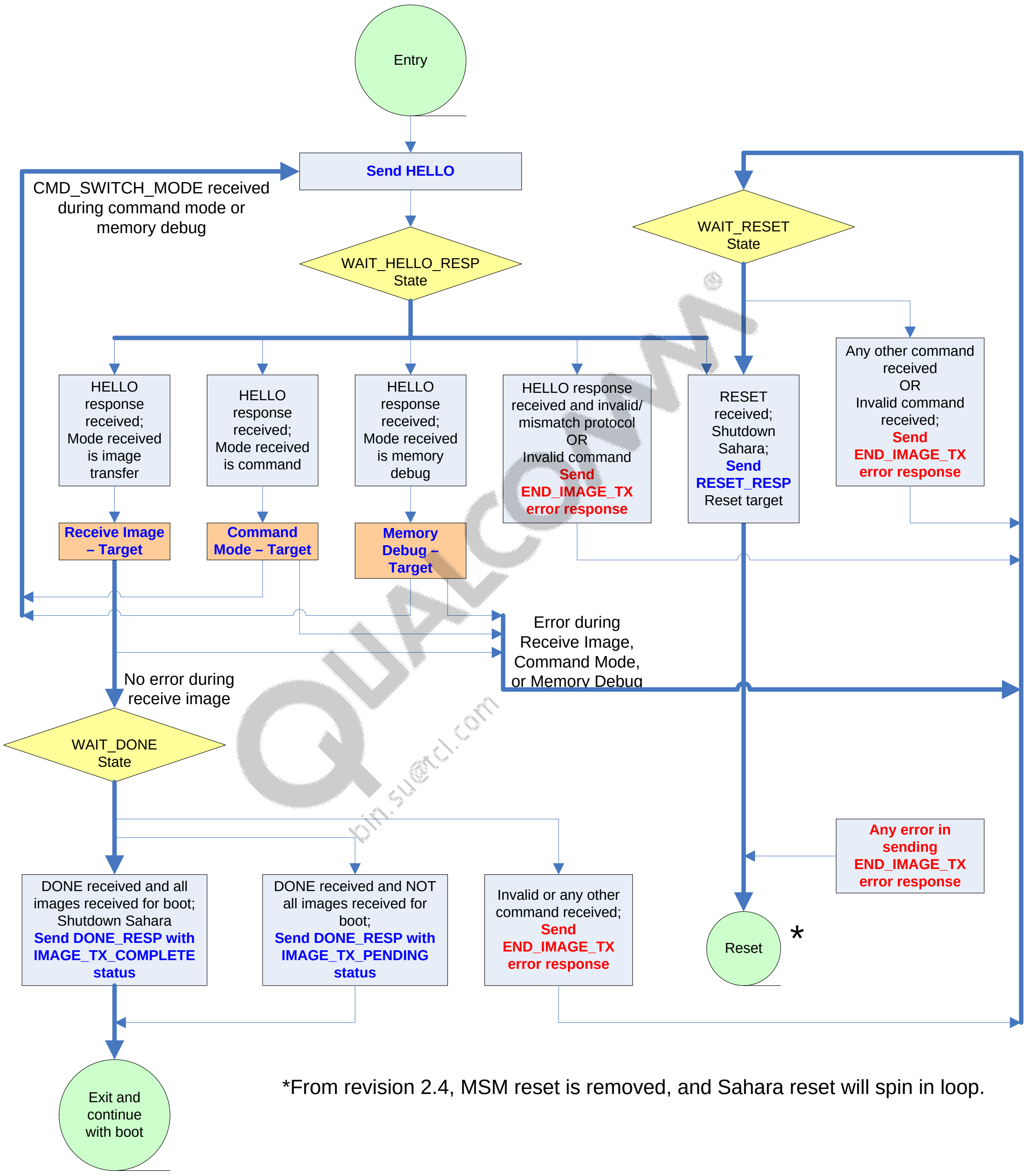
Entry
Send HELLO
CMD_SWITCH_MODE received during command mode or memory debug
WAIT_HELLO_RESP State
WAIT_RESET State
HELLO response received; Mode received is image transfer
HELLO response received; Mode received is command
HELLO response received; Mode received is memory debug
HELLO response received and invalid/ mismatch protocol OR Invalid command Send END_IMAGE_TX error response
RESET received; Shutdown Sahara; Send RESET_RESP Reset target
Any other command received OR Invalid command received; Send END_IMAGE_TX error response
Receive Image – Target
Command Mode – Target
Memory Debug – Target
Error during Receive Image, Command Mode, or Memory Debug
No error during receive image
WAIT_DONE State
DONE received and all images received for boot; Shutdown Sahara Send DONE_RESP with IMAGE_TX_COMPLETE status
DONE received and NOT all images received for boot; Send DONE_RESP with IMAGE_TX_PENDING status
Invalid or any other command received; Send END_IMAGE_TX error response
Any error in sending END_IMAGE_TX error response
Reset
*
Exit and continue with boot
*From revision 2.4, MSM reset is removed, and Sahara reset will spin in loop.

