User Manual

DA16200 DA16600 FreeRTOS OTA Update

UM-WI-048

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 to programming environments, debugging tools, and software engineering process in general is assumed.

Contents

Figures

Tables

Terms and Definitions

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

References

[1] DA16200, Datasheet, Renesas Electronics
[2] UM-WI-046, DA16200 DA16600, FreeRTOS SDK Programmer Guide, Renesas Electronics
[3] lwIP, Lightweight IP stack, https://www.nongnu.org/lwip/2_1_x/group__httpc.html
[4] UM-WI-055, DA16200 FreeRTOS Example Application Guide, Renesas Electronics
[5] UM-WI-052, DA16600 FreeRTOS Example Application Manual, Renesas Electronics
[6] UM-WI-056, DA16200 DA16600 FreeRTOS Getting Started Guide, Renesas Electronics

1 Introduction

The DA16200 (DA16600) provides support for over the air (OTA) firmware update using the HTTP protocol. The DA16200 (DA16600) operates as an HTTP client which can download and update new firmware from an HTTP server.

The DA16200 firmware image set consists of Bootloader (secondary bootloader) and RTOS. The boot loader cannot be updated via OTA, only RTOS. This product enables application programmer to develop an OTA FW update application that uses the OTA APIs.

In addition, users can update certificates (TLS Certificate Key #1 and TLS Certificate Key #2) and support firmware update of MCU connected by UART1.

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

NOTE
When DPM mode is enabled and an OTA (firmware) update is in progress, DPM Sleep Mode will not be entered due to SFLASH write operations. Once the firmware update is complete, DPM Sleep Mode will return to normal operation.

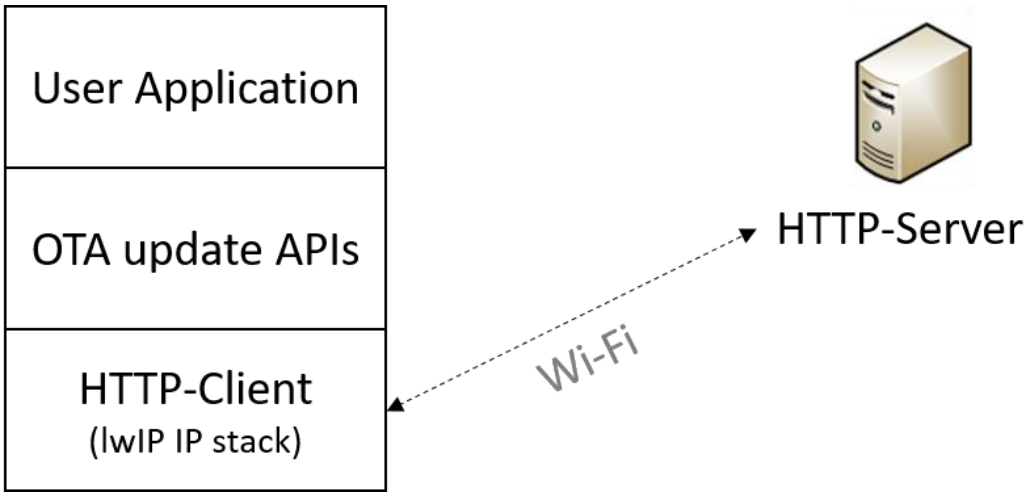


Figure 1: OTA Update Layer

2 SFLASH Memory Area

The DA16200 (DA16600) does not support file systems, so the firmware is stored in the SFLASH memory. The 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 areas:
  - RTOS #0
  - RTOS #1
  - TLS Certificate #1
  - TLS Certificate #2
  - User Area

