User Manual

DA16200 DA16600 ThreadX OTA Update

UM-WI-036

Abstract

This OTA update User Manual intends to assist software developers that implement applications with the DA16200 (DA16600) SDK. A certain degree of reader familiarity with programming environments, debugging tools and software engineering process in general is assumed.

Contents

Figures

Tables

Terms and Definitions

OTA	Over the Air
API	Application Programming Interface
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol over Secure Socket Layer
MQTT	Message Queuing Telemetry Transport
TLS	Transport Layer Security
AWS	Amazon Web Services
CLI	Command Line Interface
SDK	Software Development Kit
NVRAM	Non-Volatile RAM
CRC	Cyclic Redundancy Check

References

[1] DA16200, Datasheet, Dialog Semiconductor
[2] DA16200 DA16600, SDK Programmer Guide, Dialog Semiconductor
[3] lwIP, Lightweight IP stack, https://www.nongnu.org/lwip/2_1_x/group__httpc.html

1 Introduction

The DA16200 (DA16600) can update the firmware over the air using the HTTP protocol. The DA16200 (DA16600) operates as an HTTP client, it can download and update new firmware from the HTTP server.

Users can easily develop these functions using the API provided by the DA16200 (DA16600) SDK.

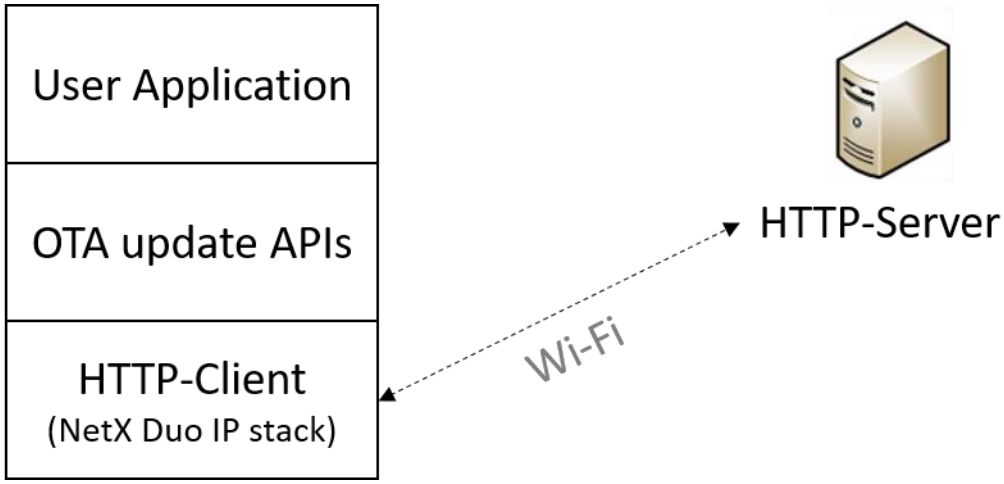

Figure 1: OTA Update Layer

2 SFLASH Memory Area

The DA16200 (DA16600) does not support file systems, so that the firmware is stored in the SFLASH memory area. SFLASH is divided into several areas as shown in the Table 1. Among them, the areas that users can directly access are as follows:

- User accessible SFLASH area:
  - RTOS #0
  - RTOS #1
  - User Area #0
  - User Area #1
  - TLS Certificate #0
  - TLS Certificate #1

NOTE
If other areas are accessed incorrectly, serious failure may occur in the system.

Table 1: 2 MB SFLASH Memory Map

Address	Name		Size (byte)
0x0000_0000	ADDRESS START		-
0x0000_0000	Bootloader		36,864
0x0000_9000	Boot Index		4,096
0x0000_A000	RTOS #0		946,176
0x000F_1000	SLIB #0		53,248
0x000F_E000	RTOS #1		946,176
0x001E_5000	SLIB #1		53,248
0x001F_2000	User Area		12,288
0x001F_5000	Debug / RMA Certificate		4,096
0x001F_6000	TLS Certificate #0 (MQTT)	CA	16,384
0x001F_7000		Cert	
0x001F_8000		Private key	
0x001F_9000		DH	
0x001F_A000	TLS Certificate #1 (HTTPS / OTA)	CA	16,384
0x001F_B000		Cert	
0x001F_C000		Private key	
0x001F_D000		DH	
0x001F_E000	NVRAM #0		4,096
0x001F_F000	NVRAM #1		4,096
0x0020_0000	ADDRESS END		-

Table 2: 4 MB SFLASH Memory Map

Address	Name		Size (byte)
0x0000_0000	ADDRESS START		-
0x0000_0000	Bootloader		36,864
0x0000_9000	Boot Index		4,096
0x0000_A000	RTOS #0		1,572,864
0x0018_A000	SLIB #0		65,536
0x0019_A000	User Area #0		372,736
0x001F_5000	Debug / RMA Certificate		4,096
0x001F_6000	TLS Certificate #0 (MQTT)	CA	16,384
0x001F_7000		Cert	
0x001F_8000		Private key	
0x001F_9000		DH	
0x001F_A000	TLS Certificate #1 (HTTPS / OTA)	CA	16,384
0x001F_B000		Cert	
0x001F_C000		Private key	
0x001F_D000		DH	
0x001F_E000	NVRAM #0		4,096
0x001F_F000	NVRAM #1		4,096
0x0020_0000	RTOS #1		1,572,864
0x0038_0000	SLIB #1		65,536
0x0039_0000	User Area #1		458,752
0x0040_0000	ADDRESS END		-

3 HTTP Protocol

The DA16200 (DA16600) supports HTTP/HTTPS 1.1. DA16200 (DA16600) requests firmware download to the HTTP server by using the GET method of the HTTP client.

The OTA update application must know the URL of the HTTP server in advance before requesting a download. How to obtain the URL depends on the user's preference. Therefore, it is not mentioned in this manual. When using HTTPS, the DA16200 (DA16600) must have at least 36 kB of heap memory for TLS encryption and decryption. The user can print the current memory usage on the terminal.

- CLI commands

```
[/DA16200] # sys.os.heap
[/DA16200] # sys.os.pool
```

- API

```
extern void memoryPoolInfo(void);
extern void   cmd_heapinfo_func(int argc, char *argv[]);
memoryPoolInfo();
cmd_heapinfo_func(0, NULL);
```

4 OTA Update Function

DA16200 (DA16600) firmware is divided into two firmwares: SLIB and RTOS. To update the firmware, SLIB and RTOS firmware must be downloaded.

The OTA update function is divided into two stages: DOWNLOAD and RENEW.

DOWNLOAD refers to the process of downloading new firmware from the OTA server. In this case, the new firmware is not yet applied.

RENEW is the process of applying to operate with the successfully downloaded firmware. To do this, some rules and information are required.

4.1 Header

Figure 2 shows DA16200 (DA16600) header information as an example. Header information is 96 bytes and automatically inserted when the firmware is built. Users do not need to understand all the contents of the header. The red box in Figure 2 is the magic number and version information. The yellow box is information for checking firmware CRC. Users only need to understand the version information in the red box.

