Qualcomm Technologies, Inc.

Application to MSDC

Interface Specification for MSDC Release 4.4

80-72115-1 Rev. A

October 26, 2023

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

Revision History

Revision	Date	Description
A	Oct 2023	New version

Contents

List of Figures

1 Introduction

1.1 Purpose

This document defines the specification of the I-1 interface that exists between the Multicast Service Device Client (MSDC) and the application (app) on the user equipment (UE). It is assumed that users of this document are familiar with Android app development (including related concepts) and the media player interaction of the app.

The following concepts are outside the scope of this document:

- Enhanced Multimedia Broadcast Multicast Services (eMBMS)
- Dynamic Adaptive Streaming over HTTP (DASH)

1.2 Environment

The following software environments are required for a device to run an application with MSDC middleware:

- Android Lollipop operating system (5.0 or later)
- DASH Player (recommend Qualcomm DASH Player which is included in the LTE Broadcast SDK package)
- MSDC Release 4.3

1.3 Conventions

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

Code variables appear in angle brackets, for example, `<number>`.

Commands and command variables appear in a different font, for example, **`copy a:*.* b:`**.

1.4 Support

For support, visit the LTE Broadcast SDK web page on the Qualcomm® Developer Network (QDN): https://developer.qualcomm.com/ltebroadcast.

2 Functional Overview

The MSDC uses its I-1 interface to communicate with the app. The app also has an interface with the media player, which plays data from the streaming service. The following figure shows the overall architecture of these interfaces on the Android device.

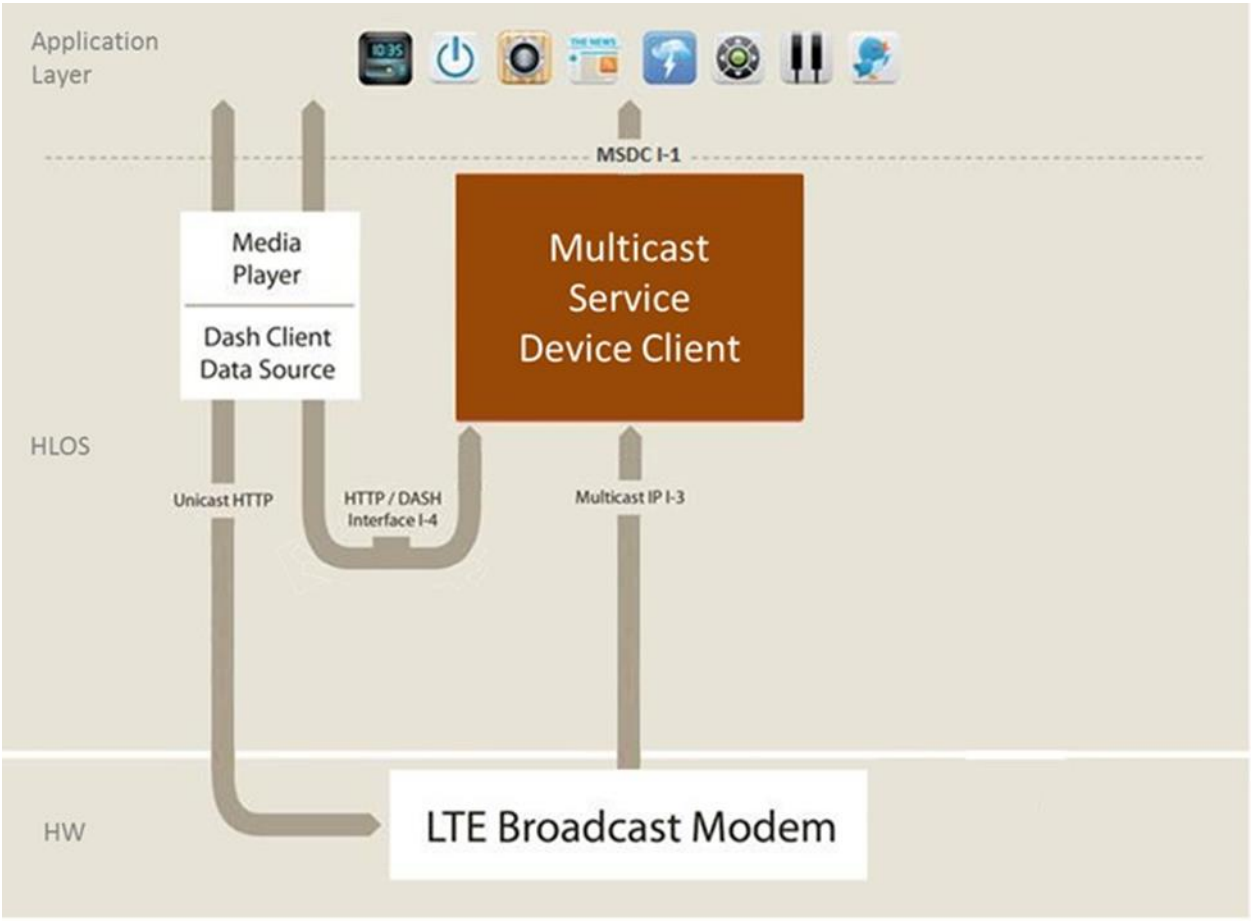


Figure 2-1 MSDC and app interfaces

Broadcast services (called user services in eMBMS broadcast networks) identify the user-visible/accessible definition of content delivered over a broadcast network. Content can typically be delivered in the form of individual files, e.g., clips and software images, or as streaming media (consumed/played shortly after reception).

2.1 File delivery services

Some examples of file delivery services include:

- Streaming service applications (see Section 8.1)
- Top 10 You Tube videos (see Section 8.2)
- Firmware Over The Air (FOTA) (see Section 8.3)
- Weekly magazine (see Section 8.4)
- Live sports games and TV channels

The content associated with file delivery services is delivered in broadcast networks via IP packets and logical access network channels. The broadcast network also provides a service announcement function to describe the services available in the network. Apps do not have to be aware of how services are delivered in the network, but use the I-1 interface to do the following:

- Discover available services
- Request that the MSDC activate reception of data for the available services

The MSDC identifies each service by a Service ID. The app must use the correct service ID in any service-specific request. For more information on how the application can get the Service ID, see Sections 4.4.8.2 (Streaming) and 5.4.6.1 (File Delivery).

To use the MSDC API, the app should have the following information from the carrier/operator:

- App ID – The unique ID of the app.
- Service class information – A set of services may be grouped together according to a common classification. This group is called a service class. For example, there could be a service class named “sports” which refers to all sports-related services.

The list of service classes that the app is interested in must be given to the MSDC while using the MSDC API. The MSDC can only give data for service that belongs to its respective set of service classes.

3 MSDC API

The MSDC API provides model and controller implementation for a typical MVC-based UI application. This architecture facilitates quick development of client applications. The MSDC API allows developers to concentrate on creating great apps for live streaming and file delivery services, instead of worrying about the underlying Android Services and management of communication with those services.

This chapter describes the main interfaces and components that an app needs. For more detailed information on classes and functions, see Chapter 9.

Note:

- Any MSDC Android service is referred to as a Module.
- In case of errors, and depending on the error case, the MSDC informs the app via asynchronous error notifications.

3.1 MSDC Manager module

The functions from the MSDC Manager module classes are used by all applications, regardless of whether they provide Streaming or File Delivery services.

The functions in these classes help the app establish and manage overall communication with the MSDC. The app must use these functions to start any communication with the MSDC.

Interface class	Description
IMSDCAppManager	Contains all the functions the app uses to send a request to the MSDC that are not specific to any kind of service. These functions provide the following functionalities: • Initialization/terminatation – Initializes or terminates connection with the MSDC Manager module. • Adding/removing event listener – Adds or removes event listener for the MSDC Manager module. • Getting controller and model instances – Streams module, file delivery module, and network module classes. • Other get functions – getAPIVersion().
IMSDCAppManagerEventListener	• Contains the generic event notifications pertaining to MSDC but not specific to a particular type of service (streaming or file delivery). • App must implement this class and add the event listener (use addMSDCEventListener()) in the IMSDCAppManager before it can receive these event notifications.

3.2 Streaming module

To use Streaming services from the MSDC, the app implements the Streaming module interface classes to communicate with the Streaming module of the MSDC.

The app can use the Streaming module classes in conjunction with functions in the Network module classes (see Section 3.4).

Interface class	Description
IMSDCStreamingController	Contains all the functions the app uses to send a streaming service related request to the MSDC. These functions provide the following functionalities: • Initialization/terminatation – Initializes or terminates connection to the Streaming module. • Adding/removing event listener – Adds or removes event listener for the Streaming module. • Streaming service control
IMSDCStreamingControllerEventListener	• Contains all event notifications related to streaming services. • App must implement this class and add the event listener in the IMSDCStreamingController before it can receive these event notifications.
IMSDCStreamingModel	Contains the get functions that the app can use any time to retrieve information from the MSDC.

3.3 File Delivery module

To use File Delivery services from the MSDC, the app implements the File Delivery module interface classes to communicate with the File Delivery module of the MSDC.

The app can use the File Delivery module classes in conjunction with functions in the Network module classes (see Section 3.4).

Interface class	Description
IMSDCFileDeliveryController	Contains all the functions the app uses to send a file delivery service related request to the MSDC. These functions provide the following functionalities: • Initialization/terminatation – Initializes or terminates connection to the File Delivery module. • Adding/removing event listener – Adds or removes event listener for the File Delivery module. • File Delivery service control
IMSDCFileDeliveryControllerEventListener	• Contains all event notifications related to file delivery services. • App must implement this class and add the event listener in the IMSDCFileDeliveryController before it can receive these event notifications.
IMSDCFileDeliveryModel	Contains the get functions that the app can use any time to retrieve information from the MSDC.

3.4 Network module

The Network module interface classes may be used by the Streaming or File Delivery app to get network-related notifications from the MSDC. The functions under this class are not specific to the type of service that the app intends to render and communicate with the Network module of MSDC.

An application that is only interested in network notifications but does not render Streaming or File Delivery services to the user can also use the Network module classes and the MSDC Manager module interface classes (see Section 3.1).

Interface class	Description
IMSDCNetworkController	Contains all the common functions that the applications implement to send a network notification-related request to the MSDC. Together, these functions provide the following functionalities: • Initialization/termination – Initializes or terminates connection to the Network module. • Adding/removing event listener – Adds or removes event listener for the Network module.
IMSDCNetworkControllerEventListener	• Contains the network-related event notifications common to all applications. • App must add the event listener in the IMSDCNetworkController before the app can receive event notifications such as: – Broadcast coverage – Signal-level
IMSDCNetworkModel	Contains the get functions that are useful for all apps to retrieve Broadcast coverage and Signal-level information at any time from the MSDC.

3.5 Group Call module

To use Group Call services from the MSDC, the app implements the Group Call module interface classes to communicate with the Group Call module of the MSDC.

The app can use the Group Call module classes in conjunction with functions in the Network module classes (see Section 3.4).

Interface class	Description
IMSDCGroupCallController	Contains all the functions the app uses to send a group call service related request to the MSDC. These functions provide the following functionalities: • Initialization/termination – Initializes or terminates connection to the Group Call module. • Adding/removing event listener – Adds or removes event listener for the Group Call module. • Group Call service control
IMSDCGroupCallControllerEventListener	• Contains all the event notifications related to Group Call services. • App must implement this class and add the event listener in the IMSDCGroupCallController before it can receive these event notifications such as:
IMSDCGroupCallModel	Contains get functions that the app can use to get information about active Group Call services from the MSDC.

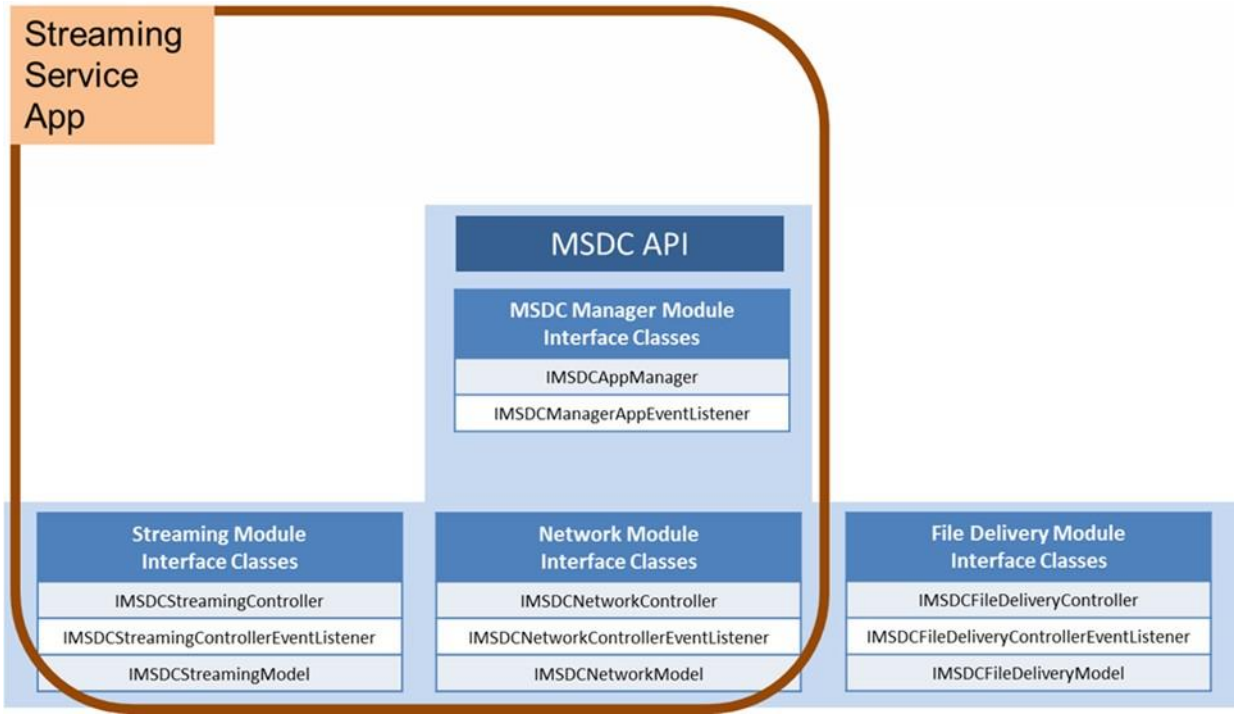
3.6 Use cases

The following sections provide an overview of typical application use cases and the classes that are used, based on application need.

3.6.1 Streaming service app with network notifications

A Streaming service app that wants to receive network notifications uses the following interface classes:

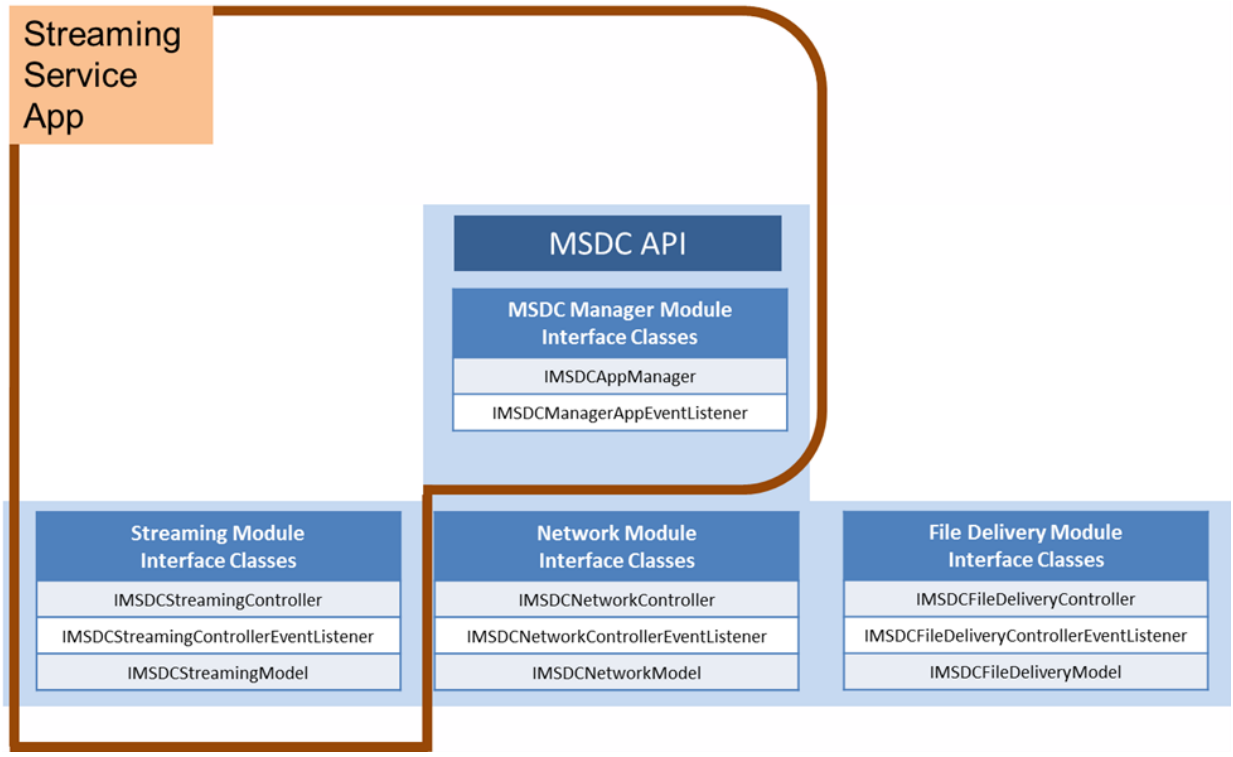
- MSDC Manager module (see Section 3.1)
- Streaming module (see Section 3.2)
- Network module (see Section 3.4)



3.6.2 Streaming service app without network notifications

A Streaming service app that does not want to receive network notifications uses the following interface classes:

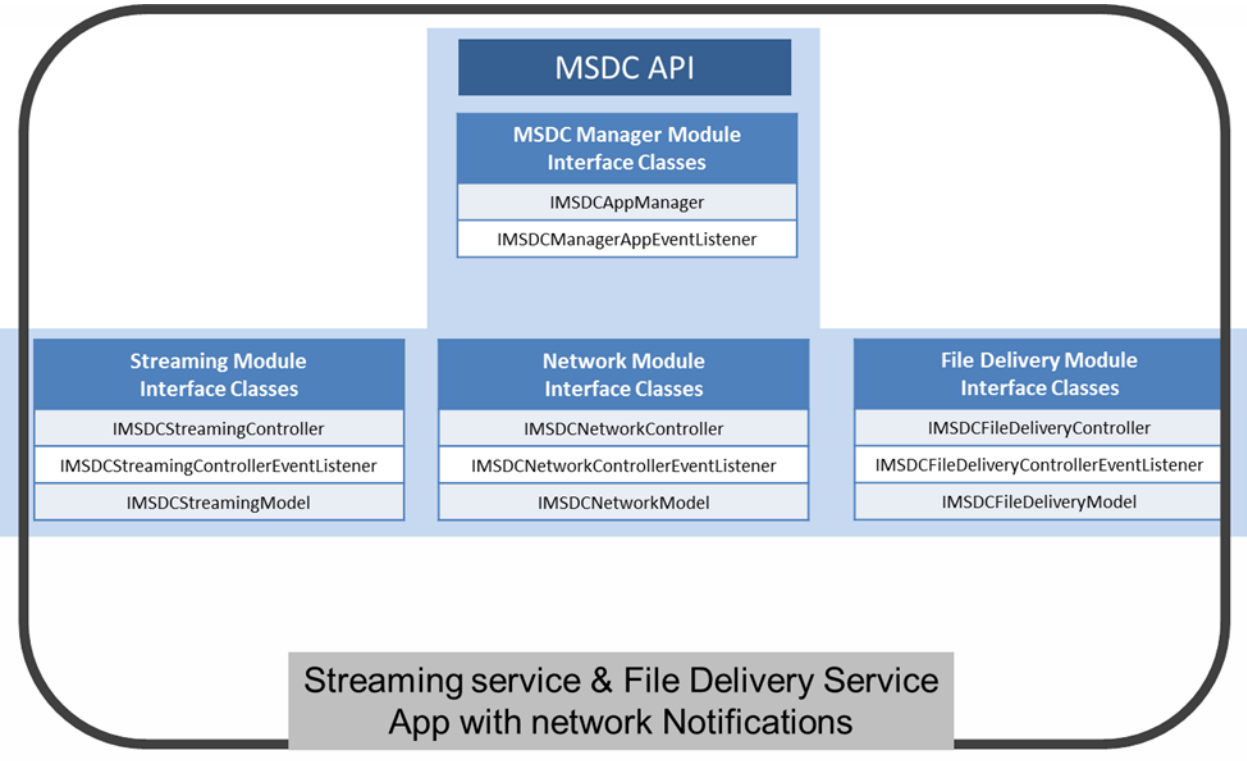
- MSDC Manager module (see Section 3.1)
- Streaming module (see Section 3.2)



3.6.3 Streaming and File Delivery app with network notifications

An app that renders both Streaming and File Delivery services and wants to receive network notifications must use the entire set of MSDC API interface classes:

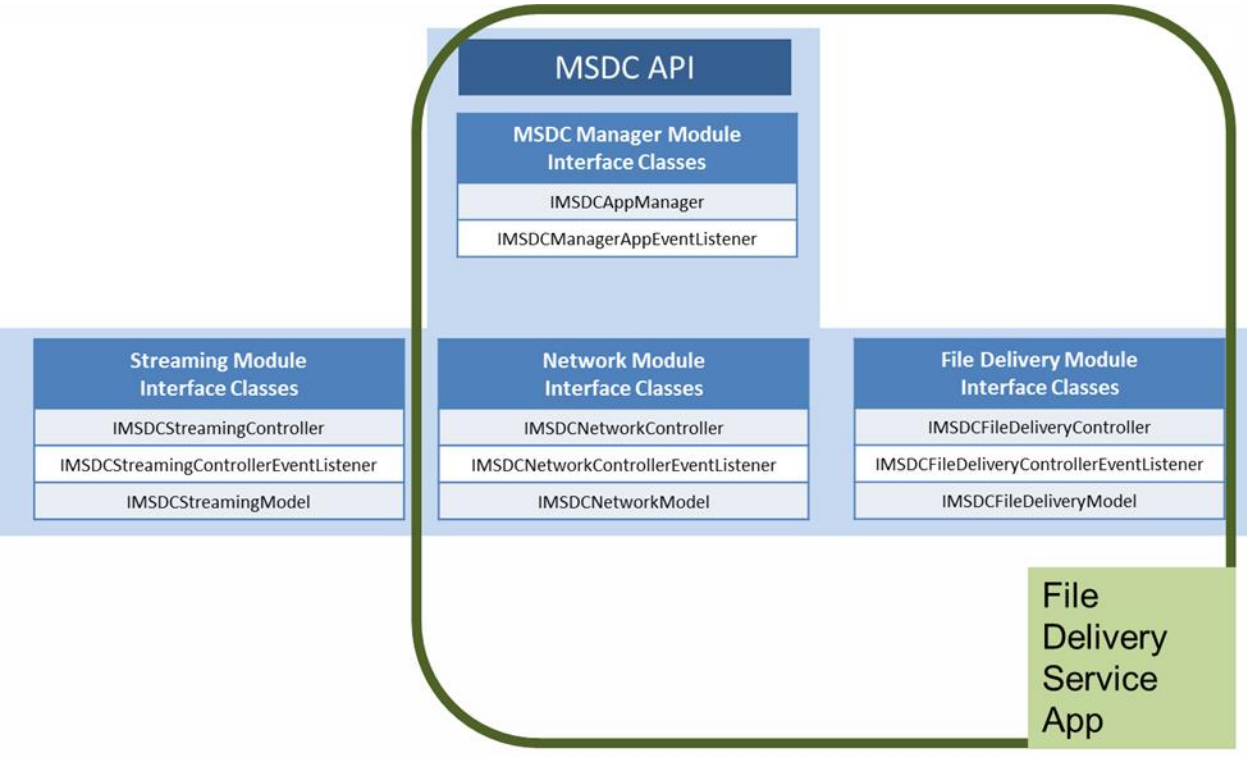
- MSDC Manager module (see Section 3.1)
- Streaming module (see Section 3.2)
- File Delivery module (see Section 3.3)
- Network module (see Section 3.4)



3.6.4 File Delivery app with network notifications

A File Delivery service app that wants to receive network notifications uses the following interface classes:

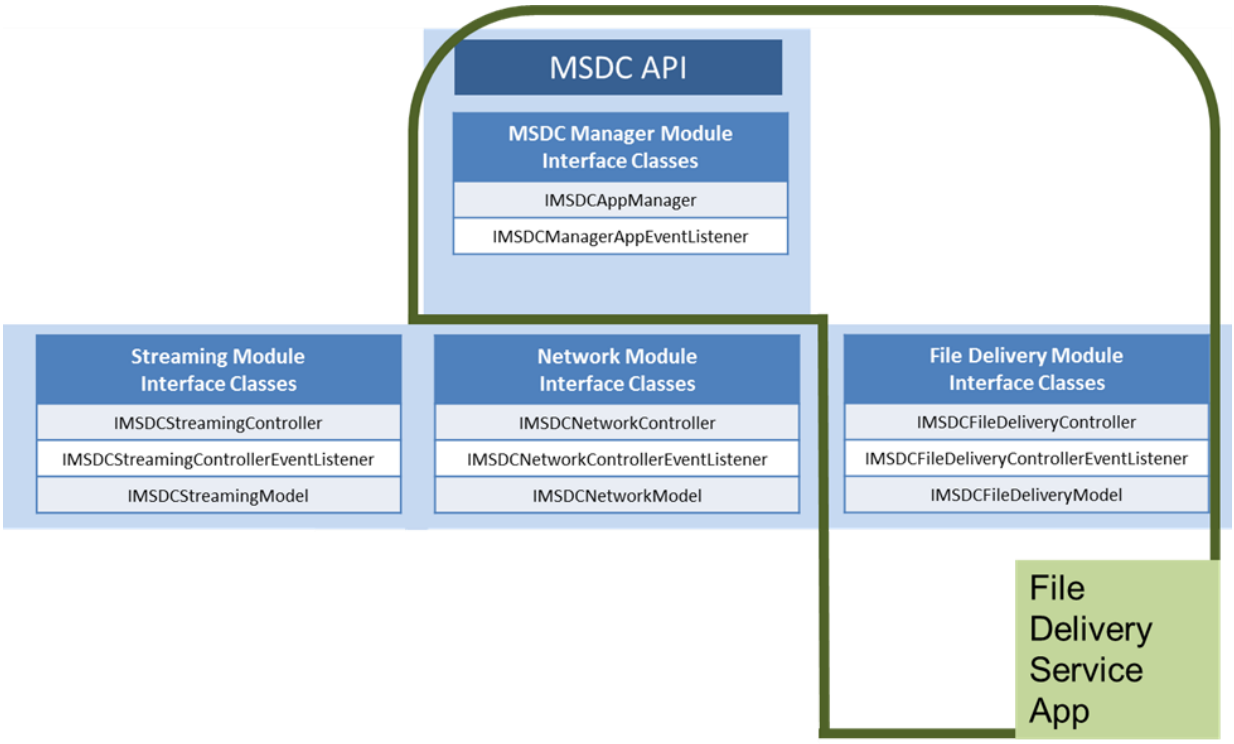
- MSDC Manager module (see Section 3.1)
- File Delivery module (see Section 3.3)
- Network module (see Section 3.4)



3.6.5 File Delivery app without network notifications

A File Delivery service app that does not want to receive network notifications uses the following interface classes:

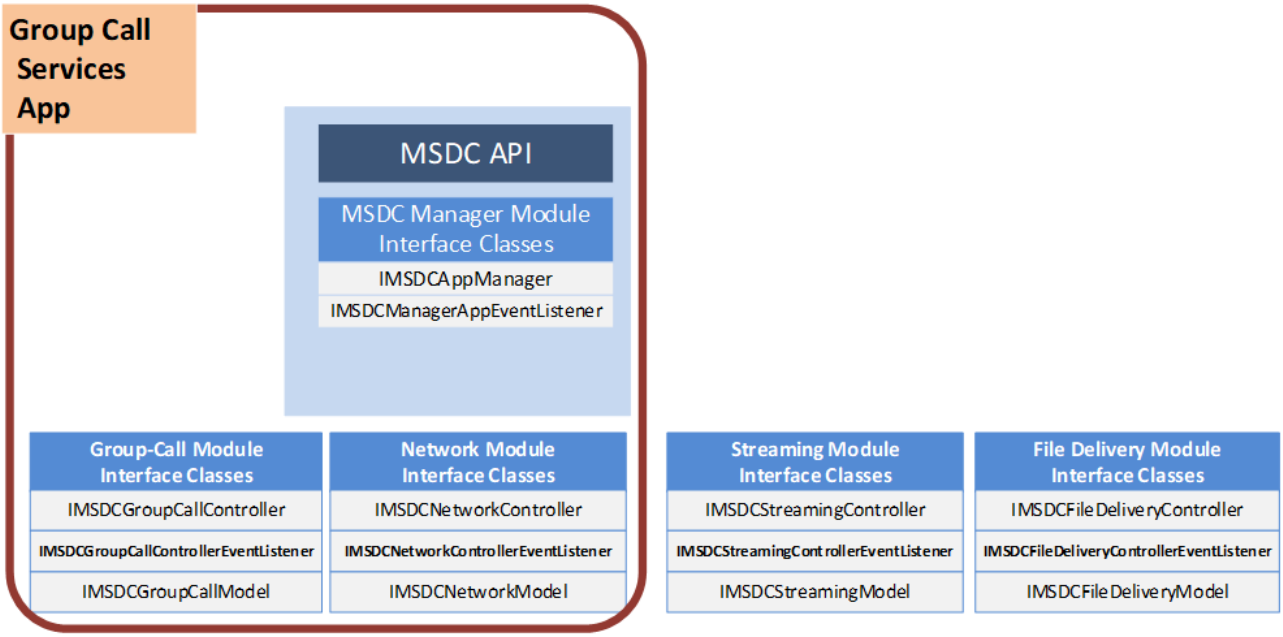
- MSDC Manager module (see Section 3.1)
- File Delivery module (see Section 3.3)



3.6.6 Group Call app with network notifications

A Group Call service app that wants to receive network notifications uses the following classes:

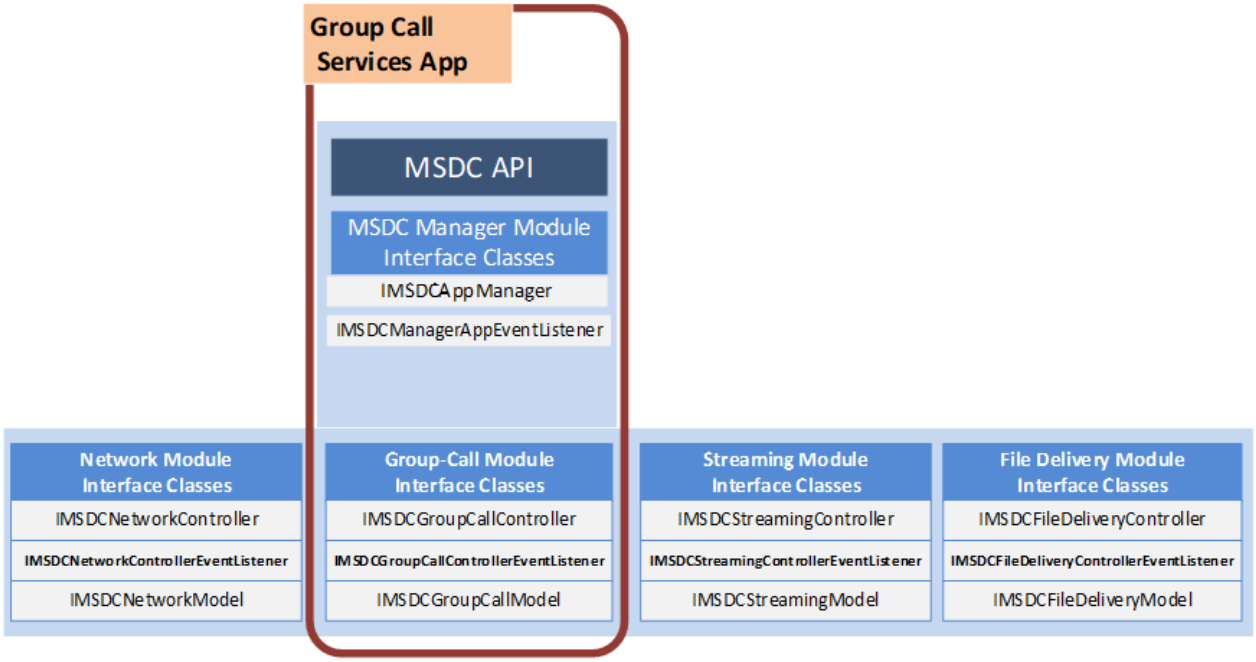
- MSDC Manager module (see Section 3.1)
- Network module (see Section 3.4)
- Group Call module (see Section 3.5)



3.6.7 Group Call app without network notifications

A Group Call service app that does not want to receive network notifications uses the following interface classes:

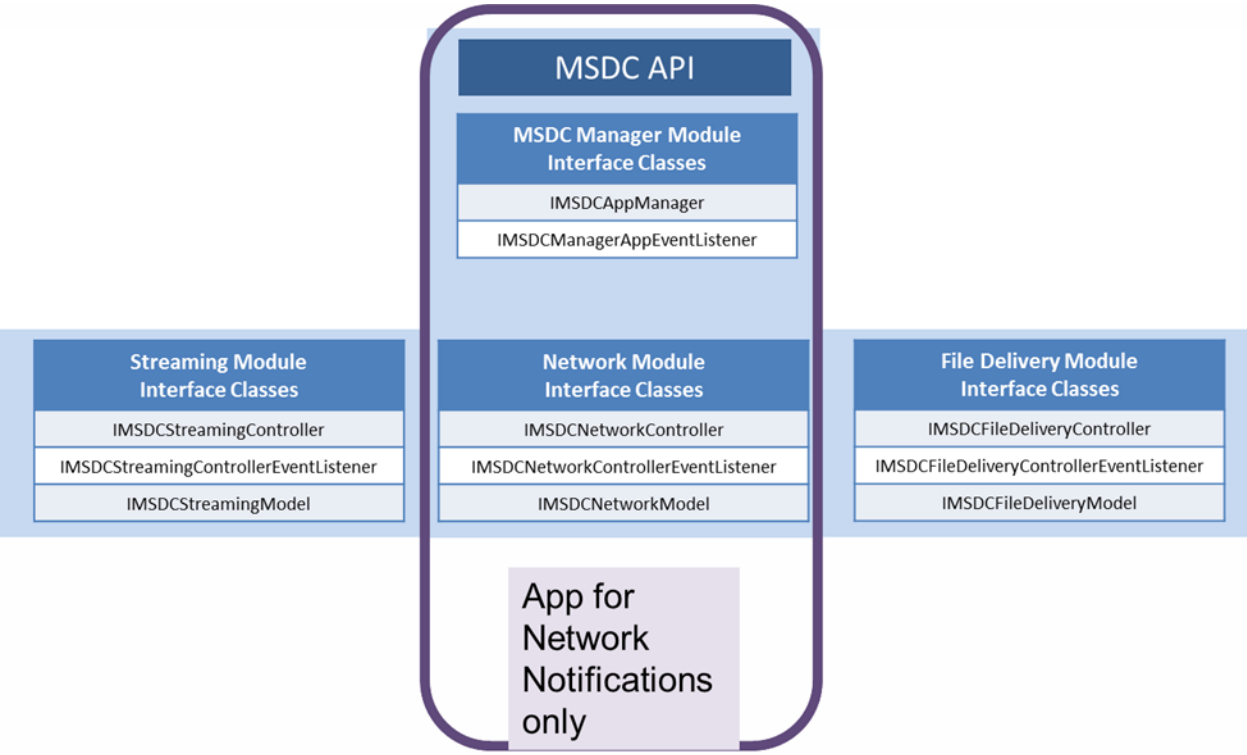
- MSDC Manager module (see Section 3.1)
- Group Call module (see Section 3.5)



3.6.8 App that only receives network notifications

An application that only wants to receive network notifications must use the following classes:

- MSDC Manager module (see Section 3.1)
- Network module (see Section 3.4)



4 Streaming Service

This chapter describes the MSDC API call flow sequences for an application that provides a Streaming service to users.

