GSFS Standard Web Call API Documentation

Version1.0.1 - 2025-02-11

About This Document

Describes the standard API of the GSFS Web Call

Document Revision History

Version	Description	Date	Author
1.0.0	Initalize	2025.01.23	khonggu@lgcns.com
1.0.1	Add TrackRepair API response data	2025.02.11	khonggu@lgcns.com

Acronyms

For the purpose of the present documentation, the following abbreviations apply:

표1

Acronym	Description
GSFS	Global Service Front System
BMS	Business Management System
DSC	Direct Service Center (same meaning as LGC=LGE Service Center)
ASC	Authorized Service Center
ESC	Exclusive Service Center (repair LGE product only)
WIP	Work In Progress
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
JSON	JavaScript Object Notation
REST	Representational State Transfer
URL	Uniform Resource Locator
XML	Extensible Markup Language
AES	Advanced Encryption Standard
TBD	To be Determined

Table of Contents

1. Introduction

This document describes API specifications between the LG Electronics GSFS system and ASC BMS. All APIs described in this standard are in REST format based on HTTP standards.

2. Protocol Rule

API method is implemented according to REST (Representational State Transfer) method. Make a request according to the HTTP method and URL for the resource, and use JSON as the response data type. Also conforms to the HTTP 1.1 Spec standard

2.1. Resource URL

Request URL of GSFS System is as follows. The interface uses http or https

When connecting from LG Electronics' external network, you can connect through a proxy server. At this time, the SSL verify option must be disabled. Please refer to section 2.3 SSL Verify Disable for sample code.

2.1.1. America (AIC)

Server	Domain (or IP)	Database	SSL Verify	Remark
Dev (LGE Ext)	https://3.36.170.162:9420	Dev	**False**	AWS Proxy Server
Dev (LGE Int)	https://gsfsplus-america-dev.lge.com:8543	Dev	True	
Production	https://gsfsplus-america.lge.com	Production	True	

2.1.2. Europe (EIC)

Server	Domain (or IP)	Database	SSL Verify	Remark
Dev (LGE Ext)	https://3.36.170.162:9430	Dev	**False**	AWS Proxy Server
Dev (LGE Int)	https://gsfsplus-eu-dev.lge.com:8543	Dev	True	
Production	https://gsfsplus-eu.lge.com	Production	True	

2.1.3. Asis Pacific (APIC)

Server	Domain (or IP)	Database	SSL Verify	Remark
Dev (LGE Ext)	https://3.36.170.162:9440	Dev	**False**	AWS Proxy Server
Dev (LGE Int)	https://gsfsplus-dev.lge.com	Dev	True	
Production	https://gsfsplus.lge.com	Production	True	

2.1.4. India

Server	Domain (or IP)	Database	SSL Verify	Remark
Dev (LGE Ext)	https://3.36.170.162:9440	Dev	**False**	AWS Proxy Server
Dev (LGE Int)	https://gsfsplus-dev.lge.com	Dev	True	
Production	https://gsfsplus-india.lge.com	Production	True	

2.1.5. China

Server	Domain (or IP)	Database	SSL Verify	Remark
Dev (LGE Ext)	https://3.36.170.162:9440	Dev	**False**	AWS Proxy Server
Dev (LGE Int)	https://gsfsplus-dev.lge.com	Dev	True	
Production	https://gsfsplus-china.lge.com	Production	True	

2.1.6. Russia (RU)

Server	Domain (or IP)	Database	SSL Verify	Remark
Dev (LGE Ext)	https://3.36.170.162:9450	Dev	**False**	AWS Proxy Server
Dev (LGE Int)	https://gsfsplus-ru-dev.lge.com	Dev	True	
Production	https://gsfsplus-ru.lge.com	Production	True	

