Qualcomm Technologies, Inc.

Application to MBMS API Extension

Interface Specification for Release 5.3

80-16892-1 Rev. A

August 6, 2020

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

Revision history

Revision	Date	Description
A	August 2020	Initial release

Contents

Figures

1 Introduction

This document defines the specification of the I-1 interface for mission critical data (MCData) APIs that exist between the multicast service device client (MSDC) and the application (app) on the user equipment (UE). It is assumed that the reader of this document is familiar with Android app development and related concepts.

The following concept is outside the scope of this document:

- eMBMS

1.1 Environment

The following software environment is required on the device to run the app with MSDC SDK:

- Android Q operating system (10.0 or later)
- MSDC Release 5.3

1.2 Acronyms and terms

Term	Definition
eMBMS	Enhanced multimedia broadcast multicast services
LTE	Long term evolution
MBMS	Multimedia broadcast multicast services
MCData	Mission critical data
MSDC	Multicast service device client
UE	User equipment

1.3 Technical assistance

For support information, please visit the following LTE Broadcast SDK webpage on the Qualcomm® Developer Network (QDN) site:

https://developer.qualcomm.com/ltebroadcast

2 Overview

MSDC uses the Android MBMS APIs to communicate with the application. Apart from the Android MBMS APIs, the MSDC SDK can be used by the App to access MSDC for additional feature support. Receiving MCData over eMBMS feature is not supported by the Android MBMS API. Receiving MCData using MSDC SDK is available from Andoid Q onwards. Figure 2-1 illustrates the overall architecture on the Android device.

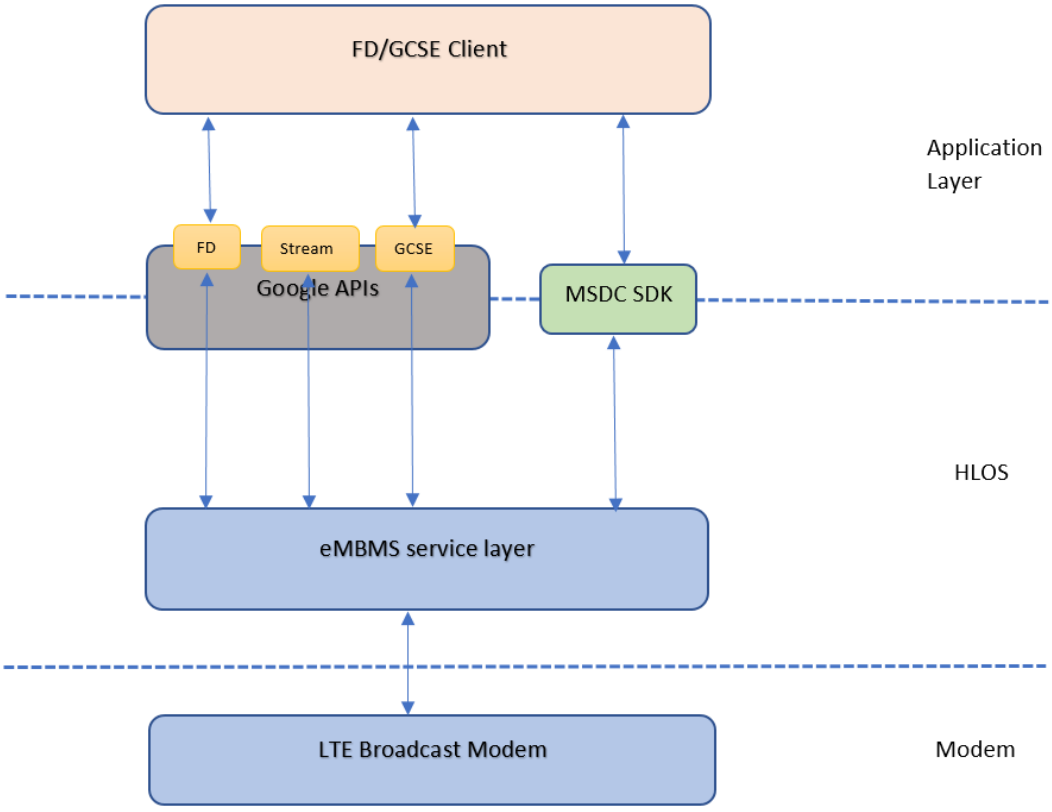

Figure 2-1 Interfaces of the app and MSDC

MSDC SDK provides API for UI application to send service announcement files for business or mission critical services. These services would be reflected in File Download services and users can start file download from file download UI.