In these call flows, it is assumed that the Streaming service application does not need network-related notifications (see Section 3.6.2). If the application wants to use network notifications, see Appendix A for code examples and implementation.

The application may also need to render File Delivery services. In this case, see Chapter 5.

All functions in this chapter, i.e., request calls and notifications, are part of one of the following classes:

- IMSDCAppManager() (see Section 3.1)
- IMSDCAppManagerEventListener() (see Section 3.1)
- IMSDCStreamingController() (see Section 3.2)
- IMSDCStreamingControllerEventListener() (see Section 3.2)
- IMSDCStreamingModel() (see Section 3.2)

4.1 Overview

To support Streaming, the app must talk to the MSDC and DASH-enabled media player (see Figure 2-1). The app's communication with the MSDC is essentially a control path, while its communication with the media player is a data path.

Figure 4-1 through Figure 4-3 show an overview of the call flow for a typical Streaming service app. The detailed call flow sequence for individual functions and other scenarios is described in the subsequent sections.

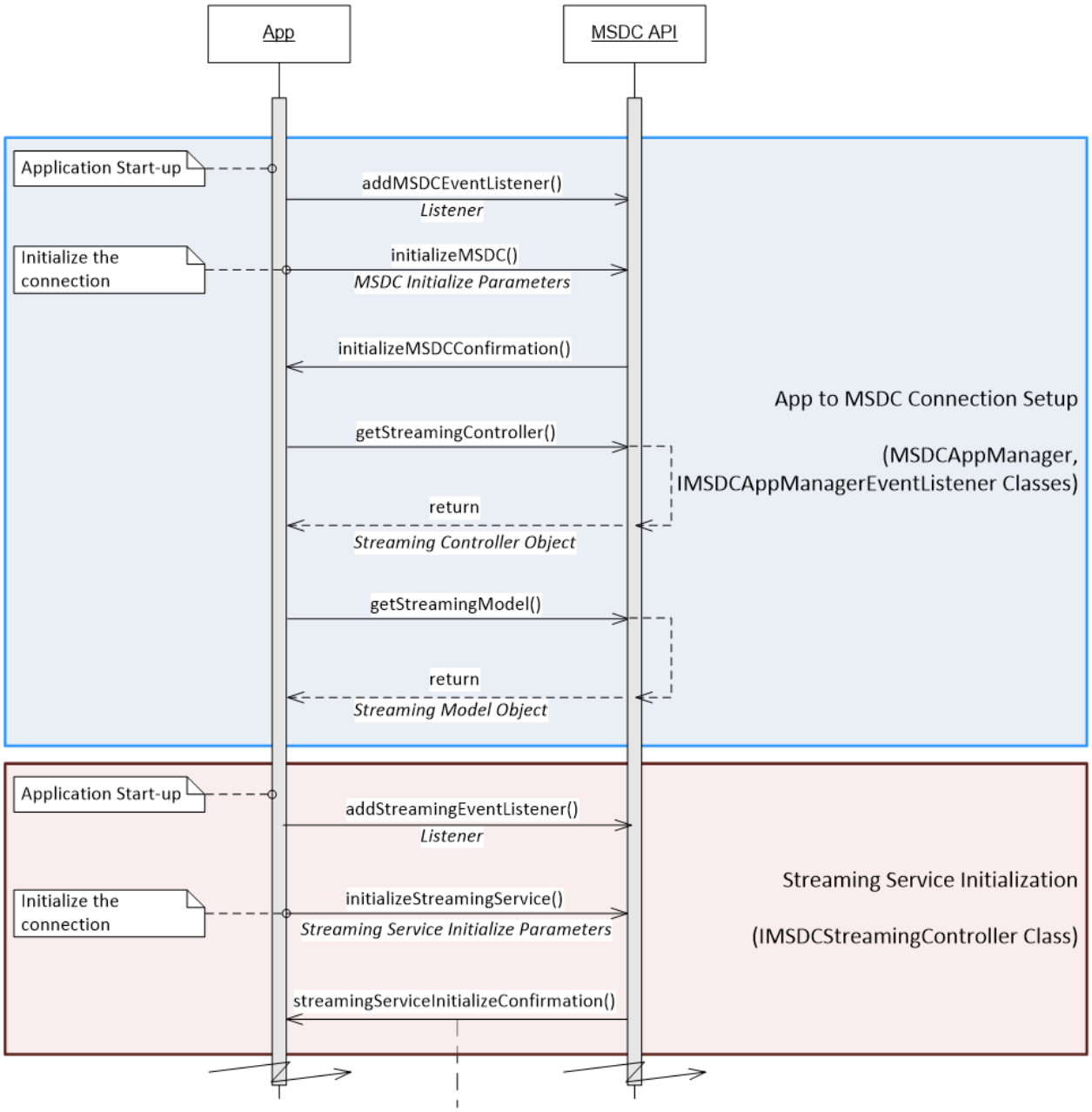


Figure 4-1 Streaming service call flow (1 of 3)

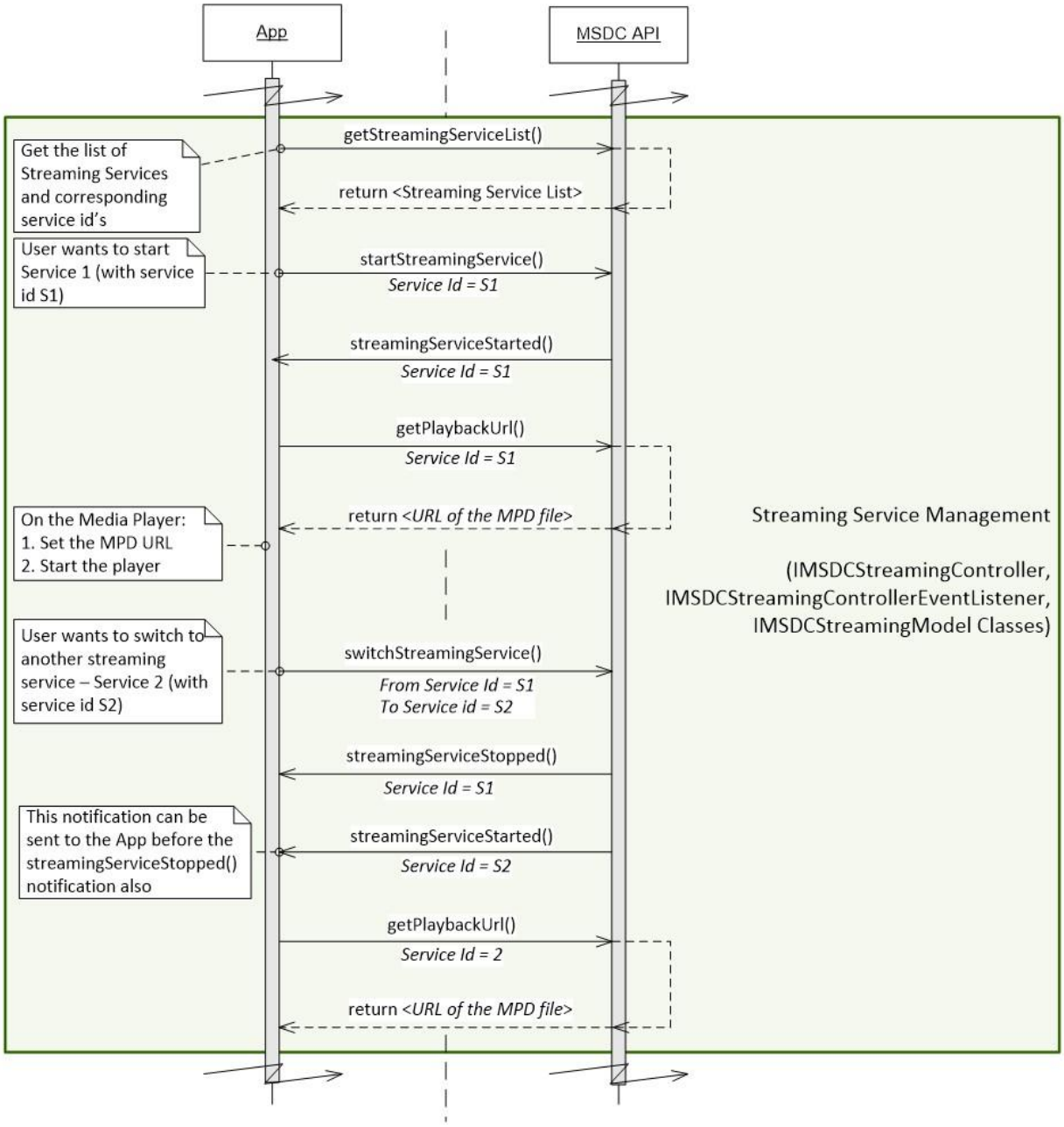


Figure 4-2 Streaming service call flow (2 of 3)

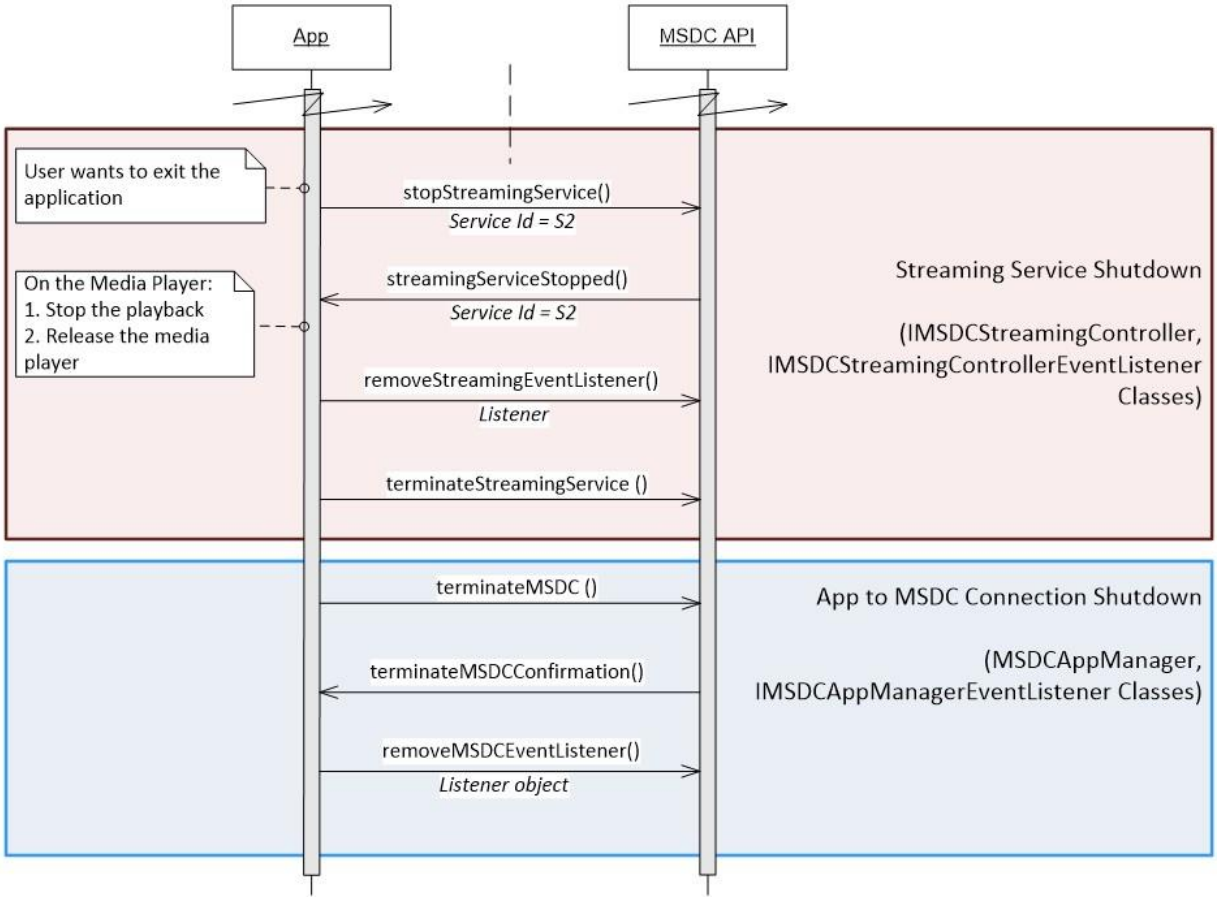


Figure 4-3 Streaming service call flow (3 of 3)

4.2 App-to-MSDC connection setup

The app starts here to begin communication with the MSDC.

4.2.1 Add MSDC Manager module event listener

4.2.1.1 Interface function

```
void addMSDCEventListener (IMSDCAppManagerEventListener listener)
```

4.2.1.2 Description

To make events generic to all types of services from the MSDC Manager module, the app must add the event listener using `addMSDCEventListener()`.

4.2.1.3 Call flows

Figure 4-4 shows the call flow for adding the MSDC Manager module event listener.

Figure 4-4 MSDC – Event listener registration

4.2.2 MSDC Manager module connection initialization

4.2.2.1 Interface function

```
void initializeMSDC (MSDCAppManagerInitParams params)
void initializeMSDCConfirmation ()
void msdcError (int errorCode, String message)
```

4.2.2.2 Prerequisites

Add MSDC Manager module event listener

4.2.2.3 Description

After adding the event listener, the app must initialize the connection to the MSDC Manager module using `initializeMSDC()`. Through `initializeMSDC()`, the app provides the following:

- App ID – The unique ID of the app, which the MSDC uses to authenticate the app.
- Reception Reporting Opt in – Informs the MSDC if the app is interested in opting in for receptions reporting. If the app wants to opt in, this value should be set to TRUE so that the MSDC can collect some or all of the following reception-related information:
  - Streaming service
    - Client ID – Mobile Directory Number of the device, or a randomly generated ID.
    - Service ID – Identifies the service to which the session belongs.
    - Session start and end time – Required to identify logging start and stop times.
    - Number of objects received or lost – The middleware periodically collects the number of objects that failed to be decoded and the number of objects it attempted to decode within each measurement period.
    - Network Cell ID – Global cell ID where the device is camped (collected once per measurement period).

  - File Delivery service
    - Client ID – Mobile Directory Number of the device, or a randomly generated ID.
    - Service ID – Identifies the service to which the session belongs.
    - Session start and end time – Required to identify logging start and stop times.
    - File URI – Identifier of the files that the app tried to download or capture.
    - URLs and delivery status of objects – The middleware logs of URLs and the delivery status of objects for which reception succeeded or failed.
    - Network Cell ID – Global cell ID where the device is camped (collected once per measurement period).

Note: An app that is only interested in Group Call service must set `Reception Reporting Opt in` to FALSE, because it cannot opt in for the reception reporting feature of the MSDC. Unless defined by the app, the MSDC considers the reception reporting opt-in value to be FALSE.

- Targeted MSDC middleware package name – This optional parameter identifies the targeted package name in case multiple MSDC middleware is installed on the device. If no package name is specified, the default is `com.qualcomm.ltebc`, which is the released Qualcomm® MSDC middleware APK package name.
- MSDC middleware connection mode – Optional parameter that identifies connection preference.
  - Local only – Mode default value. It is assumed that the MSDC middleware is on the same device as the UI client and the UI application is interested only in using the MSDC middleware on the device. The app may select this mode when no remote MSDC middleware running on mobile broadband product exists.
  - Remote only – Indicates the UI application is interested in connecting to remote MSDC middleware running on a mobile broadband product. The app may select this mode when it can leverage remote MSDC middleware running on a mobile broadband product at the home/office.
  - Remote Preferred – Indicates that the UI application prefers a connection to remote MSDC middleware running on a mobile broadband product. However, if the remote MSDC is not available, the UI application can connect to local MSDC middleware on the same device as the UI until the remote MSDC middleware becomes available again.
- Remote Retry timer in milliseconds – This optional parameter is only applicable in Remote or Remote Preferred connection mode. It specifies the retry period for connecting to remote MSDC middleware when the connection is lost. The default value is 5000 milliseconds.

If the MSDC API accepts the request, and if the connection initialization is successful, the MSDC API responds with `initializeMSDCConfirmation`.

Recommendation: To reduce startup time, the app should call `initializeMSDCConfirmation` as early as possible.

4.2.2.4 Call flows

4.2.2.4.1 Connection initialization

Figure 4-5 shows the call flow sequence for initializing the MSDC Manager module connection.

For code examples, see Section A.2.2.

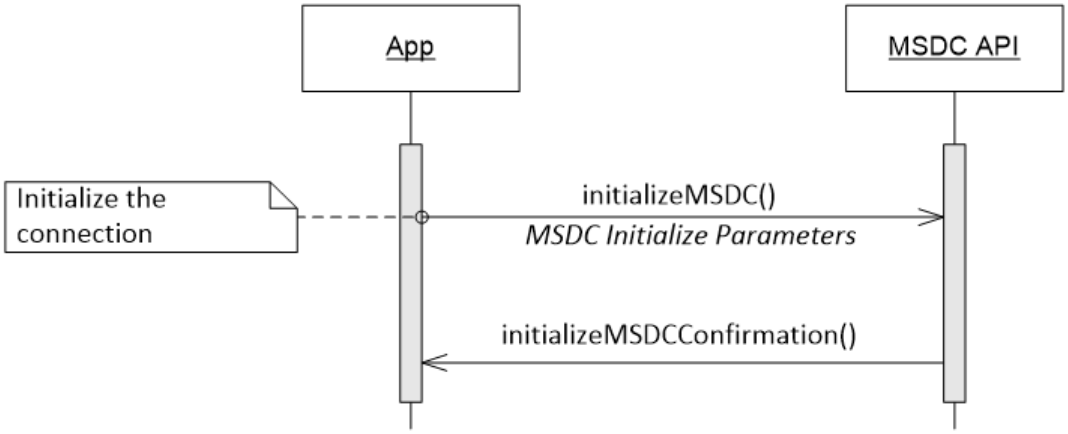


Figure 4-5 MSDC – Connection initialization

4.2.2.4.2 Connection initialization fails due to version mismatch

If a connection initialization fails because the MSDC API version used by the app is not supported by the MSDC, the MSDC API responds with `msdcError()` and the error code `ERROR_MSDC_VERSION_INCOMPATIBLE`.

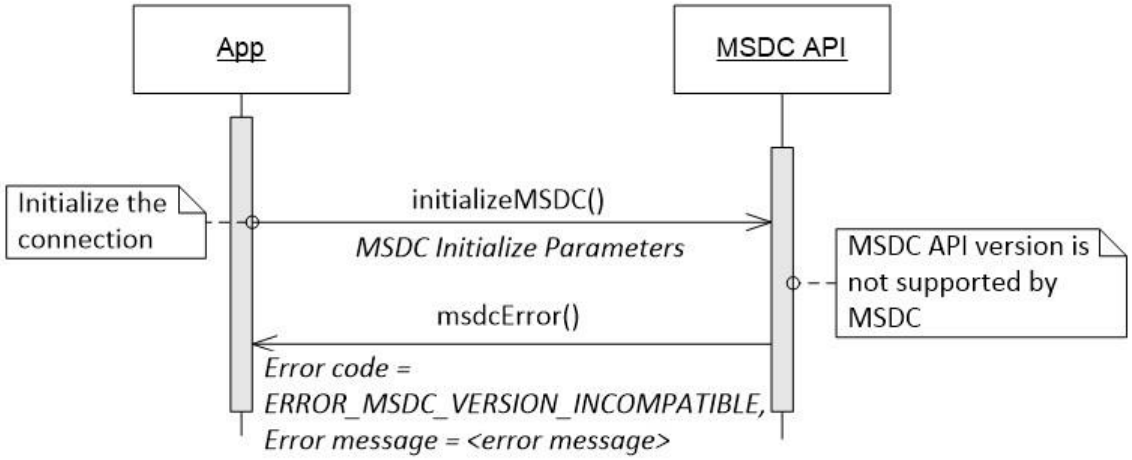


Figure 4-6 MSDC – Connection initialization fails due to version mismatch

4.2.2.4.3 Connection initialization with limited feature set support

A connection initialization may succeed even with a supported version mismatch of the MSDC API between the app and the MSDC. This can happen when the app uses a later version of the MSDC API than the version used by the MSDC.

In this case, the MSDC would only support limited features. The MSDC API notifies the app with `msdcWarning()` and the warning code `WARNING_MSDC_REDUCED_FEATURE_SET`.

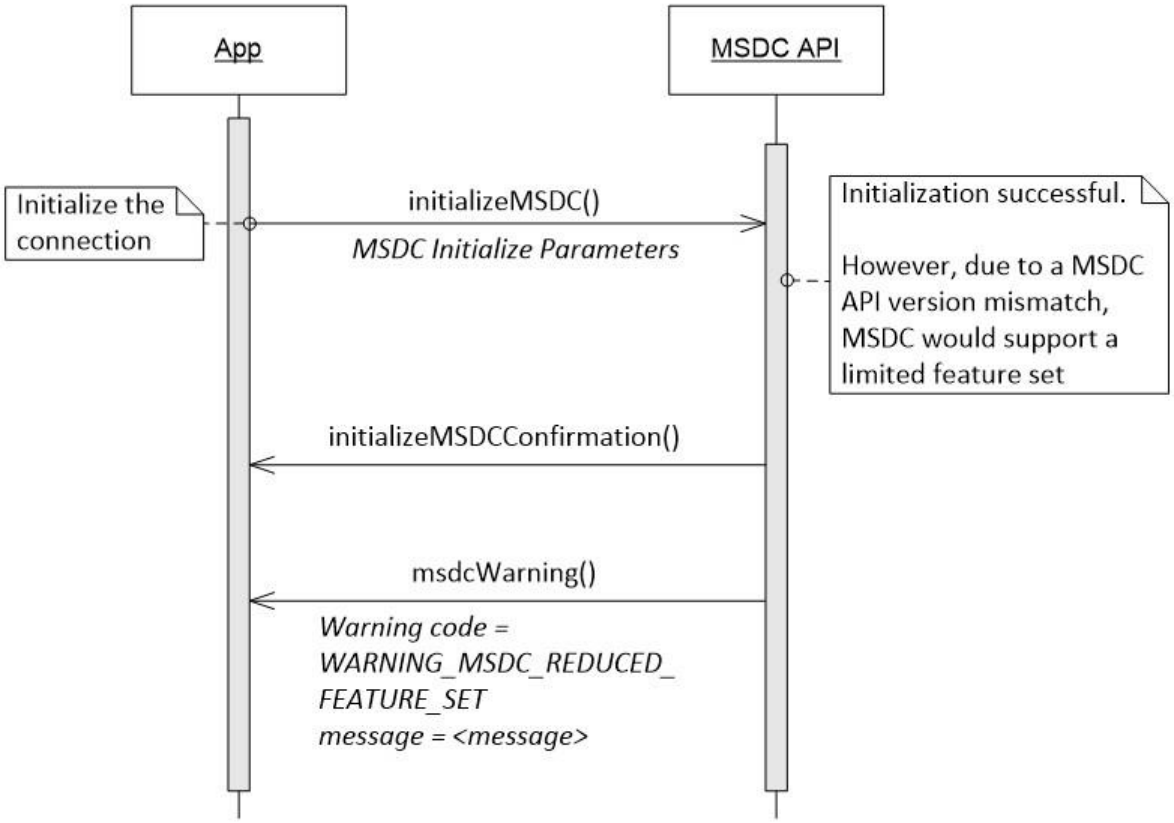


Figure 4-7 MSDC – Connection initialization with limited feature set support

4.2.2.4.4 Connection initialization fails due to carrier network not allowed

If the MSDC middleware has been configured to only run in a certain carrier network, the connection initialization fails if the device is operating in a network other than the one for which the MSDC middleware is configured. In this case, the app will receive the error code `ERROR_MSDC_CARRIER_CHANGE_NOT_ALLOWED`.

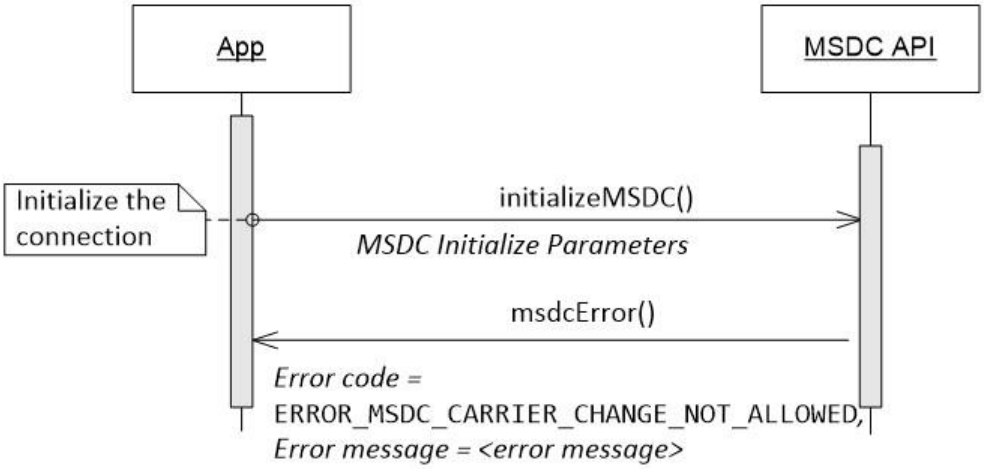


Figure 4-8 MSDC – Carrier change not allowed, connection initializaton fails

4.2.2.4.5 Connection initialization fails due to MSDC middleware not installed

If the application requests a specific MSDC middleware package name during connection initialization and it is not installed on the device, the app will receive the error code `ERROR_MSDC_MIDDLEWARE_NOT_INSTALLED`.

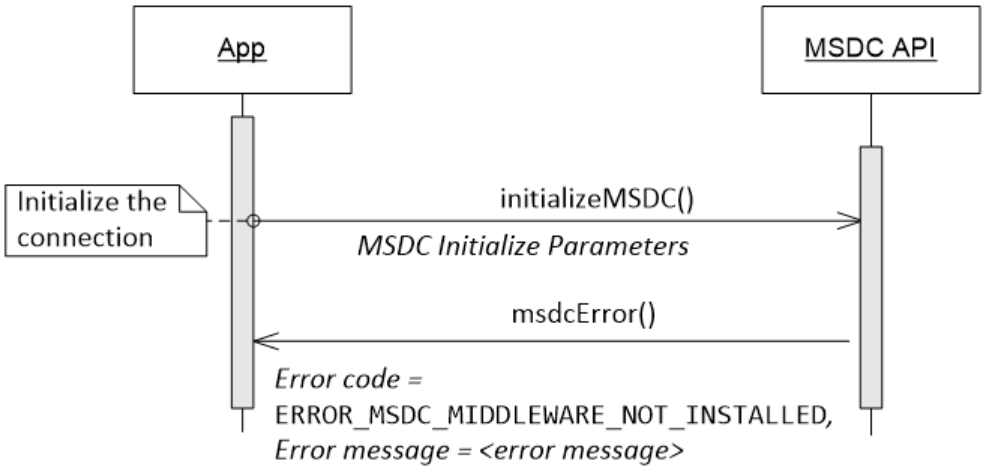


Figure 4-9 MSDC – MSDC middleware not installed, connection initializaton fails

4.2.2.4.6 Connection initialization fails due to MSDC middleware permissions

For Android M HLOS devices, the MSDC middleware requires permissions to be granted. There are scenarios where permissions are either not granted by default or denied by the user.

If the required permissions are denied or are not granted during initialization, the eMBMS application will receive the error code `ERROR_MSDC_APP_PERMISSIONS_NOT_GRANTED`.

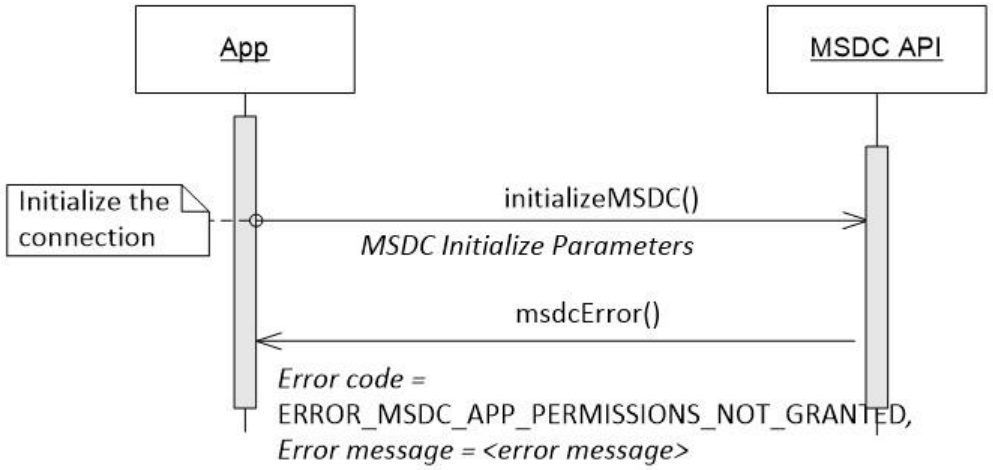


Figure 4-10 MSDC – Connection initialization failed due to MSDC middleware permissions

4.2.2.4.7 Connection initialization fails – other reasons

If a connection initialization fails for any other reason, the MSDC API responds with `msdcError()` and error code `ERROR_MSDC_UNABLE_TO_INITIALIZE`.

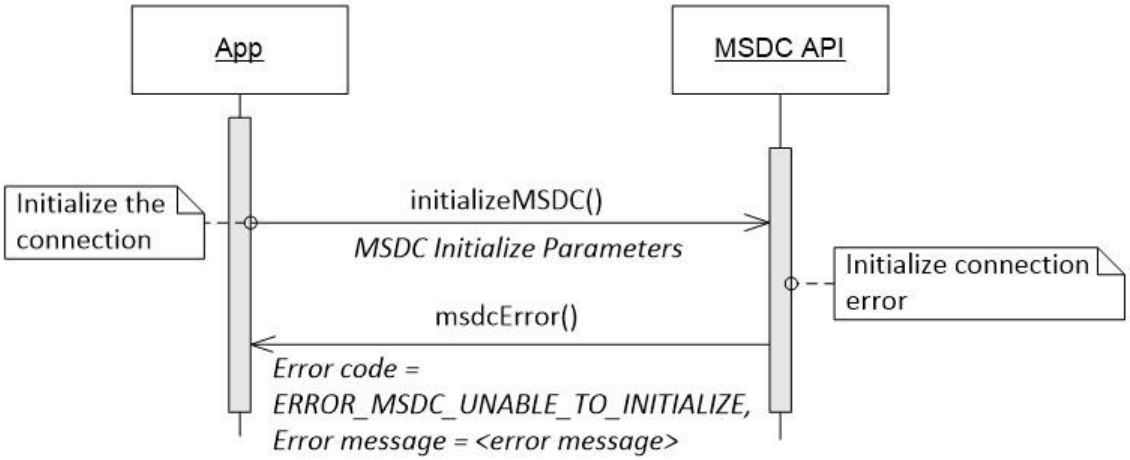


Figure 4-11 MSDC – Connection initialization fails due to other reasons

4.2.3 Get the Streaming module Controller and Model instances

4.2.3.1 Interface functions

```
IMSDCStreamingController getStreamingController()
IMSDCStreamingModel getStreamingModel()
```

4.2.3.2 Description

To send requests to the Streaming module of the MSDC, the app should use `getStreamingController()` and `getStreamingModel()` calls to get the Streaming module Controller and Model instances, respectively.