2.2. Code Sample for SSL Verify Disable in JAVA

This code is sample code for SSL disable in Java.The point is that call ingoreSsl() function before url connection.

```
url = new URL(strURL);
ignoreSsl();
con = (HttpsURLConnection)url.openConnection();
```

```
public static void ignoreSsl() throws Exception{
        HostnameVerifier hv = new HostnameVerifier() {
        public boolean verify(String urlHostName, SSLSession session)
                return true;
            }
        };
        trustAllHttpsCertificates();
        HttpsURLConnection.setDefaultHostnameVerifier(hv);
    }

 private static void trustAllHttpsCertificates() throws Exception {
        TrustManager[] trustAllCerts = new TrustManager[1];
        TrustManager tm = new miTM();
        trustAllCerts[0] = tm;
        SSLContext sc = SSLContext.getInstance("SSL");
        sc.init(null, trustAllCerts, null);
        HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
    }

    static class miTM implements TrustManager,X509TrustManager {
        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }

        public boolean isServerTrusted(X509Certificate[] certs) {
            return true;
        }

        public boolean isClientTrusted(X509Certificate[] certs) {
```

```
            return true;
        }

        public void checkServerTrusted(X509Certificate[] certs, String authType)
                throws CertificateException {
            return;
        }

        public void checkClientTrusted(X509Certificate[] certs, String authType)
                throws CertificateException {
            return;
        }
    }
```

2.3. Request Action Mapping through HTTP Method

Use the HTTP Method with the following rules

HTTP Method	Action	Meaning	Comment
GET	Read	Retrieve resource	
PUT	Update / Modify	Update resource	
POST	Create / New	Create resource	
DELETE	Delete	Delete resource	

2.4. Format of response data

All API response examples in this interface specification are written in JSON format.

```
// JSON Response {
resource : {
```

```
"string1" : "value1",
"string2" : "value2"
}
}
```

JSON does not allow comments. Therefore, in this document, '//' is used temporarily for better understanding. The order of each tag may differ from the example in the specification.

2.5. Error handling

An appropriate HTTP Status Code is returned to the calling client for errors that occur during REST request processing. Common error codes can be added depending on the purpose of the API.

HTTP Status Code / status Message	Description	
400 Bad Request	Service authentication error	common
401 Unauthorized	User not authenticated. Authentication is required, but no authentication information is required (if login is required)	common
403 Forbidden	When calling from an API that is not allowed	common
400 Bad Request	Service authentication error	common
405 Method Not Allowed	When using the wrong HTTP Method	common
406 Not Acceptable	Business logic error (error code defined per request) If the number of registered devices is exceeded, it is also processed as 406.	common
411 Length Required	If the parameters do not match	common
412 Precondition Failed	When there is no Mandatory value in the HTTP Request Header (when the header value is incorrect)	common
500 Internal Server Error	Internal server error	common

2.6. Others

Describes other common rules for interfacing with the GSFS system.

2.6.1. Time

All times of the GSFS system (registration date, modification date, etc.) are saved by calculating the time difference between the time of the country where the server is located and the time of the country where the ASC belongs.

2.6.2. String Trim

In principle, all input parameter values (String) are processed without change as they are entered by the client. Therefore, since string containing space on the left and right is stored as it is, the client must trim the space on the left and right to prevent this.

2.6.3. Parameter Order

The order of JSON objects may be different from the order of parameters written in the interface specification document, and the order is meaningless.

3. Basic Information

3.1. HTTP Request Header

If Request Header is a required value, Request information must be included when other system calls GSFS API (lge.com api).

- **Request Header**

Name	Required	Description
SYSTEM_CODE	M	기존 API의 Content-Disposition (serviceID)
PROGRAM_ID	M	기존 API의 Content-Description (serviceID2)
X-EPOCH_TIME	M	The Unix epoch time
X-HMAC_EPOCH_TIME	M	The sha256 algorithm is used to calculate HMac. Refer to sample guide code below to make X-HMAC_EPOCH_TIME using API-Key

- **Sample Value**

SYSTEM_CODE: GP1

PROGRAM_ID: TrackRepair

X-EPOCH_TIME: 1737613455

X-HMAC_EPOCH_TIME: 37C+HFHlyGeJsjKEmGKreoAm8+Tu5yOaIJfDmfVxgSo=

How Users Can Use HMAC-SHA256 in Requests

- **Secret Key:** The secret key should never be exposed in public repositories or in the request itself.
- **Generate the Signature:**

- The client generates the HMAC-SHA256 signature by hashing the request data (e.g., headers, payload, timestamp) with the secret key.

- **Send the Request:**
  - The client sends the API request with the X-HMAC_EPOCH_TIME header containing the generated HMAC-SHA256 signature.
- **Server Verification:**
  - The server verifies the received signature by generating its own HMAC-SHA256 signature using the same secret key and comparing it with the received one.
  - If the signatures match, the request is authenticated and processed.

***For inquiries regarding API keys, please contact GSFS Admin.**

3.2. Introduce HMAC-SHA256

This part explains how to use **HMAC-SHA256** (Hash-based Message Authentication Code using SHA-256) to secure API requests and authenticate users. HMAC-SHA256 is a cryptographic technique that ensures the integrity and authenticity of the message by signing it with a secret key.