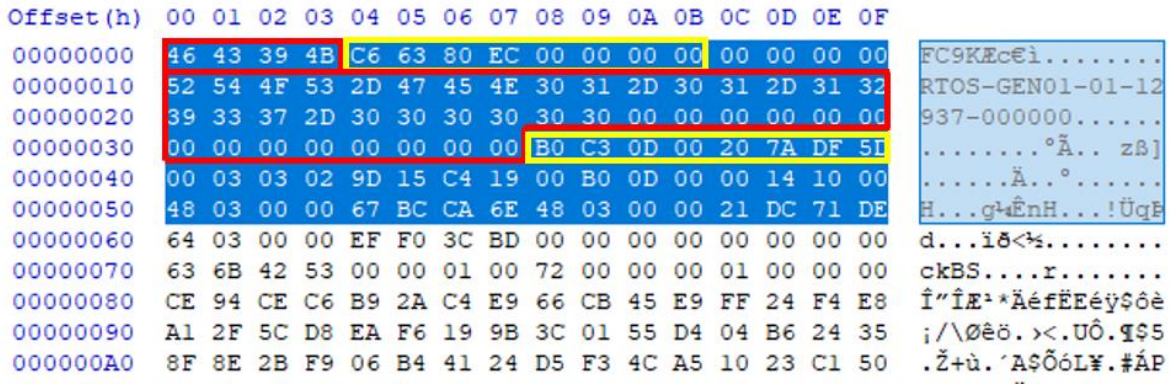

Figure 2: Firmware Header Information

4.2 Version

DA16200 (DA16600)'s firmware has unique version rules for system protection. The version string is inserted as a 26-character string in the header of the firmware at the time of build (Maximum 32 bytes).

There are five elements in the version string, separated by "-". These are: Type, Vendor, Major, Minor, and Customer. For example, RTOS-GEN01-01-12345-000001

Version String

Type-Vendor-Major-Minor-Customer

1. Type (4 bytes): Identify the type of firmware. (SLIB, RTOS)
2. Vendor (5 bytes): Vendor classification
3. Major (2 bytes): Major number to check compatibility
4. Minor (4~5 bytes): SDK patch number
5. Customer (6 bytes): User configurable version

Type-Vendor-Major determines whether DOWNLOAD or RENEW is compared to the version of firmware currently in operation. *Minor-Customer* can be used by the user for firmware version management.

Users can change the customer version by editing ..\version\3rd_customer_build_num.h. If users change the customer version and build the SDK, the customer version is applied to the image.

4.3 Result Code

All APIs provided by OTA update return the result codes shown in Table 3. It is delivered through the callback function connected with DOWNLOAD and RENEW API.

Table 3: Result Code

Result Code	Value	Description
OTA_SUCCESS	0x00	Return success.
OTA_FAILED	0x01	Return failed.
OTA_ERROR_SFLASH_ADDR	0x02	SFLASH address is wrong.
OTA_ERROR_TYPE	0x03	FW type is unknown.
OTA_ERROR_URL	0x04	Server URL is unknown.
OTA_ERROR_SIZE	0x05	FW size is too big.
OTA_ERROR_CRC	0x06	CRC is not correct.
OTA_VERSION_UNKNOWN	0x07	FW version is unknown.
OTA_VERSION_INCOMPATI	0x08	FW version is incompatible.
OTA_NOT_FOUND	0x09	FW not found on the server.
OTA_NOT_CONNECTED	0x0A	Failed to connect to server.
OTA_NOT_ALL_DOWNLOAD	0x0B	All new FWs have not been downloaded.
OTA_MEM_ALLOC_FAILED	0x0C	Failed to allocate memory.

4.4 Download

The download step is the process of downloading firmware from the OTA server and saving it into the SFLASH area.

The communication protocol with the OTA server uses HTTP and can be implemented using the HTTP API supported by lwIP. Therefore, the process of communicating with HTTP server works the same as lwIP's HTTP client.

The download sequence proceeds as follows, and both success and failure results can be delivered through the callback function (see Table 3 for results).

1. Request a query from the HTTP server.
2. Confirm that the response was successfully received from the HTTP server. If the server connection fails or receives a failure response, the download will be terminated, and the result will be transferred to the callback function. See Table 3 for result values.
3. Check the magic number and version name in the firmware header, and if they do not match, the download is terminated, and the result is transferred to the callback function. In the version name, only *Type* and *Vendor* are checked to see if they match the current FW. The *Major number* that checks the compatibility of SLIB and RTOS is checked in the Renew operation.
4. If the magic number and version name are normal, the downloaded data is written to SFLASH. The SFLASH address where the data is written is automatically determined by the boot index (see Section 4.5.1). When the download is completed successfully, the entire firmware stored in SFLASH will have a CRC check.
5. When the CRC check is successfully completed, the result value of 0x00 is transferred to the callback function and the download is terminated.

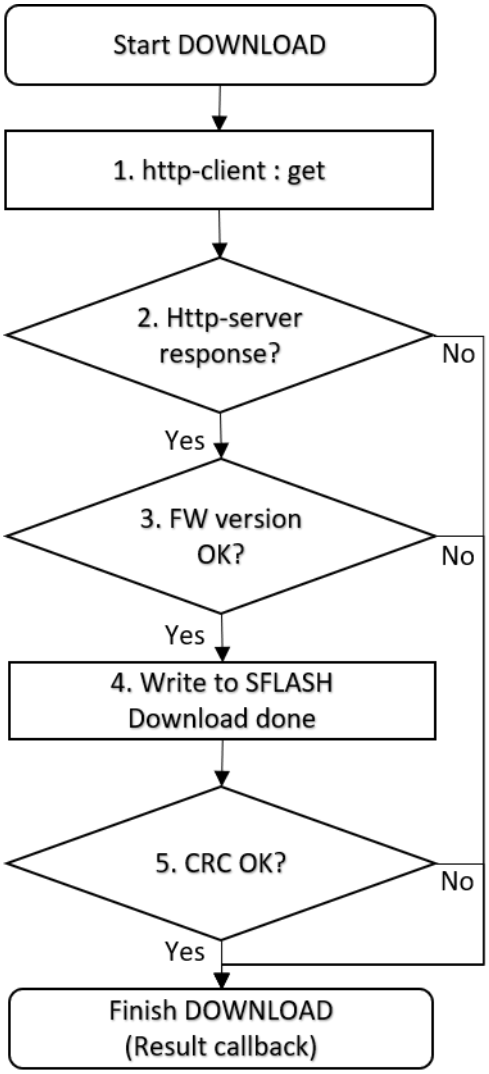

Figure 3: Firmware DOWNLOAD

NOTE
Ensure sufficient power of the system before flash write. It may cause flash corruption.