For code examples, see Section A.2.3.

4.2.3.3 Call flows

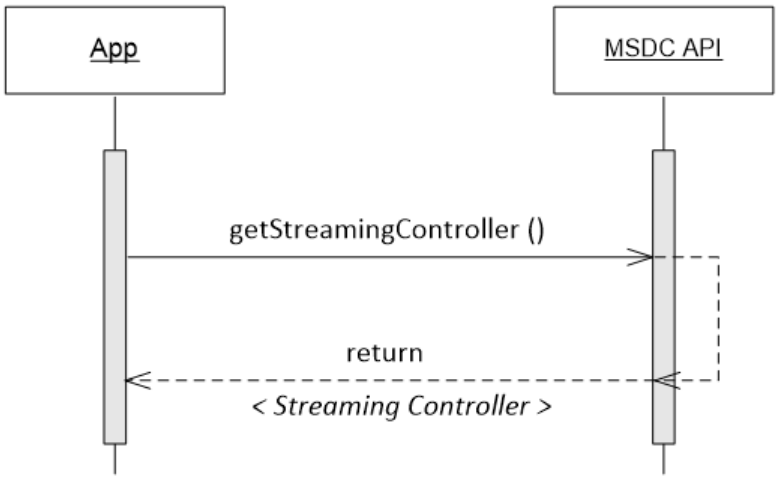


Figure 4-12 MSDC – Get Streaming module Controller instance

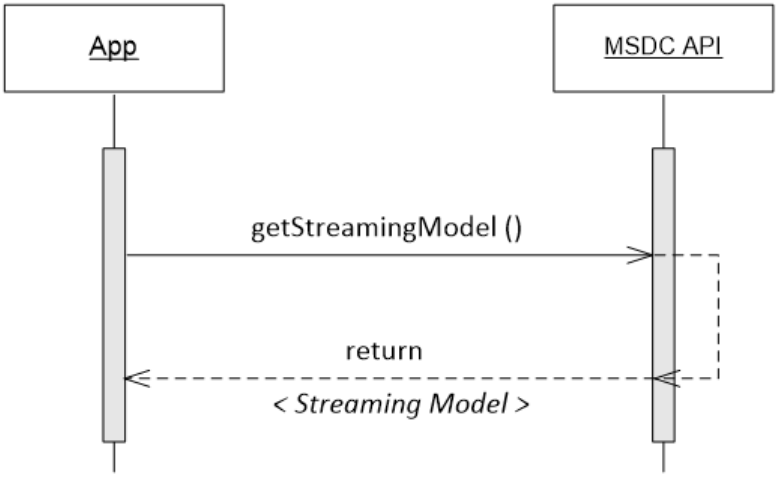


Figure 4-13 MSDC – Get Streaming module Model instance

4.3 Streaming module initialization

4.3.1 Add Streaming module event listener

4.3.1.1 Interface functions

```
void addStreamingEventListener (IMSDCStreamingControllerEventListener listener
    )
```

4.3.1.2 Prerequisites

Get the Streaming module Controller and Model instances

4.3.1.3 Description

To get Streaming module-related events from the MSDC, the app must add the event listener by using `addStreamingEventListener()`.

For code examples, see Section A.2.2.

4.3.1.4 Call flows

Figure 4-14 Streaming – Event listener registration

4.3.2 Streaming module connection initialization

4.3.2.1 Interface functions

```
void initializeStreamingService (StreamingInitParams params);
void streamingServiceConfirmation();
void streamingServiceError (int errorCode, String message, Integer serviceId);
```

4.3.2.2 Prerequisites

- MSDC Manager module connection initialization
- Add Streaming module event listener

4.3.2.3 Description

After the Streaming module event listener has been added, the app must initialize the connection to the Streaming module with `initializeStreamingService()`. This is how the app registers for the Streaming service and provides `Service Class info`.

`Service Class info` is a list of service classes that the app is interested in. The MSDC can give the app information and data only for those services that belong to this set of service classes. The list of service classes must be managed between the app provider and the carrier/operator.

Recommendation: The app can be designed to keep `Service Class info` configurable. This could help during app testing and make the app more flexible.

If the MSDC API accepts the request, and if the streaming module connection initialization is successful, the MSDC API responds with `streamingServiceInitializeConfirmation`.

For code examples, see Section A.2.2.

4.3.2.4 Call flows

4.3.2.4.1 Streaming connection initialization succeeds

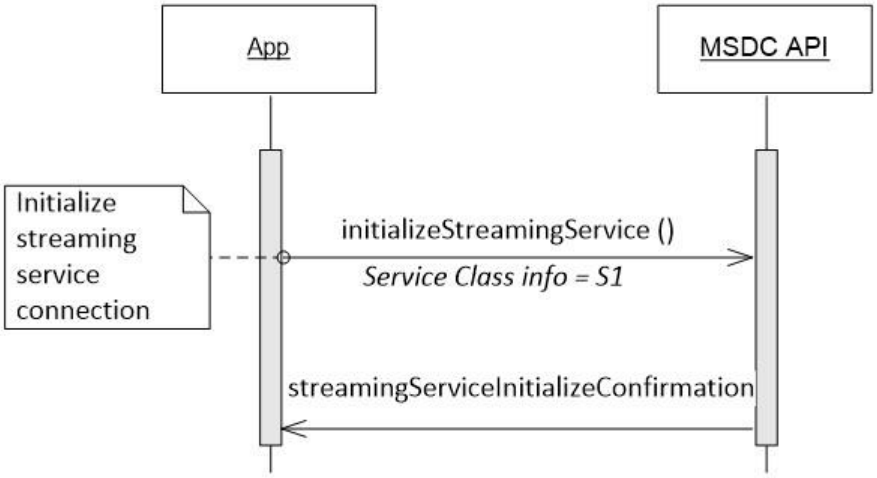


Figure 4-15 Streaming – Connection initialization succeeds

4.3.2.4.2 Connection initialization fails

If a Streaming module connection initialization fails, the MSDC API responds with `streamingServiceError()` and the error code `ERROR_S_UNABLE_TO_INITIALIZE`.

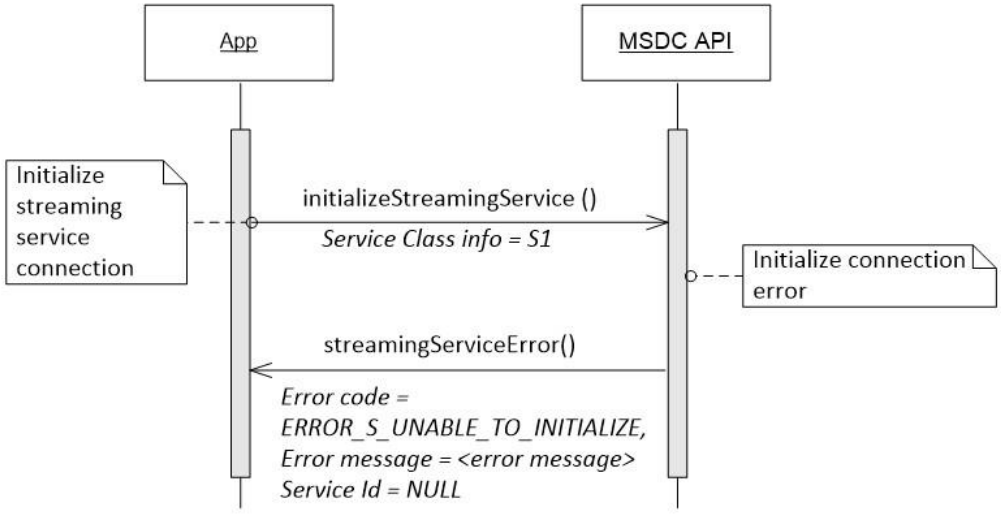


Figure 4-16 Streaming – Connection initialization fails

4.3.2.4.3 Streaming service class initialization fails

If a service class initialization fails, the MSDC API responds with `streamingServiceError()` and the error code `ERROR_S_SERVICE_CLASS_INITIALIZATION_FAILED`.

A typical reason for service class initialization failure is because the service class is already in use by another application.

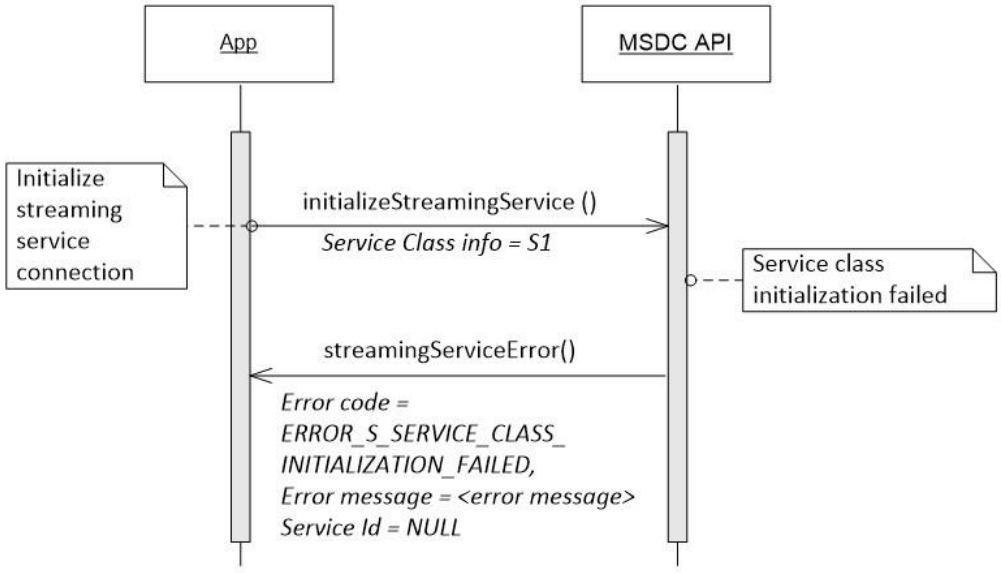


Figure 4-17 Streaming – Service class initialization fails

4.4 Streaming service management

The following sections define the set of calls to manage the start/stop sequence of a Streaming service.

4.4.1 Service states

Figure 4-18 shows the Streaming service states that are visible to the app. Depending on the actions taken by the MSDC, the service may move from one state to another.

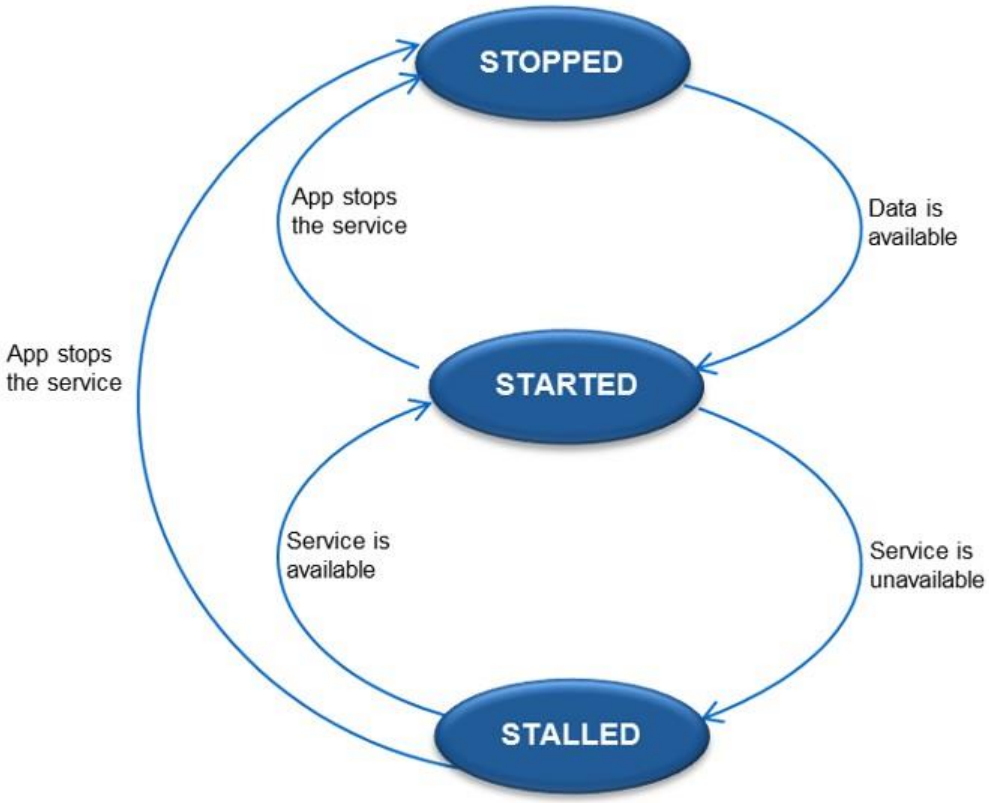


Figure 4-18 Streaming service states

- **STOPPED** – Default state of the service. The service is available for the user, but has not started yet.
  - If the user stops a service that is already STARTED or STALLED, it moves to the STOPPED state. When this happens, the app is notified by `streamingServiceStopped()` (see Section 4.4.3).
- **STARTED** – The service moves to STARTED when streaming data (if available on the network) is available on the device. When the Streaming service state moves to STARTED, the app is notified by `streamingServiceStarted()` (see Section 4.4.2).
- **STALLED** – If a service in the STARTED state has a temporary issue which causes streaming data to become unavailable to the DASH client, the MSDC moves the service to STALLED. When the streaming service state moves to STALLED, the app is notified by `streamingServiceStalled()` (see Section 4.4.5.1).
  - STALLED is expected to be a temporary state. The MSDC will move it to STARTED as soon as it can get the streaming data from the network. Alternatively, the app might choose to move the service to STOPPED by using `streamingServiceStopped()` (see Section 4.4.3).

Note: There are two temporary intermediate states not shown in Figure 4-18:

- **START_REQUESTED** — When the app requests a service to be started and it has not yet received a `streamingServiceStarted()` notification (see Sections 4.4.2 and 4.4.4).
- **STOP_REQUESTED** – When the app requests a service to be stopped and it has not yet received a `streamingServiceStopped()` notification (see Sections 4.4.3 and 4.4.4)

4.4.2 Start a Streaming service

4.4.2.1 Interface functions

```
void startStreamingService (int serviceId);
void streamingServiceStarted (int serviceId);
void streamingServiceError (int errorCode, String message, int serviceId)
```

4.4.2.2 Prerequisites

Streaming module connection initialization

4.4.2.3 Description

To start a Streaming service, the app should use `startStreamingService()`. Through this call, the app gives the service ID of the service to be started.

If starting the Streaming service is successful, the MSDC API responds with `streamingServiceStarted()` to indicate that the service has moved to the STARTED state.

4.4.2.4 Call flows

4.4.2.4.1 Starting a Streaming service

If the app wants to start a Streaming service, it must send a request with `startStreamingService()`. This function provides the service ID of the Streaming service that needs to be started. The app can get the service ID from `getStreamingServiceList()` (see Section 4.4.8.2).

After the service has successfully started, the MSDC API sends `streamingServiceStarted()` to the app and moves the service to the STARTED state.

For code examples, see Section A.3.2.

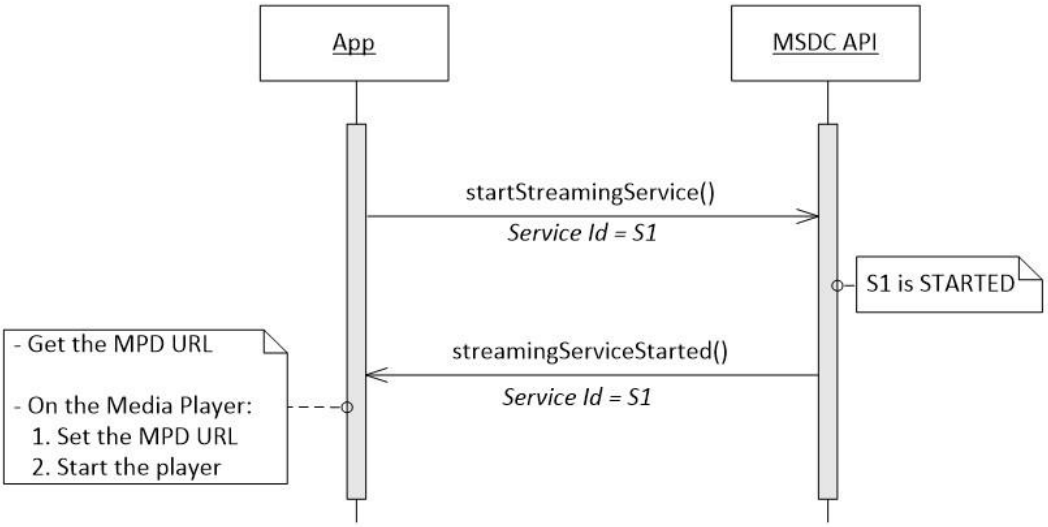


Figure 4-19 Starting a Streaming service

4.4.2.4.2 Service already in STARTED state

If the app tries to start a service that is already in the STARTED state, the MSDC API responds with `streamingServiceError()` and the error code `ERROR_S_UNABLE_TO_INITIALIZE` (see Figure 4-16).

4.4.2.4.3 Concurrent service limit reached

If the maximum allowed number of concurrent services is already running, the MSDC API responds to the app with `streamingServiceError()` and the error code `ERROR_S_CONCURRENT_SERVICE_LIMIT_REACHED`.

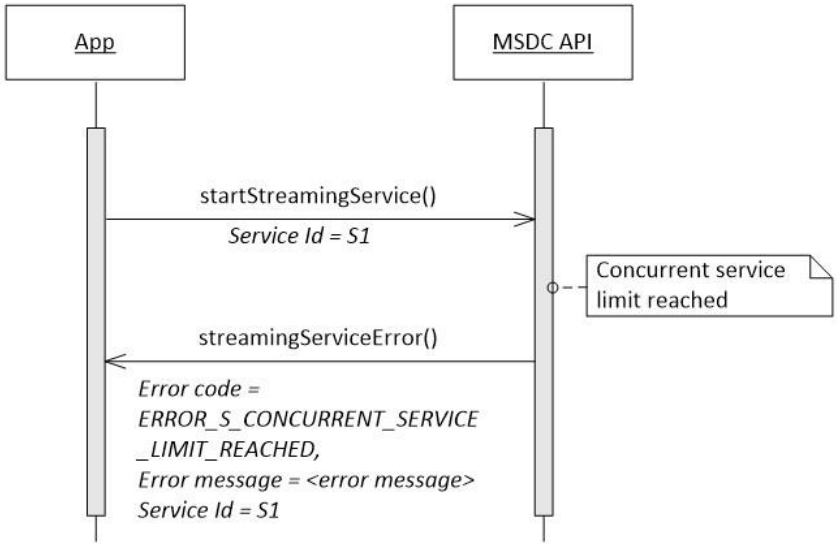


Figure 4-20 Streaming – Concurrent service limit reached

4.4.2.4.4 Unable to start Streaming service

If the MSDC cannot start the streaming service for some reason, the MSDC API responds to the app with `streamingServiceError()` and the error code `ERROR_S_UNABLE_TO_START_SERVICE`.

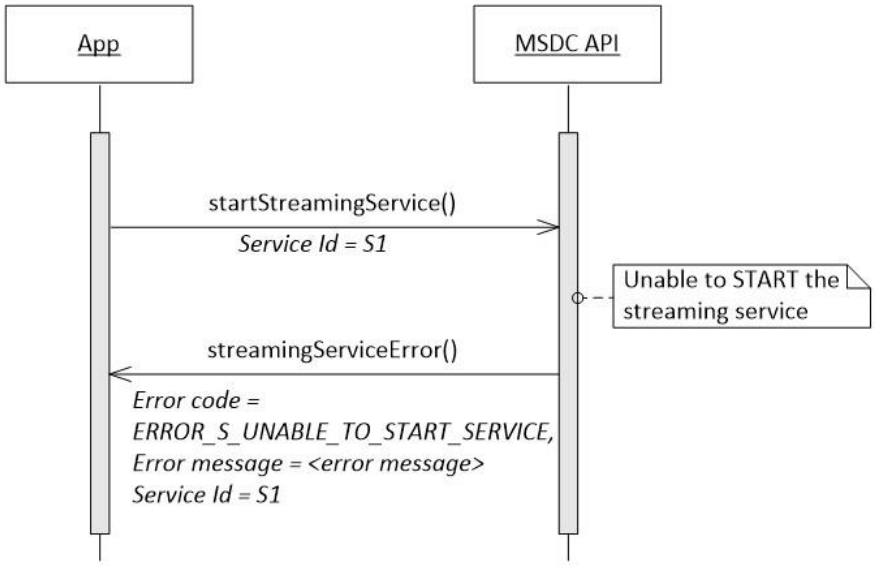


Figure 4-21 Unable to start Streaming service

4.4.3 Stop a Streaming service

4.4.3.1 Interface functions

```
void stopStreamingService (int serviceId);
void streamingServiceStopped (int serviceId);
void streamingServiceError (int errorCode, String message, int serviceId)
```

4.4.3.2 Prerequisites

Streaming module connection initialization

4.4.3.3 Description

To stop a Streaming service that is in the STARTED or STALLED state, the app should use `stopStreamingService()`. By calling this function, the app gives the service ID of the service to be stopped.

If stopping the Streaming service succeeds, the MSDC API responds with `streamingServiceStopped()` to indicate that the service has moved to the STOPPED state.

For code examples, see Section A.3.2.

4.4.3.4 Call flows

4.4.3.4.1 Stopping a Streaming service

To stop a service in the STARTED or STALLED state, the app uses the `stopStreamingService()`. If stopping the service is successful, the MSDC API sends `streamingServiceStopped()` to the app.

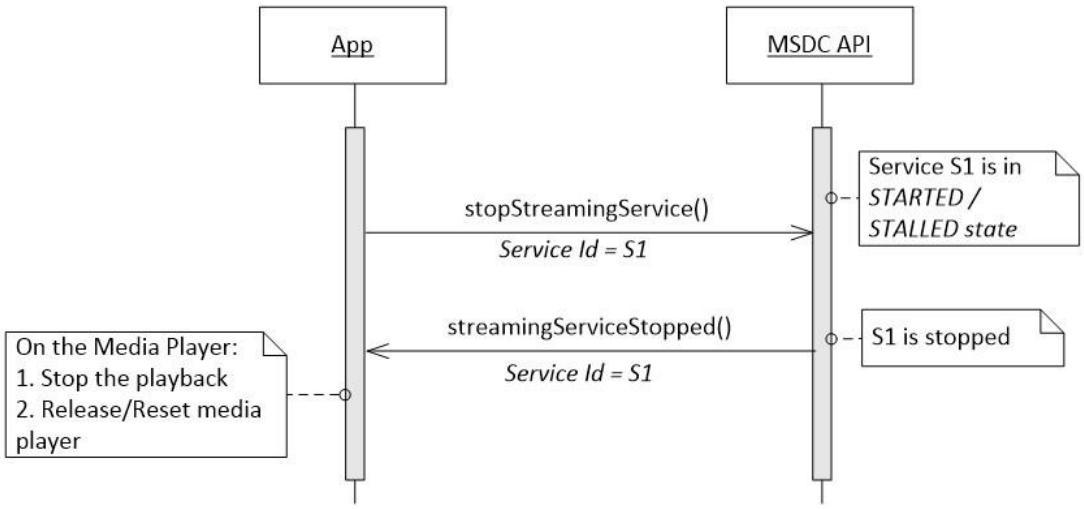


Figure 4-22 Stopping a Streaming service

4.4.3.4.2 Stopping a service already in STOPPED state

See Figure 4-22.

4.4.3.4.3 Unable to stop Streaming service

If the MSDC cannot stop the streaming service for some reason, the MSDC API responds to the app with `streamingServiceError()` and the error code `ERROR_S_UNABLE_TO_STOP_SERVICE`.

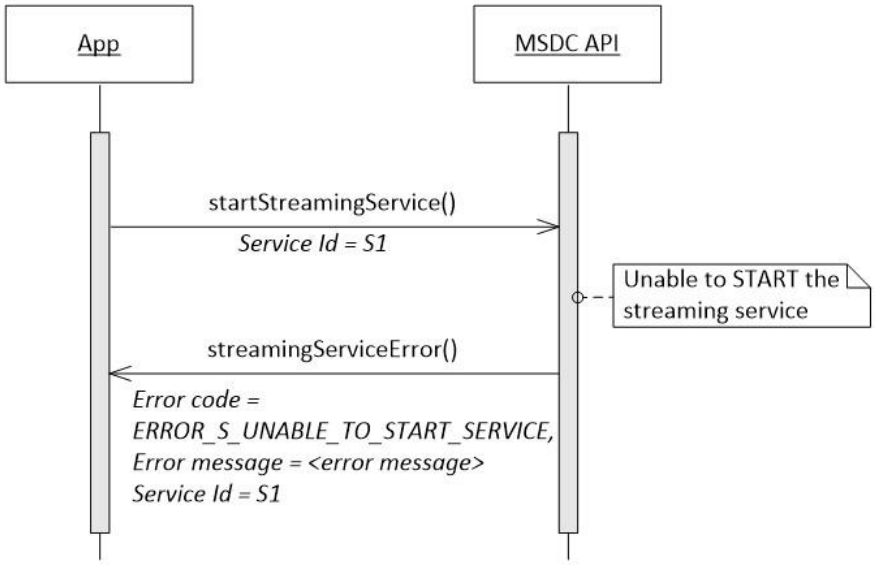


Figure 4-23 Unable to start Streaming service

4.4.4 Switch Streaming service

4.4.4.1 Interface functions

```
void switchStreamingService (int fromServiceId, int toServiceId);
void streamingServiceStopped (int serviceId);
void streamingServiceStarted (int serviceId);
```

4.4.4.2 Prerequisites

- Streaming module connection initialization
- Service S1 is in the STARTED state

4.4.4.3 Description

The app can use `switchStreamingService()` when it wants to switch from one service to another, e.g., S1 to S2. Through this call, the app gives the service ID of both services.

If the service switch is successful, the MSDC API responds with `streamingServiceStopped()` for S1 and `streamingServiceStarted()` for S2. This indicates that S1 has moved to the STOPPED state, while S2 has moved to the STARTED state.

Note: When the app uses the `switchStreamingService()`, the `streamingServiceStopped()` and `streamingServiceStarted()` notifications can be sent from MSDC in any order.

For code examples, see Section A.3.3.

4.4.4.4 Call flows

Figure 4-24 shows the call flow sequence for switching from the S1 service to the S2 service.

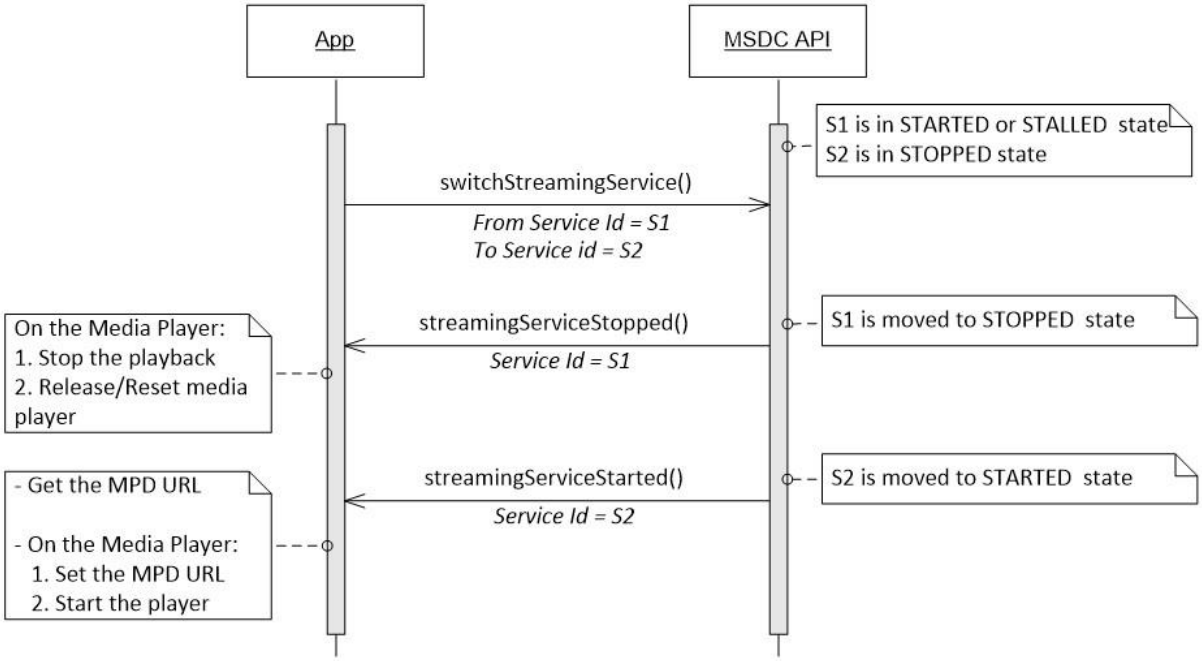


Figure 4-24 Switching from one Streaming service to another

4.4.5 Other information notifications

4.4.5.1 Streaming service stalled

4.4.5.1.1 Interface functions

```
void streamingServiceStalled (int serviceId);
```

4.4.5.1.2 Prerequisites

- Streaming module connection initialization
- Service is in the STARTED state

4.4.5.1.3 Description

If a service that is already in the STARTED state is temporarily unavailable, it is moved to the STALLED state. In this case, the MSDC API notifies the app with `streamingServiceStalled()`.

If the MSDC can resume the service, the MSDC API sends `streamingServiceStarted()`, to indicate that the service has moved back to the STARTED state.

Typically, the service moves to a STALLED state if the content of the service cannot be received due to radio control channel unavailability or adverse coverage conditions.

For code examples, see Section A.3.2.

4.4.5.1.4 Call flows

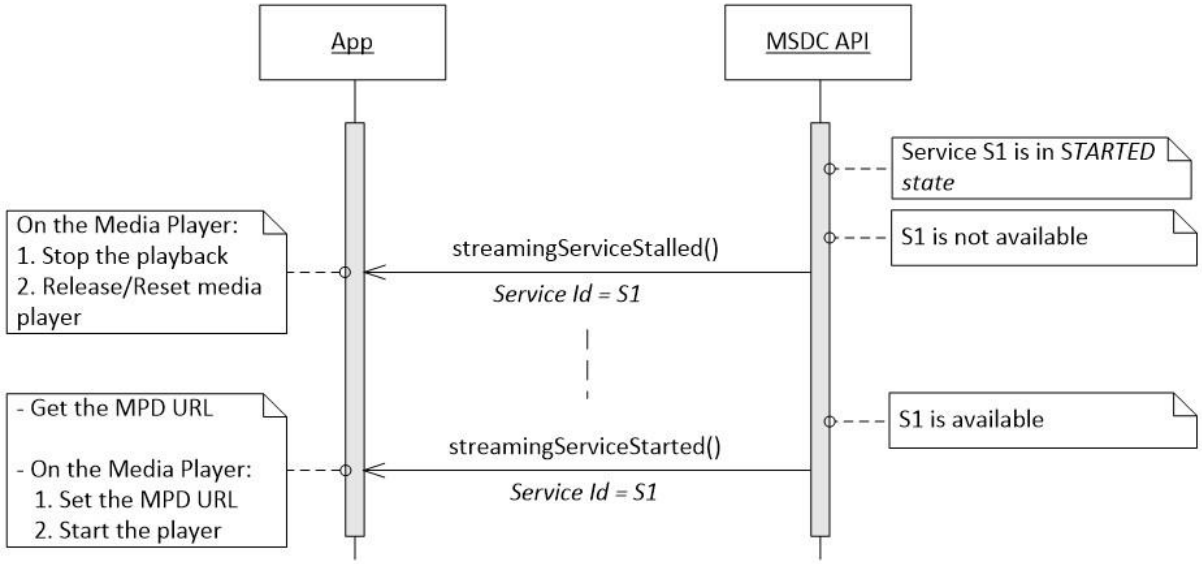


Figure 4-25 Streaming service is stalled

4.4.5.2 MPD update

4.4.5.2.1 Interface functions

```
void mpdUpdated(int ServiceId);
```

4.4.5.2.2 Prerequisites

Streaming module connection initialization

4.4.5.2.3 Description

If there is an updated MPD file available for a particular service, the MSDC API notifies the app with `mpdUpdated()`. The app can then use `getPlaybackUrl()` to get the updated MPD URL (see Section 4.4.8.1).

Note: `mpdUpdated()` is only sent when content is streamed in a loop. This is usually the case in demonstrations and tests. The app does not get this notification if there is a Service Announcement file update or MPD inband update in MSDC.

For code examples, see Section A.3.2.