- ### HMAC-SHA256
  - **HMAC-SHA256** is a hashing algorithm that combines a secret key with the message being sent. It generates a unique signature (digest) that verifies the integrity of the message and ensures that it has not been tampered with.
  - **SHA-256** is a cryptographic hash function that produces a 256-bit (32-byte) hash value.
  - **HMAC** ensures that the sender of the message is authenticated, and the message itself is not altered during transmission.

This technique is often used in API authentication to secure communication between a client and a server.

- ## How HMAC-SHA256 Works
  - **Client Side:**
    - The client generates an HMAC-SHA256 signature using the request data (e.g., headers, payload) and a secret key.
    - The signature is sent as part of the HTTP request (typically in headers).
  - **Server Side:**
    - The server uses the same secret key to generate the HMAC-SHA256 signature of the incoming request data.
    - If the signature generated by the server matches the one sent by the client, the request is considered authentic. If they don't match, the request is rejected.
- ## How to make X-EPOCH_TIME and X-HMAC_EPOCH_TIME in JAVA.

```java
/**
     * epochTime 값을 가져온다.
     * 초단위 epochTime 을 가져오는 모든 방법에 대해서 허용한다.
     * ex ) Instant.now().getEpochSecond()
     */
    private long getEpochTime()  {
        return System.currentTimeMillis() / 1000;
    }

    /**
     * sha256 알고리즘을이용하여 HMac 을계산한다.
     * 결과적으로 epochTime 의 sha256 의 암호화값의 base64 인코딩된 데이터면 중간과정이 달라도 무관하다.
     * @param epochTime: API 의 Header 파라미터중 X-EPOCH_TIME (UnixTimeStampex: 1405916593)
     * @param apiKey: API 의 Header 파라미터중 X-Application-Key 또는 Internal App Key
     * @return
     * @throwsSignatureException
     */
    private String getHMacSHA256(long epochTime, String apiKey) throwsSignatureException {
        String hmac;
        try {
            byte[] keyBytes = apiKey.getBytes("UTF-8");
            SecretKeySpec secretKeySpec = new SecretKeySpec(keyBytes, "HmacSHA256");
            Mac mac = Mac.getInstance("HmacSHA256");
            mac.init(secretKeySpec);

            byte[] hmacBytes = mac.doFinal(String.valueOf(epochTime).getBytes("UTF-8"));
```

```
            hmac = new String(Base64.encodeBase64(hmacBytes), "UTF-8");
        } catch (Exception e) {
                // LogHelper.loggingException(logger, e);
                LOGGER.error(e.getMessage(), e);
                thrownewSignatureException("Failed to generate HMAC : ");
        }

        return hmac;
    }
```

-

3.3. Steps to Create HMAC-SHA256 Signed Request

- **Generate Secret Key**

Before making API calls, you need to generate and securely store a **secret key**.

The secret key will be shared between the client and the server.

- **Key Length:** 256 bits (32 bytes) is recommended for security.
- **Key Generation:** The key must be kept secure and never exposed to unauthorized parties.

- Generate HMAC-SHA256 Signature in Java (Client-Side)

- Send HMAC-SHA256 Signature in API Request

- Server Side: Verify the HMAC-SHA256 Signature

3.4. IP based Access Control

GSFS Server has an AccessControl system based on IP address. If you want to access the GSFS server, ask the GSFS Admin to register an IP address.

Allowed Request Format : 10.10.10.10, 10.10.10.0/24, 10.10.10.10/32

3.5. Response Encryption (AES 128)

The GSFS system can respond with an encrypted value. The data can be decrypted using the AES128 algorithm.