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

Table 1: 4 MB SFLASH Memory Map

Address	Name		Size (KB)
0x0000_0000	2nd Bootloader		136
0x0002_2000	Boot Index		4
0x0002_3000	RTOS #0		1788
0x001E_2000	RTOS #1		1788
0x003A_1000	Reserved Area		4
0x003A_2000	Debug / RMA Certificate		4
0x003A_3000	TLS Certificate #1 (MQTT)	CA	4
0x003A_4000		Cert	4
0x003A_5000		Private key	4
0x003A_6000		Diffie-Hellmann key	4
0x003A_7000	TLS Certificate #2 (HTTPs / OTA)	CA	4
0x003A_8000		Cert	4
0x003A_9000		Private key	4
0x003A_A000		Diffie-Hellmann key	4
0x003A_B000	NVRAM #0		4
0x003A_C000	NVRAM #1 (Backup)		4
0x003A_D000	User Area (including DA14531 image, See NOTE)		256
0x003E_D000	TLS Certificate Key #3 (WPA Enterprise)	CA	4
0x003E_E000		Cert	4
0x003E_F000		Private	4
0x003F_0000		Diffie-Hellmann key	4
0x003F_1000	TLS Certificate Key #4 (Reserved)	CA	4
0x003F_2000		Certificate	4
0x003F_3000		Private Key	4
0x003F_4000		Diffie-Hellmann key	4
0x003F_5000	NVRAM FOOTPRINT		4
0x003F_6000	AT-CMD TLS Certificate Key #0 ~ #9		40

NOTE

For DA16600, the DA14531 image is stored in User Area (0x003A_D000 ~ 0x003C_1FFF). See the Table 32 in [6] for further details.

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 from 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

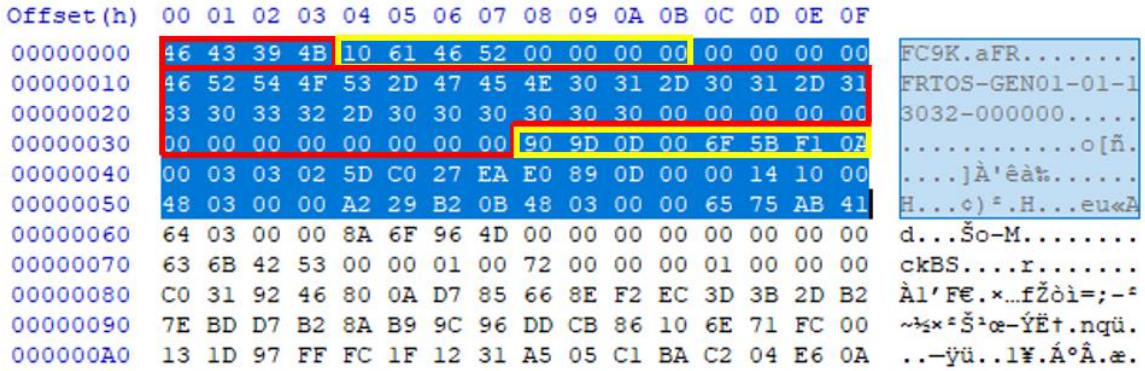
The OTA update function is divided into two stages: DOWNLOAD and RENEW.

DOWNLOAD refers to the process of downloading the 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 is 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.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	46	43	39	4B	10	61	46	52	00	00	00	00	00	00	00	00	FC9K.aFR........
00000010	46	52	54	4F	53	2D	47	45	4E	30	31	2D	30	31	2D	31	FRTOS-GEN01-01-1
00000020	33	30	33	32	2D	30	30	30	30	30	30	00	00	00	00	00	3032-000000.....
00000030	00	00	00	00	00	00	00	00	90	9D	0D	00	6F	5B	F1	02	............o[ñ.
00000040	00	03	03	02	5D	C0	27	EA	E0	89	0D	00	00	14	10	00	....]À'êà‰......
00000050	48	03	00	00	A2	29	B2	0B	48	03	00	00	65	75	AB	41	H...¢)².H...eu«A
00000060	64	03	00	00	8A	6F	96	4D	00	00	00	00	00	00	00	00	d...Šo–M........
00000070	63	6B	42	53	00	00	01	00	72	00	00	00	01	00	00	00	ckBS....r.......
00000080	C0	31	92	46	80	0A	D7	85	66	8E	F2	EC	3D	3B	2D	B2	À1'F€.×…fŽòì=;-²
00000090	7E	BD	D7	B2	8A	B9	9C	96	DD	CB	86	10	6E	71	FC	00	~½×²Š¹œ–ÝË†.nqü.
000000A0	13	1D	97	FF	FC	1F	12	31	A5	05	C1	BA	C2	04	E6	0A	..—ÿü..1¥.Á°Â.æ.