4.4.5.2.4 Call flows

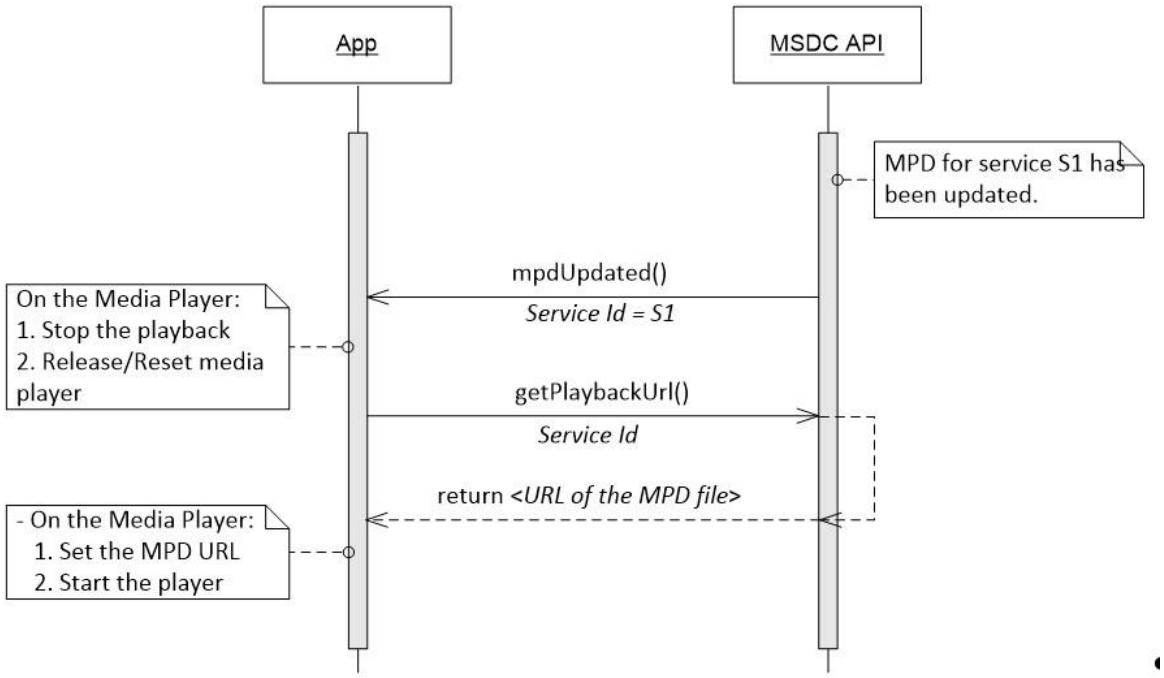


Figure 4-26 Streaming – MPD update notification

4.4.5.3 Service list update

4.4.5.3.1 Interface functions

```
void streamingServiceListUpdate();
```

4.4.5.3.2 Prerequisites

Streaming module connection initialization

4.4.5.3.3 Description

The MSDC API notifies the app with `streamingServiceListUpdate()` if there is a change in the available Streaming service list. This can mean one or both of the following:

- There may be an addition or deletion of service to the available Streaming services list
- There may be a change in the mode of availability of some services in the available Streaming services list

To get the updated service list, the app must invoke `getStreamingServiceList()` of the MSDC API (see Section 4.4.8.2).

Note: The MSDC API sends a `streamingServiceListUpdate()` to the app if there is any change in the camped group or the service groups. For more information on camped and service groups, see Section 4.4.8.5.

The Streaming service list has the `serviceAvailability` flag which can take the following values:

- 0 (STREAMING_SERVICE_AVAILABLE) – Service is available in broadcast mode.
- 1 (STREAMING_SERVICE_NOT_AVAILABLE_IN_BC) – Service is available in unicast mode.
- 2 (STREAMING_SERVICE_NOT_AVAILABLE) – Service is unavailable.

After receiving a `streamingServiceListUpdate()` indication from the MSDC, the app must use `getStreamingServiceList()` to get the updated available Streaming services list. You can pause or resume the active Streaming services depending on the value of the `serviceAvailability` flag.

For an active Streaming service, if the `serviceAvailability` flag changed from `STREAMING_SERVICE_AVAILABLE` to `STREAMING_SERVICE_NOT_AVAILABLE_IN_BC`, the MSDC middleware will trigger the service continuity feature to signal the DASH player for switching to Unicast mode. The service would then continue in Unicast mode.

Note: When you start a service for the first time, the Media Player should start playing only after receiving notification that the serve has started.

4.4.5.3.4 Call flows

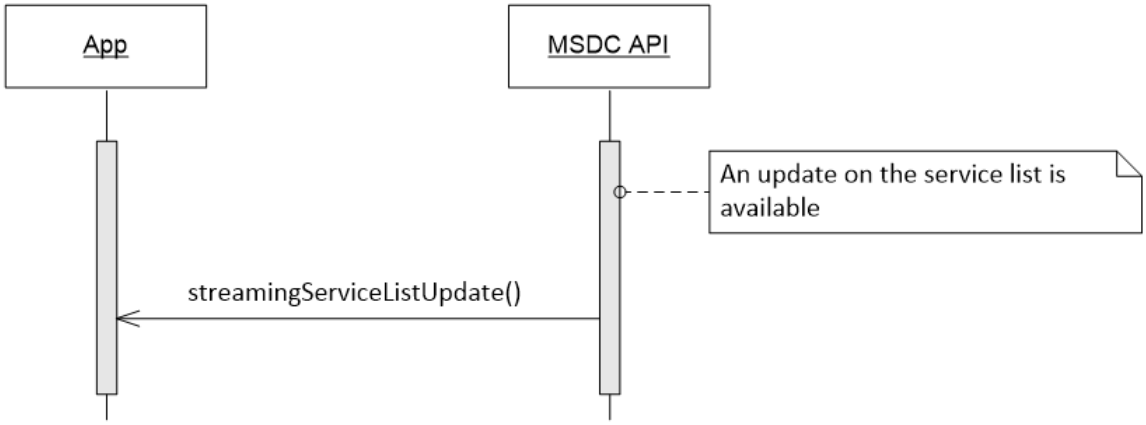


Figure 4-27 Streaming – Service list update notification

4.4.6 Other error notifications

4.4.6.1 Interface functions

```
void streamingServiceError(int service Id,
                           int errorCode,
                           String message);
```

4.4.6.2 Prerequisites

Add Streaming module event listener

4.4.6.3 Description

If the MSDC API wants to notify the app of any Streaming service error, it uses `streamingServiceError()`. For more information on the different types of error notifications, see Section 9.4.

For code examples, see Section A.3.2.

4.4.6.4 Call flows

If the app tries to request a Streaming service operation (starting, stopping, or switching a service) with an invalid service ID, the MSDC API responds to the app with `streamingServiceError()` and the error code `ERROR_S_INVALID_SERVICE_ID`.

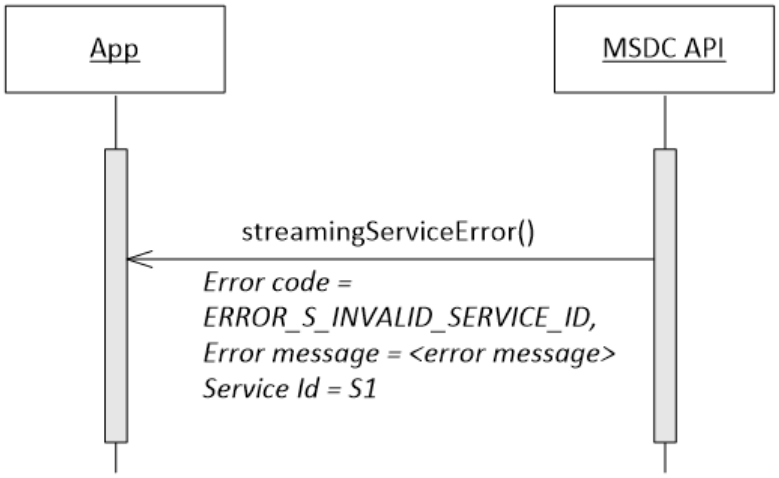


Figure 4-28 Streaming – Invalid service ID error notification

If a Streaming service is not available, the MSDC API responds to the app with `streamingServiceError()` and the error code `ERROR_S_SERVICE_UNAVAILABLE` along with the service ID of the affected service.

However, if all Streaming services are unavailable, the service ID would be NULL.

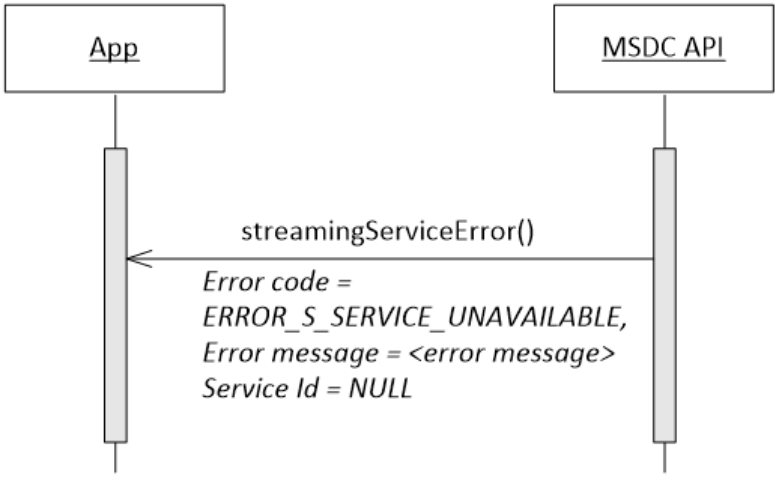


Figure 4-29 Error notification – Streaming module of MSDC is unavailable

The error code `ERROR_S_SERVICE_RESET` tells the app that the service related information (like service ID) has been reset and the app should fetch the new service information using `getStreamingServiceList()` (see Section 4.4.8.2).

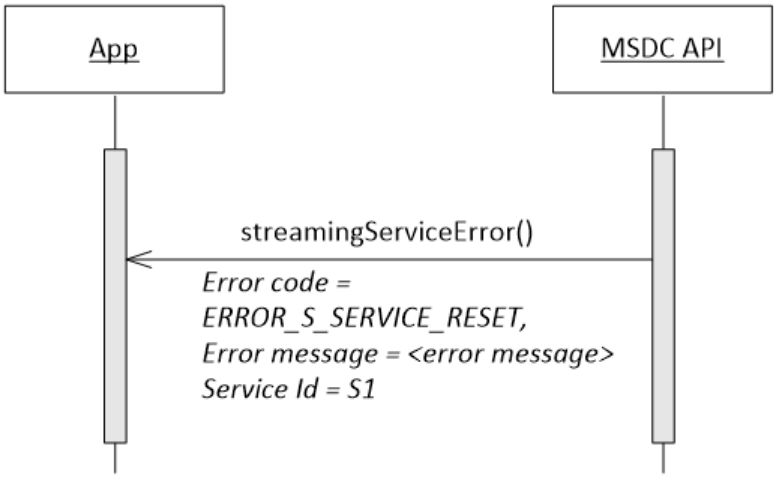


Figure 4-30 Error notification – Streaming service-related data has been reset

When trying to start a service (e.g., S1), the app may get an error notification with error code `ERROR_S_FREQUENCY_CONFLICT`. This means that another app is already using a Streaming service (e.g., S2) which falls in a different service group than the one which has S1 in it (see Section 4.4.8.5). In this case, the MSDC is unable to start the service.

4.4.7 Warning notifications

4.4.7.1 Interface functions

```
void streamingServiceWarning (int warningCode,
                              String warningMsg,
                              Integer serviceId);
```

4.4.7.2 Description

If the MSDC API wants to notify the app of any Streaming service warnings, it uses the overloaded `streamingServiceWarning()` notification.

For more information on the different types of warning notifications, see Section 9.4.

4.4.8 Information calls

4.4.8.1 Get the Playback URL

4.4.8.1.1 Interface functions

```
String getPlaybackUrl(int serviceId);
```

4.4.8.1.2 Prerequisites

Streaming module connection initialization

4.4.8.1.3 Description

To get the URL of the MPD file, the app can use `getPlaybackUrl()`. The return value has the URL of the MPD file.

This function is typically used when the app starts a Streaming service or when it is notified that an updated MPD file is available (see Section 4.4.5.2).

4.4.8.1.4 Call flows

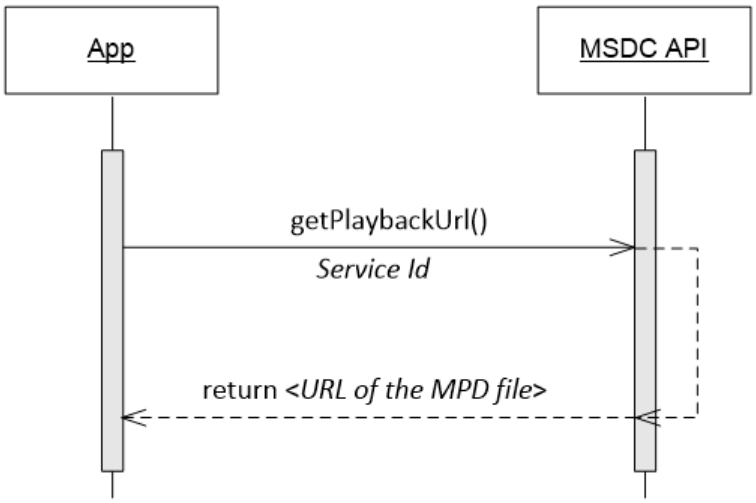


Figure 4-31 Streaming – Get Playback URL

4.4.8.2 Get service list

4.4.8.2.1 Interface functions

```
Map<Integer, Streamingservice> getStreamingServiceList ();
```

4.4.8.2.2 Prerequisites

Streaming module connection initialization

4.4.8.2.3 Description

To get the list of available Streaming services, the app should use `getStreamingServiceList()`.

The return value has data including the list of services and corresponding service IDs. For each service in the list, MSDC among other fields includes the mode of availability in the field “serviceAvailability”. Values it can take are:

- 0 (STREAMING_SERVICE_AVAILABLE) – Service is available in Broadcast mode.
- 1 (STREAMING_SERVICE_NOT_AVAILABLE_IN_BC) – Service not available in Broadcast mode but is available in Unicast mode.
- 2 (STREAMING_SERVICE_NOT_AVAILABLE) – Service is not available in either Unicast or Broadcast modes.

For the complete list of returned data, see Section 9.17. For code examples, see Section A.3.2.

`getStreamingServiceList()` provides the list of all services in all the service groups. For more information on service groups, see Section 4.4.8.5.

4.4.8.2.4 Call flows

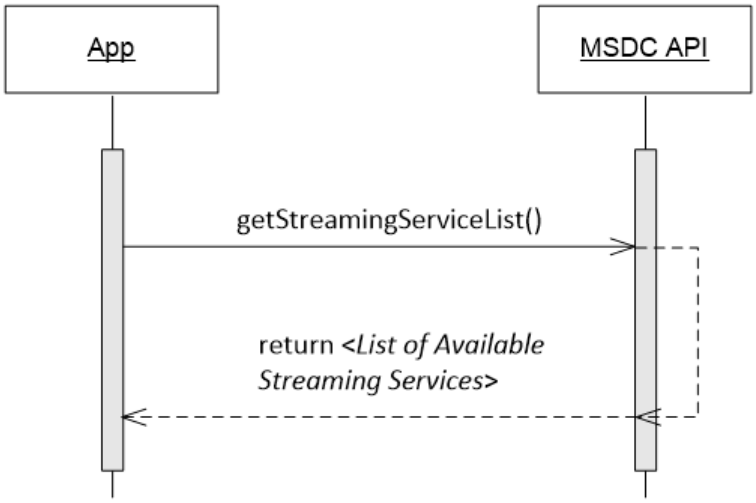


Figure 4-32 Get list of available Streaming services

4.4.8.3 Get running services

4.4.8.3.1 Interface functions

```
List<Integer> getRunningStreamingService ();
```

4.4.8.3.2 Prerequisites

Streaming module connection initialization

4.4.8.3.3 Description

To get the list of Streaming services in the STARTED state, the app should use `getRunningStreamingService()`. The return value has the list of service IDs.

4.4.8.3.4 Call flows

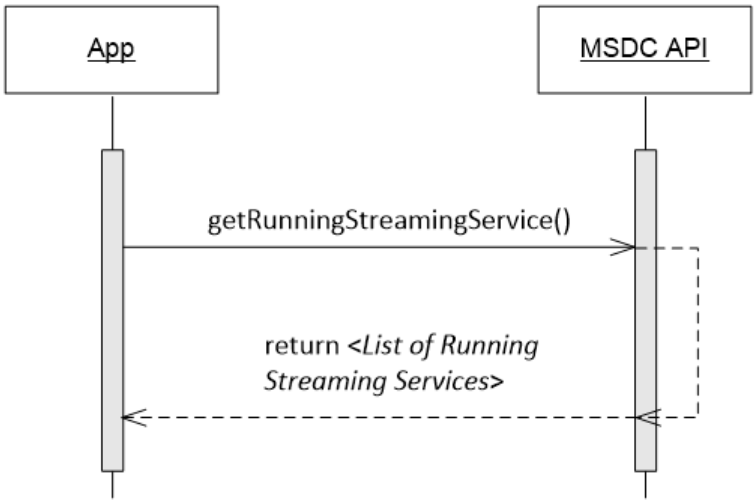


Figure 4-33 Get list of running Streaming services

4.4.8.4 Get Streaming service state

4.4.8.4.1 Interface functions

```
StreamingServiceState getStreamingServiceList ();
```

4.4.8.4.2 Prerequisites

Streaming module connection initialization

4.4.8.4.3 Description

To get the state of a particular streaming service, the app should use `getStreamingServiceState`. The return value has the state of the service.

For more information on the various Streaming service states, see Section 4.4.1.

4.4.8.4.4 Call flows

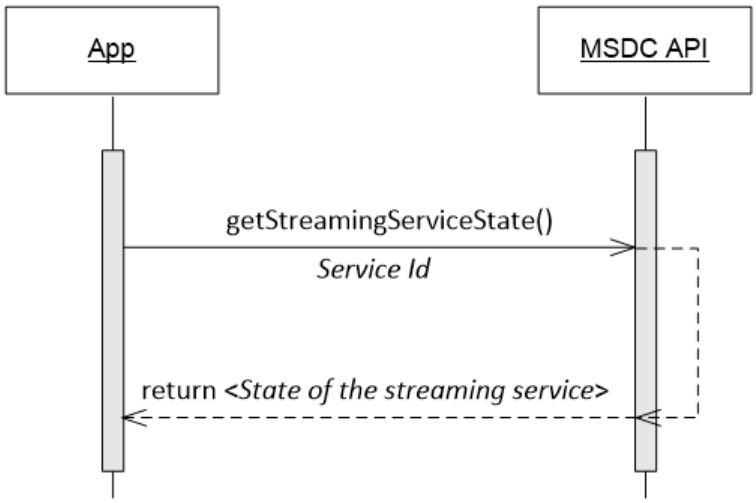


Figure 4-34 Get Streaming service state

4.4.8.5 Get camped group

4.4.8.5.1 Interface functions

```
GroupItem getCampedGroup();
```

`getCampedGroup()` returns the *GroupItem* object which contains the following member variables:

```
String groupName;
List<Integer> serviceAreaIdList;
List<Integer> serviceHandleList;
```

The member variable *groupName* is the name of the currently camped network type. The *serviceHandleList* is the list of streaming services which are available/valid in the Service Areas listed in the *serviceAreaIdList* of that group.

4.4.8.5.2 Prerequisites

Streaming module connection initialization

4.4.8.5.3 Description

The UE may be able to access services broadcast on multiple frequency carriers in a given geographical location even though it may be camped on a single frequency.

Services on the camped frequency are called the camped group. Similarly, the services on all frequencies that the UE can access can be grouped per frequency into multiple service groups. Thus, a camped group is the same or a subset of all the service groups.

Service groups are a set of services the UE can concurrently access. The services in the camped group are immediately accessible without the UE having to switch/acquire another frequency carrier.

To get camped group information from the MSDC, the app should use `getCampedGroup()`. The return value defines the following:

- Group name
- Service Area ID list
- Service handle list

Note:

- If the app tries to access a service outside the camped group, an additional delay ensues.
- In a multiview UI of the app, services from different service groups cannot be displayed.

4.4.8.5.4 Call flows

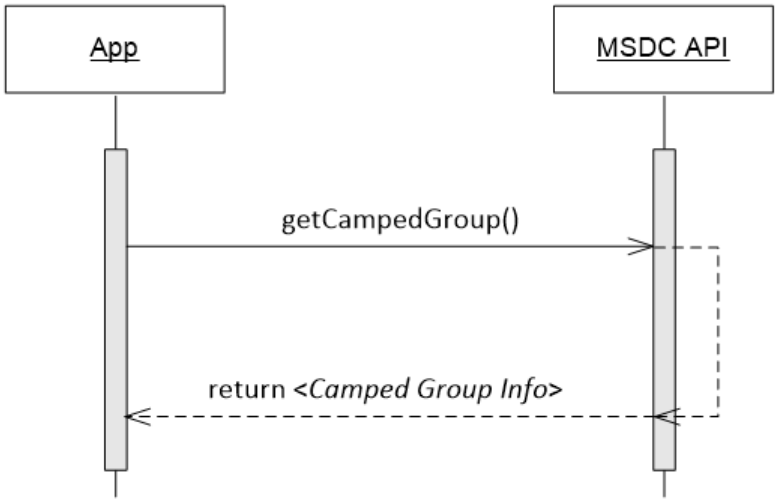


Figure 4-35 Streaming – Get camped group information

4.4.8.6 Get Streaming group list

4.4.8.6.1 Interface functions

```
List< GroupItem > getStreamingGroupList ()
```

`getStreamingGroupList()` returns the list of available groups (*GroupItem*) where Streaming services are available. Each *GroupItem* class contains the following member variables:

```
String groupName;
List<Integer> serviceAreaIdList;
List<Integer> serviceHandleList;
```

The member variable *groupName* is the name of the currently camped network type. The *serviceHandleList* is the list of streaming services which are available/valid in the Service Areas listed in the *serviceAreaIdList* of that group.

4.4.8.6.2 Prerequisites

Streaming module connection initialization

4.4.8.6.3 Description

To get the list of all service groups other than the camped group, the app should use `getStreamingGroupList()` (see Section 4.4.8.5). The return value has the list of service groups other than the camped group.

4.4.8.6.4 Call flows

Figure 4-36 shows the call flow sequence to get the Streaming group list.

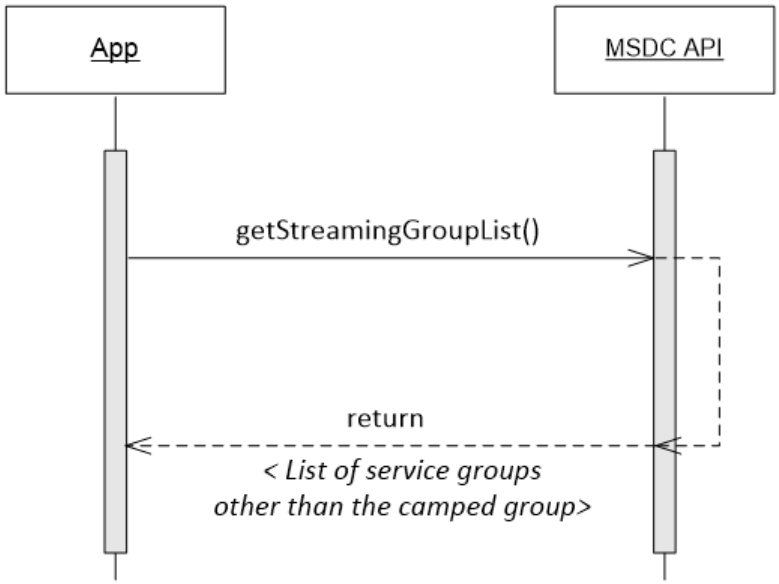


Figure 4-36 Get Streaming group list

4.4.8.7 Get Streaming service list by group

4.4.8.7.1 Interface functions

```
Map <Integer, StreamingService> getStreamingServiceListByGroup
                                    (String groupName)
```

4.4.8.7.2 Prerequisites

Streaming module connection initialization

4.4.8.7.3 Description

To get the list of all Streaming services for a particular service group, the app should use `getStreamingServiceListByGroup()`. For more information on service groups, see Section 4.4.8.5.

4.4.8.7.4 Call flows

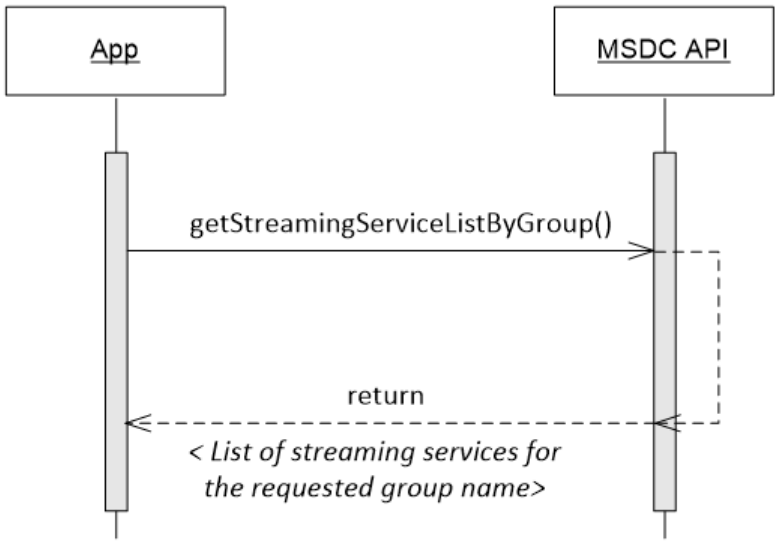


Figure 4-37 Get Streaming service list by group

4.5 MSDC Manager module connection management

4.5.1 MSDC error notifications

4.5.1.1 Prerequisites

Add MSDC Manager module event listener

4.5.1.2 Description

The MSDC uses `msdcError()` to notify the app of an generic MSDC error. For more information on the different types of error notifications, see Section 9.4.

4.5.1.3 Call flows

If the MSDC does not get enough memory from the OS, the MSDC API responds to the app with `msdcError()` and the error code `ERROR_MSDC_UNABLE_TO_ALLOCATE_MEMORY`.

In this case, check if the device that is running the MSDC has enough free memory.

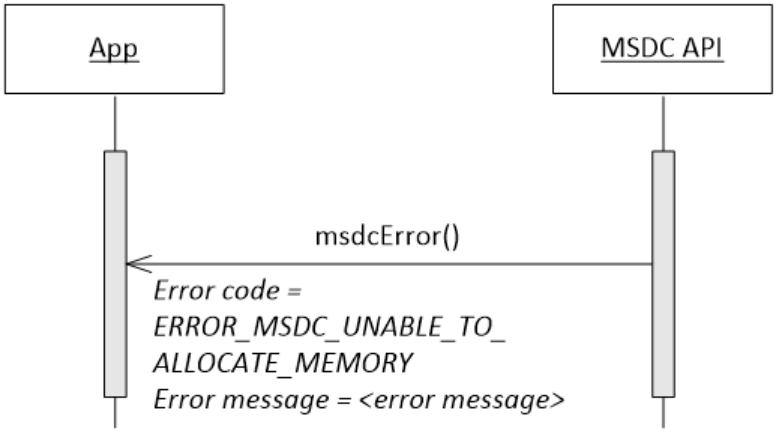


Figure 4-38 MSDC – Error notification, unable to allocate memory

If the MSDC API notifies the app using `msdcError()` with any of the following operator-specific error codes, contact the operator:

- ERROR_MSDC_UNABLE_TO_LOCATE_PROVISIONING
- ERROR_MSDC_PROVISIONING_FILE_INCOMPATIBILITY
- ERROR_MSDC_UNABLE_TO_PARSE_PROVISIONING_FILE
- ERROR_MSDC_UNABLE_TO_BIND_PORT
- ERROR_MSDC_TIME_SYNC
- ERROR_MSDC_MODEM_VERSION_INCOMPATIBLE
- ERROR_MSDC_BOOTSTRAP_CONNECTION_FAIL
- ERROR_MSDC_UNABLE_TO_PARSE_BOOTSTRAP
- ERROR_MDSC_NOT_CONNECTED_TO_HOMECARRIER_LTE
- ERROR_MDSC_FAILED_BOOTSTRAP_SERVER_DISCOVERY
- ERROR_MDSC_SIM_READ
- ERROR_MSDC_UNABLE_TO_ENABLE_MODEM
- ERROR_MSDC_MIDDLEWARE_NOT_INSTALLED
- ERROR_MSDC_EMBMS_SERVICE_NOT_AVAILABLE

4.5.2 MSDC warning notifications

4.5.2.1 Prerequisites

Add MSDC Manager module event listener

4.5.2.2 Description

If the MSDC wants to notify the app of any generic MSDC warning, it uses the overloaded `msdcWarning()`. For more information on the different types of warning notifications, see Section 9.4.

4.5.2.3 Call flows

If the MSDC detects a change in the network on which the device operates on, the MSDC API warns the app using `msdcWarning()` with the warning code WARNING_MSDC_NETWORK_CHANGE_DETECTED.

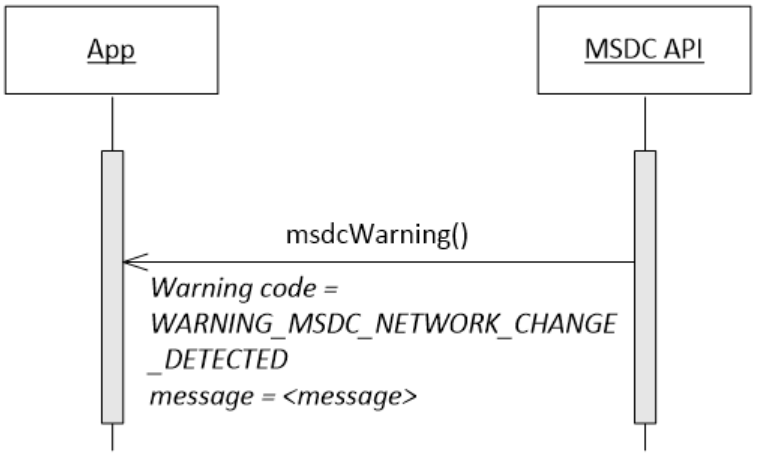


Figure 4-39 MSDC – Warning notification, network change detected

The MSDC API notifies the app using `msdcWarning()` with the following possible warnings:

- WARNING_MSDC_NETWORK_CHANGE_DETECTED
- WARNING_MSDC_REDUCED_FEATURE_SET

4.5.3 Enhanced 911 notifications

4.5.4 Interface functions

```
void e911Indication(int state);
```

4.5.4.1 Prerequisites

MSDC Manager module connection initialization

4.5.4.2 Description

When you dial 911, the phone should ideally stop all activities until the end of both the Enhanced 911 (E911) call and the subsequent callback period. The callback period is usually 5 minutes long.

The middleware halts all processing and sends an E911 indication with status IN to the application when the modem indicates that it is in the E911 state.

The middleware resumes normal operations and sends an E911 indication with status OUT to the application when the modem indicates the end of the E911 state.

On receiving an E911 indication from the middleware, it is recommended that the application informs the user of the E911 state and exit.

4.5.4.3 Call flows

If you dial 911 and the app is running in the background, the app receives the E911 notification. If you launch the app during the callback period, the app receives the E911 notification after the MSDC initialization and requests to initialize other modules would fail.

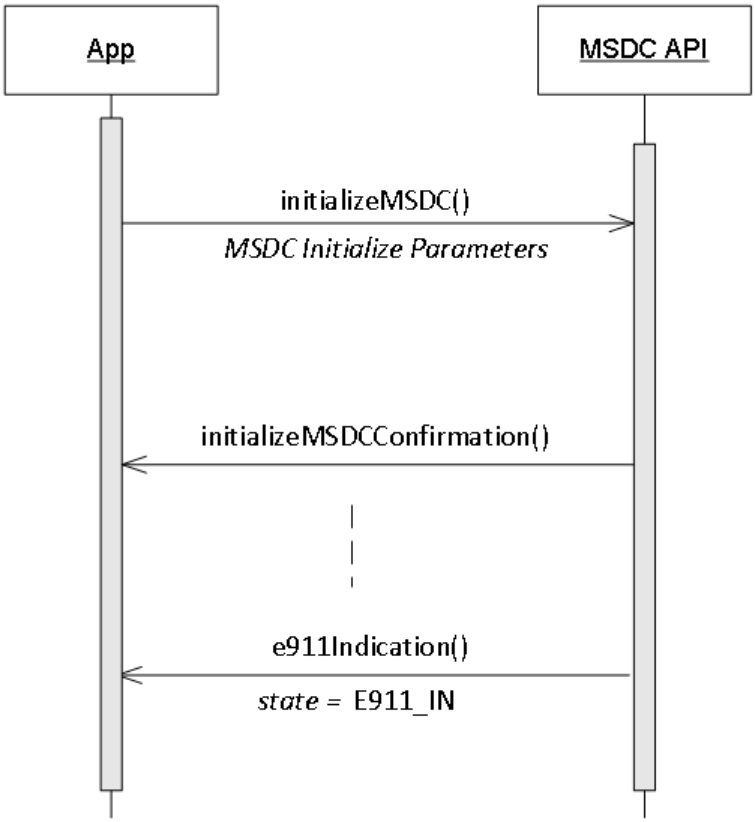


Figure 4-40 MSDC – E911 call flow

4.5.5 MSDC unavailable notifications

4.5.6 Interface functions

```
void msdcUnavailableNotification(int reason);
```

4.5.6.1 Prerequisites

Add MSDC Manager module event listener

4.5.6.2 Description

The UI application receives this notification when there is no connection to MSDC middleware. This notification includes a possible reason for the connection loss:

- MSDC_NONE – There is no MSDC middleware found based on the UI application preference stated in the `initializeMSDC()` call.
- MSDC_NO_WIFI – There is no Wi-Fi connection to connect to remote MSDC middleware.
- MSDC_REMOTE_AVAILABLE – Remote MSDC middleware is available and the UI application is disconnected from local MSDC middleware on the device while connecting to remote MSDC middleware. The app should receive `msdcAvailableNotification()` shortly after this notification.