Figure 4-4 Sahara state machine (target side)

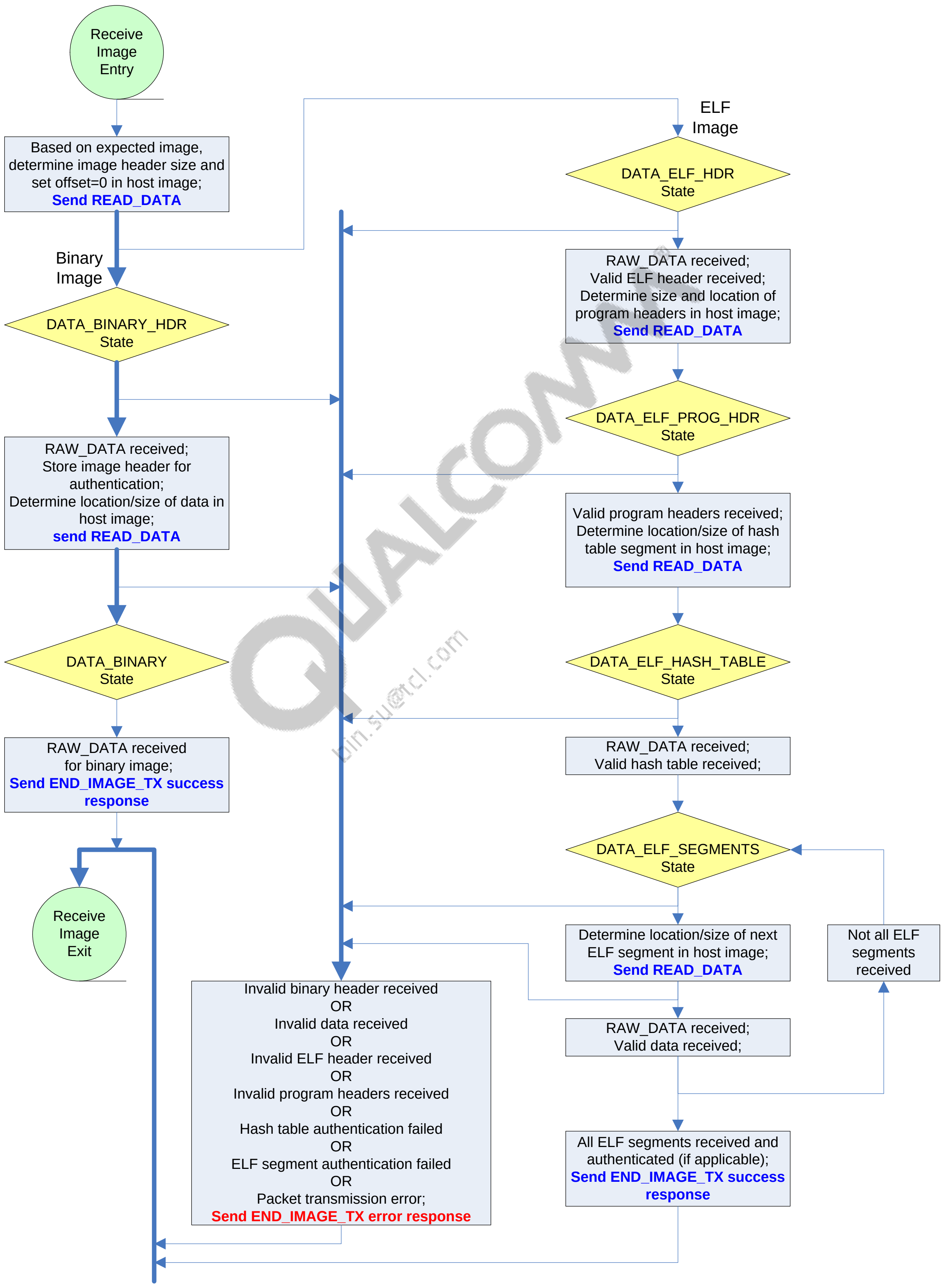
Receive Image Entry
Based on expected image, determine image header size and set offset=0 in host image; Send READ_DATA
Binary Image
DATA_BINARY_HDR State
RAW_DATA received; Store image header for authentication; Determine location/size of data in host image; send READ_DATA
DATA_BINARY State
RAW_DATA received for binary image; Send END_IMAGE_TX success response
Receive Image Exit
ELF Image
DATA_ELF_HDR State
RAW_DATA received; Valid ELF header received; Determine size and location of program headers in host image; Send READ_DATA
DATA_ELF_PROG_HDR State
Valid program headers received; Determine location/size of hash table segment in host image; Send READ_DATA
DATA_ELF_HASH_TABLE State
RAW_DATA received; Valid hash table received;
DATA_ELF_SEGMENTS State
Determine location/size of next ELF segment in host image; Send READ_DATA
Not all ELF segments received
RAW_DATA received; Valid data received;
All ELF segments received and authenticated (if applicable); Send END_IMAGE_TX success response
Invalid binary header received OR Invalid data received OR Invalid ELF header received OR Invalid program headers received OR Hash table authentication failed OR ELF segment authentication failed OR Packet transmission error; Send END_IMAGE_TX error response