4.5 Renew

RENEW only operates when the firmware download is successful. Additionally, the download success can be checked, based on the power ON/OFF standard of the DA16200 (DA16600), DA16200 (DA16600) should have the download history after power ON.

1. Check whether the download of both SLIB and RTOS firmware is successful. After turning on the power, check the download history.
2. Check the CRC of the firmware stored in the SFLASH. In case of failure, RENEW ends and the result is transmitted to the callback function.
3. Check the *Type, Vendor* and *Major* of the firmware version names stored in the flash. Check the compatibility of RTOS and SLIB by *Major number*. In case of failure, RENEW ends and the result is transmitted to the callback function.
4. It is determined that the new firmware is normal, and the boot index is changed to the new firmware location.

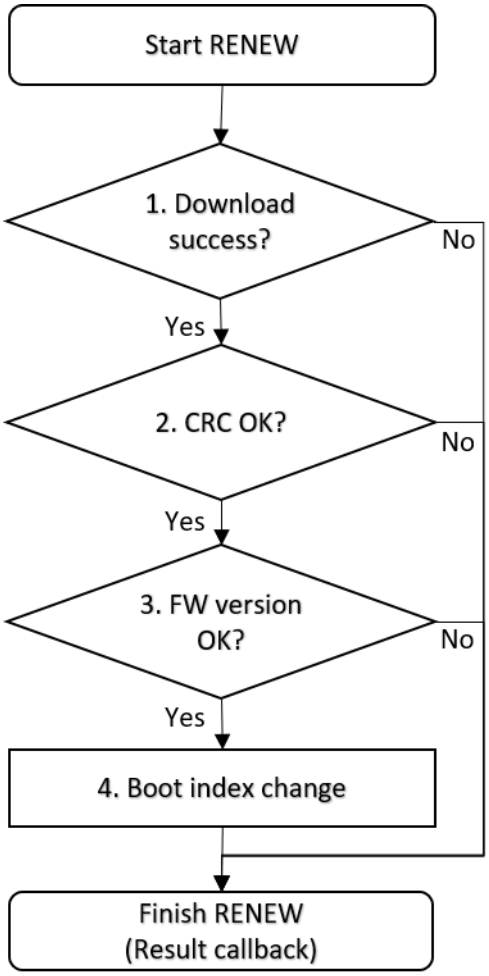

Figure 4: Firmware RENEW

4.5.1 Boot Index

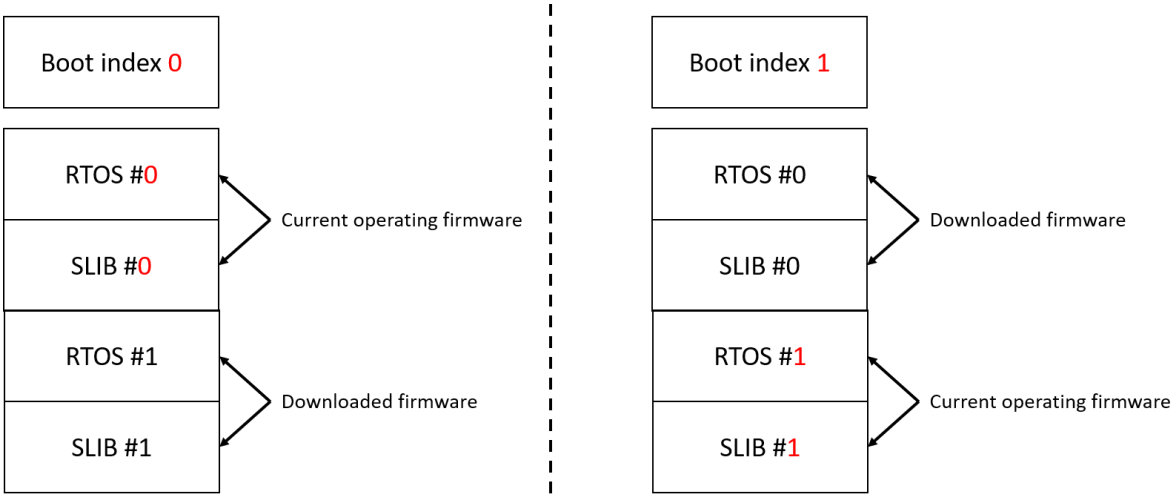

Figure 5: Boot Index Operation

The DA16200 (DA16600) is divided into firmware download area and current area for the OTA update function. The two areas are toggled on each other by the boot index. For example, if the boot index value is 0, it operates as the firmware stored in the SFLASH RTOS #0 area upon booting, and the newly downloaded firmware is stored in RTOS #1. After that, if RENEW is operated successfully, the boot index value is changed to 1, rebooted, and the firmware stored in the SFLASH RTOS#1 area is operated (see Table 1 and Table 2).

5 Application Programming

Describes the structures and APIs required for the OTA firmware update application.

5.1 Type

OTA update task is operated based on the type defined in the OTA update type. The operation sequence is tailored to the specified type.

Table 4: OTA Update Type

Name	ota_update_type
Description	Identify and specify targets for OTA updates.

	/// Operation step of process
	enum ota_update_fw_type
	{
	  /// Init value
	  OTA_TYPE_INIT,
	  /// SLIB
	  OTA_TYPE_SLIB,
	  /// RTOS
	  OTA_TYPE_RTOS,
	  /// User mcu firmware
	  OTA_TYPE_MCU_FW,
	  /// Certificate or key
	  OTA_TYPE_CERT_KEY,
	  /// Unknown value
	  OTA_TYPE_UNKNOWN
	};

5.2 Structure

OTA UPDATE CONFIG sets the necessary parameters when calling OTA firmware update API.

Table 5: OTA_UPDATE_CONFIG

Name	OTA_UPDATE_CONFIG
Description	Contains information to be passed as argument values to OTA update APIs.
	/// OTA update configuration structure typedef struct { /// Pointer variable to the server address where the SLIB exists. char *uri_slib; /// Pointer variable to the server address where the RTOS exists. char *uri_rtos; /// Pointer variable to the server address where the others (MCU FW, Certi) exist. char *uri_other; /// Set the type of uri_other. /// Must be OTA_TYPE_MCU_FW or OTA_TYPE_CERT_KEY. UINT other_type; /// Callback function pointer to call HTTP-client API. void (*http_configure)(NX_HTTP_CLIENT *http); /// Callback function pointer to check the download status. void (*download_complete_notify)(UINT fw_type, UINT ret_status); /// Callback function pointer to check the renew state. void (*renew_notify)(UINT ret_status); /// If the value is true, /// if the download of a new FW succeeds, the new FW is rebooted. UINT auto_renew; /// Address of sflash where other FW or certificate are stored. UINT download_sflash_addr; /// If the value is true, /// firmware download does not check compatibility by version. /// (Use only in debugging or special situations) UINT disable_version_check; } OTA_UPDATE_CONFIG;

5.3 APIs

Describes the API required for the OTA firmware update application.