Note: It is recommended to release mediaPlayer after receiving the MSDC unavailable notification.

4.5.7 MSDC available notifications

4.5.7.1 Interface functions

```
void msdcAvailableNotification(int which, Object obj);
```

4.5.7.2 Prerequisites

Add MSDC Manager module event listener

4.5.7.3 Description

The UI application receives this notification when it is connected to MSDC middleware. This notification includes a connection type:

- MSDC_LOCAL – UI application is connected to local MSDC middleware found on the same device as the UI.
- MSDC_REMOTE – UI application is connected to remote MSDC middleware found on mobile broadband product.

4.5.8 Information calls

4.5.8.1 Get the MSDC API version

4.5.8.1.1 Interface functions

```
String getAPIVersion();
```

4.5.8.1.2 Prerequisites

MSDC Manager module connection initialization

4.5.8.1.3 Description

To get the MSDC API version, the app can use `getAPIVersion()`. The return value has the MSDC API version.

4.5.8.1.4 Call flows

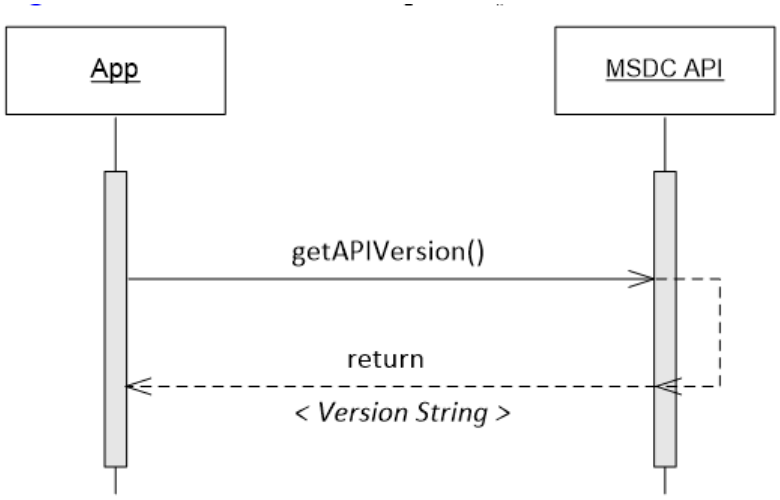


Figure 4-41 MSDC – Get API version

4.5.8.2 Get the MSDC server version

4.5.8.2.1 Interface functions

```
String getServerAPIVersion ();
```

4.5.8.2.2 Description

To get the MSDC server version, the app can use `getServerAPIVersion()`. The return value has the MSDC server version.

4.5.8.2.3 Call flows

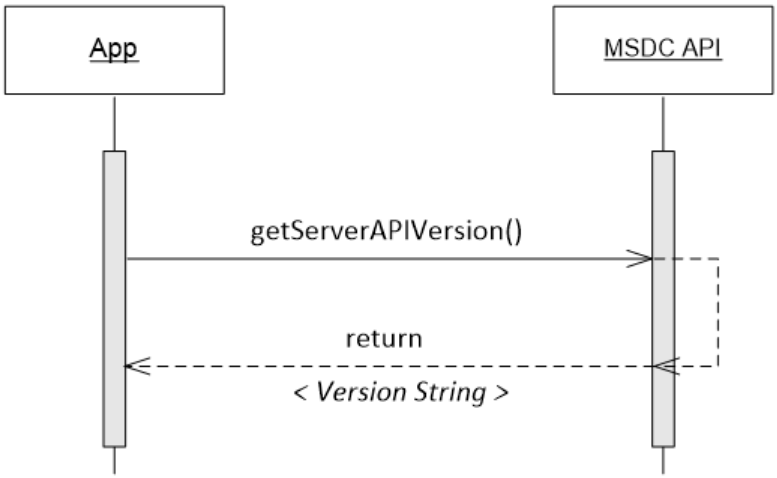


Figure 4-42 MSDC – Get Server version

4.5.8.3 Check MSDC initialization status

4.5.8.3.1 Interface functions

```
boolean isMSDCInitialized();
```

4.5.8.3.2 Prerequisites

Add MSDC Manager module event listener

4.5.8.3.3 Description

To identify whether the connection to the MSDC has been initialized, the app uses `isMSDCInitialized()`. If the return boolean value is TRUE, the MSDC connection has been initialized.

4.5.8.3.4 Call flows

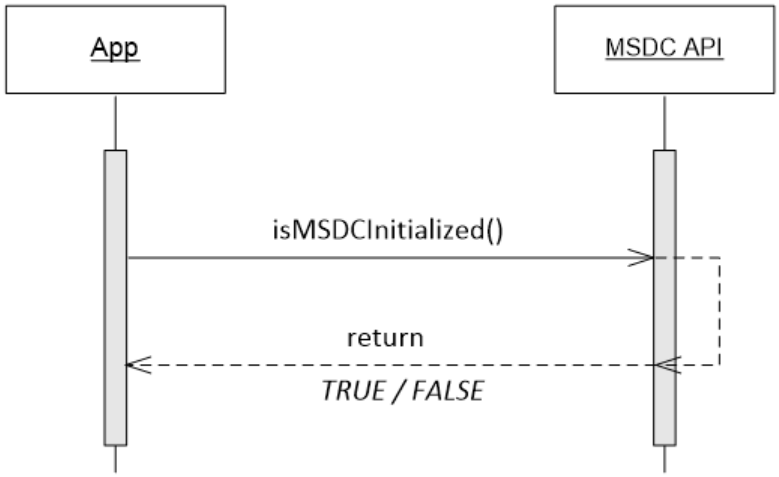


Figure 4-43 MSDC – Check MSDC initialization status

4.6 App-to-MSDC connection shutdown

This section defines the calls the app uses to shut down the connection with the MSDC.

4.6.1 Close Streaming module connection

4.6.1.1 Interface functions

```
void terminateStreamingService()
```

4.6.1.2 Prerequisites

Streaming module connection initialization

4.6.1.3 Description

To close the connection to the streaming module of the MSDC, the app uses `terminateStreamingService()`.

4.6.1.4 Call flows

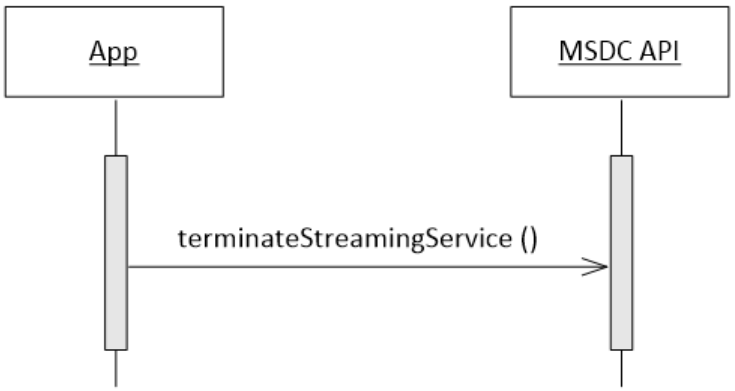


Figure 4-44 Close Streaming module connection

4.6.2 Remove Streaming module event listener

4.6.2.1 Interface functions

```
void removeStreamingEventListener
       (IMSDCStreamingControllerEventListener listener);
```

4.6.2.2 Prerequisites

Add Streaming module event listener

4.6.2.3 Description

To stop getting events from the Streaming module of the MSDC, the app must remove the event listener that it added earlier (see Section 4.3.1) by using `removeStreamingEventListener()`.

For code examples, see Section A.3.2.

4.6.2.4 Call flows

Figure 4-45 Remove Streaming event listener

4.6.3 Close MSDC Manager module connection

4.6.3.1 Interface functions

```
void terminateMSDC ()
void terminateMSDCConfirmation();
```

4.6.3.2 Prerequisites

MSDC Manager module connection initialization

4.6.3.3 Description

To close all connections to the MSDC, the app uses `terminateMSDC()`. If the MSDC successfully closes the connection, it returns `terminateMSDCConfirmation()`.

Note: `terminateMSDC()` closes the connections to all the modules that the app is using: Streaming (Chapter 4), File Delivery (Chapter 5), Network (Chapter 6), and Group Call (Chapter 7).

4.6.3.4 Call flows

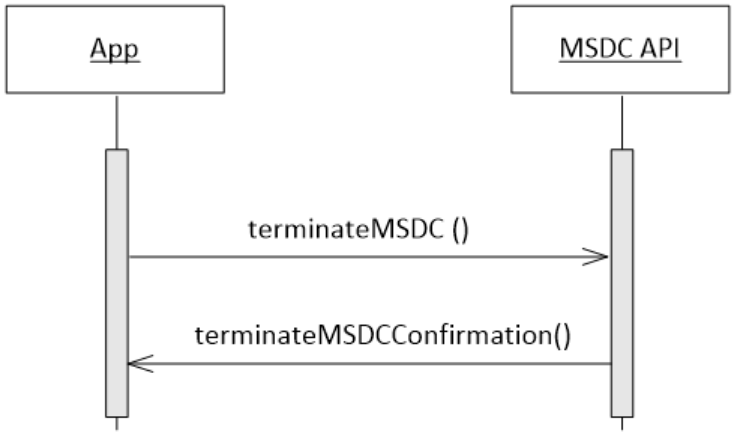


Figure 4-46 Close MSDC connection

4.6.4 Remove MSDC Manager module event listener

4.6.4.1 Interface functions

```
void removeMSDCEventListener (IMSDCAppManagerEventListener
                              listener)
```

4.6.4.2 Prerequisites

Add MSDC Manager module event listener

4.6.4.3 Description

To stop getting events from the MSDC, the app removes the event listener that it added earlier (see Section 4.2.1) by using `removeEventListener()`.

4.6.4.4 Call flows

Figure 4-47 MSDC – Remove MSDC event listener

5 File Delivery Service

This chapter describes the app-to-MSDC interface call flow sequences for a File Delivery service application.

All functions in this chapter, i.e., request calls and notifications, are part of one of the following classes:

- IMSDCAppManager() (see Section 3.1)
- IMSDCAppManagerEventListener() (see Section 3.1)
- IMSDCFileDeliveryController() (see Section 3.3)
- IMSDCFileDeliveryControllerEventListener() (see Section 3.3)
- IMSDCFileDeliveryModel() (see Section 3.3)

5.1 Overview

Figure 5-1 and Figure 5-2 show a typical call flow sequence of an application that supports File Delivery service. Subsequent sections include the call flow sequences for individual functions and other scenarios.

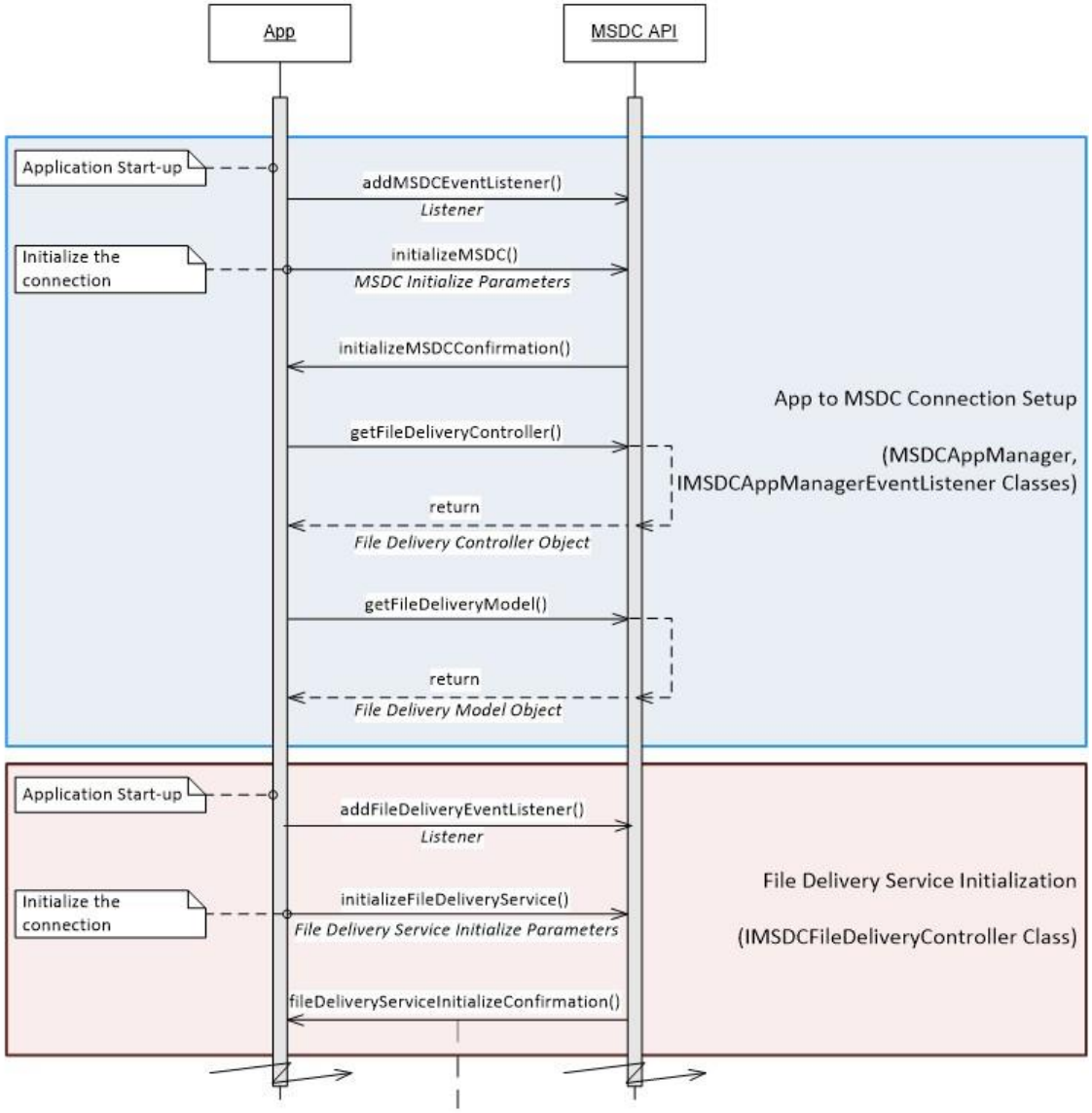


Figure 5-1 File Delivery service app call flow (1 of 2)

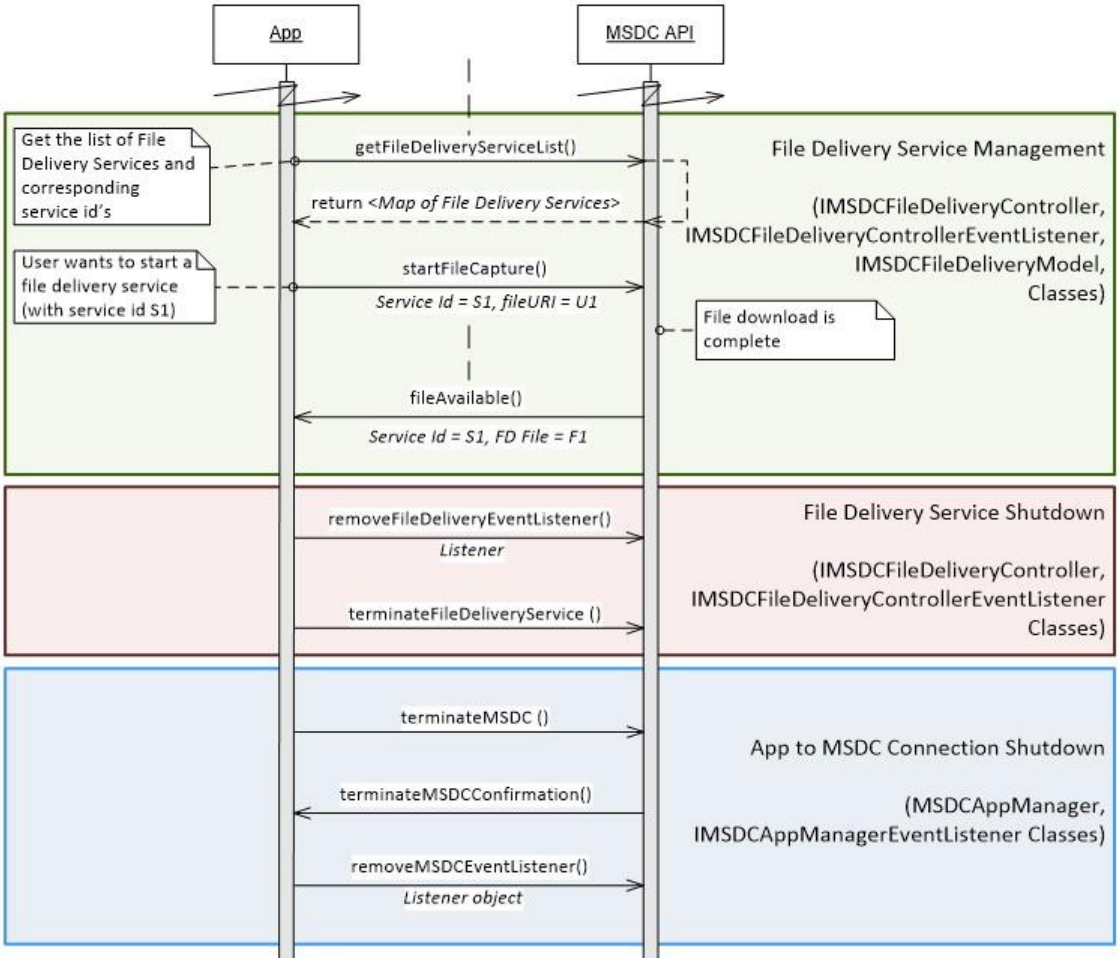


Figure 5-2 File Delivery service app call flow (2 of 2)

5.2 App-to-MSDC connection setup

The app starts here to begin communication with the MSDC.

5.2.1 Add MSDC Manager module event listener

See Section 4.2.1.

5.2.2 MSDC Manager module connection initialization

See Section 4.2.2.

5.2.3 Get the File Delivery module Controller and Model instances

5.2.3.1 Interface functions

```
IMSDCFileDeliveryController getFileDeliveryController()
IMSDCFileDeliveryModel getFileDeliveryModel()
```

5.2.3.2 Description

To send requests to the File Delivery module of the MSDC, the app should use `getFileDeliveryController()` and `getFileDeliveryModel()` calls to get the File Delivery module Controller and Model instances, respectively.

For code examples, see Section A.2.3.

5.2.3.3 Call flows

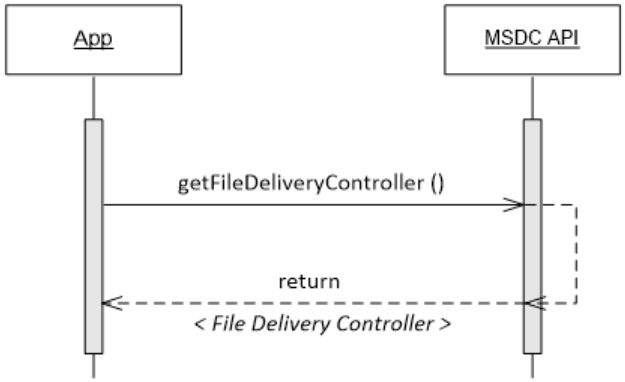


Figure 5-3 MSDC – Gets File Delivery module Controller instance

Figure 5-4 MSDC – Gets File Delivery module Model instance

5.3 File Delivery module initialization

5.3.1 Add File Delivery module event listener

5.3.1.1 Interface functions

```
void addFileDeliveryEventListener (IMSDCFileDeliveryControllerEventListener
    listener);
```

5.3.1.2 Prerequisites

Get the File Delivery module Controller and Model instances

5.3.1.3 Description

To get the events related to the File Delivery module from the MSDC, the app must add the event listener by using `addStreamingEventListener()`.

For code examples, see Section A.2.2.

5.3.1.4 Call flows

Figure 5-5 shows the call flow for adding a Streaming module event listener.

Figure 5-5 File Delivery – Event listener registration

5.3.2 File Delivery module connection initialization

5.3.2.1 Interface functions

```
void initializeFileDeliveryService (FileDeliveryInitParams params);
void fileDeliveryServiceConfirmation();
void fileDeliveryServiceError (int errorCode,
                               String message,
                               Integer serviceId);
```

5.3.2.2 Prerequisites

- MSDC Manager module connection initialization
- Add File Delivery module event listener

5.3.2.3 Description

After the File Delivery module event listener has been added, the app must initialize the connection to the File Delivery module with `initializeFileDeliveryService()`. Through `initializeFileDeliveryService()`, the app provides the following:

- Registration Time To Live (Reg TTL) – This timeout defines the length of time that the app wants the MSDC to retain the app information, e.g., App ID and open File capture requests, even after the app terminates the connection (see Section 5.6.1).

Holding on to the app registration means that the MSDC continues to download service updates for the service classes that the app requested and continues any pending or scheduled file downloads. The network operation typically defines a maximum value of registration. Characteristically, the registration hold timeout can have the following values:

  - Timeout ≤ 0 – No timer is set.
  - 0 ≤ Timeout ≤ Maximum value of timeout – Timer is set to a positive value.
  - Timeout ≥ Maximum value of timeout – Timer is set to maximum value of timeout.

- Cancel capture history – If set to TRUE, the app wants the MSDC to clear all file captures initiated during a previous session.
- Service Class names – A list of service classes that the app is interested in; the MSDC can give app information and data specifically for those services that belong to this set of service classes.
- Storage location – The location where the app is expected to get the downloaded file. This location must be an absolute path. If no location is set, the MSDC uses the emulated external storage to download the files, and the same location is passed to the app in the `fileAvailable()` notification. This option allows the app to download and store large files in the removable storage (SD card). This feature is especially useful for the devices with limited emulated storage (low end devices).
- Enable copy of downloaded files – If set to TRUE, the downloaded file gets copied to the internal location of the UI application even if it needs to be fetched from the HTTP server. The `fileAvailable()` notification indicates the file location after copy, unless the copy or fetching from the HTTP server fails. In that case, the `fileAvailable()` notification has the original location of the downloaded file.
- Enable Wakeup notification – If set to TRUE, the MSDC would broadcast the wakeup notification when the background file download is complete. The application receives a `fileAvailable()` notification when it is running; otherwise, it broadcasts a wakeup notification.
  - To receive this wakeup call, the app must register the receiver to listen for the action `<UI package name>.WAKE_UP_CALL` in AndroidManifest.xml:

```
<receiver
 android:name=<Name of receiver>
 android:enabled="true" >
<intent-filter>
 <action android:name="<UI␣package␣name>.WAKE_UP_CALL" />
</intent-filter>
</receiver>
```

If the MSDC API accepts the request and if the connection initialization is successful, the MSDC responds with `fileDeliveryServiceInitializeConfirmation`.

For code examples, see Sections A.2.2 and A.4.2.

5.3.2.4 Call flows

5.3.2.4.1 Connection initialization succeeds

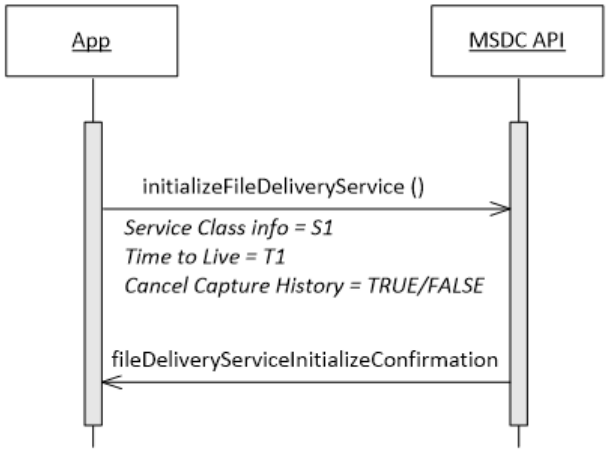


Figure 5-6 File Delivery – Connection initialization succeeds

5.3.2.4.2 Connection initialization fails

If a File Delivery module connection initialization fails, the MSDC API responds with `fileDeliveryServiceError()` and the error code `ERROR_FD_UNABLE_TO_INITIALIZE`.

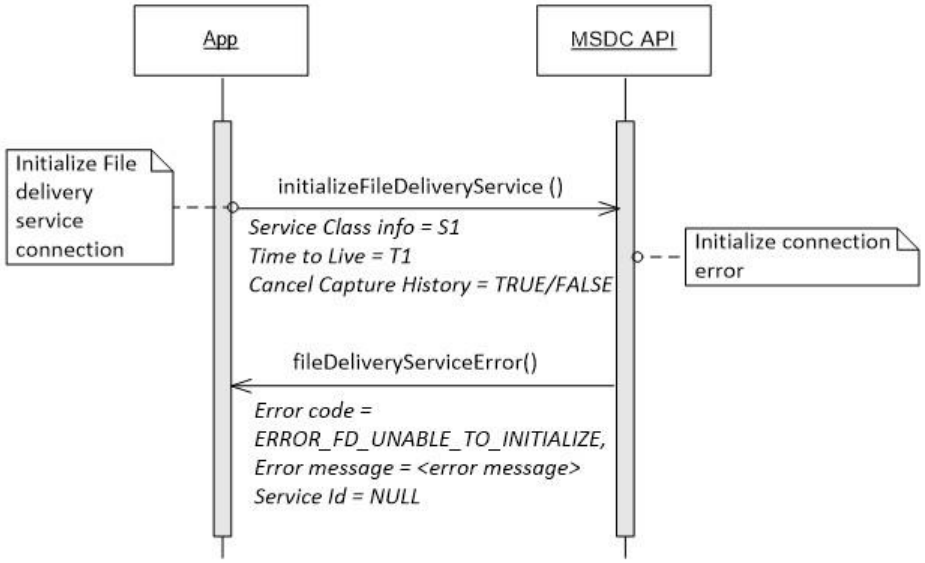


Figure 5-7 File Delivery – Connection initialization fails

5.3.2.4.3 Service class initialization fails

If a service class initialization fails, the MSDC API responds with `fileDeliveryServiceError()` and the error code `ERROR_FD_SERVICE_CLASS_INITIALIZATION_FAILED`.

A typical reason for service class initialization failure is that another application is already using this service class.

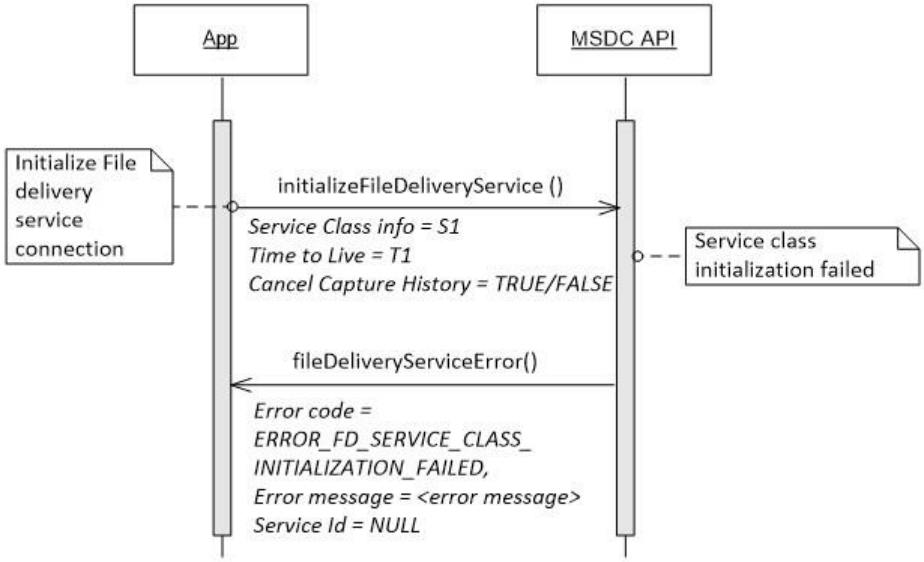


Figure 5-8 File Delivery – Service class initialization fails

5.4 File Delivery service management

5.4.1 Start a file capture

5.4.1.1 Interface functions

```
void startFileCapture (int serviceId);
void startFileCapture (int serviceId,
                       String fileURI);
public void startFileCapture(FileCaptureParams params);
void fileAvailable (int serviceId,
                    FDFile file);
void fileDeliveryServiceError (int errorCode,
                               String message,
                               Integer serviceId);
```

5.4.1.2 Prerequisites

File Delivery module connection initialization

5.4.1.3 Description

To start a file capture, the app should send a `startFileCapture()` request to the MSDC. The app provides a Service ID to the MSDC API in this request. The Service ID is the ID of the File Delivery service for which the app is requesting to capture the file. The app gets this ID from `ServiceInfo.serviceHandle`.

The app may also provide the following:

- Enable file download progress report – When set to TRUE, the UI application receives the notification regarding file download progress.
- File URI – The unique path identifier of the file the app wants. For example: `http://file-uri.com/abc/file1.txt`.

  If the app does not specify the file name in the path, e.g., `http://file-uri.com/abc/`, the MSDC tries to capture all the files under this path.

  If the user sets the prefix `/URI` to `/CNN`, the MSDC captures all the files that start with that prefix, such as `/CNN/s1/file1.xml` on the selected service, but it does not match `/XYZ/CNN/file2.xml`.

  – You can specify the prefix URI with or without the forward slash (/).

  – The URI path should contain the complete folder or name. For example, if you specify the prefix URI as `/CN`, the MSDC does not capture files under `/CNN`. Instead, it looks for files under `/CN` which may not exist.

  – The MSDC captures all files when no file URI is specified in the request.

  – Example: User sets prefix as `/CNN/s1`

    ◦ MSDC would capture these files:

      · /CNN/s1/s2/1.txt

      · /CNN/s1/s3/s4/s5/1.txt

      · /CNN/s1/1.txt

    ◦ MSDC would not capture these files:

      · /CNN/s2/1.txt

      · /CNN/1.txt

      · /CNN/s1/s2/1.txt

If file capture is successful, the MSDC responds with `fileAvailable` for each captured file and provides the following information:

- Service ID – The ID of the File Delivery service for which the file was captured.
- File – An `FDfile` object with all the required information on the captured file. For more information on `FDfile`, see Section 9.22.

Once the MSDC sends the `fileAvailable` notification to the app, the file belongs to the app. The app is then responsible for the file and actions such as file deletion.

After MSDC downloads a file, it does not download it again if the file (with the same file name and MDS)

reappears. An exception to this is if the file is deleted by `deleteFile()`, `deleteAllCapturedFiles()` (see Section 5.4.4) or the MSDC itself.

Recommendation: Once the app gets the `fileAvailable` notification, it is recommended that the app creates a copy of the file. If the file remains too long at the location specified by the `fileAvailable` notification, the MSDC may delete the file. The UI application can also enable the copy of downloaded files during `initializeFileDeliveryService` and set storage location (optional).

5.4.1.4 Call flows

5.4.1.4.1 Start file capture

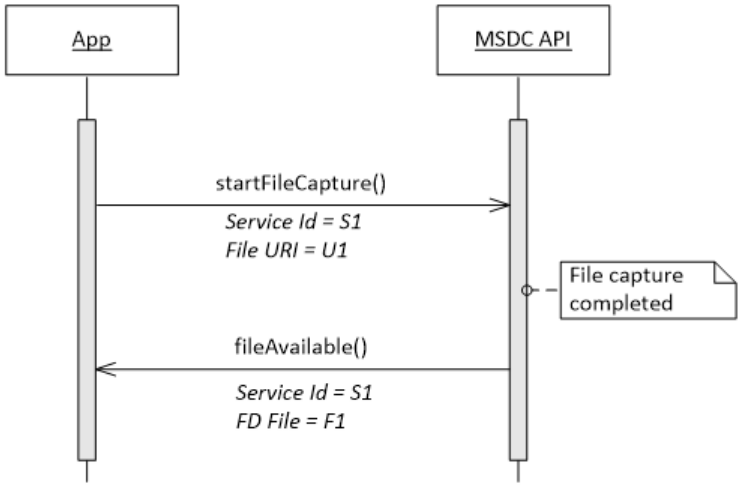


Figure 5-9 File Delivery – Start file capture

5.4.1.4.2 Start file capture fails

If the requested file capture is for a service ID that is already being used by another app, the MSDC API responds with a `fileDeliveryServiceError()` notification and the error code `ERROR_FD_SERVICE_ALREADY_USED_BY_ANOTHER_APP`.

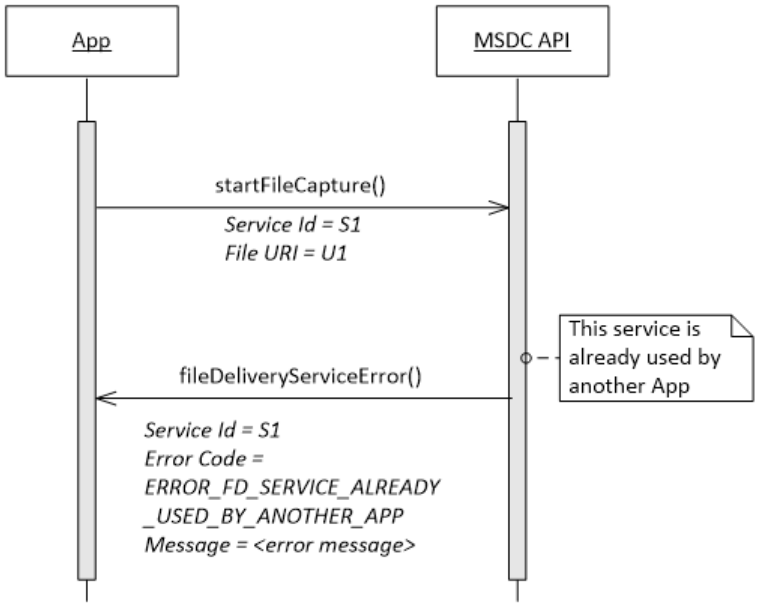


Figure 5-10 File Delivery – Start file capture fails, service used by another app

5.4.2 Stop a file capture

5.4.2.1 Interface functions

```
void startFileCapture (int serviceId);
void startFileCapture (int serviceId,
              String fileURI);
public void startFileCapture(FileCaptureParams params);
void stopFileCapture (int serviceId,
              String fileURI);
void fileDeliveryServiceError(int serviceId,
                              int errorCode,
                              String message
void fileDownloadFailure(int serviceHandle,
                         String uri);
```

5.4.2.2 Prerequisites

File Delivery module connection initialization

5.4.2.3 Description

If the app wants to stop a file capture that was started earlier, it should use the `stopFileCapture()`.