Figure 4-5 Sahara state machine (target side) – Receive Image

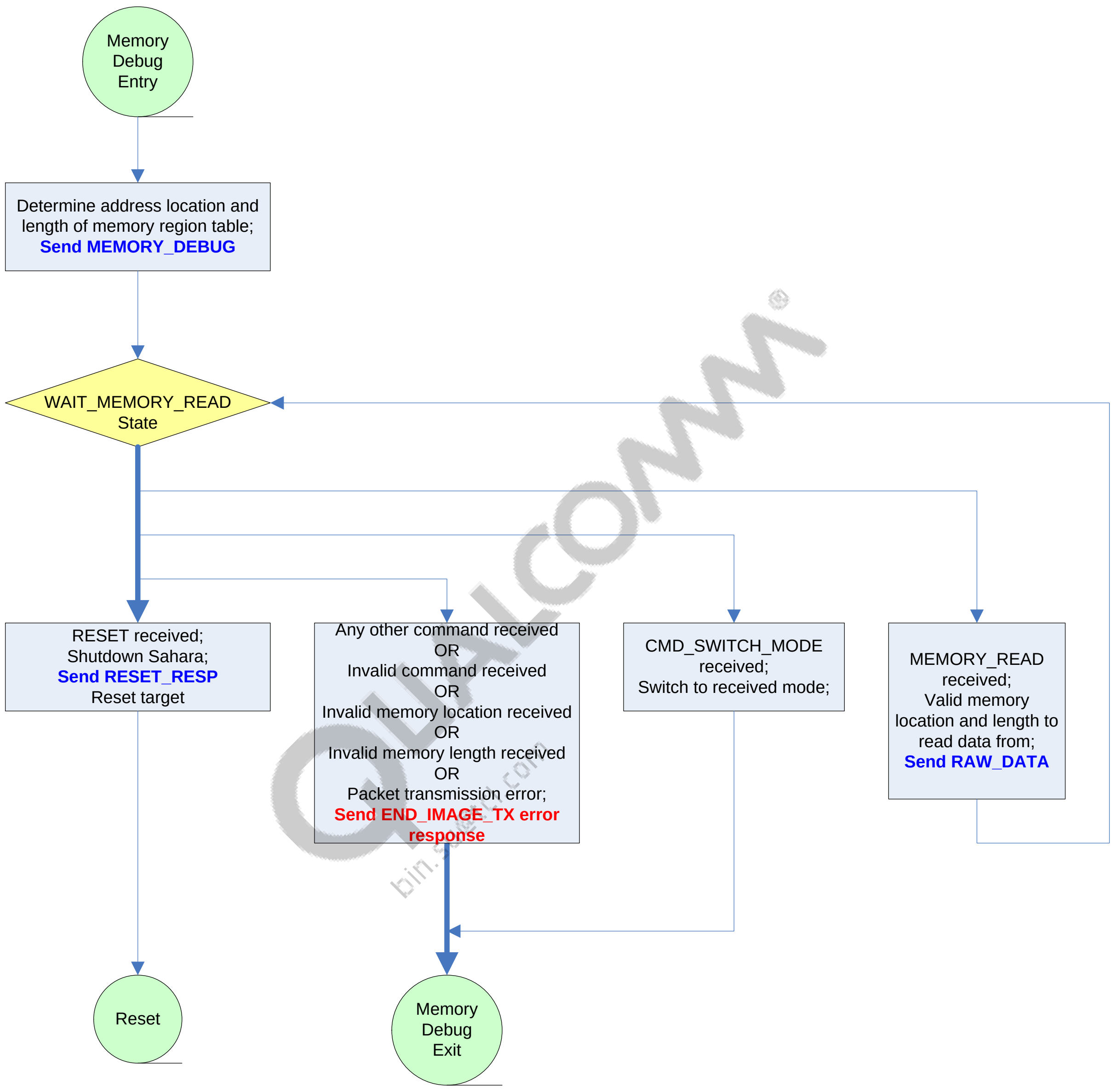
Memory Debug Entry
Determine address location and length of memory region table; Send MEMORY_DEBUG
WAIT_MEMORY_READ State
RESET received; Shutdown Sahara; Send RESET_RESP Reset target
Any other command received OR Invalid command received OR Invalid memory location received OR Invalid memory length received OR Packet transmission error; Send END_IMAGE_TX error response
CMD_SWITCH_MODE received; Switch to received mode;
MEMORY_READ received; Valid memory location and length to read data from; Send RAW_DATA
Reset
Memory Debug Exit