3 MSDC SDK API

The MSDC SDK API provides interface for UI application to send service announcement file for mission critical services to the eMBMS service layer. The MSDC SDK API lets application developers concentrate on creating great apps instead of worrying about the underlying Android services and management of communication with those services.

The following sections describe the main interfaces and components required by the app.

NOTE

1. In case of errors, and depending on the error case, the MSDC informs the app via asynchronous error notifications.

3.1 LTEMSDCService

This class provides APIs for creating session and further communication with eMBMS service layer.

- public static LTEMSDCService create(Context context, Executor executor, ILTEMSDCServiceCallback callback)

  This function creates and returns a new `LTEMSDCService` instance. `LTEMSDCService` instance can be used later for any subsequent request. Application context must be passed with this function. Context argument should not be NULL. `ILTEMSDCServiceCallback` callback and `Executor` argument should not be NULL. If it is NULL, the application cannot get any callback. All future callbacks will use executor for callback task execution.

NOTE: If the app calls this method while an active instance of `LTEMSDCService` is in process (in other words, one that has not had `close()` called on it), this method throws an IllegalStateException. Note that initialization may fail asynchronously. If the app intends to try again after it receives such an asynchronous error, it must call close() on the instance of `LTEMSDCService` that it received before calling this method again.

- public int addSA(String filePath)

  This function sends service annoucement file to the eMBMS service layer. It can pass absolute path of SA file. This function can be called with `LTEMSDCService` instance. It can return the following values.

```
public static final int SUCCESS = 0;
public static final int ERROR_NO_UNIQUE_MIDDLEWARE = 1;
public static final int ERROR_MIDDLEWARE_NOT_BOUND = 2;
```

```
public static final int ERROR_MIDDLEWARE_LOST = 3;
public static final int ERROR_FILE_READ_FAILED = 4;
```

- public void close()

  It deletes `LTEMSDCService` instance, which was created using `create` function.

3.2 ILTEMSDCServiceCallback

The UI application must implement this callback interface and pass it during `LTEMSDCService create` function call.

- void onMiddlewareReady()

  This callback is invoked by middleware after middleware is ready and UI application can send request to middleware. Only after receiving this callback, the application should call any request API, otherwise it returns error.

- void addSAResponse(int result, String msg)

  This callback function is invoked as a response of addSA request. The parameter result is SUCCESS[=0], otherwise error message appears. One possible error scenario is when service announcement file is not correct or corrupted and parsing of SA file is failed.

- void onError(int errorCode, String message)

  This function is called by the MSDC SDK to check for any errors. The possible error it can return is:

  ```
  ERROR_LTE_MSDC_SVC_INIT_FAILED = 1;
  ```

4 Examples on how to use the MSDC SDK API

4.1 Prerequisite

The UI app should implement the Google `[MbmsGroupCallSession]create` API to establish groupcall session. Also, the UI should implement File Download using Google APIs because middleware sends MCData services with File Download services after UI sends service annoucement to middleware and it is parsed succesfully. Mission critical services are available with File Download services.

4.2 Create LTEMSDCService session

The app can get the `LTEMSDCService` instance using its static `create` method, as shown in the following example. It should be called after groupcall session is started. Try to not use main thread, otherwise it can block main thread since it is a synchronous call and may take time for initialization.

```
ILTEMSDCServiceCallback mILTEMSDCServiceCallback = new ILTEMSDCServiceCallback() {

        @Override
        public void addSAResponse(int result, String msg) {

        }

        @Override
        public void onMiddlewareReady() {

        }

        @Override
        public void onError(int errorCode, String message) {

        }
    };
LTEMSDCService mLTEMSDCService = LTEMSDCService.create(getApplicationContext(), new
ScheduledThreadPoolExecutor(2), mILTEMSDCServiceCallback);
```

4.3 Add SA file

The app can send service annoucement file to middleware as shown in this example:

```
public void addServiceAnnouncement() {
        final String filePath = getApplicationContext().getExternalFilesDir(null) +
"/msdcui/sa.txt";;
        mLTEMSDCService.addSA(filePath);
}
```

4.4 Stop connection

The app can close the connection with middleware as shown in this example:

```
mLTEMSDCService.close();
```

Figure 4-1 shows how the app can register and add MCData services SA file to MSDC SDK.