5.4.2.4 Call flows

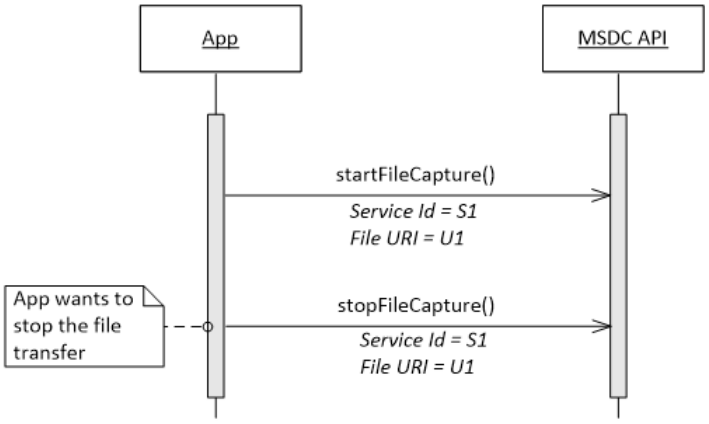


Figure 5-11 File Delivery – Stop file capture for a single file

5.4.3 Set storage location

5.4.3.1 Interface functions

```
public void setStorageLocation(String path);
```

5.4.3.2 Prerequisites

File Delivery module connection initialization

5.4.3.3 Description

If the app wants to change the file download location, it can use `setStorageLocation()`. The path must be an absolute path. This function allows the app to change the download location if it gets an `insufficientStorage()` or `inaccessibleLocation()` notification.

5.4.3.4 Call flows

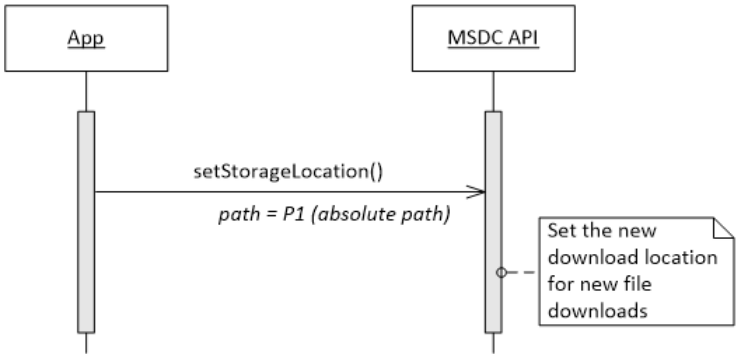


Figure 5-12 File Delivery – Set storage location

5.4.4 Delete a captured file

5.4.4.1 Interface functions

```
int deleteFile (int serviceId,
       String fileURI);
int deleteAllCapturedFiles (int serviceId);
void fileDeliveryServiceError(int serviceId,
                     int errorCode,
                     String message
```

5.4.4.2 Prerequisites

File Delivery module connection initialization

5.4.4.3 Description

If the app wants to delete a captured file from the device, it can use `deleteFile()`. To delete all captured files, the app can use `deleteAllCapturedFiles()`.

These functions delete the files from the device and clear the MSDC history related to the files. The next time the file shows up, the MSDC downloads it. However, if the app deletes the file by directly accessing it, the MSDC will not be aware of it and does not download the same file again.

For code examples, see Section A.4.3.

5.4.4.4 Call flows

5.4.4.4.1 Delete a captured file

To delete a captured file for a particular service, the app can use `deleteFile()`.

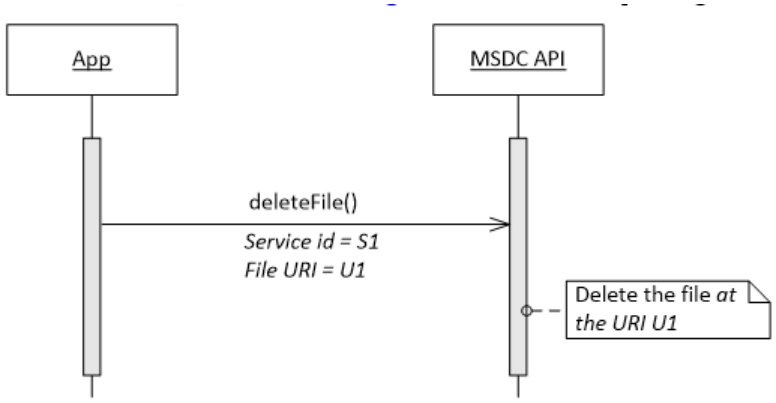


Figure 5-13 File Delivery – Delete a captured file

5.4.4.4.2 Delete all captured files

To delete all captured files for a particular service, the app can use `deleteAllCapturedFiles()`.

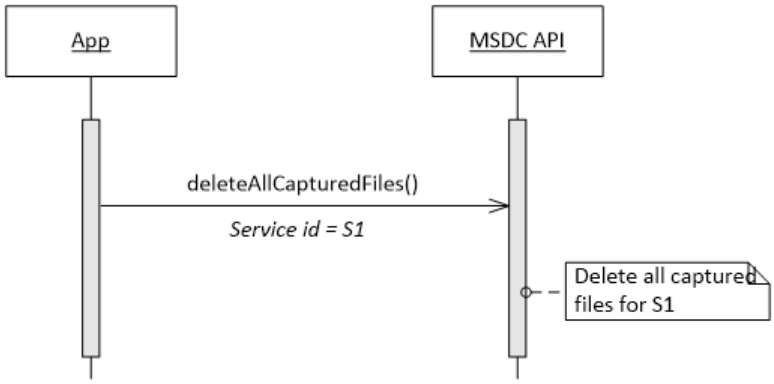


Figure 5-14 File Delivery – Delete all captured files

5.4.4.4.3 Delete a file fails

If the app requests the MSDC to delete a file and it is unable to do so, the MSDC API responds with a `fileDeliveryServiceError()` notification and the error code `ERROR_FD_UNABLE_TO_DELETE_FILE`.

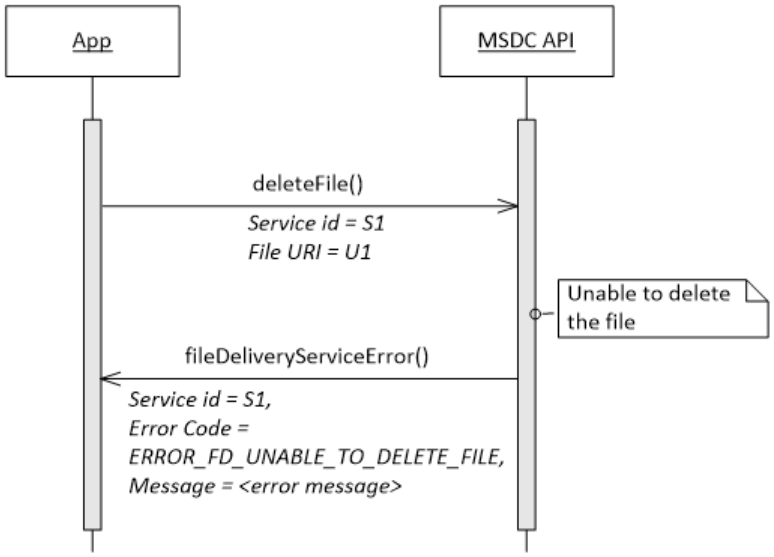


Figure 5-15 File Delivery – Delete a file fails

5.4.4.4.4 Delete all captured files fails

If the app requests the MSDC to delete all captured files for a service and it is unable to do so, the MSDC API responds with a `fileDeliveryServiceError()` notification and the error code `ERROR_FD_UNABLE_TO_DELETE_ALL_FILES`.

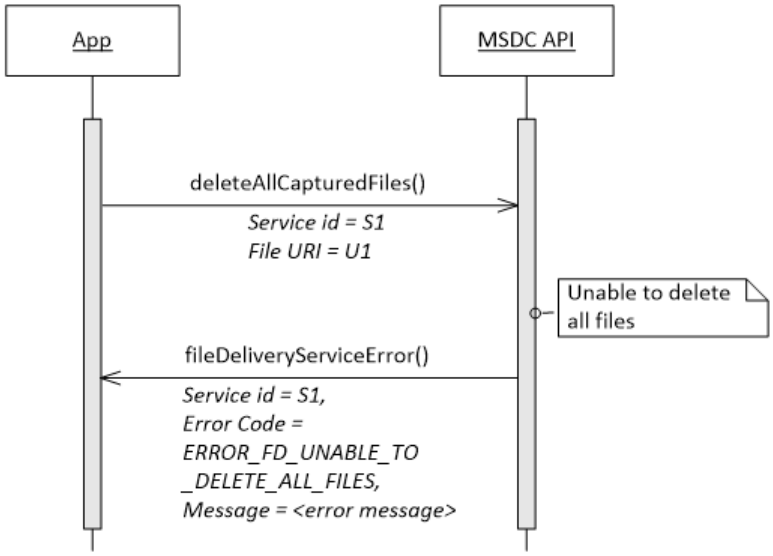


Figure 5-16 File Delivery – Delete all files fails

5.4.5 Other information notifications

5.4.5.1 Service list update

5.4.5.1.1 Interface functions

```
void fileDeliveryServiceListUpdate();
```

5.4.5.1.2 Prerequisites

File Delivery module connection initialization

5.4.5.1.3 Description

If there is an update on the File Delivery service list, the MSDC notifies the app with `fileDeliveryServiceListUpdate()`, to indicate its availability. The app can then get the updated list using `getFileDeliveryServiceList()` (see Section 5.4.6.1).

For code examples, see Section A.4.2.

Note: The MSDC sends a `fileDeliveryServiceListUpdate()` to the app if there is any change in the camped group or the service groups. For more information on camped and service groups, see Section 5.4.6.4.

5.4.5.1.4 Call flows

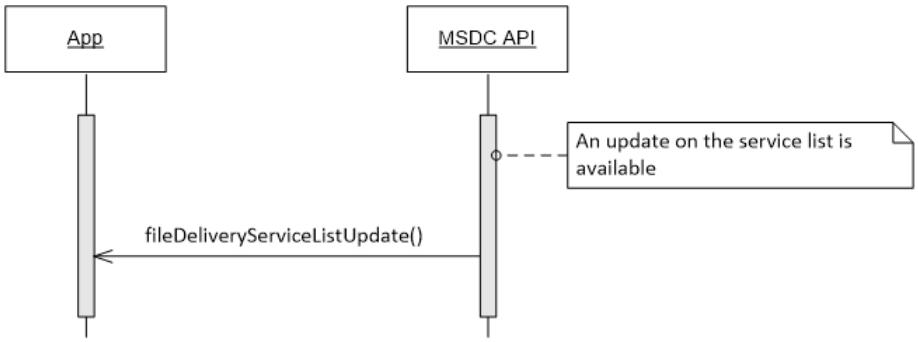


Figure 5-17 File Delivery service update notification

5.4.5.2 File download failure

5.4.5.2.1 Interface functions

```
void fileDownloadFailure(int serviceHandle,
                         String uri);
```

5.4.5.2.2 Prerequisites

- File Delivery module connection initialization
- Start a file capture

5.4.5.2.3 Description

If the MSDC fails to download the requested file after trying all possible options, it notifies the app with `fileDownloadFailure()` to indicate the failure.

5.4.5.2.4 Call flows

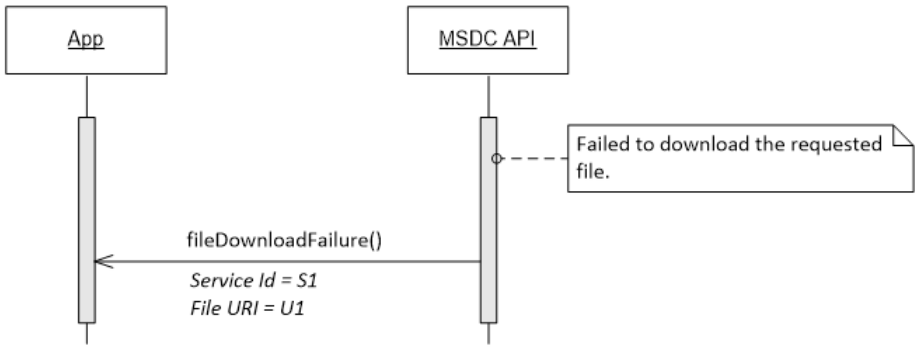


Figure 5-18 File Delivery – File download failure notification

5.4.5.3 Active file download state list update

5.4.5.3.1 Interface functions

```
public void activeFileDownloadStateInfoListUpdate();
public Map<String, Enum<ActiveFileDownloadState>>
    getActiveFileDownloadInfoList(int serviceHandle, String fileUri);
```

5.4.5.3.2 Prerequisites

- File Delivery module connection initialization
- Start a file capture

5.4.5.3.3 Description

The MSDC provides the state of active file downloads to show whether the files are in progress. Active files are the files which the app has requested to download by calling `startFileCapture()`.

The MSDC notifies the app with `activeFileDownloadStateInfoListUpdate()` to indicate a list change. The app can then get the updated list using `getActiveFileDownloadInfoList()` (see Section 5.4.6.7).

The active download list can be updated for the following reasons:

- A new file download gets started within the broadcast schedule.
- A file download is completed successfully. In this case, the app also receives a `fileAvailable()` notification.
- Failure to download a file. In this case, the app also receives a `fileDownloadFailure notification()` notification.
- The schedule for the active file is canceled by the Broadcast Multicast Service Center (BM-SC).

5.4.5.3.4 Call flows

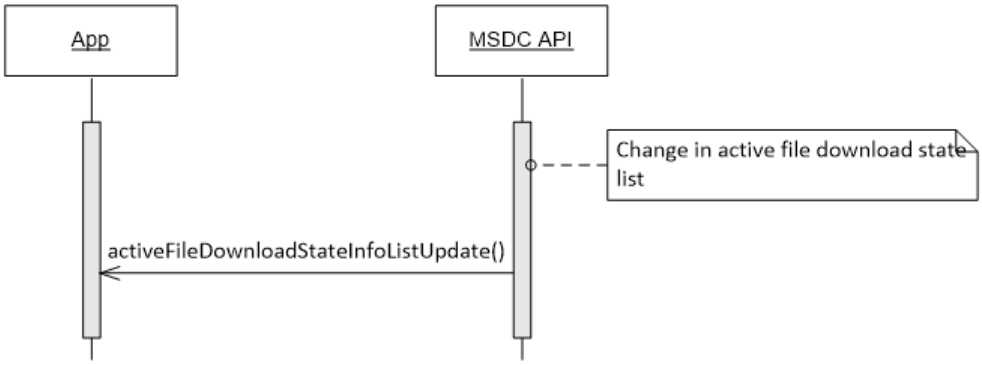


Figure 5-19 File Delivery – Active file download state list update notification

5.4.5.4 File list available

5.4.5.4.1 Interface functions

```
public void fileListAvailable(int serviceHandle);
void initializeFileDeliveryService (FileDeliveryInitParams params);
```

5.4.5.4.2 Prerequisites

File Delivery module connection initialization

5.4.5.4.3 Description

If the app wants to continue monitoring File Delivery services after exiting, the app receives a `fileListAvailable(0` notification after relaunch (if any file was downloaded while it was gone). The app can then use `getAvailableFileList()` to get the list of downloaded files.

5.4.5.4.4 Call flows

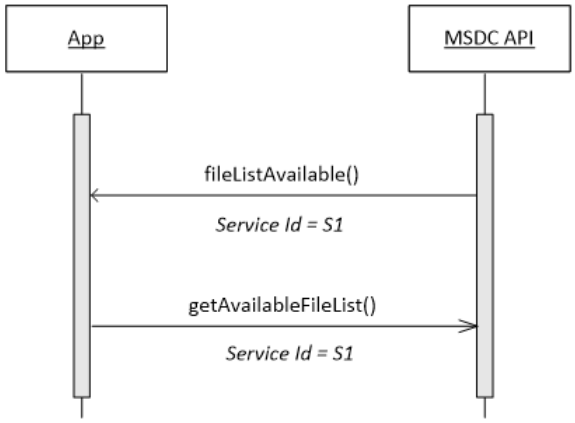


Figure 5-20 File Delivery file list available notification

5.4.5.5 File download progress

5.4.5.5.1 Interface functions

```
public void fileDownloadProgress (int serviceHandle, String fileUri, Long
    receivedBytes, Long receivedBytesTarget,Long decodedBytes, Long
    decodedBytesTarget, int receptionType);

void fileDownloadProgressSuspended(int serviceHandle , String uri);
```

5.4.5.5.2 Prerequisites

- File Delivery module connection initialization
- Start a file capture

5.4.5.5.3 Description

If the UI application indicates that it is interested in receiving a file download progress report when calling `startFileCapture()`, it receives a File Download Progress notification while the file is being downloaded. There are two phases for a complete file download:

- Collection – The MSDC middleware collects the symbols required to decode the file successfully.
- Decode – The MSDC middleware decodes the collected symbols and constructs the file. When decoding is complete, the `fileAvailable()` notification is sent to the application.

The UI application receives a file download progress notification during both phases. The following information is included in the notification:

- Service ID – The ID of the File Delivery service for which the file was captured.
- File URI – The unique path identifier of the downloaded file.
- Received bytes – Number of bytes received so far for this file (Collection phase).
- Received bytes target – Total number of bytes needed to be received before starting to decode the file.
- Decoded bytes – Number of bytes decoded so far for this file. When it is non-zero, it means the Decode phase has started.
- Decoded bytes target – Total number of bytes for decoding the file.

The UI application can use the above information to calculate the download progress percentage in each phase.

5.4.5.6 File download progress suspended

5.4.5.6.1 Interface functions

```
public void fileDownloadProgress (int serviceHandle, String fileUri, Long
    receivedBytes, Long receivedBytesTarget,Long decodedBytes, Long
    decodedBytesTarget, int receptionType);

void fileDownloadProgressSuspended(int serviceHandle , String uri);
```

5.4.5.6.2 Prerequisites

- File Delivery module connection initialization
- Start a file capture

5.4.5.6.3 Description

If file download progress is stopped for any reason, the application receives a File Download Progress Suspended notification.

The following information is included in the notification:

- Service ID – The ID of the File Delivery service for which the file was captured.
- File URI – The unique path identifier of the downloaded file.

5.4.5.7 Insufficient storage

5.4.5.7.1 Interface functions

```
public void insufficientStorage(int serviceHandle ,
                                String uri,
                                String path,
                                Long totalStorageNeeded);
```

5.4.5.7.2 Prerequisites

- File Delivery module connection initialization
- Start a file capture

5.4.5.7.3 Description

If the MSDC determined there is not enough space available to download the requested files in the specified location, it notifies the app with `insufficientStorage()`.

5.4.5.7.4 Call flows

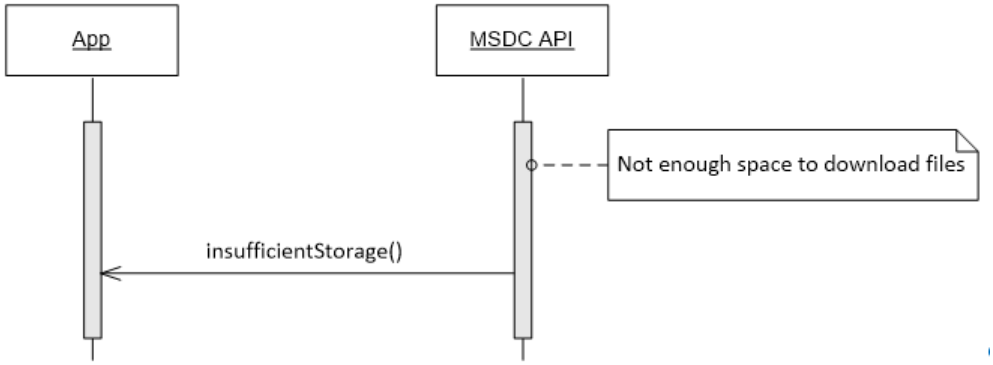


Figure 5-21 File Delivery insufficient storage notification

5.4.5.8 Inaccessible location

5.4.5.8.1 Interface functions

```
public void insufficientStorage(int serviceHandle ,
                                String uri,
                                String path,
                                Long totalStorageNeeded);
```

5.4.5.8.2 Prerequisites

File Delivery module connection initialization

5.4.5.8.3 Description

If the MSDC fails to access the specified location to store the downloaded files, it notifies the app with `inaccessibleLocation()`.

5.4.5.8.4 Call flows

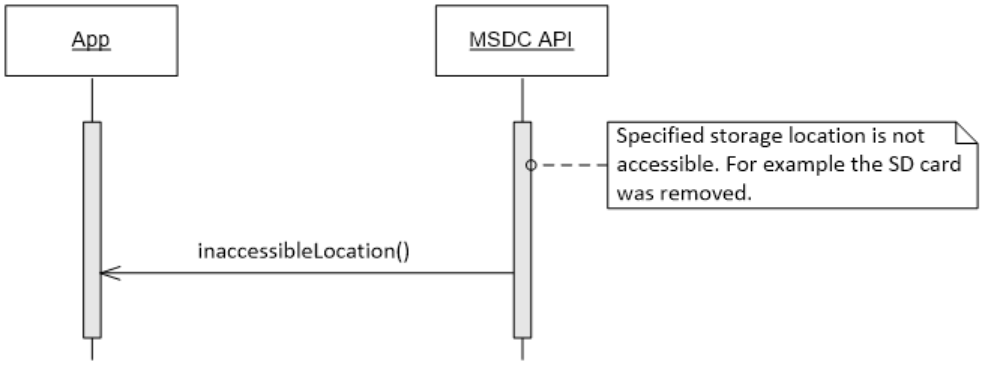


Figure 5-22 File Delivery inaccessible location notification

5.4.5.9 Other error notifications

5.4.5.9.1 Interface functions

```
void fileDeliveryServiceError(int errorCode,
                     String message
                     int service Id);
```

5.4.5.9.2 Prerequisites

Add File Delivery module event listener

5.4.5.9.3 Description

If the MSDC API wants to notify the app of any File Delivery service error, it uses `fileDeliveryServiceError()`. For more information on the different types of error notifications, see Section 9.4.

For code examples, see Section A.4.2.

5.4.5.9.4 Call flows

If the app tries to request a File Delivery service operation, the MSDC API responds with `fileDeliveryServiceError()` and the error code `ERROR_FD_INVALID_SERVICE_ID`.

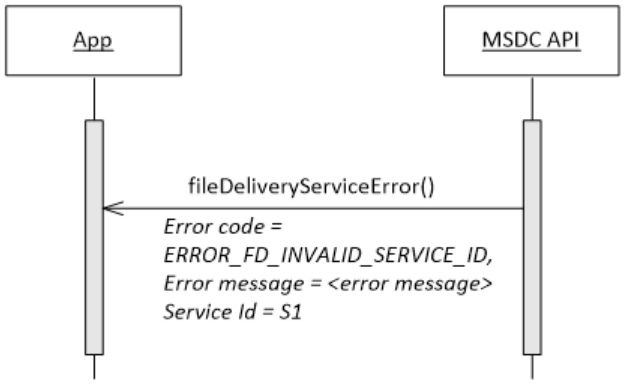


Figure 5-23 File Delivery – Invalid Service ID error notification

If a File Delivery service is not available, the MSDC API responds to the app with `fileDeliveryServiceError()` and the error code `ERROR_FD_SERVICE_UNAVAILABLE` along with the service ID of the affected service.

However, if all File Delivery services are unavailable, the service ID would be NULL.

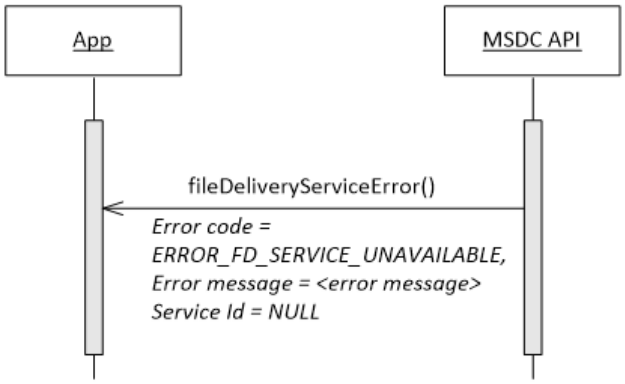


Figure 5-24 Error notification when File Delivery module of MSDC is unavailable

The error notification with error code `ERROR_FD_SERVICE_RESET` tells the app that the service related information (like service ID) has been reset and the app should fetch the new service information using `getFileDeliveryServiceList()` (see Section 5.4.6.1).

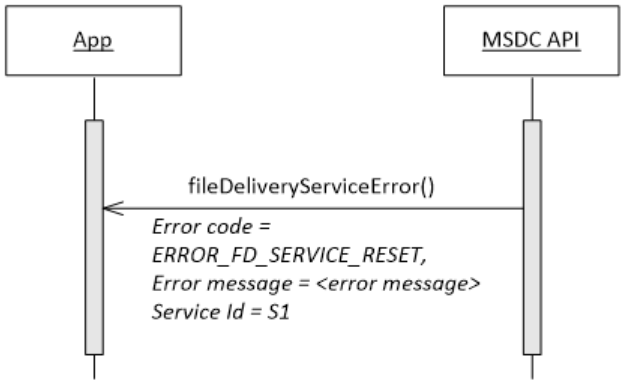


Figure 5-25 Error notification – File Delivery service-related data has been reset

5.4.5.10 Warning notifications

5.4.5.10.1 Interface functions

```
void fileDeliveryServiceWarning(int warningCode,
          String message,
          Integer serviceId);
```

5.4.5.10.2 Prerequisites

Add File Delivery module event listener

5.4.5.10.3 Description

If the MSDC API wants to notify the app of any File Delivery service warnings, it uses the overloaded `fileDeliveryServiceWarning()` notification.

For more information on the different types of error notifications, see Section 9.4.

5.4.5.10.4 Call flows

When trying to start a file download (e.g., S1), the app may get an error notification with the warning code `WARNING_FD_FREQUENCY_CONFLICT`. This means that another app is already using a File Delivery service (e.g., S2) which falls in a different service group than the one which has S1 in it (see Section 5.4.6.4).

This is a warning because the MSDC continues to try starting the file download if the frequency conflict is resolved. If it is not resolved, the MSDC is unable to start the download.

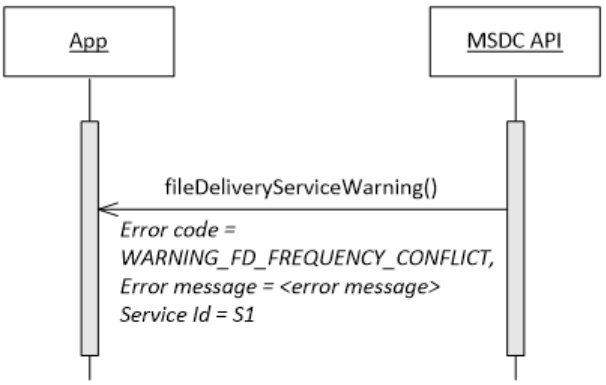


Figure 5-26 File Delivery – Warning notification when another service is active from a different service group

If File Delivery is stalled for any reason after a file capture is initiated by the app, the app gets a warning notification with warning code `WARNING_FD_STALLED`.

This is a warning because the MSDC continues to try downloading the file in case the service is available again. If the service is not available at a later time, the MSDC is unable to complete the download.

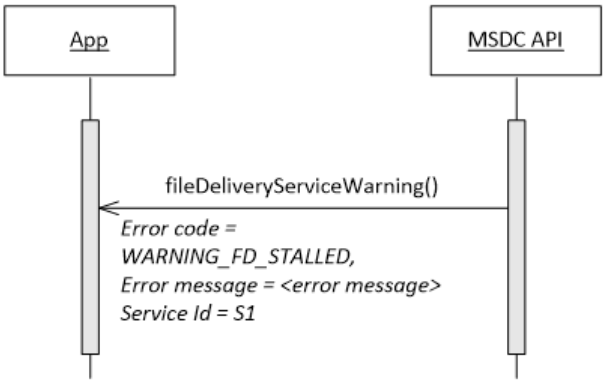


Figure 5-27 Warning notification when the File Delivery service is stalled

A warning notification with `WARNING_FD_STORAGE_LOCATION_COPY_FAILED` is sent to the UI application if the MSDC fails to copy the file to the specified storage location.

5.4.6 Information calls

5.4.6.1 Get service list

5.4.6.1.1 Interface functions

```
Map<Integer, FDService> getFileDeliveryServiceList ();
```

5.4.6.1.2 Prerequisites

File Delivery module connection initialization

5.4.6.1.3 Description

To get the list of available File Delivery services, the app should use `getFileDeliveryServiceList()`. The return value has the list of services and corresponding service IDs.

`getFileDeliveryServiceList()` provides the list of all services in all the service groups. For more information on service groups, see Section 5.4.6.4 .

For code examples, see Section A.4.1.

5.4.6.1.4 Call flows

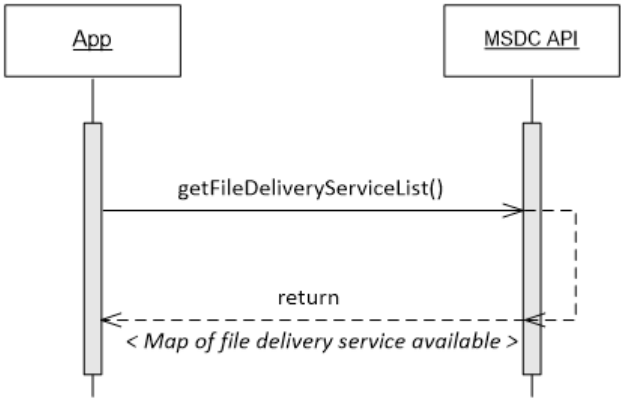


Figure 5-28 Get list of available File Delivery services

5.4.6.2 Get available file list

5.4.6.2.1 Interface functions

```
List<FDFile> getAvailableFileList (int serviceId);
```

5.4.6.2.2 Prerequisites

File Delivery module connection initialization

5.4.6.2.3 Description

To get the list of available files already downloaded by a File Delivery service, the app should use `getAvailableFileList()`. The return value has the list of available files.

5.4.6.2.4 Call flows

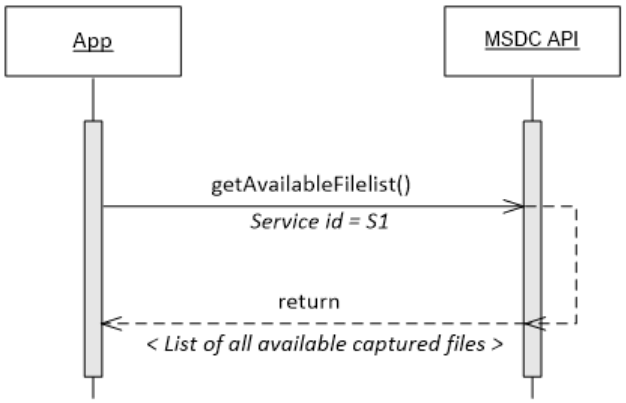


Figure 5-29 File Delivery – Get list of available files

5.4.6.3 Get running File Delivery services

5.4.6.3.1 Interface functions

```
List<Integer> getRunningFileDeliveryService ();
```

5.4.6.3.2 Prerequisites

File Delivery module connection initialization

5.4.6.3.3 Description

To get the list of File Delivery services that have active file downloads, the app should use `getRunningFileDeliveryService()`. The return value has the list of service IDs.

5.4.6.3.4 Call flows

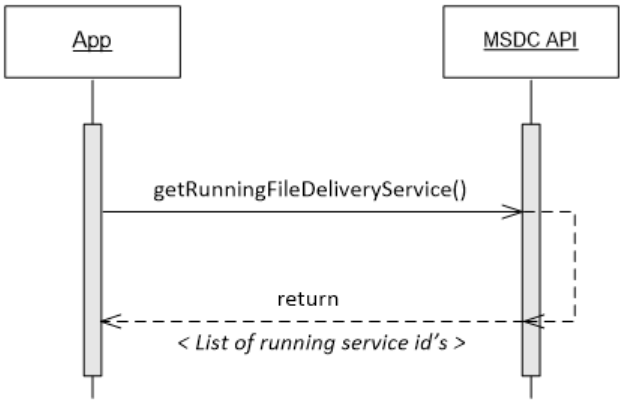


Figure 5-30 Get list of running File Delivery services

5.4.6.4 Get camped group

5.4.6.4.1 Interface functions

```
GroupItem getCampedGroup();
```

`getCampedGroup()` returns the *GroupItem* object which contains the following member variables:

```
String groupName;
List<Integer> serviceAreaIdList;
List<Integer> serviceHandleList;
```

The member variable *groupName* is the name of the currently camped network type. The *serviceHandleList* is the list of streaming services which are available/valid in the Service Areas listed in the *serviceAreaIdList* of that group.