Figure 2: Firmware Header Information

4.2 Version

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

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

Version String

Type-Vendor-Major-Minor-Customer

1. Type (4 bytes): Identify the type of firmware.
1. Vendor (5 bytes): Vendor classification.
2. Major (2 bytes): Major number to check compatibility.
3. Minor (4~5 bytes): SDK patch number.
4. 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 as shown in the Table 2. It is delivered through the callback function connected with DOWNLOAD and RENEW API.

Table 2: 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 was not found on the server.
OTA_NOT_CONNECTED	0x0A	Failed to connect to the 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 2 for results):

5. Request a query from the HTTP server.

6. 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 2 for result values.
7. 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.
8. 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.
9. When the CRC check is successfully completed, the result value of 0x00 is transferred to the callback function and the download is terminated.

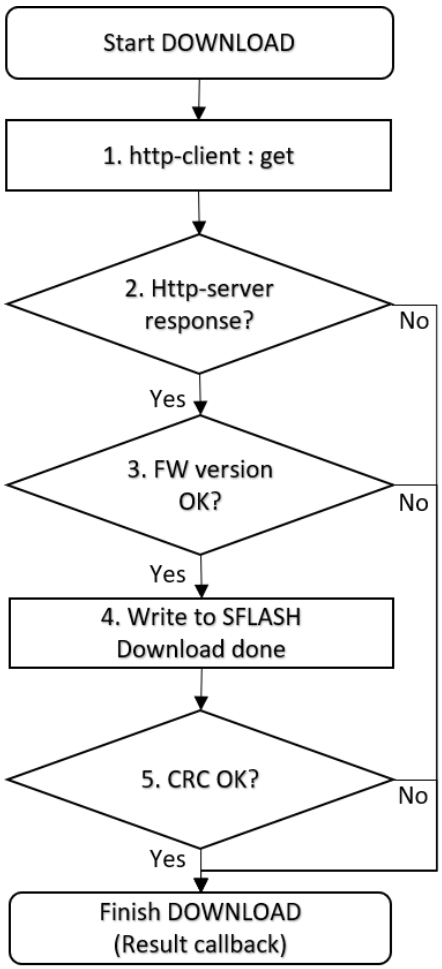


Figure 3: Firmware DOWNLOAD

4.5 Renew

RENEW only operates when the firmware download is successful. DA16200 (DA16600) should have the download history after power on.

1. Check whether the download was 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 firmware version stored in the flash. In case of failure, RENEW ends and the result is transmitted to the callback function.
4. Determine if the new firmware is normal and change the boot index to the new firmware location.

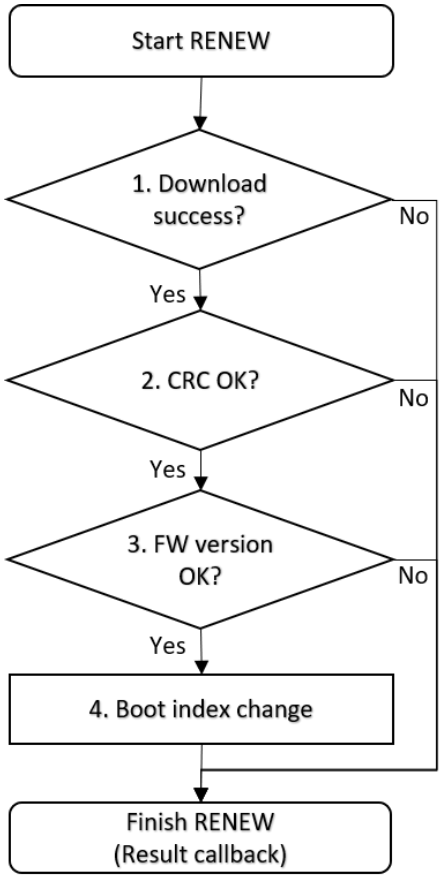


Figure 4: Firmware RENEW

4.5.1 Boot Index

DA16200 (DA16600) is divided into firmware download area and current area for 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.