Table 6: Lists for OTA APIs

UINT ota_update_start_download(OTA_UPDATE_CONFIG *fw_conf)		
Parameter	[in] ota_update_conf	The pointer of OTA_UPDATE_CONFIG structure. *uri_slib: Pointer variable to the server address where the SLIB exists. *uri_rtos: Pointer variable to the server address where the RTOS exists. *uri_other: Pointer variable to the server address where the others (MCU FW, Certi) exist. other_type: Set the type of uri_other. Must be OTA_TYPE_MCU_FW or OTA_TYPE_CERT_KEY. (*http_configure)(NX_HTTP_CLIENT *http): Callback function pointer to call HTTP-client API. (*download_complete_notify)(UINT fw_type, UINT ret_status) : Callback function pointer to check the download status. (*renew_notify)(UINT ret_status): Callback function pointer to check the renew state. auto_renew: If the value is true, if the download of a new FW succeeds, the new FW is rebooted. download_sflash_addr: Address of sflash where other FW or certificate are stored. disable_version_check: If the value is true, firmware download does not check compatibility by version. (Use only in debugging or special situations.)
Return		Returns 0x00 on success. See Table 3.
Description		HTTP-client task is created and sent query to HTTP server. It checks the version compatibility of the firmware received from the server and writes it to the download area of SFLASH.

UINT ota_update_stop_download(void)		
Parameter	Void	None.
Return		Returns 0x00 on success. See Table 3.
Description		A download can be stopped while downloading from the HTTP server.

UINT ota_update_get_download_progress(UINT fw_type)		
Parameter	[in] fw_type	Specifies the type to be updated.
Return		Returns a value between 0 and 100. If the download was successful, it returns 100.
Description		Checks the progress while downloading or after completion.

UINT ota_update_start_renew(OTA_UPDATE_CONFIG *fw_conf)		
Parameter	[in] ota_update_conf	The pointer of OTA_UPDATE_CONFIG structure.
Return		Returns 0x00 on success. See Table 3.

UINT ota_update_start_renew(OTA_UPDATE_CONFIG *fw_conf)	
Description	Check the version compatibility and CRC, change the boot index to the new firmware location, and then reboot automatically.

UINT ota_update_get_new_sflash_addr(UINT fw_type)		
Parameter	[in] fw_type	Specifies the type to be updated.
Return		Returns the SFLASH address.
Description		The user can know the address of SFLASH where the data (firmware) downloaded from the server is stored.

UINT ota_update_read_flash(UINT addr, VOID *buf, UINT len)		
Parameter	[in] addr [out] buf [in] len	SFLASH address (hex). Buffer pointer to store read data. Length to read.
Return		Returns 0x00 on success. See Table 3.
Description		Read SFLASH as much as the input address and length.

UINT ota_update_erase_flash(UINT addr, UINT len)		
Parameter	[in] addr [in] len	SFLASH address (hex). Length to erase.
Return		Returns erased length.
Description		Erases SFLASH as much as the input address and length.

UINT ota_update_copy_flash(UINT dest_addr, UINT src_addr, UINT len)		
Parameter	[in] dest_addr [in] src_addr [in] len	dest_addr: Destination SFLASH address (hex). src_addr: Source SFLASH address (hex). Length to copy.
Return		Returns 0x00 on success. See Table 3.
Description		Copies as much as the length from SFLASH address src_addr to dest_addr.

UINT ota_update_set_mcu_fw_name(char *name)		
Parameter	[in] name	Input the firmware name (version). Maximum eight bytes.
Return		Returns 0x00 on success. See Table 3.
Description		Sets the name (version) of MCU FW to be downloaded to SFLASH. If not set, it is set as the default string.

UINT ota_update_get_mcu_fw_name(char *name)		
Parameter	[out] name	Pointer to get the name (version) of MCU FW.
Return		Returns 0x00 on success. See Table 3.
Description		Gets name (version) of MCU FW downloaded to SFLASH.

UINT ota_update_get_mcu_fw_info(char *name, UINT *size, UINT *crc)		
Parameter	[out] name [out] size [out] crc	Pointer to get the name (version) of MCU FW. Pointer to get the size of MCU FW. Pointer to get the CRC32 value of MCU FW.
Return		Returns 0x00 on success. See Table 3.
Description		Get name (version), size and CRC32 of MCU FW downloaded to SFLASH.

UINT ota_update_uart_read_mcu_fw(UINT sflash_addr, UINT size)		
Parameter	[in] sflash_addr [int] size	sflash_addr: Start address for reading. Read size.
Return		Returns 0x00 on success. See Table 3.
Description		Starts transmission of MCU FW stored in flash through UART1 as much as the set size.

UINT ota_update_uart_trans_mcu_fw(void)		
Parameter	void	None.
Return		Returns 0x00 on success. See Table 3.
Description		Starts transmission of MCU FW stored in flash through UART1.

UINT ota_update_erase_mcu_fw(void)		
Parameter	void	None.
Return		Returns 0x00 on success. See Table 3.
Description		Delete MCU FW saved in SFLASH.

UINT ota_update_calcu_mcu_fw_crc(int sflash_addr, int size)		
Parameter	[in] sflash_addr [int] size	sflash_addr: CRC calculation start address. CRC calculation size.
Return		Returns 0x00 on success. See Table 3.
Description		Calculate CRC32 of MCU FW stored in SFLASH.

void ota_update_uart1_init(void)		
Parameter	void	None.
Return		Void.
Description		UART1 initialization.

5.4 Example

To update the DA16200 (DA16600)'s firmware, the user should always download two images, SLIB and RTOS.

1. Configure each server URL and set auto_renew to renew automatically when download is successful. If auto_renew = 1 is set, there is no need to call the ota_update_start_renew() API separately.
2. Register the callback function to receive the result in download and renew.
3. Call ota_update_start_download().

DA16200 (DA16600) FW update Pseudo code. See the sample code for more detailed usage.

```
conf.uri_slib = "http://ota_server/DA16200_RLIB-GEN01-01-12345-000001.img"
conf.uri_rtos = "http://ota_server/DA16200_RTOS-GEN01-01-12345-000001.img"
conf.auto_renew = 1;
conf.download_complete_notify = download_complete_notify();
conf.renew_notify = renew_notify();


ota_update_start_download(conf);
```

5.4.1 Test Command

The DA16200 (DA16600) SDK includes sample code and CLI commands to make it easier for users to use the OTA update. It is possible to program directly by referring to the sample code, but the user can simply check the network status with the OTA server by using the CLI command before that.