5.4.6.4.2 Prerequisites

File Delivery module connection initialization

5.4.6.4.3 Description

The UE may be able to access services broadcast on multiple frequency carriers in a given geographical location even though it may be camped on a single frequency.

Services on the camped frequency are called the camped group. Similarly, the services on all the frequencies that the UE can access can be grouped per frequency into multiple service groups. Thus, a camped group is the same or a subset of all the service groups.

Service groups are a set of services the UE can concurrently access. The services in the camped group are immediately accessible without the UE having to switch/acquire another frequency carrier.

To get camped group information from the MSDC, the app should use `getCampedGroup()`. The return value defines the following:

- Group name
- Service Area ID list
- Service handle list

Note:

- If the app tries to access a service outside the camped group, an additional delay ensues.
- In a multiview UI of the app, services from different service groups cannot be displayed.

5.4.6.4.4 Call flows

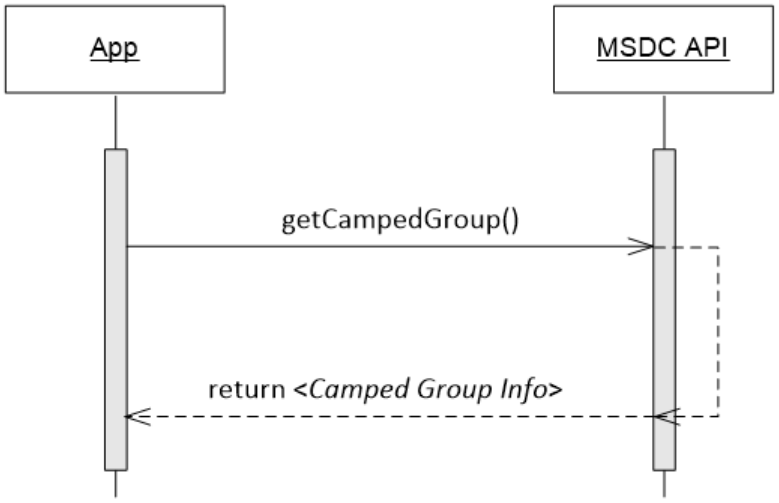


Figure 5-31 File Delivery – Get camped group information

5.4.6.5 Get File Delivery group list

5.4.6.5.1 Interface functions

```
List< GroupItem > getFileDeliveryGroupList ()
```

`getFileDeliveryGroupList()` returns the list of available groups (*GroupItem*) where File Delivery services are available. Each *GroupItem* class contains the following member variables:

```
String groupName;
List<Integer> serviceAreaIdList;
List<Integer> serviceHandleList;
```

The member variable *groupName* is the name of the currently camped network type. The *serviceHandleList* is the list of File Delivery services which are available/valid in the Service Areas listed in the *serviceAreaIdList* of that group.

5.4.6.5.2 Prerequisites

File Delivery module connection initialization

5.4.6.5.3 Description

To get the list of all service groups other than the camped group (see Section 5.4.6.4), the app should use `getFileDeliveryGroupList()`. The return value has the list of service groups other than the camped group.

5.4.6.5.4 Call flows

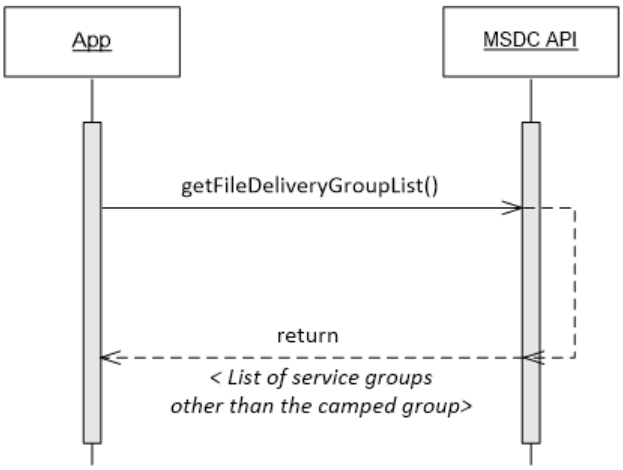


Figure 5-32 Get File Delivery group list

5.4.6.6 Get File Delivery service list by group

5.4.6.6.1 Interface functions

```
Map <Integer, FDService> getFileDeliveryServiceListByGroup
                                  (String groupName)
```

5.4.6.6.2 Prerequisites

File Delivery module connection initialization

5.4.6.6.3 Description

To get the list of all File Delivery services for a particular service group, the app should use `getFileDeliveryServiceListByGroup()`. For more information on service groups, see Section 5.4.6.4.

5.4.6.6.4 Call flows

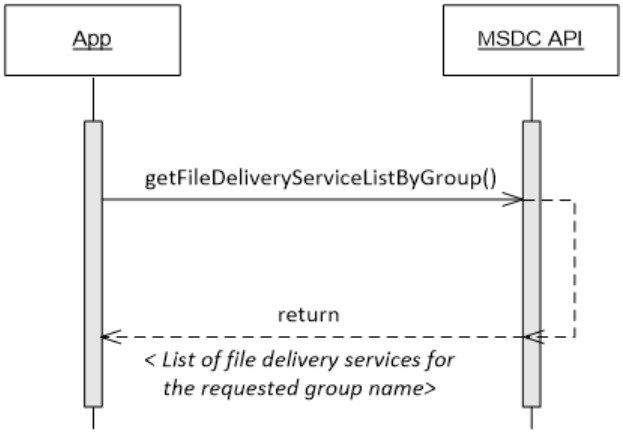


Figure 5-33 Get File Delivery service list by group

5.4.6.7 Get active file download state list

5.4.6.7.1 Interface functions

```
public Map<String, Enum<ActiveFileDownloadState>>
getActiveFileDownloadInfoList(int serviceHandle, String fileUri);
```

5.4.6.7.2 Prerequisites

File Delivery module connection initialization

5.4.6.7.3 Description

To get the list of active file download states, the app should use `getActiveFileDownloadInfoList()`. The return value has the file URI map and the corresponding state for the requested service.

If the app does not specify any file URI, the map contains the state for all active files for the requested service. If the app requests the state for a specific file URI, the return map is filtered based on the requested file URI and Service ID.

Example

The result of this function can be used to determine if a requested file is still in progress or if it has been canceled by BM-SC. Suppose the app starts capture on F1 (listed in the file URI list). Later, the app receives a service list update that F1 is no longer there. F1 can be removed from the file URI list in the service list update for the following reasons:

- F1 schedule expired
- F1 schedule was canceled by BM-SC

After F1 has been removed from the service list update, the app can check the state of F1 to ensure that the file download is still in progress and is pending for file repair.

5.4.6.7.4 Call flows

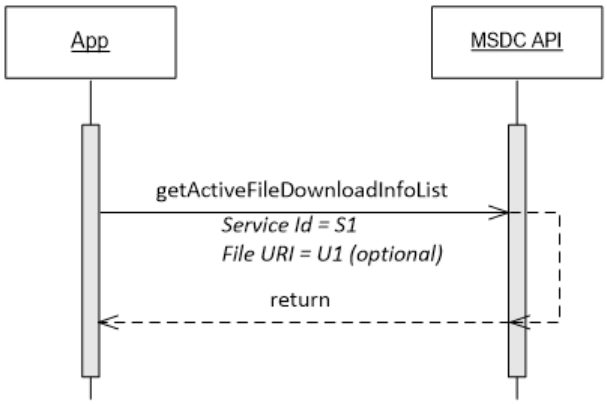


Figure 5-34 File Delivery – Get list of active file download states

5.5 MSDC Manager module connection management

See Section 4.4.8.6.

5.6 App-to-MSDC connection shutdown

This section defines the calls the app uses to shut down the connection with the MSDC.

5.6.1 Close File Delivery module connection

5.6.1.1 Interface functions

```
int terminateFileDeliveryService();
```

5.6.1.2 Prerequisites

File Delivery module connection initialization

5.6.1.3 Description

To close the connection to the File Delivery module of the MSDC, the app uses `terminateFileDeliveryService()`. After this call, the app is unable to communicate with the File Delivery module unless it reinitializes the connection (see Section 5.3.2).

In `terminateFileDeliveryService()`, the app can redefine the value of Registration Time to Live (see Section 5.3.2). If the app does not want to change the Registration Time to Live value defined by `initializeFileDeliveryService()`, it can use a NULL value for a timeout parameter in the `terminateFileDeliveryService()` call.

5.6.1.4 Call flows

5.6.1.4.1 Terminate connection and Registration Time to Live expires

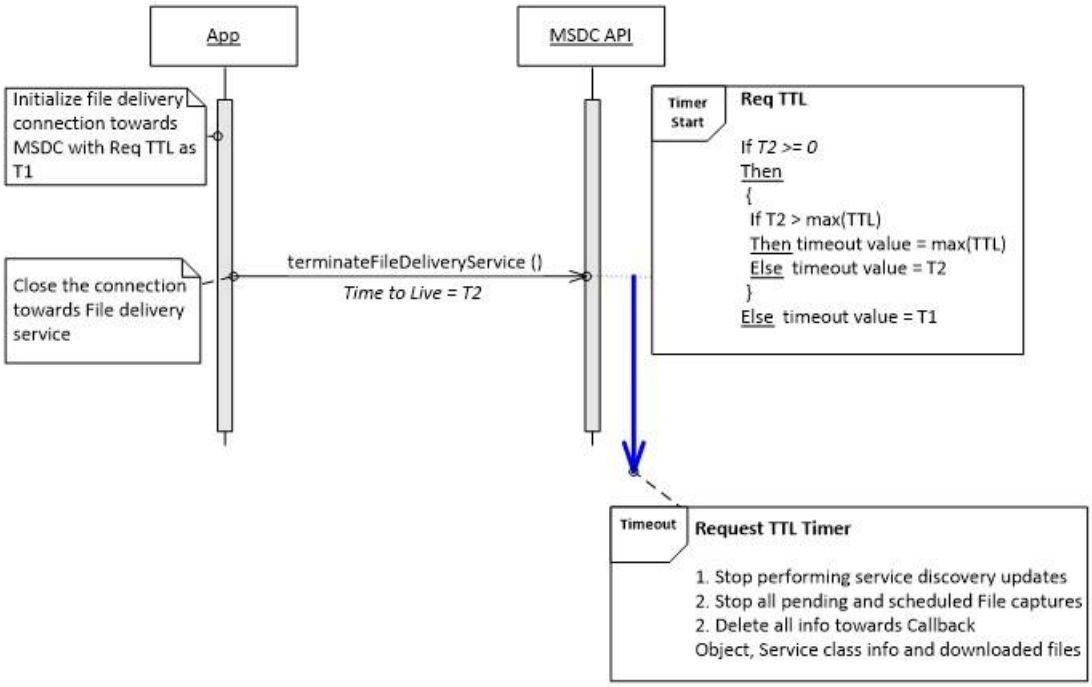


Figure 5-35 File Delivery – Terminate connection and Registration Time to Live expires

5.6.1.4.2 Terminate the connection and reinitialize before timeout

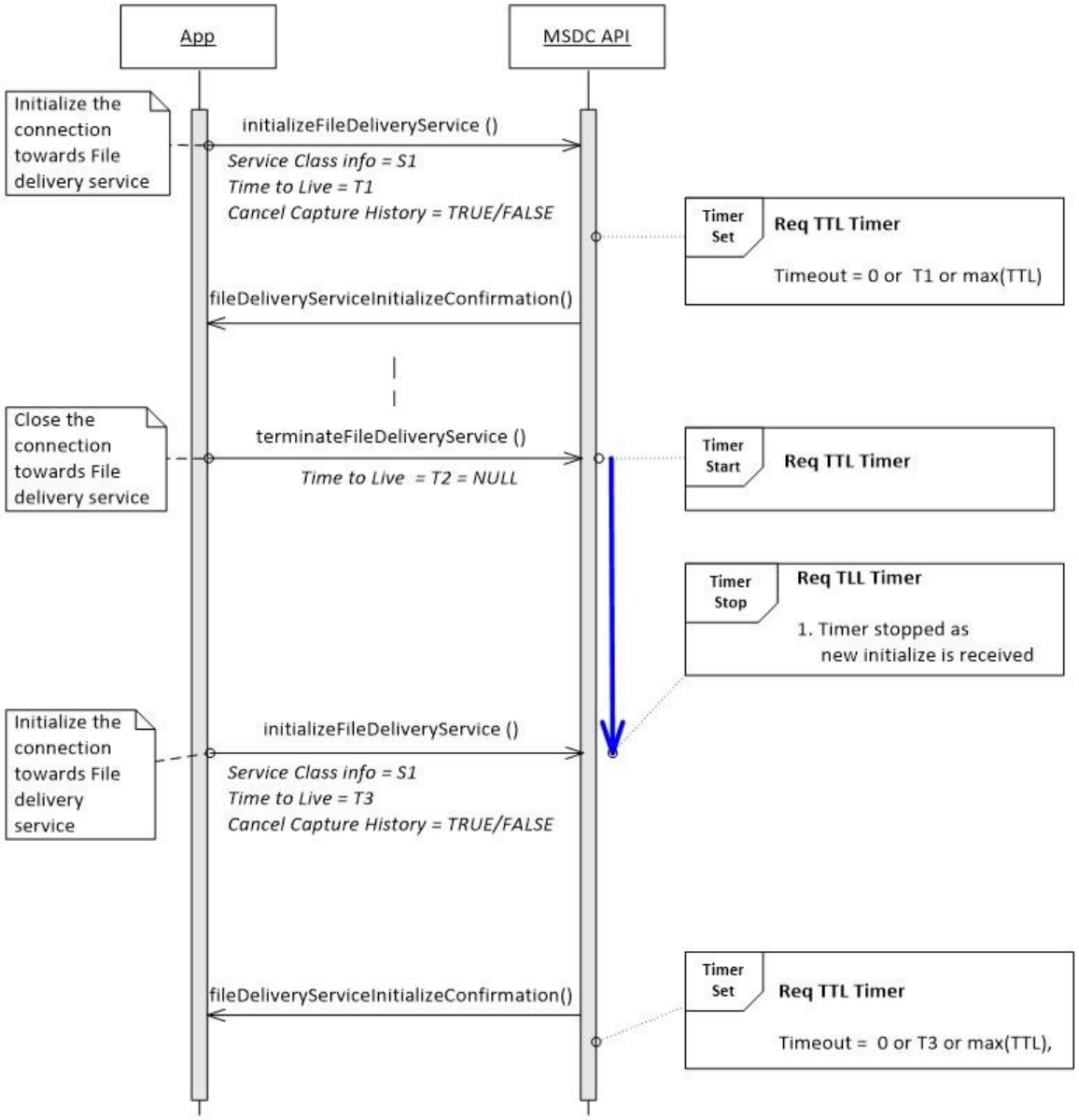


Figure 5-36 File Delivery – Terminate connection and reinitialize before Registration Time to Live timer expires

5.6.2 Remove File Delivery module event listener

5.6.2.1 Interface functions

```
void removeFileDeliveryEventListener (IMSDCFileDeliveryControllerEventListener
     listener)
```

5.6.2.2 Prerequisites

Add File Delivery module event listener

5.6.2.3 Description

To stop getting events from the File Delivery module of the MSDC, the app must remove the event listener that it added earlier by using `removeFileDeliveryEventListener()` (see Section 5.3.1).

5.6.2.4 Call flows

Figure 5-37 Remove File Delivery event listener

5.6.3 Close MSDC Manager module connection

See Section 4.6.3.

5.6.4 Remove MSDC Manager module event listener

See Section 4.6.4.

6 Network Module Management

This chapter describes the app-to-MSDC interface call flow sequences for an application interested in Network services.

All functions in this chapter, i.e., request calls and notifications, are part of one of the following classes:

- IMSDCAppManager() (see Section 3.1)
- IMSDCAppManagerEventListener() (see Section 3.1)
- IMSDCNetworkController() (see Section 3.4)
- IMSDCNetworkControllerEventListener() (see Section 3.4)
- IMSDCNetworkModel() (see Section 3.4)

6.1 App-to-MSDC connection setup

The app starts here to begin communication with the MSDC.

6.1.1 Add MSDC Manager module event listener

See Section 4.2.1.

6.1.2 MSDC Manager module connection initialization

See section 4.2.2.

6.1.3 Get the Network module Controller and Model instances

6.1.3.1 Interface functions

```
IMSDCNetworkController getNetworkController()
IMSDCNetworkModel getNetworkModel()
```

6.1.3.2 Description

To send requests to the Network module of the MSDC, the app should use `getNetworkController()` and `getNetworkModel()` calls to get the Network Controller and Model instances, respectively.

For code examples, see Section A.2.3.

6.1.3.3 Call flows

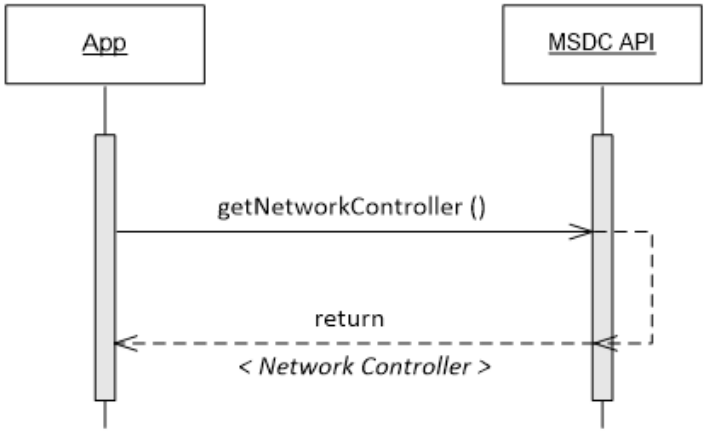


Figure 6-1 MSDC – Gets Network module Controller instance

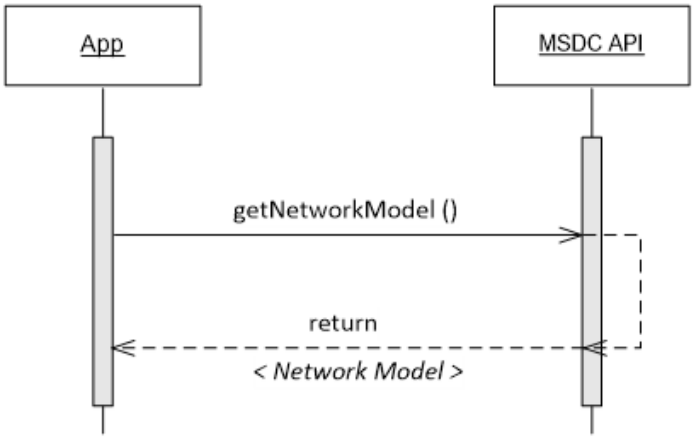


Figure 6-2 MSDC – Gets Network module Model instance

6.2 Network module initialization

6.2.1 Add Network module event listener

6.2.1.1 Interface functions

```
void addNetworkEventListener (NetworkServiceEventListener
                      listener);
```

6.2.1.2 Description

To get the events from the Network module of the MSDC API, the app must add the Network module event listener by using `addNetworkEventListener()`.

For code examples, see Section A.2.4.

6.2.1.3 Call flows

Figure 5-5 shows the call flow for adding a Network module event listener.

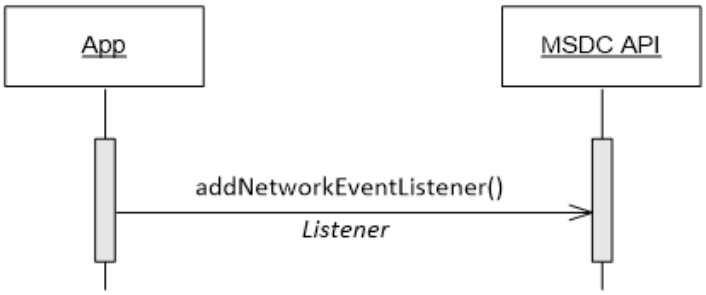


Figure 6-3 Network – Event listener registration

6.2.2 Network module connection initialization

6.2.2.1 Interface functions

```
void initializeNetworkService ();
void networkServiceConfirmation();
void networkServiceError (int errorCode,
                          String message);
```

6.2.2.2 Prerequisites

Add Network module event listener

6.2.2.3 Description

After the Network module event listener has been added, the app must initialize the connection to the Network module with `initializeNetworkService()`.

If the MSDC API accepts the request and if the connection initialization is successful, the MSDC responds with `networkServiceInitializeConfirmation`.

For code examples, see Section A.2.2 and A.5.1.

6.2.2.4 Call flows

6.2.2.4.1 Connection initialization succeeds

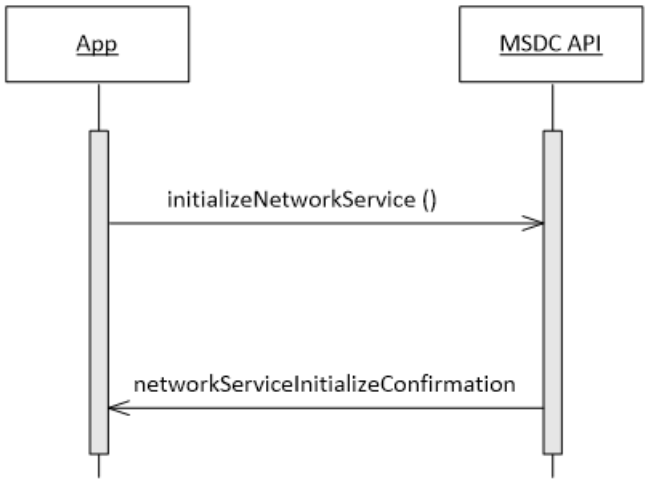


Figure 6-4 Network – Connection initialization succeeds

6.2.2.4.2 Connection initialization fails

If a Network module connection initialization fails, the MSDC API responds with `networkServiceError()` and the error code `ERROR_NW_UNABLE_TO_INITIALIZE`.

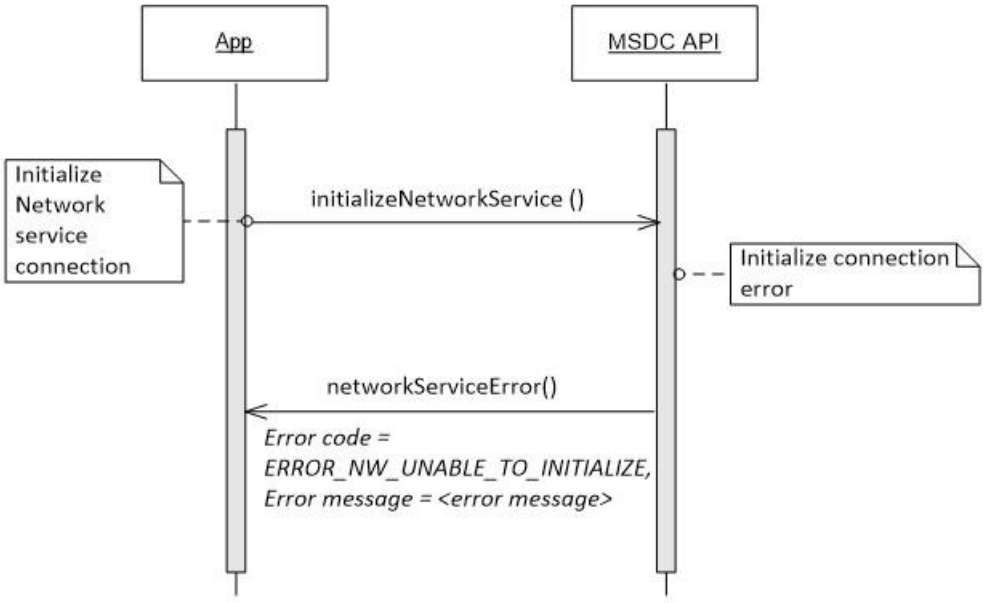


Figure 6-5 Network – Connection initialization fails

6.3 Notifications and feature opt-ins

6.3.1 Broadcast coverage notification

6.3.1.1 Interface functions

```
void broadcastCoverageNotification(int state);
int getBroadcastCoverage();
```

6.3.1.2 Prerequisites

Network module connection initialization

6.3.1.3 Description

If the device running the MSDC moves in or out of the LTE broadcast coverage area, the MSDC API notifies the app with a `broadcastCoverageNotification()` to indicate this change.

6.3.1.4 Call flows

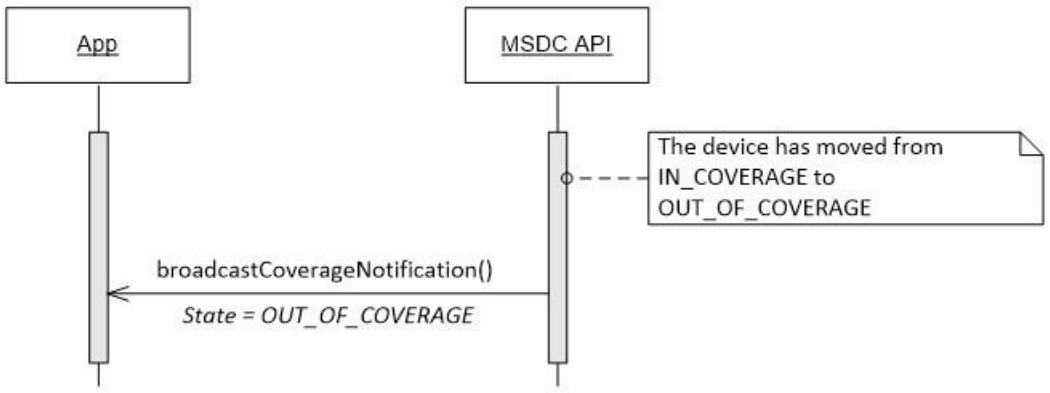


Figure 6-6 Network – Broadcast coverage notification

6.3.2 Signal level notification

6.3.2.1 Interface functions

```
void enableSignalLevelMonitoring(int periodicity);
void signalLevelNotification();
void disableSignalLevelMonitoring();
void networkServiceError(int errorCode,
                         String message);
```

6.3.2.2 Prerequisites

Network module connection initialization

6.3.2.3 Description

The app can request regular signal level notification to display by using `enableSignalLevelMonitoring()`. It also provides the Monitoring Interval parameter value in seconds.

The MSDC API responds with the `signalLevelNotification()`, which has signal level values ranging from 0 (lowest) to 5 (highest). The signal level during the Out of Coverage event is -1.

If the notification interval given by the app is a positive value, the MSDC API continues sending `signalLevelNotification()` messages with updated signal level values. The time between each notification is defined by the periodicity value.

To disable the signal level, use `disableSignalLevelMonitoring()`.

6.3.2.4 Call flows

6.3.2.4.1 Enable signal level

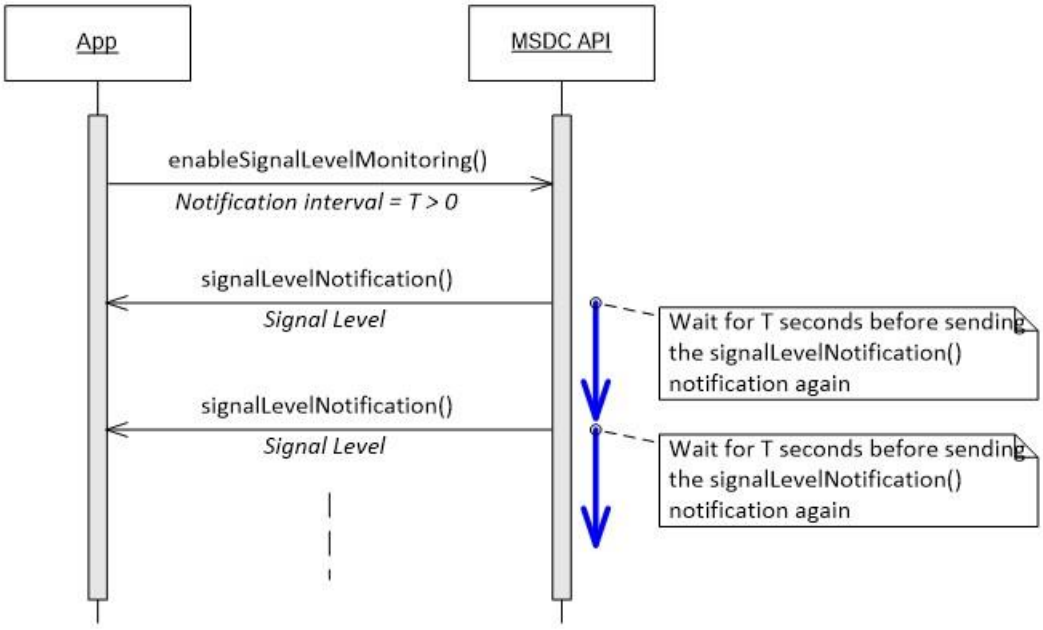


Figure 6-7 Network – Enable repetitive signal notification

If the app gives a notification interval value that is less than or equal to zero, the MSDC API sends the `signalLevelNotification()` message once.

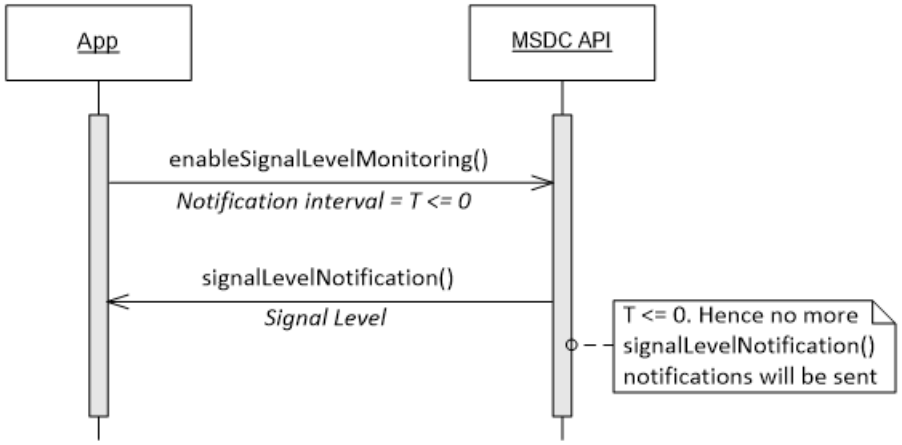


Figure 6-8 Network – Enable a single signal notification

If the MSDC cannot enable the signal level notification, the MSDC API responds with `networkServiceError()` and the error code `ERROR_NW_ UNABLE_TO_ENABLE_SIGNAL_LEVEL`.

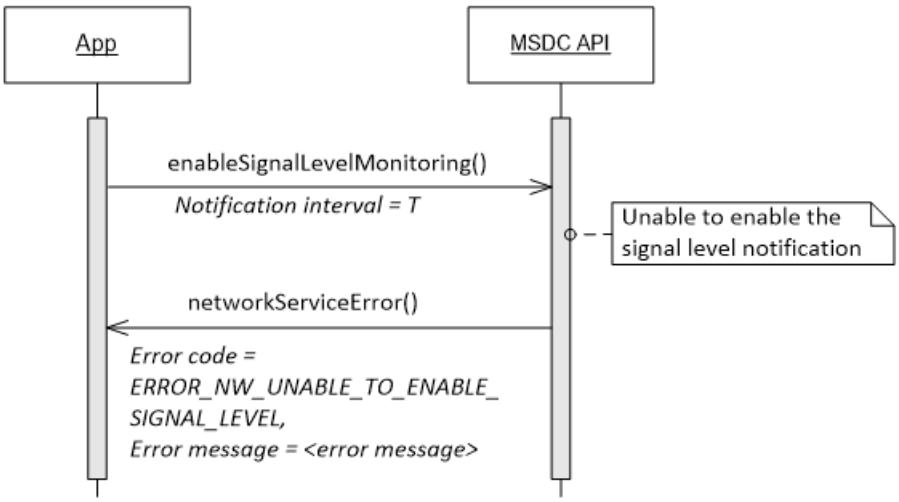


Figure 6-9 Network – Error with enabling a signal level notification

6.3.2.4.2 Disable signal level notification

To disable the signal level notification, the app can use `disableSignalLevelNotification()`. If the MSDC API accepts the request, it returns ACCEPTED to the app.

The MSDC API sends no disable confirmation notification to the app as a response to this request.

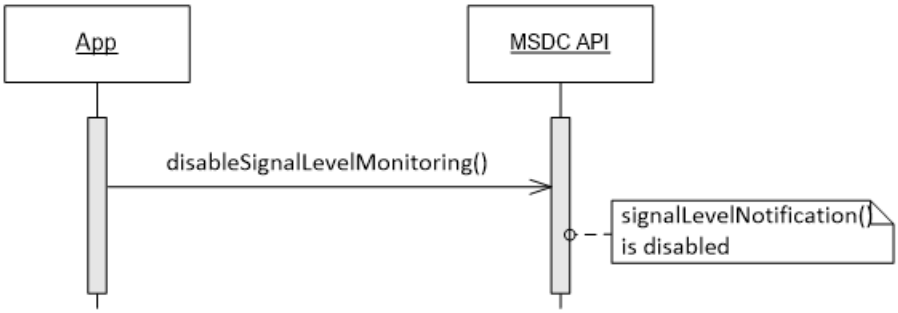


Figure 6-10 Network – Disable signal level notification

If the MSDC is unable to disable the signal level notification, the MSDC API responds with `networkServiceError()` and the error code `ERROR_NW_UNABLE_TO_DISABLE_SIGNAL_LEVEL`.