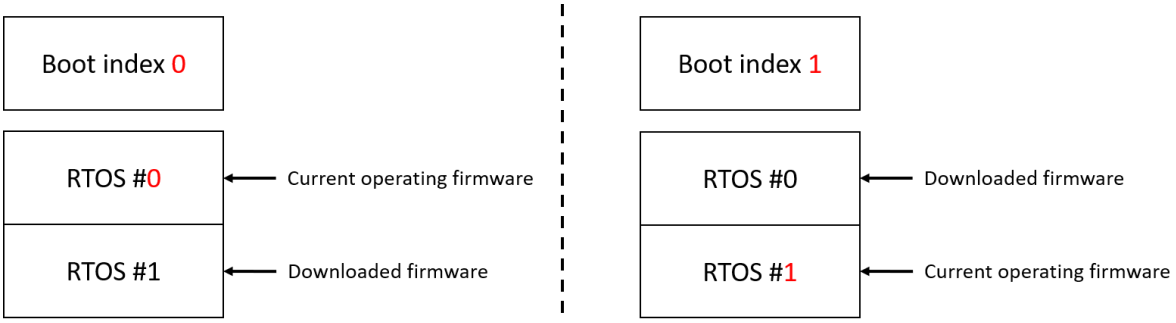


Figure 5: Boot Index Operation

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).

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 3: OTA Update Type

Name	ota_update_type
Description	Identify and specify targets for OTA updates.
	/// Operation step of process typedef enum { /// Init value OTA_TYPE_INIT, /// RTOS OTA_TYPE_RTOS, /// MCU firmware, not DA16x OTA_TYPE_MCU_FW, /// Certificate or Key OTA_TYPE_CERT_KEY, /// Unknown value OTA_TYPE_UNKNOWN } ota_update_type;

5.2 Structure

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

Table 4: 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 { /// Update type. ota_update_type update_type; /// Pointer variable to the server address where the RTOS exists. char uri[HTTP_MAX_RESOURCE]; /// Callback function pointer to check the download status. void (*download_notify)(ota_update_type update_type, UINT ret_status, UINT progress); /// Callback function pointer to check the renew state. Only for RTOS. void (*renew_notify)(UINT ret_status); /// If the value is true, if the new firmware download is successful, it will reboot with the new firmware. Only for RTOS UINT auto_renew; /// Address of sflash where other_fw is stored. Only for MCU_FW and CERT_KEY UINT download_sflash_addr; } OTA_UPDATE_CONFIG;

5.3 APIs

Describes the API required for the OTA firmware update application.

Table 5: Lists for OTA APIs

UINT ota_update_start_download(OTA_UPDATE_CONFIG *ota_update_conf)		
Parameter	[in] ota_update_conf	The pointer of OTA_UPDATE_CONFIG structure. update_type: Update type. uri: Server address where firmware exists. (*download_notify)(ota_update_type update_type, UINT ret_status, UINT progress): Callback function pointer to check the download status. (*renew_notify)(UINT ret_status): Callback function pointer to check the renew state. Only for RTOS. auto_renew: If the value is true, if the new firmware download is successful, it will reboot with the new firmware. Only for RTOS. download_sflash_addr: Address of sflash where other_fw is stored. Only for MCU_FW and CERT_KEY.
Return		Returns 0x00 on success. See Table 2.

UINT ota_update_start_download(OTA_UPDATE_CONFIG *ota_update_conf)	
Description	HTTP-client task is created and sent a query to the 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 2.
Description		A download can be stopped while downloading from the HTTP server.

UINT ota_update_get_download_progress(ota_update_type update_type)		
Parameter	[in] update_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 *ota_update_conf)		
Parameter	[in] ota_update_conf	The pointer of OTA_UPDATE_CONFIG structure.
Return		Returns 0x00 on success. See Table 2.
Description		Checks the version compatibility and CRC, changes the boot index to the new firmware location, and then reboots automatically.

UINT ota_update_get_new_sflash_addr(UINT update_type)		
Parameter	[in] update_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 2.

UINT ota_update_read_flash(UINT addr, VOID *buf, UINT len)	
Description	Reads 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 2.
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 8 bytes.
Return		Returns 0x00 on success. See Table 2.
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 2.
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 2.
Description		Gets 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 2.
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 2.
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 2.
Description		Deletes 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 2.
Description		Calculates 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) firmware, the user should always download RTOS.