Figure 4-6 Sahara state machine (target side) – Memory Debug mode

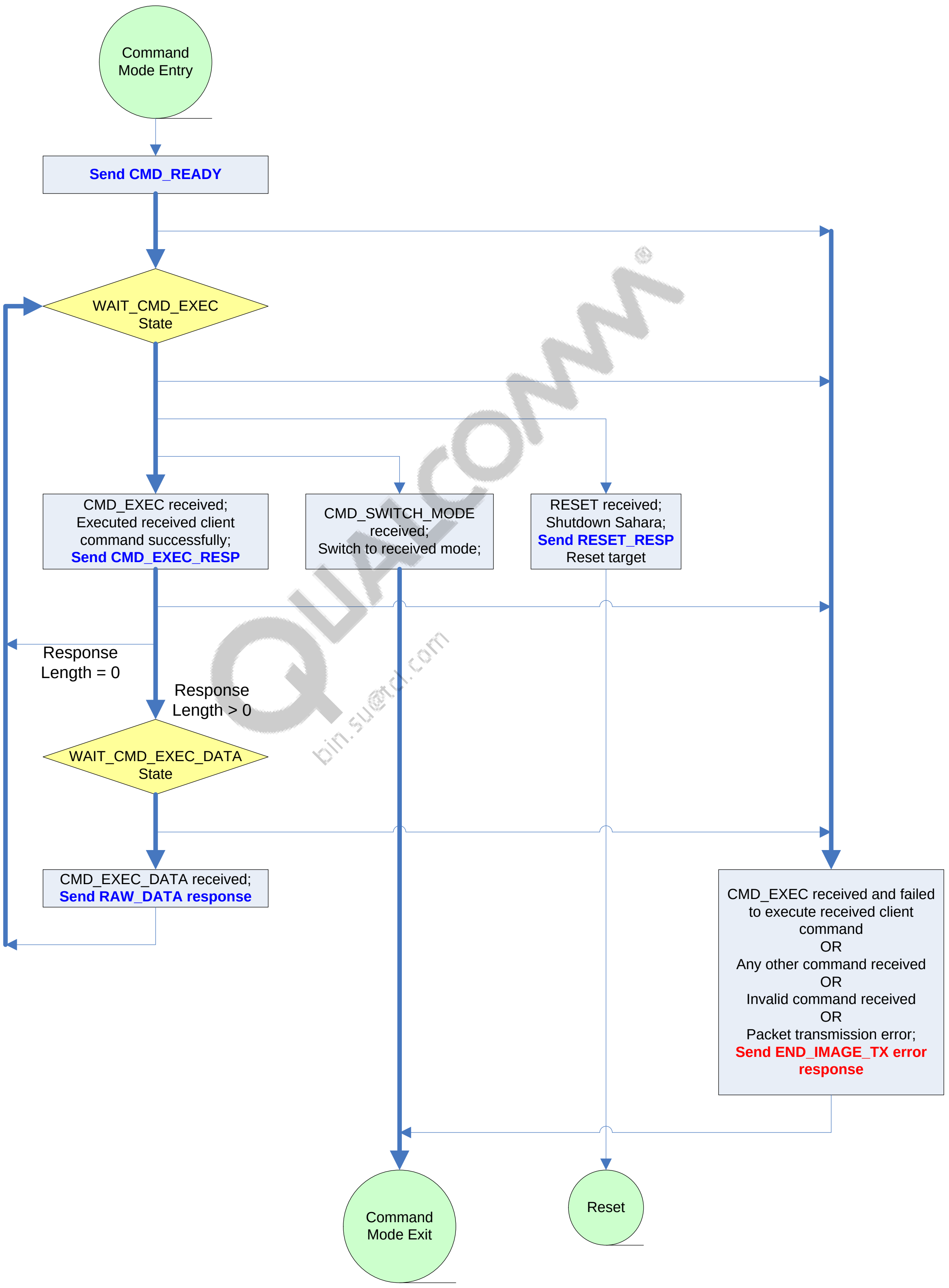
Command Mode Entry
Send CMD_READY
WAIT_CMD_EXEC State
CMD_EXEC received; Executed received client command successfully; Send CMD_EXEC_RESP
CMD_SWITCH_MODE received; Switch to received mode;
RESET received; Shutdown Sahara; Send RESET_RESP Reset target
Response Length = 0
Response Length > 0
WAIT_CMD_EXEC_DATA State
CMD_EXEC_DATA received; Send RAW_DATA response
CMD_EXEC received and failed to execute received client command OR Any other command received OR Invalid command received OR Packet transmission error; Send END_IMAGE_TX error response
Command Mode Exit
Reset