- ### AES 128 Decryption Code Guide in JAVA

```java
    /**
     * aes128 알고리즘을이용하여 복호화한다.
     * @param epochTime: encryptedText 암호화된 상태의 데이터
     * @param key: 암호화/복호화 키
     * @return
     * @throwsSignatureException
     */
    private String decrypt(String encryptedText, String key) throws Exception {
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        SecretKeySpec keySpec = new SecretKeySpec(key.getBytes("UTF-8"), "AES");
        cipher.init(Cipher.DECRYPT_MODE, keySpec);
        byte[] decodedBytes = Base64.decodeBase64(encryptedText);
        byte[] decrypted = cipher.doFinal(decodedBytes);
        return new String(decrypted, "UTF-8");
    }
```

4. API Lists

4.1. TrackRepair

- Desc: Call reservation inquiry / repair inquiry
- Content-Type: application/json
- HTTP Method: POST
- URL: /standardWebCall.do
- Input Parameters

Field	Mandatory	Description	Type	Format	Length
P_CORPORATION_CODE	Y	Corporation Code	String		
P_COUNTRY_CODE		Country Code	String		
P_LANGUAGE_CODE	Y	Language Code	String		
P_PHONE_NO		Phone No	String		
P_SERIAL_NO	Y or N	Serial No (P_SERIAL_NO is required if P_SERVICE_REQUEST_NO does not exist)	String		
P_SERVICE_REQUEST_NO	Y or N	ServiceReceiptNo or ConsultationNo (P_SERVICE_REQUEST_NO is required if P_SERIAL_NO does not exist)	String		

1. Sample input data

```
{
    "P_CORPORATION_CODE":"LGEIL",
    "P_LANGUAGE_CODE":"en",
    "P_SERVICE_REQUEST_NO":"RNP250108067301",
    "P_COUNTRY_CODE":"IN",
    "P_PHONE_NO":"9871976305",
    "P_SERIAL_NO":""
}
```

2. Return value

Field	Description	Type	Format	Length
resultCode	Row Count	String		
resultMsg	Type	String		
data	object	object		
dataCount	Length of list	number		
datalist	Data list	list		
ASC_ADDRESS	Address	String		
ASC_CODE	Code	String		
ASC_FAX_NO	Fax No	String		
ASC_NAME	Name	String		
ASC_PHONE_NO	Phone No	String		
AWB_NO2	Awb No2	String		
BR_MC_FLAG	Br Mc Flag	String		
CALLBACK_DATE	Callback Date	String		
OMPLETED_DATE	Completed Date	String		
CONSULTATION_NO	Consultation No	String		
CUST_ADDRESS	Customer Address	String		
CUST_CELLULAR_NO	Customer Cellular No	String		
CUST_EMAIL_ADDR	Customer Email Addr	String		
CUST_FAX_NO	Customer Fax No	String		
CUST_PHONE_NO	Customer Phone No	String		
CUST_POSTAL_CODE	Customer Postal Code	String		
CUSTOMER_NAME	Customer Name	String		

DATA_FROM_CODE	Data From Code	String		
DR_ENTRY_NO	Dr Entry No	String		
DR_FLAG	Dr Flag	String		
DR_REQUEST_NO	Dr Request No	String		
E_TICKET_NO	E Ticket No	String		
ERP_ORDER_NO	Erp Order No	String		
ESN_IMEI_NO	EsnImei No	String		
FIRST_PROMISE_DATE	First Promise Date	String		
O_ENGINEER_CODE	O Engineer Code	String		
O_ENGINEER_NAME	O Engineer Name	String		
O_LAST_UPDATE_DATE	O Last Update Date	String		
O_MC_PRC_FLAG	O Mc Prc Flag	String		
OB_UNIT_TRACKING_NO	Ob Unit Tracking No	String		
PRIMARY_REPAIR_CODE	Primary Repair Code	String		
PRODUCT_CODE	Product Code	String		
PRODUCT_GROUP_CODE	Product Group Code	String		
PRODUCT_NAME	Product Name	String		
PROMISE_DATE	Promise Date	String		
PROMISE_TIME_FROM	Promise Time From	String		
PROMISE_TIME_TO	Promise Time To	String		
PROMISE_TIME_ZONE_NAME	Promise Time Zone Name	String		
PURCHASE_DATE	Purchase Date	String		
REASON	Reason	String		
REASON_CODE	Reason Code	String		
RECEIPT_DATE	Receipt Date	String		
REMARK	Remark	String		
REQUEST_DATE	Request Date	String		

SALES_MODEL_CODE	Sales Model Code	String		
SERIAL_NO	Serial No	String		
SERVICE_REPAIR_NO	Service Repair No	String		
SERVICE_TYPE_CODE	Service Type Code	String		
STATUS_MESSAGE	Status Message	String		
SUB_SYMP_NAME	Sub Symp Name	String		
SUPPLIER_CODE	Supplier Code	String		
SVC_CENTER_TYPE_CODE	Svc Center Type Code	String		
SVC_TYPE_CODE	Svc Type Code	String		
SYMP_NAME	Symp Name	String		
TECH_MOBILE_NO	Tech Mobile No	String		
TECH_PHONE_NO	Tech Phone No	String		
TRANSFER_SEQ_NO	Transfer Seq No	String		
WARRANTY_FLAG	Warranty Flag	String		
WARRANTY_STATUS	Warranty Status	String		
PIN_CODE	Pin Code	String		