Table 7: Download Example Using CLI Command

[/DA16200/NET] # ota_update rtos https://ota-server/DA16200_RTOS-GEN01-01-12345-000000.img > Server FW version: RTOS-GEN01-01-12345-000000 >> HTTP(s) Client Downloading... 100 % (800000/800000 Bytes) - OTA Update: <RTOS> Download - Success

Table 8: OTA Test Command

Command		Option	Description
ota_update		[update_type] [url]	Start to FW download. * update_type slib: update_type of SLIB rtos: update_type of RTOS cert_key: update_type of cert or key mcu_fw: update_type of MCU FW * url: Server URL where FW exists ex) ota_update rtos http://192.168.0.1/rtos.img
	stop		Stop to firmware download. ex) ota_update stop

Command		Option	Description
	renew		Change current firmware to new firmware. ex) ota_update renew
	info		Show firmware information. ex) ota_update info
	crc	[addr]	Check CRC of firmware. ex) ota_update crc 0x23000
	read_sflash	[addr] [size]	Read SFLASH data. ex) ota_update read 0x1f2000 128
	copy_sflash	[dst_addr] [src_addr] [size]	Copy from sflash data src_add to dst_add. ex) ota_update copy_sflash 0x1f6000 1f2000 4096
	erase_sflash	[addr] [size]	Erase sflash data. ex) ota_update erase_sflash 1f2000 4096
	init_mcu		UART1 initialization. ex) ota_update init_mcu
	set_name_mcu		Set the name (version) of MCU FW to be downloaded to sflash. ex) ota_update set_name_mcu MCU_FW
	get_name_mcu		Get name (version) of MCU FW downloaded to sflash. ex) ota_update get_name_mcu
	read_mcu		Read the firmware as much as the size from the read_addr and transmit it. ex) ota_update read_sflash 0x1f2000 4096
	trans_mcu		Transmit a firmware to MCU through UART1. ex) ota_update trans_mcu
	erase_mcu		Erase the firmware stored in serial flash of DA16200 (DA16600). ex) ota_update erase_mcu
	get_boot_index		Get current boot index info. ex) ota_update get_boot_index
	toggle_boot_index		Toggle boot index. ex) ota_update toggle_boot_index

5.4.2 Sample Code

The DA16200 (DA16600) SDK provides sample code and user guide:

- Sample code

The sample code includes not only the DA16200 (DA16600)'s firmware update, but also a sample of the MCU firmware and certificate update.

..\sample\Network\OTA_Update\src\ota_update_sample.c

- User Guide

  PDF\UM-WI-007_DA16200_Example_Application_Manual.pdf

6 OTA Update Extension

The OTA update function supports updating not only the DA16200 (DA16600) firmware, but also the firmware of the MCU chip or the certificate for TLS protocol.

6.1 Certificates

To update the SFLASH *TLS Certificate #0* and *TLS Certificate #1* areas:

1. Download directly to SFLASH *TLS Certificate #0* or *TLS Certificate #1* or download to User Area and copy to SFLASH *TLS Certificate #0* or *TLS Certificate #1*.
2. Set *uri_other* and *download_complete_notify* to suit user environment. And *other_type* should be set to *OTA_TYPE_CERT_KEY*. For *download_sflash_addr*, enter the SFHLASH address where the downloaded certificate will be saved. If *download_sflash_addr* is not set, the default is *User Area* (see Table 1 and Table 2).
3. Call the *ota_update_start_download()* API with the set parameter value.
4. If a user confirmed that the download was successful through download_complete_notify, user can copy, read, or erase the certificate using the *ota_update_copy_flash()*, *ota_update_read_flash()*, and *ota_update_erase_flash()* APIs.

Certificate update Pseudo code. See the sample code for more detailed usage.

```
conf.uri_other = "http://ota_server/cert.pem"

conf.other_type = OTA_TYPE_CERT_KEY;

conf.download_sflash_addr = SFLASH address;

conf.download_complete_notify = download_complete_notify();


ota_update_start_download(conf);


If Download success
      ota_update_copy_flash(dest_addr, src_addr, length);
```

6.2 MCU Firmware

To update the firmware of the MCU connected to the DA16200 (DA16600) UART:

1. Set *uri_other* and *download_complete_notify* to suit the user environment. And *other_type* should be set as *OTA_TYPE_MCU_FW*. For *download_sflash_addr*, input SFHLASH address where downloaded MCU FW will be saved. If *download_sflash_addr* is not set, the default is *User Area* (see Table 1 and Table 2).
2. Call the *ota_update_start_download()* API with the set parameter value.
3. If a user confirmed that the download was successful through *download_complete_notify*, transfer FW to the MCU using UART1.
4. Initialize UART1 by calling the *ota_update_uart1_init()* API before sending.
5. Call the *ota_update_uart_trans_mcu_fw()* API to start the transfer. If the user needs hardware configuration, contact our customer service.

MCU FW update Pseudo code. See the sample code for more detailed usage.

```
conf.uri_other = "http://ota_server/mcu_fw.img"
conf.other_type = OTA_TYPE_MCU_FW;
conf.download_sflash_addr = SFLASH address;
conf.download_complete_notify = download_complete_notify();

ota_update_start_download(conf);

If Download success
        ota_update_uart1_init();
        ota_update_uart_trans_mcu_fw();
```

6.2.1 UART Protocol with MCU

The DA16200 (DA16600) and MCU uses a simple UART protocol.

Table 9: UART Control Characters

Define	Value	Remarks
SOH	0x01	Start of heading
STX	0x02	Start of Text
ACK	0x06	Acknowledge
NAK	0x15	Negative Acknowledge

Steps:

1. Transfer
   The MCU firmware stored in SFLASH of DA16200 (DA16600) is transmitted to the MCU.

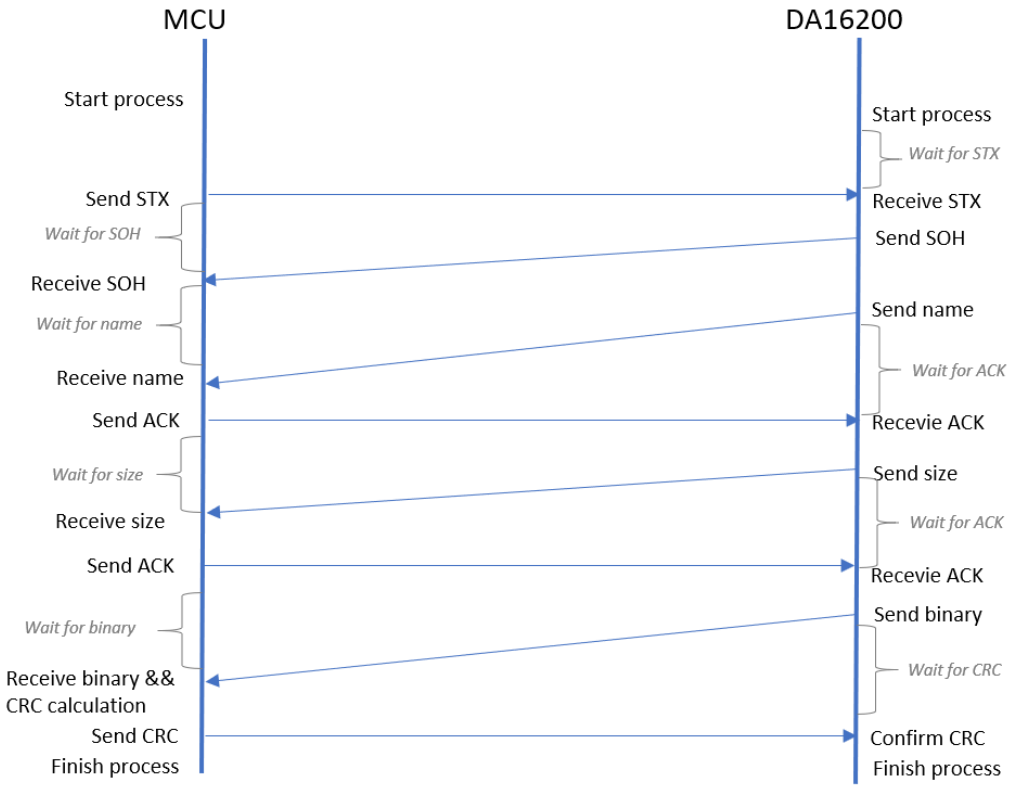

Figure 6: Transfer

a. STX and SOH announce the start of UART1 transmission.
b. Receive name: Check if the MCU FW is correct.
c. Receive size: Prepare a buffer to receive FW.
d. Receive binary && CRC Calculation: Calculate CRC when binary reception is completed.
e. Send CRC: The calculated CRC is transmitted to the DA16200 (DA16600).

2. Read.

   Used to inquire the downloaded MCU FW. It reads as much as the read_size from the read_addr and sends it to MCU through UART1. Unlike Transferring procedure, there is no process to check the name and CRC.