Figure 4-7 Sahara state machine (target side) – Command mode

4.5.2 Host state machine

Figure 4-8, Figure 4-9, and Figure 4-10 show a state machine which implements the host side of the packet flows shown in Figure 4-1, Figure 4-2, and Figure 4-3.

The following list describes each state in the state machine and how the host reacts to incoming packets when in these states:

- WAIT_HELLO – The host waits for the target to initiate the protocol. Once the Hello is received and the protocol version validated, the host sends a Hello Response with "success" status. If the host receives an invalid packet (or any other packet), it sends a Reset packet to the target. It also sends a Reset packet if the target protocol version is not compatible with the host.
- WAIT_COMMAND – If the host receives a Read Data packet, it reads and transfers data from the corresponding image in a data packet. If the host receives an End of Image Transfer packet with "success" status, it sends a Done packet. If the host receives a Command Ready packet, it enters the Command sequence. If the host receives a Memory Debug packet, it enters the Memory Debug sequence. If the host receives an invalid command (or any other command), it sends a Reset packet. It also sends a Reset packet if it receives an End of Image Transfer packet with an error code.
- WAIT_DONE_RESP – The host waits for a Done Response. If all images have not been transferred, the host waits for another Hello packet. If all images have been transferred, the host exits the protocol. If the host receives an invalid command (or any other command), it sends a Reset packet.
- WAIT_RESET_RESP – After the host sends a Reset packet, it waits for the target to send a Reset Response. If the host receives a Reset Response, it exits the protocol. If the host receives an invalid command (or any other command), it sends another Reset packet.
- WAIT_MEMORY_TABLE – Once the host sends a Memory Read packet to read the memory debug table, it waits for a Raw Data packet with the contents of the table.
- WAIT_MEMORY_REGION – After receiving the memory debug table, the host repeatedly sends Memory Read packets to dump each memory region from the target. Once all the memory regions are received, it sends either a Reset command or Command Switch Mode command to the target.
- WAIT_CMD_EXEC_RESP – After receiving the Command Ready packet, the host proceeds to execute a series of client commands by sending Command Execute packets to the target. Once all commands have been executed and the corresponding data received, the host can switch the mode of the target and re-initiate the protocol by waiting for a Hello packet. If the host receives invalid raw data or an End of Image Transfer packet, it sends a Reset packet.