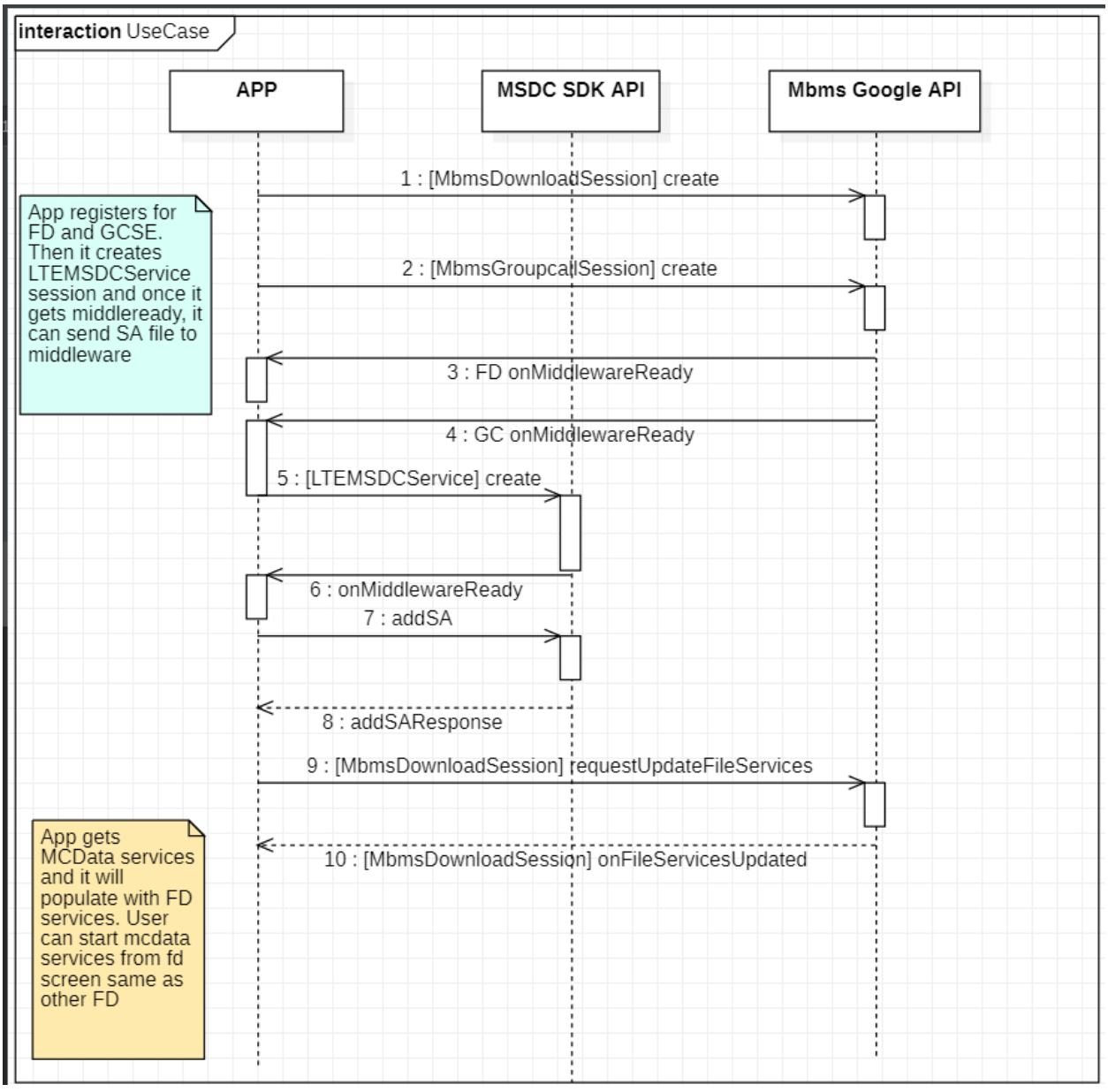

Figure 4-1 App and MSDC SDK interaction diagram

5 Use cases

5.1 Mission critical data use case

This feature can be used by organisations in the mining, oil and gas industry. This is because they have a need for MCData communications to enable remote monitoring and control of their operations.