1. Configure 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

```
char uri[256] = "https://ota_server/DA16200_FRTOS-GEN01-01-12345-000001.img";
ota_update_conf->update_type = OTA_TYPE_RTOS;
memcpy(ota_update_conf->uri, uri, HTTP_MAX_RESOURCE);
ota_update_conf->download_notify = download_notify;
ota_update_conf->renew_notify = renew_notify;

ota_update_start_download(ota_update_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.

- 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 6: OTA Test Command

Command		Option	Description
ota_update		[update_type] [url]	Start to FW download. * update_type 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
	renew		Change current firmware to new firmware. ex) ota_update renew
	info		Show FW information. ex) ota_update info
	crc	[addr]	Check CRC of firmware. ex) ota_update crc 0x1e2000
	read_sflash	[addr] [size]	Read sflash data. ex) ota_update read_sflash 0x1e2000 128

Command		Option	Description
	copy_sflash	[dst_addr] [src_addr] [size]	Copy from sflash data src_add to dst_add. ex) ota_update copy_sflash 0x3ad000 0x1e2000 4096
	erase_sflash	[addr] [size]	Erase sflash data. ex) ota_update erase_sflash 0x3ad000 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 0x3ad000 4096
	trans_mcu		Transmit a firmware to MCU through UART1. ex) ota_update trans_mcu
	erase_mcu		Erase the firmware stored in a 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
  See [1] for further details

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 #1* and *TLS Certificate #2* areas:

4. Download directly to SFLASH *TLS Certificate #1* or *TLS Certificate #2*, or download to User Area and copy to SFLASH *TLS Certificate #1* or *TLS Certificate #2*.
5. Set *uri* and *download_notify* to suit user environment. And *update_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).
6. Call the *ota_update_start_download()* API with the set parameter value.
7. If user have confirmed that the download was successful through *download_notify*, user can copy, read, or delete 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

```
char uri[256] = "https://ota_server/cert.pem";
ota_update_conf->update_type = OTA_TYPE_CERT_KEY;
memcpy(ota_update_conf->uri, uri, HTTP_MAX_RESOURCE);
ota_update_conf->download_notify = download_notify;
ota_update_conf->download_sflash_addr = SFLASH address;

ota_update_start_download(ota_update_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* and *download_notify* to suit the user environment. And *update_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).
2. Call the *ota_update_start_download()* API with the set parameter value.
3. If the user has confirmed that the download was successful through *download_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

```
char uri[256] = "https://ota_server/mcu_fw.img";
ota_update_conf->update_type = OTA_TYPE_MCU_FW;
memcpy(ota_update_conf->uri, uri, HTTP_MAX_RESOURCE);
ota_update_conf->download_notify = download_notify();
ota_update_conf->download_sflash_addr = SFLASH address;

ota_update_start_download(conf);
```

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

6.2.1 UART Protocol with MCU

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

Table 7: UART Control Characters

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

- **Procedure:**Transfer: The MCU firmware stored in SFLASH of DA16200 (DA16600) is transmitted to the MCU.