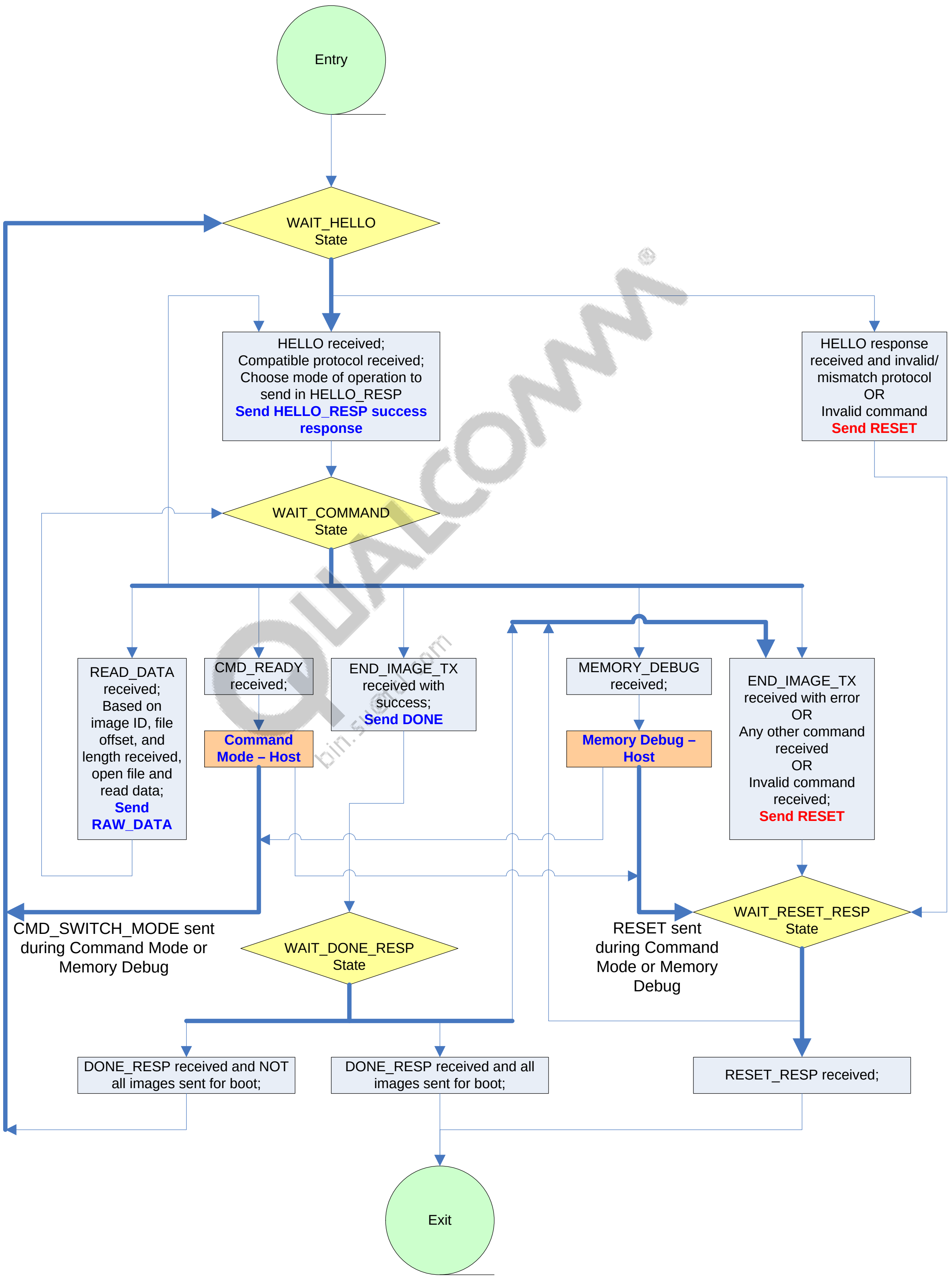
Entry
WAIT_HELLO State
HELLO received; Compatible protocol received; Choose mode of operation to send in HELLO_RESP **Send HELLO_RESP success response**
HELLO response received and invalid/ mismatch protocol OR Invalid command **Send RESET**
WAIT_COMMAND State
READ_DATA received; Based on image ID, file offset, and length received, open file and read data; **Send RAW_DATA**
CMD_READY received;
Command Mode – Host
END_IMAGE_TX received with success; **Send DONE**
MEMORY_DEBUG received;
Memory Debug – Host
END_IMAGE_TX received with error OR Any other command received OR Invalid command received; **Send RESET**
CMD_SWITCH_MODE sent during Command Mode or Memory Debug
WAIT_DONE_RESP State
RESET sent during Command Mode or Memory Debug
WAIT_RESET_RESP State
DONE_RESP received and NOT all images sent for boot;
DONE_RESP received and all images sent for boot;
RESET_RESP received;
Exit

Figure 4-8 Sahara state machine (host side)