3. Response data (success)


```
{
    "resultMsg": "success",
    "data": {
        "dataList": [
            {
                "CUSTOMER_NAME": "ANJU HANDA",
                "DR_FLAG": "N",
                "ERP_ORDER_NO": "",
                "PROMISE_DATE": "09/01/2025",
                "REMARK": "",
                "CALLBACK_DATE": "",
                "PROMISE_TIME_ZONE_NAME": "All Day",
                "STATUS_MESSAGE": "COMP",
                "PIN_CODE": "110024",
                "CUST_FAX_NO": "11",
                "PURCHASE_DATE": "17/05/2023",
                "PRIMARY_REPAIR_CODE": "C04",
                "SALES_MODEL_CODE": "OLED65B7T.ATR",
```

```
                "O_ENGINEER_NAME": "AJAY KUMAR",
                "DR_ENTRY_NO": "",
                "SUB_SYMP_NAME": "Refrigerator: Not Cooling",
                "ASC_NAME": "LGEIL Delhi Direct Service",
                "RECEIPT_DATE": "08/01/2025",
                "ASC_PHONE_NO": "",
                "BR_MC_FLAG": "N",
                "ASC_CODE": "177959",
                "SERIAL_NO": "705LPBL003661",
                "CUST_POSTAL_CODE": "DL0448",
                "PRODUCT_GROUP_CODE": "TV",
                "CUST_PHONE_NO": "9871976305",
                "TRANSFER_SEQ_NO": "001",
                "TECH_PHONE_NO": "",
                "SYMP_NAME": "Cooling Performance",
                "WARRANTY_STATUS": "",
                "PRODUCT_NAME": "LCD/LED/OLED TV ( 33\" and over )",
                "AWB_NO2": "",
                "PRODUCT_CODE": "LCL",
                "SUPPLIER_CODE": "177959",
                "ASC_ADDRESS": "LGEIL, A-27,MOHAN CO-OPERATIVE,INDUSTRIAL ESTATE,NEW DELHI    ",
                "OB_UNIT_TRACKING_NO": "",
                "ASC_FAX_NO": "",
                "CUST_ADDRESS": "25/159,, VIKRAM VIHAR,,, LAGPAT NAGAR 2ND FLOOR,, LAJPAT NAGAR IV, DELHI, DELHI",
                "E_TICKET_NO": "",
                "CUST_CELLULAR_NO": "9717587233",
                "PROMISE_TIME_TO": "2400",
                "COMPLETED_DATE": "13/01/2025 ",
                "WARRANTY_FLAG": "I",
                "SERVICE_TYPE_CODE": "IH",
                "FIRST_PROMISE_DATE": "09/01/2025",
                "O_LAST_UPDATE_DATE": " 12:15",
                "O_ENGINEER_CODE": "YIL002311",
                "DATA_FROM_CODE": "ONSC",
                "REASON": "",
                "CUST_EMAIL_ADDR": "",
                "O_MC_PRC_FLAG": "",
                "SVC_CENTER_TYPE_CODE": "01",
                "REASON_CODE": "",
                "CONSULTATION_NO": "",
                "SERVICE_REPAIR_NO": "RNP250108067301",
                "DR_REQUEST_NO": "",
                "SVC_TYPE_CODE": "In-Home Service",
                "PROMISE_TIME_FROM": "0000",
                "REQUEST_DATE": "08/01/2025",
                "TECH_MOBILE_NO": "7290099747",
                "ESN_IMEI_NO": ""
            }
        ],
        "dataCount": 1
    },
    "resultCode": "0000"
}
```

4. Response data (error)

resultCode	Description
0000	success
0001	Json parsing error (please check requestBody)
0002	Missing mandantory requestHeaderParameters
0003	Invalid IP address
0004	Invalid Program id
0005	Invalid EpochTime (if more than 5 minutes have passed)
0006	Invalid HmacEpochTime data (please check api_key or encrypt logic)
0007	Internal error caused by functionName – programId not Matched.
0008	Encrypt Error
0009	Error caused by Service Logic
9999	Undefined Error