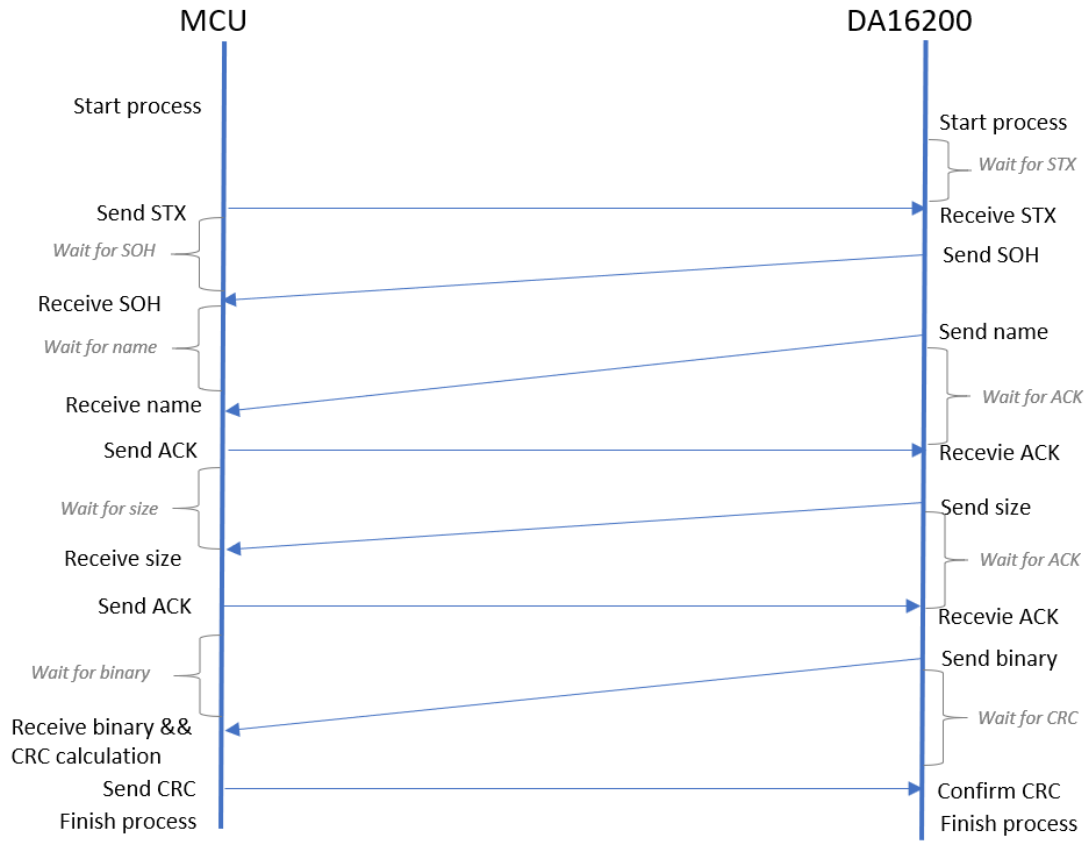


Figure 6: Transfer

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 complete.
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.

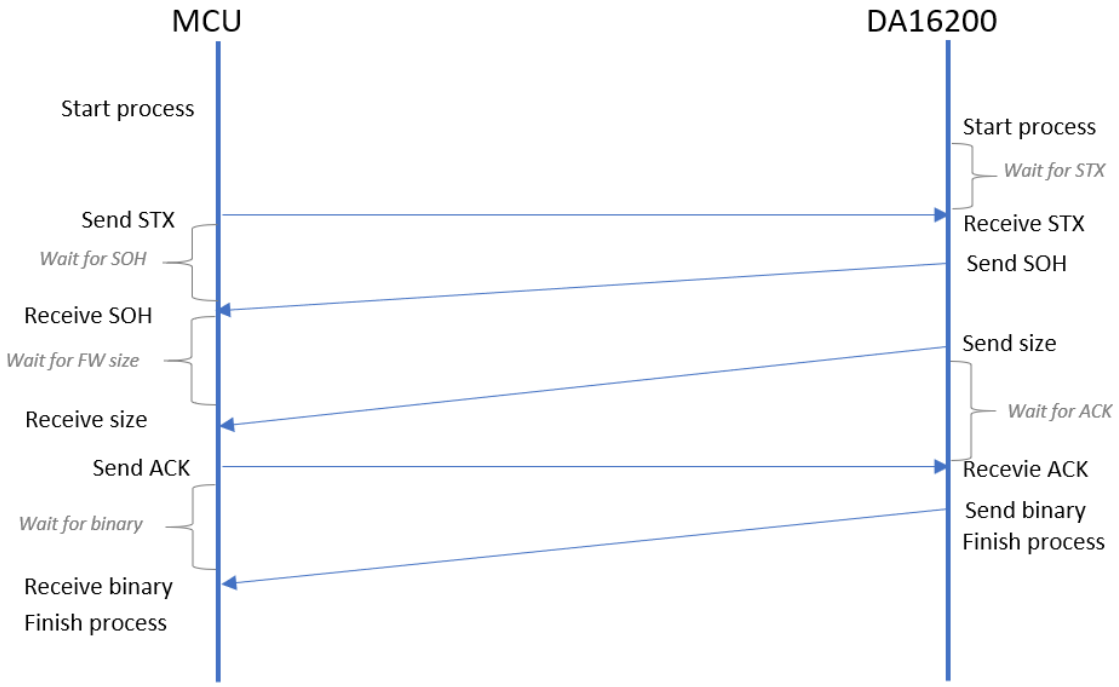


Figure 7: Read

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;