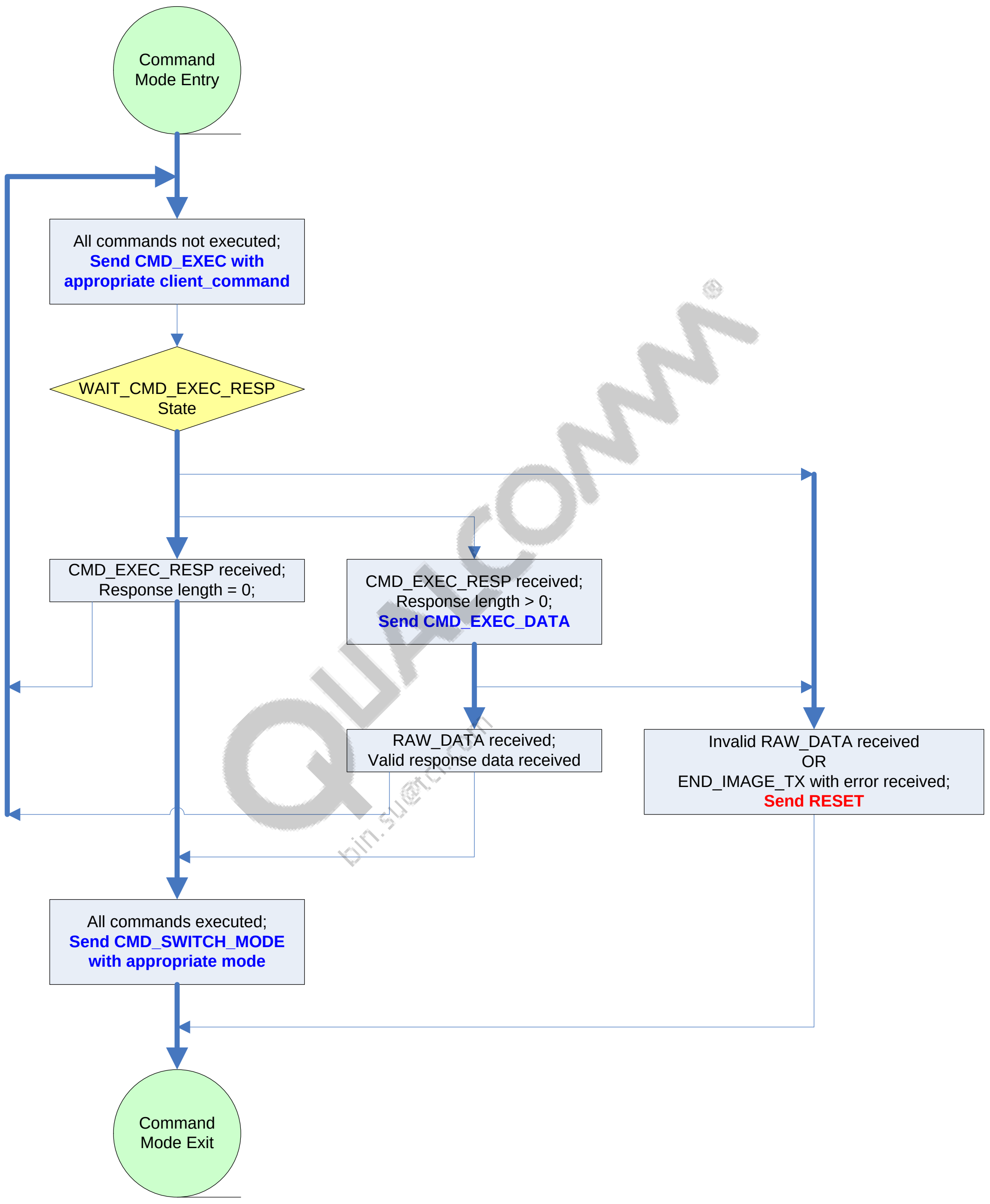
Command Mode Entry
All commands not executed; Send CMD_EXEC with appropriate client_command
WAIT_CMD_EXEC_RESP State
CMD_EXEC_RESP received; Response length = 0;
CMD_EXEC_RESP received; Response length > 0; Send CMD_EXEC_DATA
RAW_DATA received; Valid response data received
Invalid RAW_DATA received OR END_IMAGE_TX with error received; Send RESET
All commands executed; Send CMD_SWITCH_MODE with appropriate mode
Command Mode Exit