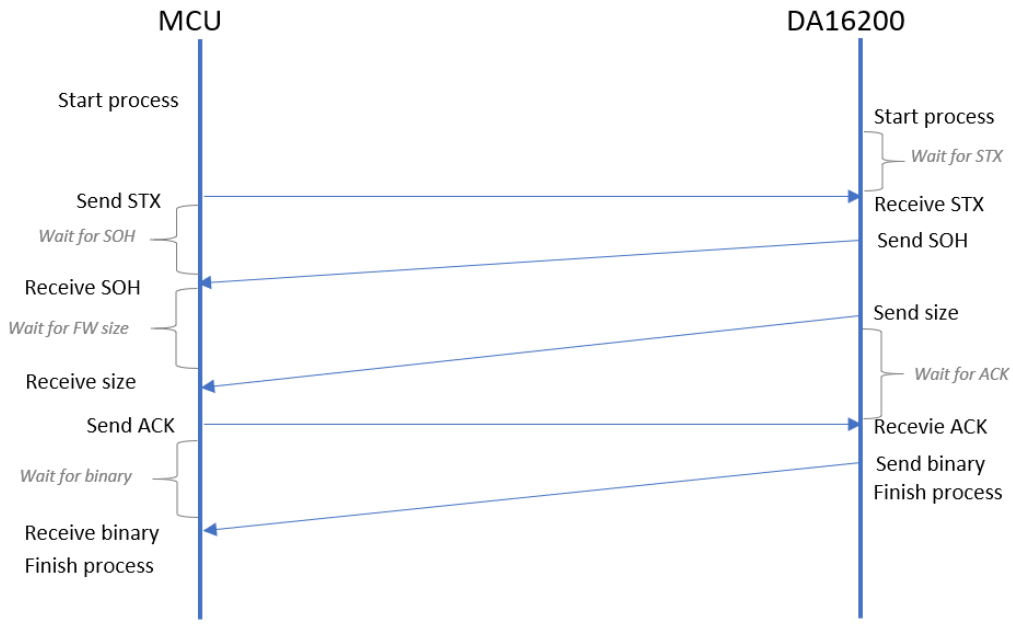

Figure 7: Read

6.2.2 Calculation CRC-32

This is an example for calculating the CRC value required in the Transfer protocol.

```
static const unsigned int ota_crc_table[] =
{
  0x00000000L, 0x77073096L, 0xee0e612cL, 0x990951baL, 0x076dc419L,
  0x706af48fL, 0xe963a535L, 0x9e6495a3L, 0x0edb8832L, 0x79dcb8a4L,
  0xe0d5e91eL, 0x97d2d988L, 0x09b64c2bL, 0x7eb17cbdL, 0xe7b82d07L,
  0x90bf1d91L, 0x1db71064L, 0x6ab020f2L, 0xf3b97148L, 0x84be41deL,
  0x1adad47dL, 0x6ddde4ebL, 0xf4d4b551L, 0x83d385c7L, 0x136c9856L,
  0x646ba8c0L, 0xfd62f97aL, 0x8a65c9ecL, 0x14015c4fL, 0x63066cd9L,
  0xfa0f3d63L, 0x8d080df5L, 0x3b6e20c8L, 0x4c69105eL, 0xd56041e4L,
  0xa2677172L, 0x3c03e4d1L, 0x4b04d447L, 0xd20d85fdL, 0xa50ab56bL,
  0x35b5a8faL, 0x42b2986cL, 0xdbbbc9d6L, 0xacbcf940L, 0x32d86ce3L,
  0x45df5c75L, 0xdcd60dcfL, 0xabd13d59L, 0x26d930acL, 0x51de003aL,
  0xc8d75180L, 0xbfd06116L, 0x21b4f4b5L, 0x56b3c423L, 0xcfba9599L,
  0xb8bda50fL, 0x2802b89eL, 0x5f058808L, 0xc60cd9b2L, 0xb10be924L,
  0x2f6f7c87L, 0x58684c11L, 0xc1611dabL, 0xb6662d3dL, 0x76dc4190L,
  0x01db7106L, 0x98d220bcL, 0xefd5102aL, 0x71b18589L, 0x06b6b51fL,
  0x9fbfe4a5L, 0xe8b8d433L, 0x7807c9a2L, 0x0f00f934L, 0x9609a88eL,
  0xe10e9818L, 0x7f6a0dbbL, 0x086d3d2dL, 0x91646c97L, 0xe6635c01L,
```

```
  0x6b6b51f4L, 0x1c6c6162L, 0x856530d8L, 0xf262004eL, 0x6c0695edL,
  0x1b01a57bL, 0x8208f4c1L, 0xf50fc457L, 0x65b0d9c6L, 0x12b7e950L,
  0x8bbeb8eaL, 0xfcb9887cL, 0x62dd1ddfL, 0x15da2d49L, 0x8cd37cf3L,
  0xfbd44c65L, 0x4db26158L, 0x3ab551ceL, 0xa3bc0074L, 0xd4bb30e2L,
  0x4adfa541L, 0x3dd895d7L, 0xa4d1c46dL, 0xd3d6f4fbL, 0x4369e96aL,
  0x346ed9fcL, 0xad678846L, 0xda60b8d0L, 0x44042d73L, 0x33031de5L,
  0xaa0a4c5fL, 0xdd0d7cc9L, 0x5005713cL, 0x270241aaL, 0xbe0b1010L,
  0xc90c2086L, 0x5768b525L, 0x206f85b3L, 0xb966d409L, 0xce61e49fL,
  0x5edef90eL, 0x29d9c998L, 0xb0d09822L, 0xc7d7a8b4L, 0x59b33d17L,
  0x2eb40d81L, 0xb7bd5c3bL, 0xc0ba6cadL, 0xedb88320L, 0x9abfb3b6L,
  0x03b6e20cL, 0x74b1d29aL, 0xead54739L, 0x9dd277afL, 0x04db2615L,
  0x73dc1683L, 0xe3630b12L, 0x94643b84L, 0x0d6d6a3eL, 0x7a6a5aa8L,
  0xe40ecf0bL, 0x9309ff9dL, 0x0a00ae27L, 0x7d079eb1L, 0xf00f9344L,
  0x8708a3d2L, 0x1e01f268L, 0x6906c2feL, 0xf762575dL, 0x806567cbL,
  0x196c3671L, 0x6e6b06e7L, 0xfed41b76L, 0x89d32be0L, 0x10da7a5aL,
  0x67dd4accL, 0xf9b9df6fL, 0x8ebeeff9L, 0x17b7be43L, 0x60b08ed5L,
  0xd6d6a3e8L, 0xa1d1937eL, 0x38d8c2c4L, 0x4fdff252L, 0xd1bb67f1L,
  0xa6bc5767L, 0x3fb506ddL, 0x48b2364bL, 0xd80d2bdaL, 0xaf0a1b4cL,
  0x36034af6L, 0x41047a60L, 0xdf60efc3L, 0xa867df55L, 0x316e8eefL,
  0x4669be79L, 0xcb61b38cL, 0xbc66831aL, 0x256fd2a0L, 0x5268e236L,
  0xcc0c7795L, 0xbb0b4703L, 0x220216b9L, 0x5505262fL, 0xc5ba3bbeL,
  0xb2bd0b28L, 0x2bb45a92L, 0x5cb36a04L, 0xc2d7ffa7L, 0xb5d0cf31L,
  0x2cd99e8bL, 0x5bdeae1dL, 0x9b64c2b0L, 0xec63f226L, 0x756aa39cL,
  0x026d930aL, 0x9c0906a9L, 0xeb0e363fL, 0x72076785L, 0x05005713L,
  0x95bf4a82L, 0xe2b87a14L, 0x7bb12baeL, 0x0cb61b38L, 0x92d28e9bL,
  0xe5d5be0dL, 0x7cdcefb7L, 0x0bdbdf21L, 0x86d3d2d4L, 0xf1d4e242L,
  0x68ddb3f8L, 0x1fda836eL, 0x81be16cdL, 0xf6b9265bL, 0x6fb077e1L,
  0x18b74777L, 0x88085ae6L, 0xff0f6a70L, 0x66063bcaL, 0x11010b5cL,
  0x8f659effL, 0xf862ae69L, 0x616bffd3L, 0x166ccf45L, 0xa00ae278L,
  0xd70dd2eeL, 0x4e048354L, 0x3903b3c2L, 0xa7672661L, 0xd06016f7L,
  0x4969474dL, 0x3e6e77dbL, 0xaed16a4aL, 0xd9d65adcL, 0x40df0b66L,
  0x37d83bf0L, 0xa9bcae53L, 0xdebb9ec5L, 0x47b2cf7fL, 0x30b5ffe9L,
  0xbdbdf21cL, 0xcabac28aL, 0x53b39330L, 0x24b4a3a6L, 0xbad03605L,
  0xcdd70693L, 0x54de5729L, 0x23d967bfL, 0xb3667a2eL, 0xc4614ab8L,
  0x5d681b02L, 0x2a6f2b94L, 0xb40bbe37L, 0xc30c8ea1L, 0x5a05df1bL,
  0x2d02ef8dL
};

/* update the CRC on the data block one byte at a time */
static unsigned int update_crc (unsigned int init, const unsigned char *buf, int len)
```

```
{
        unsigned int crc = init;

        while (len--)
                crc = ota_crc_table[(crc ^ *(buf++)) & 0xFF] ^ (crc >> 8);

        return ~crc;
}
```

Appendix A OTA Test Server

OTA update complies with HTTP protocol to download firmware. Therefore, users can easily implement an OTA server using a HTTP server. This manual does not provide a guide on configuring OTA servers. However, it explains how to configure a simple test environment for functional testing in the application development stage.

AWS S3 is recommended as the OTA test server. This allows users to easily create HTTPS servers without having to set up a server domain or TLS certificate. Use it only for simple testing during the development phase. Charges may apply depending on the amount of data used. Check out AWS' pricing policy.

Steps:

1. Sign up for an AWS account for free and log in to the console (see Figure 8).

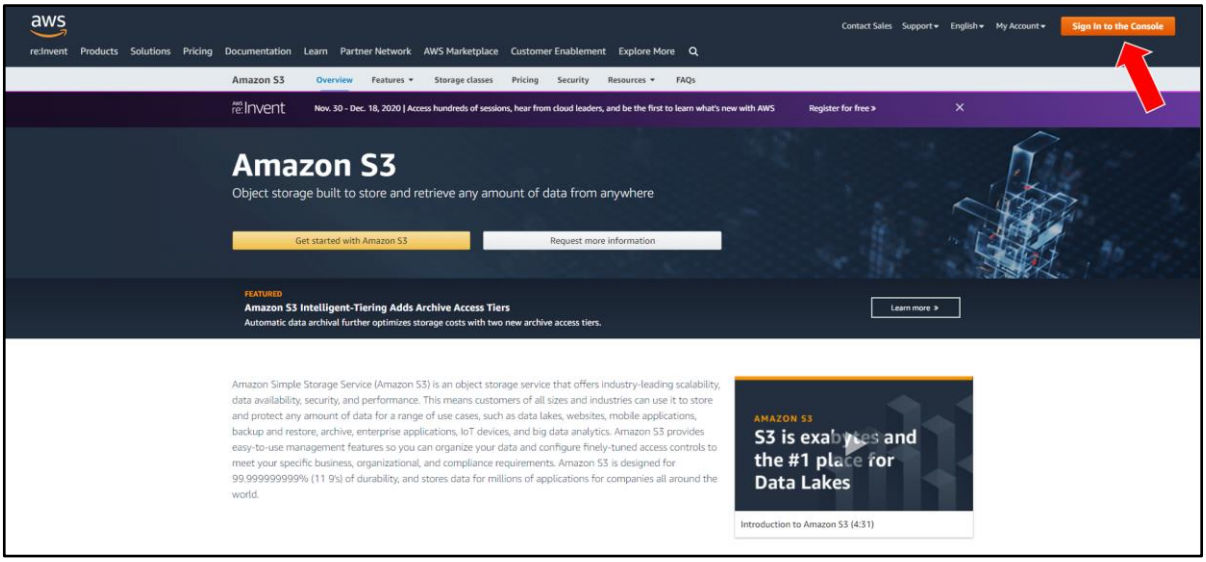

Figure 8: Sign In to AWS Console

2. In the **Storage** category, click **S3** as in Figure 9.

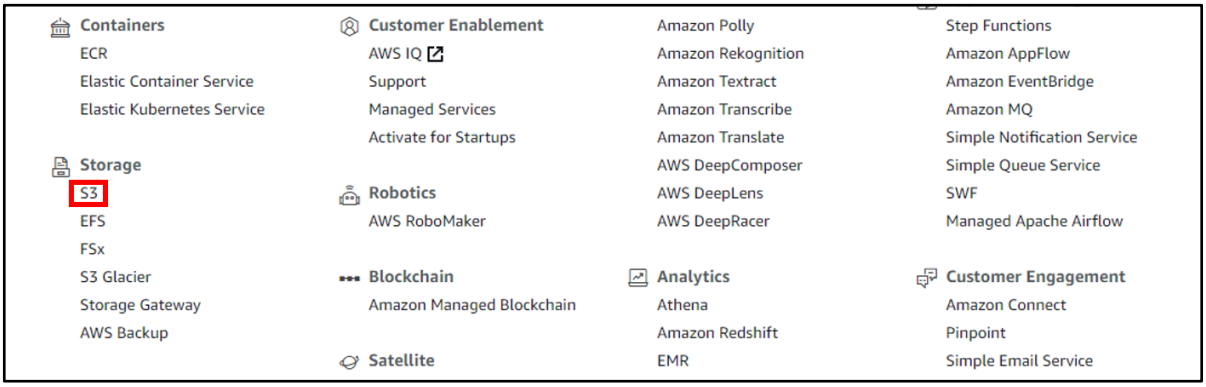

Figure 9: AWS Categories

3. To create a bucket with default settings, click **Create Bucket** as in Figure 10.

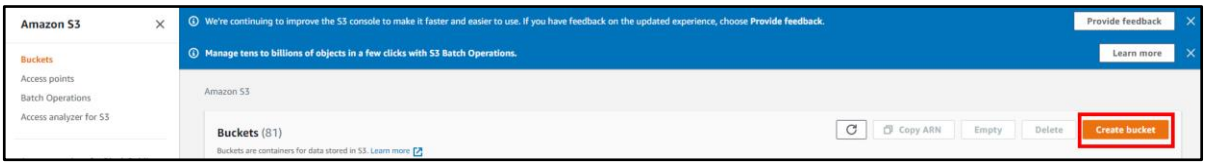

Figure 10: Create Bucket Selection Window

4. Upload the firmware to the created bucket as in Figure 11.

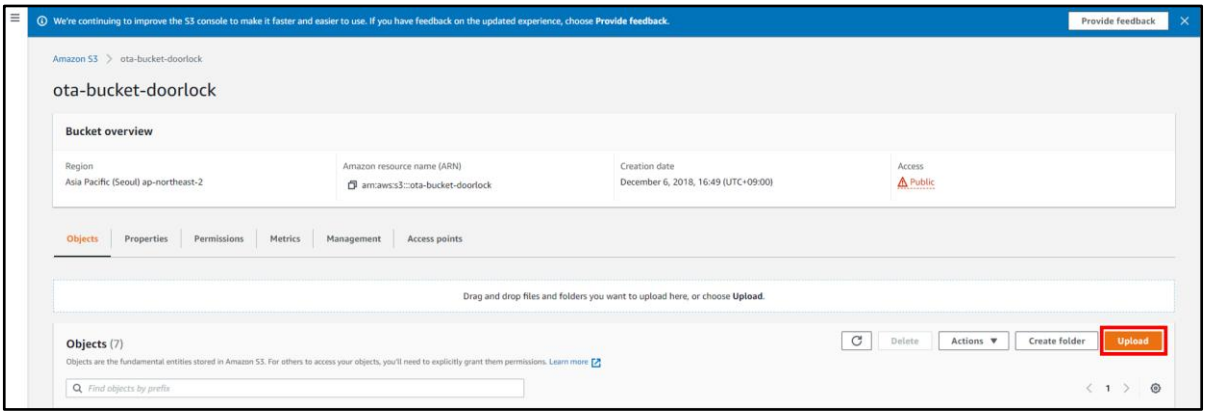

Figure 11: Upload Window

5. Check the URL (https://) of the uploaded firmware as in Figure 12.

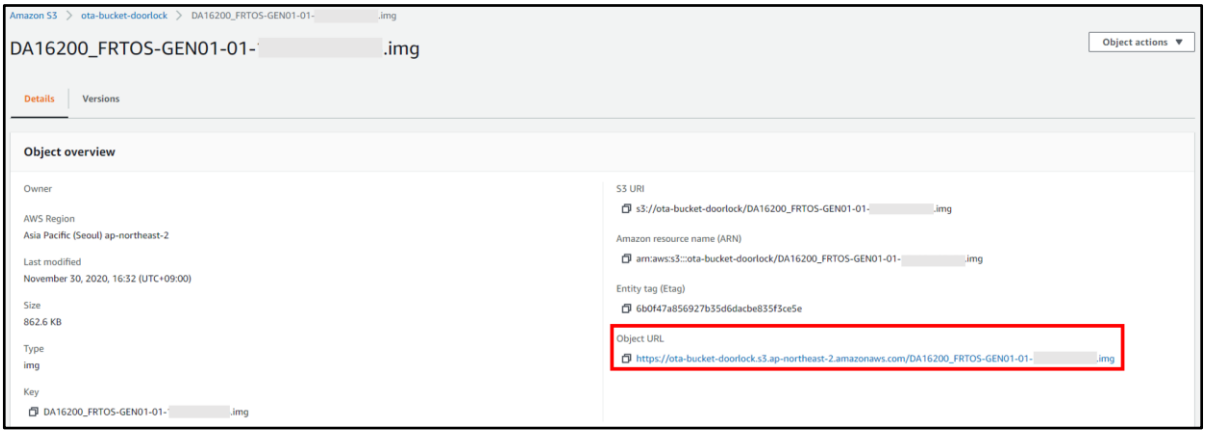

Figure 12: URL Displayed

6. Set the URL as the OTA update API parameter value and proceed with the test.

Revision History

Revision	Date	Description
2.3	28-Mar-2022	Update logo, disclaimer, copyright.
2.2	29-Nov-2021	Title was changed.
2.1	12-May-2021	Added Note for SFLASH corruption.
2.0	17-Feb-2021	Added description for: ● SFLASH Copy, Read, Delete API description ● MCU FW, Certificate update description Modified: ● Table 4 OTA Update Type ● Table 5 OTA_UPDATE_CONFIG ● Table 6 Lists for OTA APIs (new API added)
1.0	02-Dec-2020	First Release

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

Important Notice and Disclaimer

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu

Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

https://www.renesas.com/contact/