}
```


6.3 BLE Firmware update OTA

To update the BLE firmware of the DA16600 via the OTA, see the section 6.2 in [5]. After building the code of the DA14531 SDK, the following images are available to update DA14531 firmware via the OTA function.

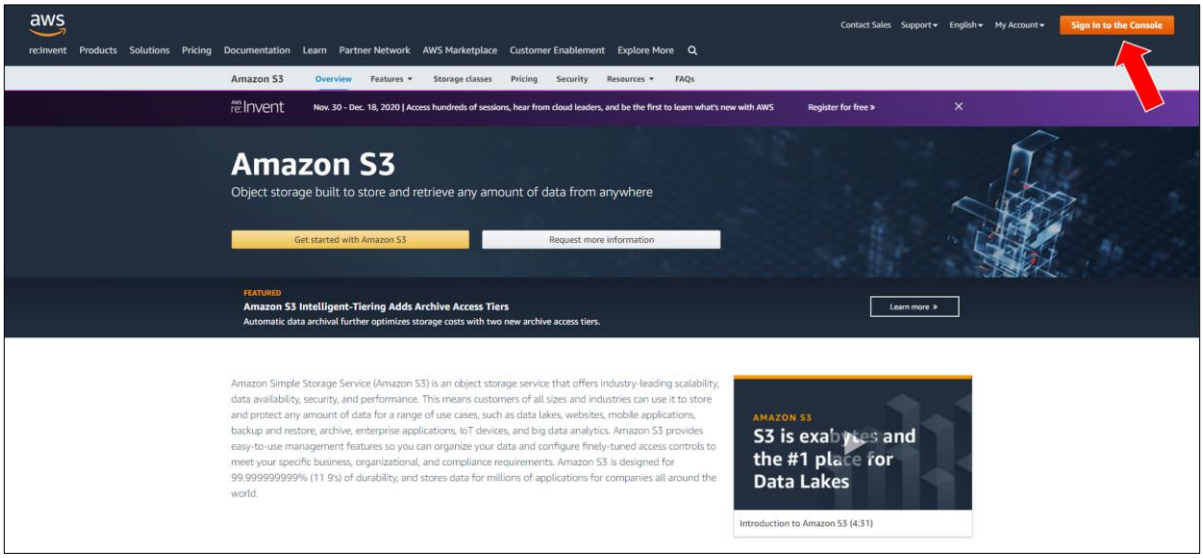
- The DA14531 SDK.
  `[DA16600 SDK ROOT]\utility\combo\da14531_sdk_v_6.0.14.1114.zip`
- The BLE OTA firmware images for the DA16600 examples (after code build):
  - IoT Sensor gateway example (central example)
    `[DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_monitor_aux_ext_coex\Keil_5\out_img\px`**`m`**`_coex_ext_531_6_0_14_1_ota.img.`
  - Rest of the DA16600 examples (peripheral examples)
    `[DA14531_SDK_ROOT]\projects\target_apps\ble_examples\prox_reporter_sensor_ext_coex\Keil_5\out_img\px`**`r`**`_sr_coex_ext_531_6_0_14_1114_1_`**`ota`**`.img.`

Appendix A OTA Test Server

OTA update complies with HTTP protocol to download firmware. Therefore, users can easily implement an OTA server using 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.