Figure 4-9 Sahara state machine (host side) – Command mode

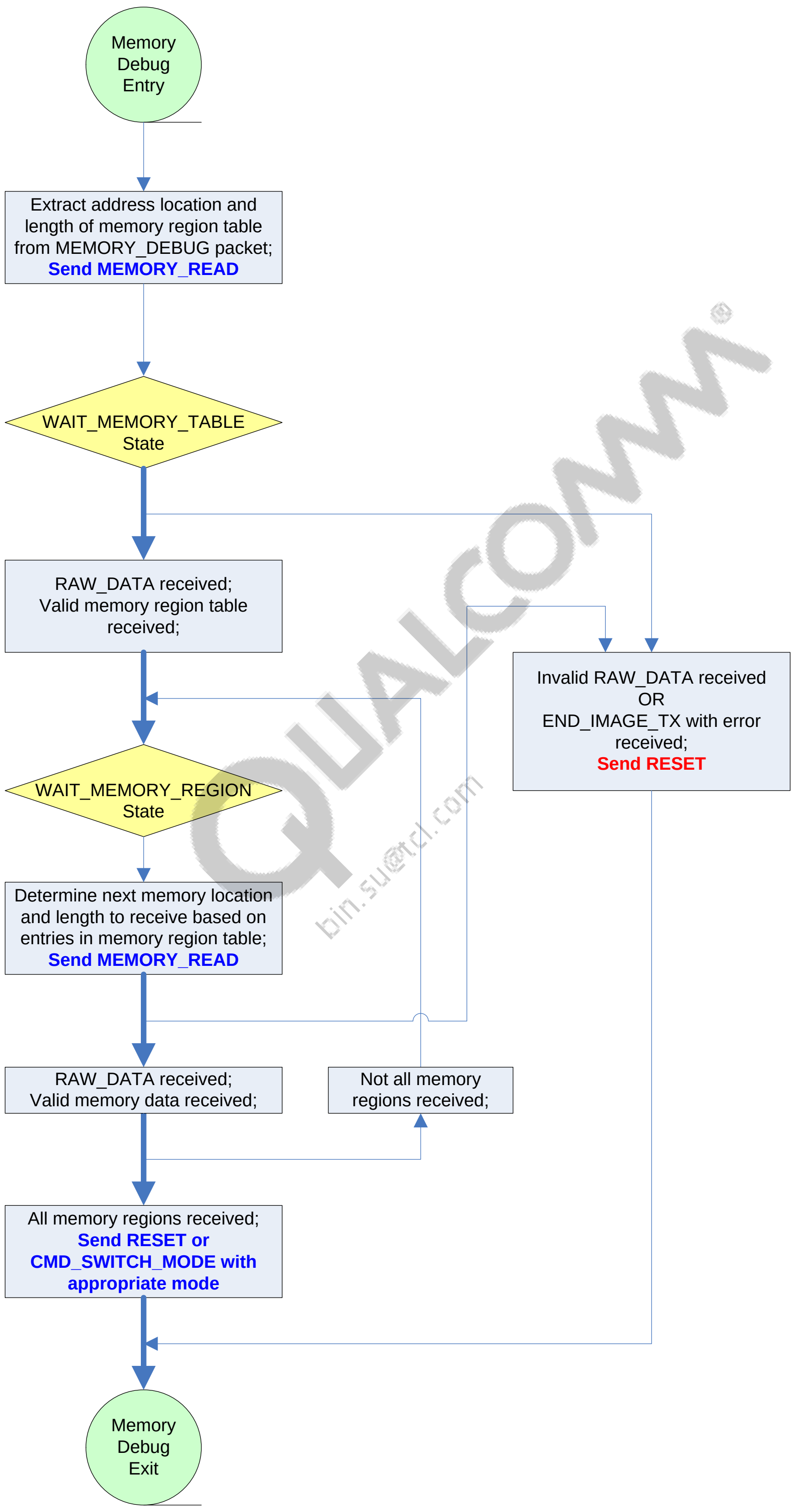
Memory Debug Entry
Extract address location and length of memory region table from MEMORY_DEBUG packet; Send MEMORY_READ
WAIT_MEMORY_TABLE State
RAW_DATA received; Valid memory region table received;
Invalid RAW_DATA received OR END_IMAGE_TX with error received; Send RESET
WAIT_MEMORY_REGION State
Determine next memory location and length to receive based on entries in memory region table; Send MEMORY_READ
RAW_DATA received; Valid memory data received;
Not all memory regions received;
All memory regions received; Send RESET or CMD_SWITCH_MODE with appropriate mode
Memory Debug Exit

Figure 4-10 Sahara state machine (host side) – Memory Debug mode

4.6 Error handling

Section 4.5 illustrates state machines that show how errors can be implemented and handled using the protocol. Timeouts and retransmission of packets are often implemented to handle possible transmission or reception errors.

Techniques for implementing such strategies are outside the scope of this protocol.

4.7 Parallel image transfers

This section describes how multiple images can be transferred in parallel through the use of multi-threaded environments.

4.7.1 Single host, multiple targets

If the host can distinguish between targets at the hardware transport layer and can route Sahara packets to the corresponding target, the host can kick off a state machine for each target on a separate thread:

- Each target would run its own state machine to transfer the images it wishes to transfer
- The host would need to run a thread manager to send/receive the Sahara packets and route them to corresponding state machine

Since the Read Data requests only need to specify the image ID, data offset, and data length, the host only needs to read from the corresponding image file and send the data to the requesting target.

4.7.2 Multiple hosts, single target

If the target needs to transfer images from multiple hosts, the connections from each host can be encapsulated in the hardware transport layer. On entering the protocol, the target can specify which hardware it wishes to use to transfer the given image. The target can choose to enter and exit the protocol for each image, allowing it to select the hardware transport layer to use for each image (effectively abstracting the host connections in the corresponding software driver by using dispatch tables). Each host would then need to run separate state machines.

4.7.3 Single host, single target, parallel images

If the target wants to transfer images in parallel from the host, a threaded environment can be used on the target (similar to the threaded environment Section 4.7.1 describes). The difference is that the threaded environment is not needed on the host. The target needs to manage the routing of packets to the state machine for each image.