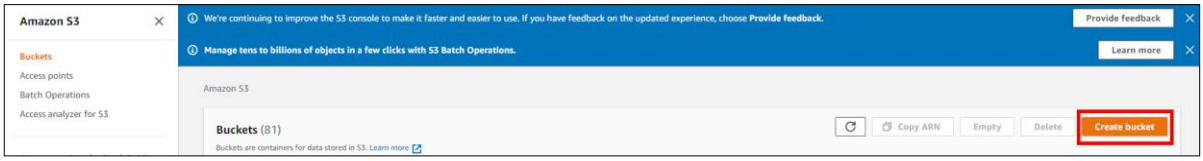
AWS S3 is recommended as the OTA test server.

1. Sign up for an AWS account for free and log in to the console.

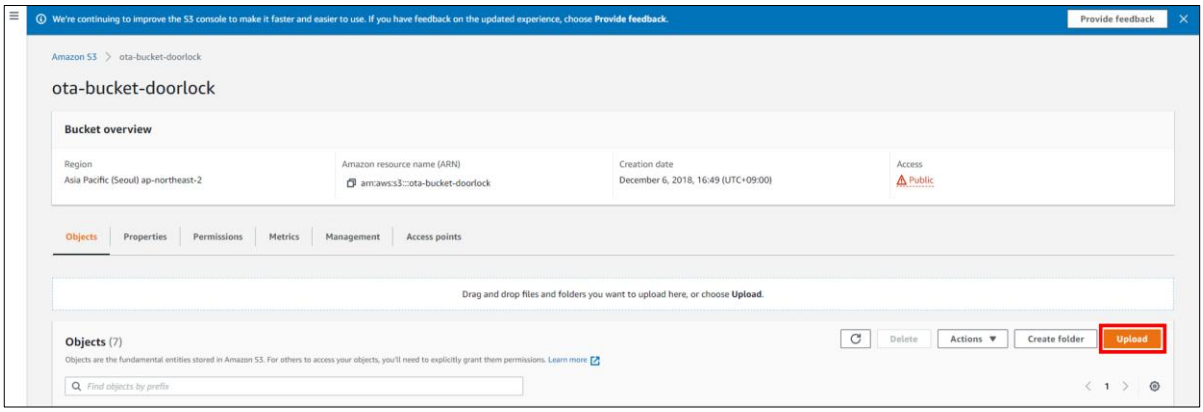


2. In the **Storage** category, select **S3**.

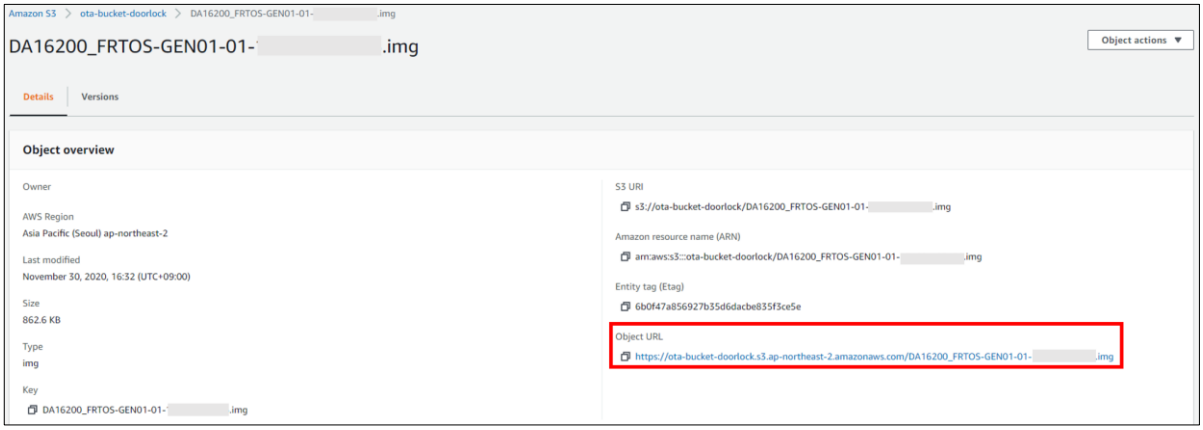
3. To create a bucket with default settings, click **Create Bucket**.



4. Upload the firmware to the created bucket.



5. Check the URL (https://) of the uploaded firmware.



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

Revision History

Revision	Date	Description
1.4	24-Aug-2022	Added 6.3 for BLE firmware update and typo corrections.
1.3	22-Jun-2022	Update the NOTE about DPM. Typo correction.
1.2	28-Mar-2022	Update logo, disclaimer, copyright.
1.1	29-Nov-2021	Tittle was changed.
1.0	26-Feb-2021	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

Renesas Electronics'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/