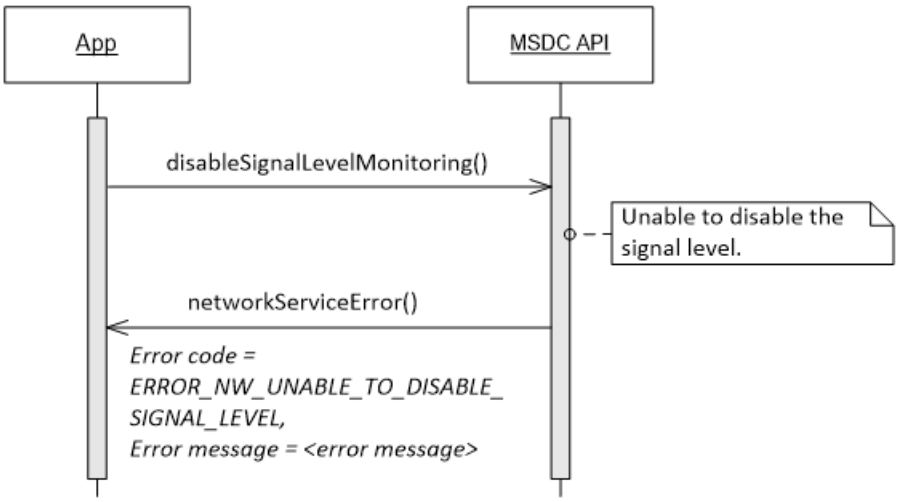


Figure 6-11 Network – Disable signal level notification error

6.3.3 Roaming notification

6.3.3.1 Interface functions

```
void roamingNotification (int state);
int getRoamingState();
State Enum: IN_NETWORK, OUT_OF_NETWORK
```

6.3.3.2 Prerequisites

Network module connection initialization

6.3.3.3 Description

If the device running the MSDC roams in or out of the LTE broadcast home operating network area, the MSDC API notifies the app with `roamingNotification()` to indicate this change.

6.3.3.4 Call flows

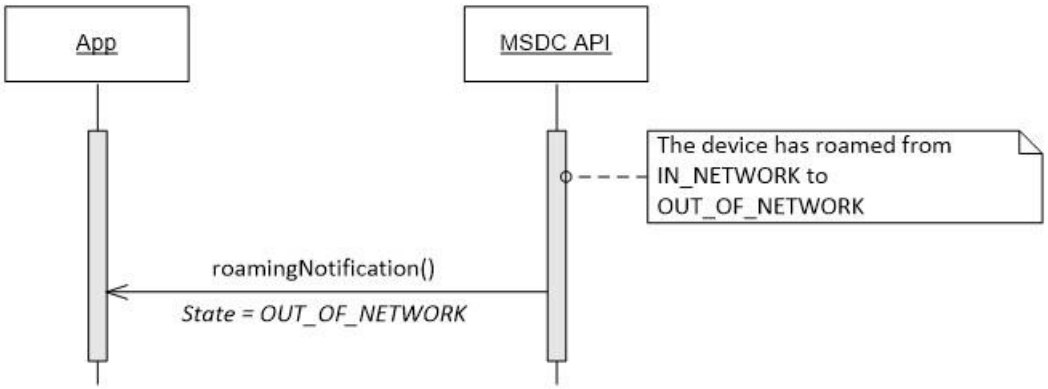


Figure 6-12 Network – Broadcast roaming notification

6.4 Other error notifications

If the MSDC API wants to notify the app of any Network module error, it uses the overloaded `networkServiceError()` notification.

For more information on the different types of error notifications, see Section 9.4.

6.4.1 Interface functions

```
void networkServiceError(int errorCode,
                         String message);
```

6.4.2 Prerequisites

Add Network module event listener

6.4.3 Call flows

If the Network module of MSDC is unavailable for any reason, the MSDC API responds to the app with `networkServiceError()` and the error code `ERROR_NW_SERVICE_UNAVAILABLE`.

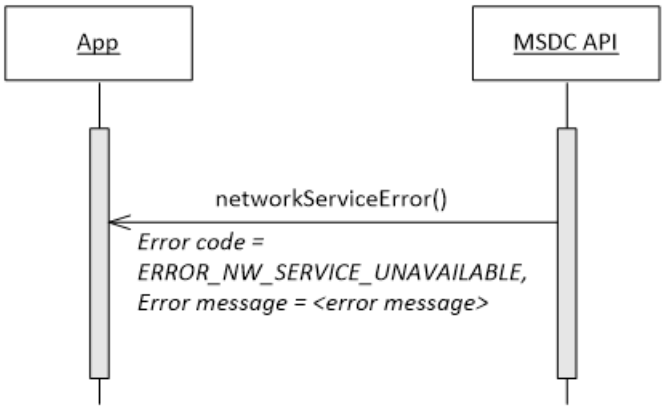


Figure 6-13 Error notification when Network module of MSDC is unavailable

6.5 App-to-MSDC connection shutdown

This section defines the calls the app uses to shut down the connection with the MSDC.

6.5.1 Close Network module connection

6.5.1.1 Interface functions

```
void terminateNetworkService()
```

6.5.1.2 Prerequisites

Network module connection initialization

6.5.1.3 Description

To close the connection to the Network module of the MSDC, the app uses `terminateNetworkService()`.

6.5.1.4 Call flows

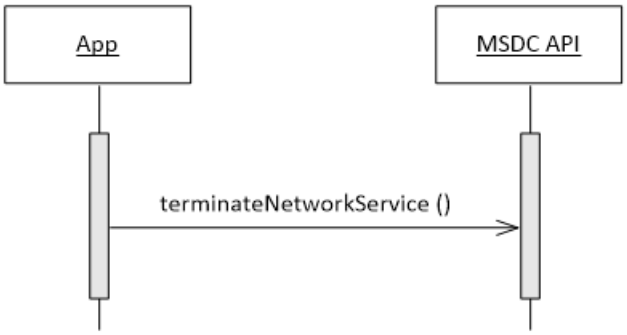


Figure 6-14 Close Network module connection

6.5.2 Remove Network module event listener

6.5.2.1 Interface functions

```
void removeNetworkEventListener (NetworkServiceEventListener
                                 listener);
```

6.5.2.2 Prerequisites

Add Network module event listener

6.5.2.3 Description

To stop getting events from the Network module, the app must remove the event listener that it added earlier by using `removeNetworkEventListener()` (see Section 6.2.1).

6.5.2.4 Call flows

Figure 6-15 Remove Network event listener

6.5.3 Close MSDC Manager module connection

See Section 4.6.3.

6.5.4 Remove MSDC Manager module event listener

See Section 4.6.4.

7 Group Call Service

This chapter describes the app-to-MSDC interface call flow sequences for a Group Call service application.

All functions in this chapter, i.e., request calls and notifications, are part of one of the following classes:

- IMSDCAppManager() (see Section 3.1)
- IMSDCAppManagerEventListener() (see Section 3.1)
- IMSDCGroupCallController() (see Section 3.5)
- IMSDCGroupCallControllerEventListener() (see Section 3.5)
- IMSDCGroupCallModel() (see Section 3.5)

7.1 Overview

To support the Group Call service, the application must talk to the MSDC, Group Call Client, and Media Player. The app's communication with the MSDC is essentially a control path, while its communication with the Media Player is a data path.

The app is responsible for configuring the data path (out of scope of the MSDC).

7.1.1 Group Call Client

The Group Call App is expected to encompass the functionality of the Group Call Client, which communicates with the Group Call Server in the Network. The app must obtain the list of all Group Call services from the Group Call server via the Group Call Client. Each Group Call service is identified by the following tuple:

```
< Group-Call Service Name;
    TMGI;
     List of Service Areas IDs where it is defined,
     List of Frequencies,
     Multicast IP,
     Multicast Port >
```

Figure 7-1 through Figure 7-3 provide an overview of the call flow for a typical Group Call service app interaction with the MSDC-SDK. More detailed call flow sequence for individual functions and other scenarios is described in subsequent sections.

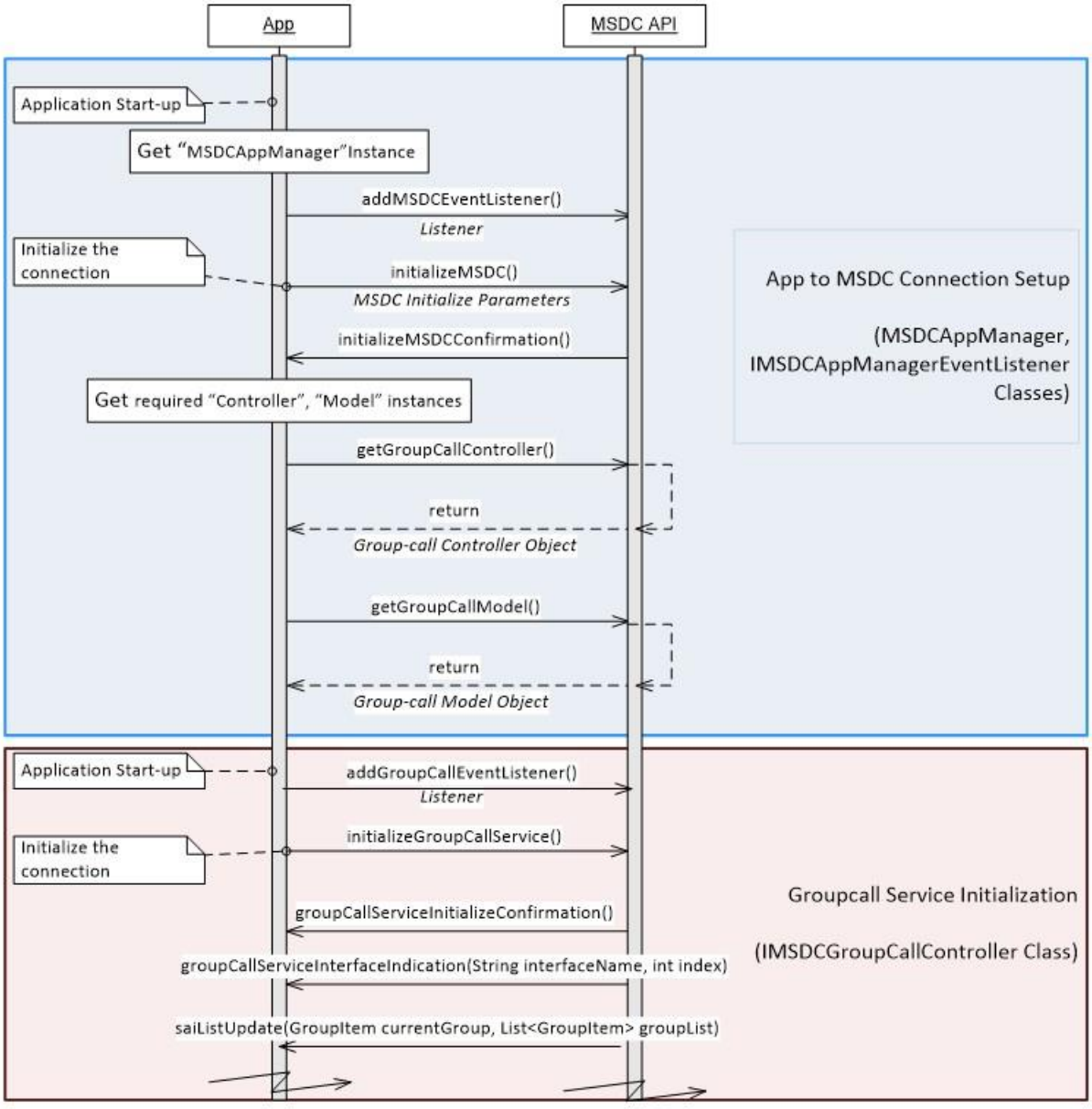


Figure 7-1 Group Call service app call flow (1 of 3)

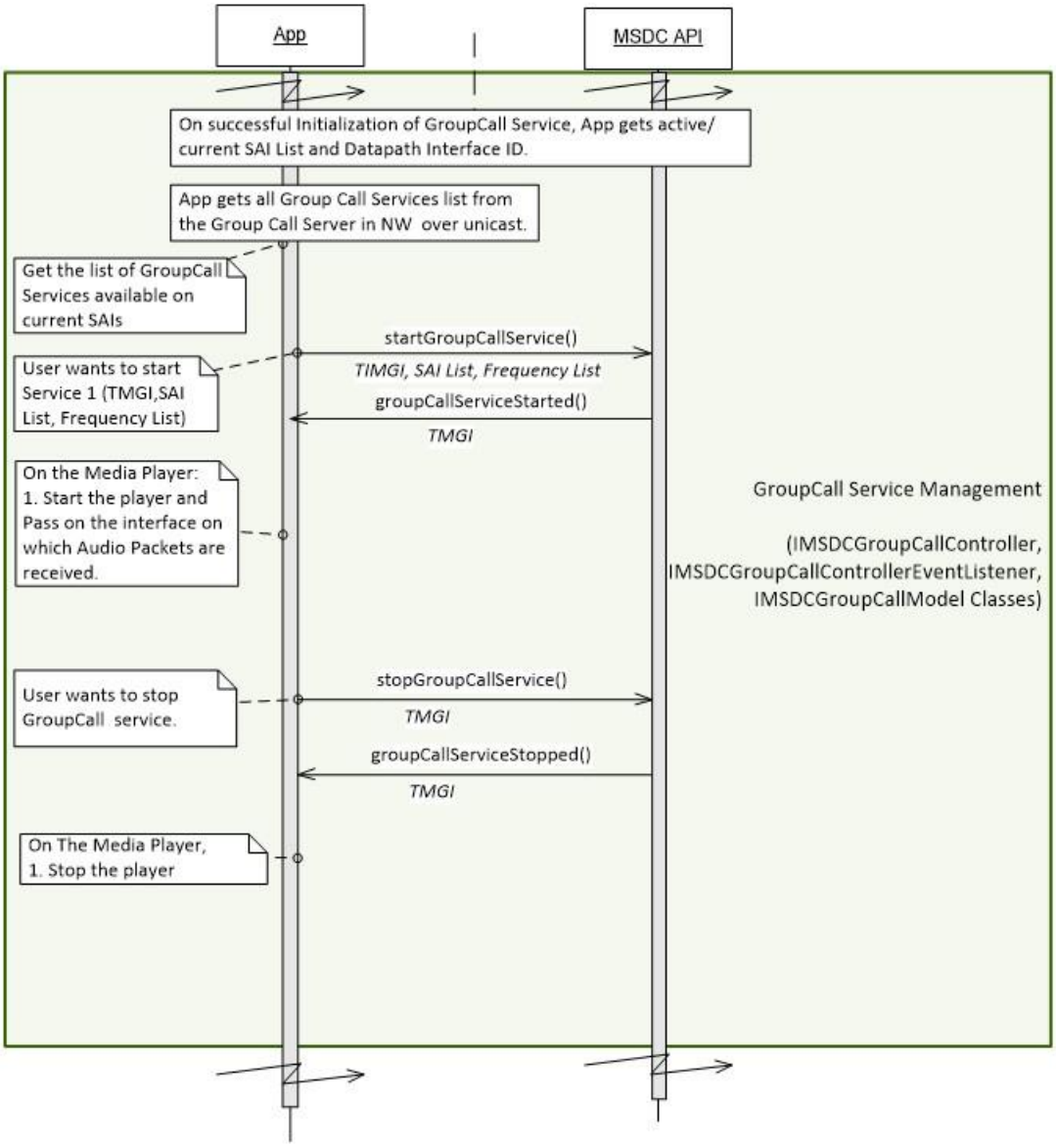


Figure 7-2 Group Call service app call flow (2 of 3)

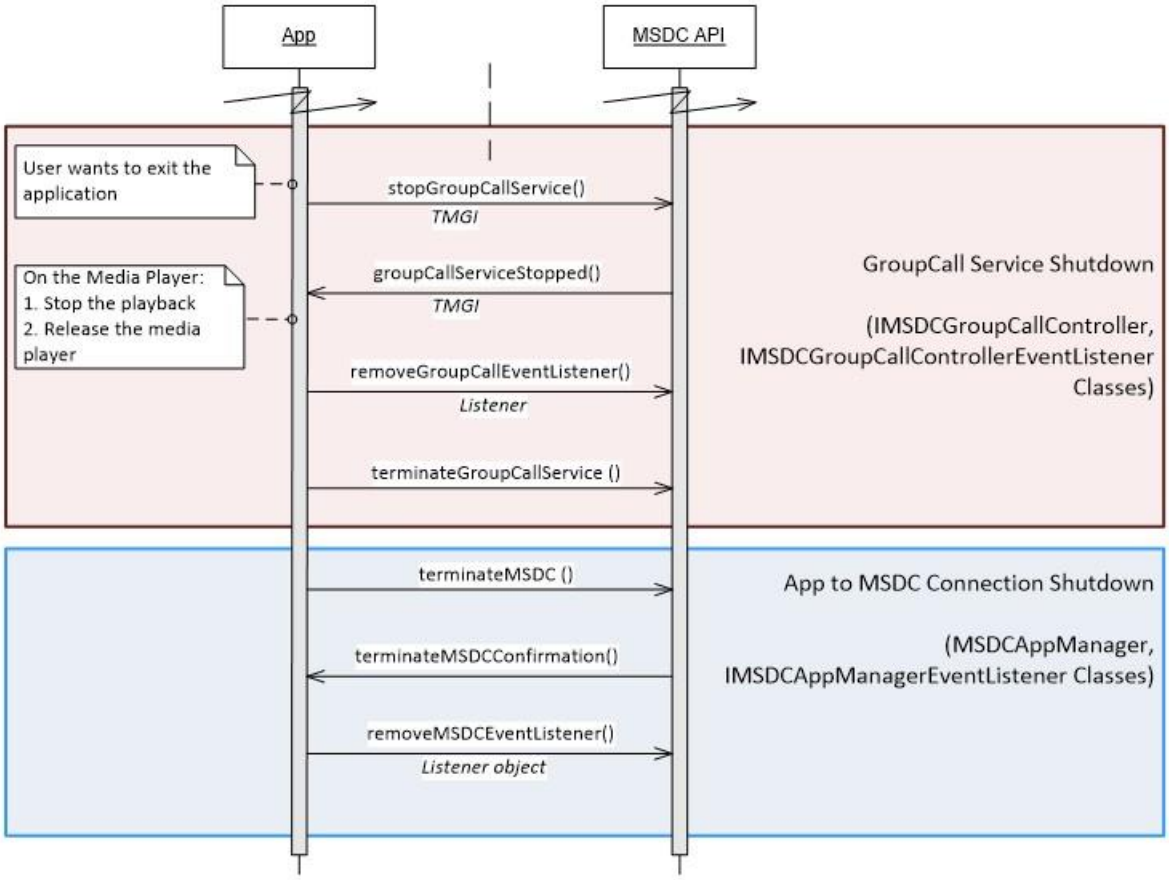


Figure 7-3 Group Call service app call flow (3 of 3)

7.1.2 Configuring the Media Player

At the end of the initialization process of the Group Call service module, the app gets an `interface-name` and `index` from the MSDC. These parameters enable the application to open multicast IP sockets on the network interface which the MSDC will push the Temporary Mobile Group ID (TMGI) bearer content. The TMGI bearer contains the multicast IP packets.

7.2 App-to-MSDC connection setup

The app starts here to begin communication with the MSDC.

7.2.1 Add MSDC Manager module event listener

See Section 4.2.1.

7.2.2 MSDC Manager module connection initialization

See Section 4.2.2.

7.2.3 Get the Group Call module Controller and Model instances

7.2.3.1 Interface functions

```
IMSDCGroupCallController getGroupCallController()
IMSDCGroupCallModel getGroupCallModel()
```

7.2.3.2 Description

To send requests to the Network module of the MSDC, the app should use `getGroupCallController()` and `getGroupCallModel()` calls to get the Network Controller and Model instances, respectively.

For code examples, see Section A.2.3.

7.2.3.3 Call flows

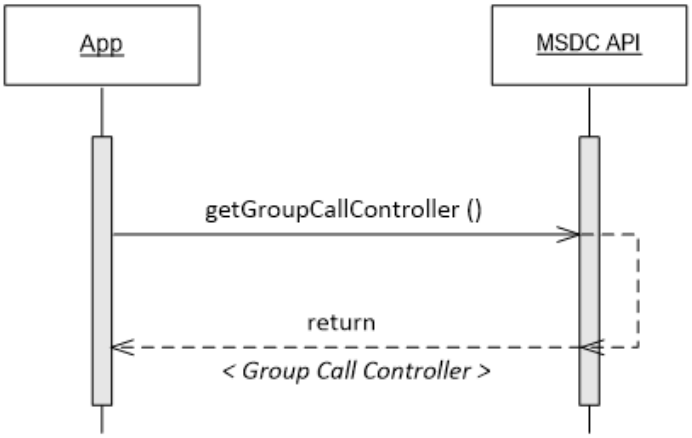


Figure 7-4 MSDC – Gets Group Call module Controller instance

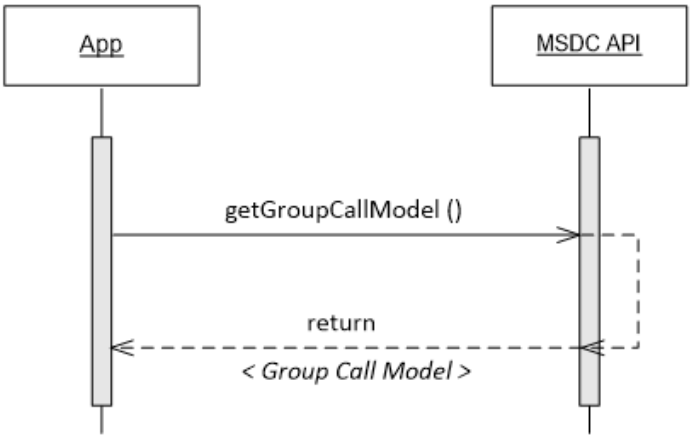


Figure 7-5 MSDC – Gets Group Call module Model instance

7.3 Group Call module initialization

7.3.1 Add Group Call module event listener

7.3.1.1 Interface functions

```
void addGroupCallEventListener (IMSDCGroupCallControllerEventListener listener
    );
```

7.3.1.2 Prerequisites

Get the Group Call module Controller and Model instances

7.3.1.3 Description

To get the events from the Group Call module of the MSDC API, the app must add/register the Group Call module event listener by using `addGroupCallEventListener()`.

For code examples, see Section A.2.4.

7.3.1.4 Call flows

Figure 7-6 shows the call flow for adding a Group Call module event listener.

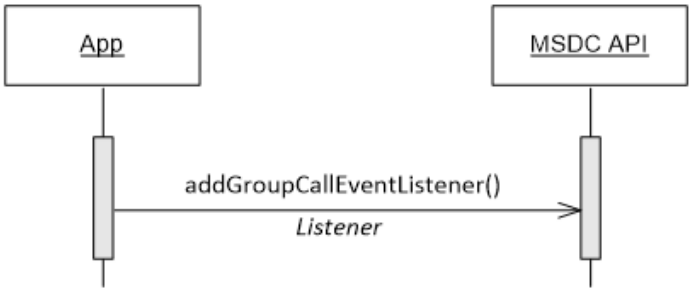


Figure 7-6 Group Call – Event listener registration

7.3.2 Group call module connection initialization

7.3.2.1 Interface functions

```
void initializeGroupCallService ();
void groupCallServiceConfirmation();
void groupCallServiceError (long TMGI, int errorCode,
                            String message);
```

7.3.2.2 Prerequisites

- Add MSDC Manager module event listener
- Add Group Call module event listener

7.3.2.3 Description

After the Group Call module event listener has been added, the app must initialize the connection to the Group Call module with `initializeGroupCallService()`.

If the MSDC API accepts the request and if the connection initialization is successful, the MSDC responds with `groupCallServiceInitializeConfirmation`.

For code examples, see Section A.2.2.

7.3.2.4 Call flows

7.3.2.4.1 Connection initialization succeeds

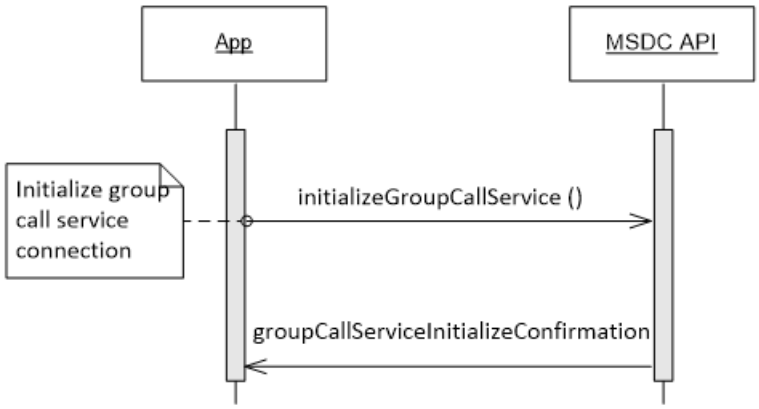


Figure 7-7 Group Call – Connection initialization succeeds

7.3.2.4.2 Connection initialization fails

If the Group Call module connection initialization fails, the MSDC API responds with `groupCallServiceError()` and the error code `ERROR_GC_UNABLE_TO_INITIALIZE`.

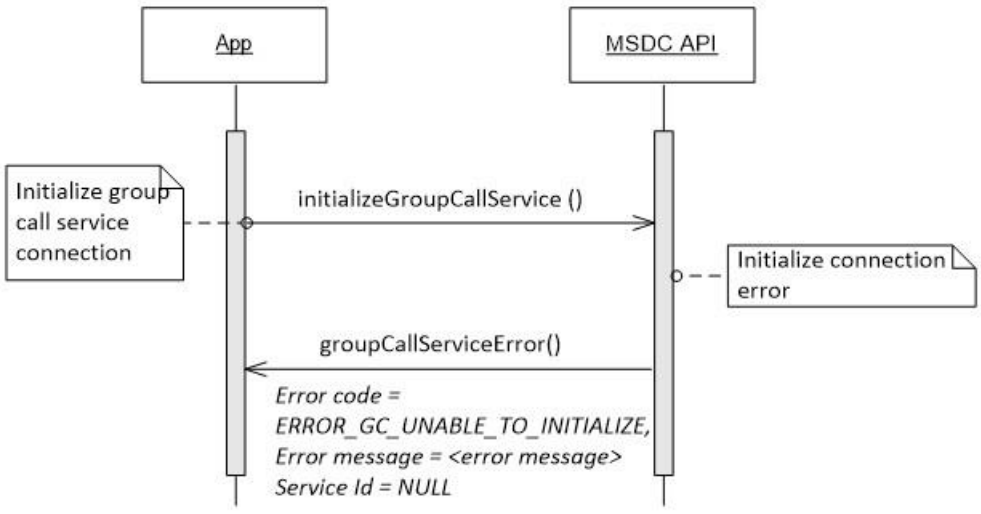


Figure 7-8 Group Call – Connection initialization fails

7.4 Group Call service management

The following sections define the set of calls to manage the start/stop sequence of a Group Call service.

7.4.1 Service states

Figure 7-9 shows the Group Call service states that are visible to the app. Depending on the actions taken by the MSDC or the user/app, the service may move from one state to another.

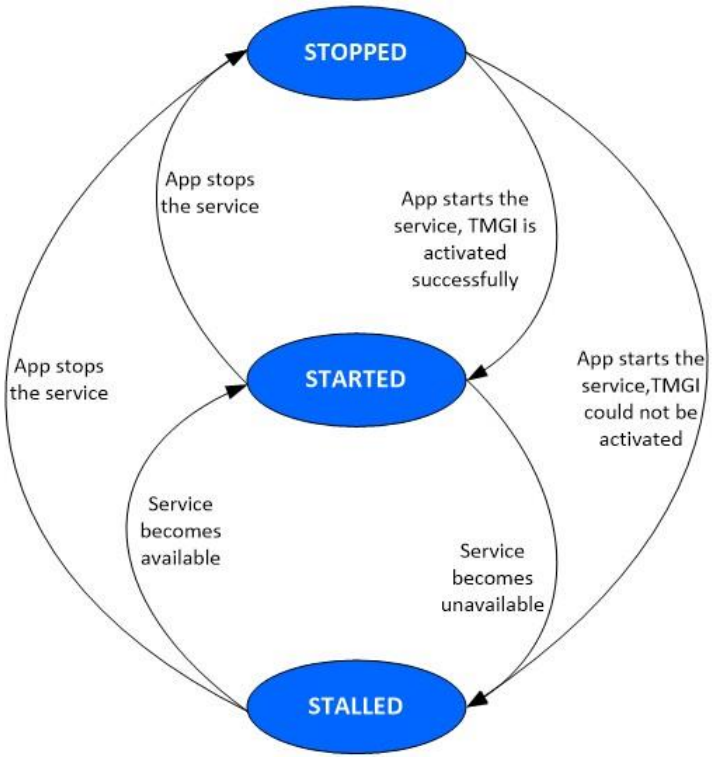


Figure 7-9 Group Call service states

- **STOPPED** – Default state of the service. The service is available for the user, but has not started yet.
  - If the user stops a service that is already STARTED or STALLED, it moves to the STOPPED state. When this happens, the app is notified by `groupCallServiceStopped()` (see Section 4.4.3).
- **STARTED** – The service moves to STARTED when the user/app starts an available service (available on the network) and TMGI activation succeeds for the service. When the Group Call service state moves to STARTED, the app is notified by `groupCallServiceStarted()` (see Section 4.4.2).
- **STALLED** – If a service in the STARTED state has a temporary issue which causes the service to become unavailable to the device, the MSDC moves the service to STALLED. When the Group Call service state moves to STALLED, the app is notified by `groupCallServiceStalled()` (see Section 4.4.5.1).
  - When the app starts a service and the TMGI that corresponds to the service cannot be activated, the MSDC also returns `groupCallServiceStalled()` to the app.
  - It is expected that this state is recoverable, and the MSDC will move to the STARTED state as soon as it becomes available. Alternatively, the app might choose to move the service to STOPPED by using `stopGroupCallService()` (see Section 4.4.3).

Note: There are two temporary intermediate states not shown in Figure 7-9:

- **START_REQUESTED** — When the app requests a service to be started and it has not yet received a `groupCallServiceStarted()` notification (see Section 7.4.2).
- **STOP_REQUESTED** – When the app requests a service to be stopped and it has not yet received a `groupCallServiceStopped()` notification (see Section 7.4.3).

7.4.2 Start a Group Call service

7.4.2.1 Interface functions

```
void startGroupCallService(long tmgi, List<Integer> saiList, List<Integer>
   freqList, String multicastIP, int multicastPort);

void groupCallServiceStarted(long tmgi, String server, int port);

void startGroupCallService (long TMGI, List <Integer> SAIList, List <Integer>
   FrequencyList); -- Deprecated from version 4.3.03.01.0

void groupCallServiceStarted (long TMGI); -- Deprecated from version
   4.3.03.01.0

void groupCallServiceError (long TMGI, int errorCode,
                            String message);
```

7.4.2.2 Prerequisites

Group Call module connection initialization

7.4.2.3 Description

To start a Group Call service, the app should use `startGroupCallService()`. Through this call, the app gives TMGI, SAI list, and Frequency list of the service to be started.

SIB15 is a system information block that provides the list of SAIs for the current cell and its neighbors. In cells where SIB15 is broadcast, the UE uses the SAI list to determine the frequency on which the TMGI is broadcast, and the Frequency list is ignored. If SIB15 is not broadcast in the current cell, the modem uses the Frequency list to activate the TMGI. The SAI list and Frequency list may be empty, indicating that the UE should attempt to activate the TMGI on the current cell.

If starting the Group Call service is successful, the MSDC API responds with `groupCallServiceStarted()` to indicate that the service has moved to the STARTED state.

7.4.2.4 Call flows

7.4.2.4.1 Starting a Group Call service

If the app wants to start a Group Call service, it must send a request with `startGroupCallService()`. This function requires the TMGI, SAI list, and Frequency list of the Group Call service that needs to be started. The app must get this information from the Group Call Client, which gets it from the Group Call Server in the network.

After the service has successfully started, the MSDC API sends `groupCallServiceStarted()` to the app and moves the service to the STARTED state.

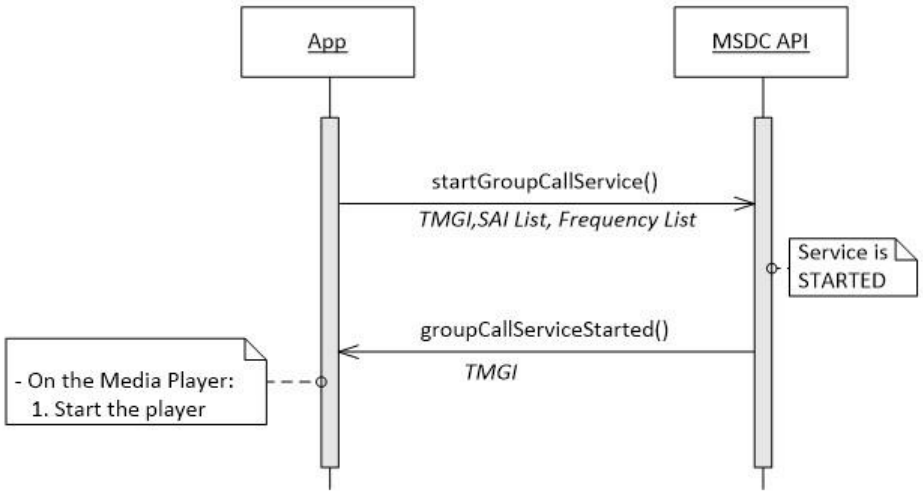


Figure 7-10 Starting a Group Call service

7.4.2.4.2 Unable to start a service

If the MSDC cannot start the Group Call service for any reason, the MSDC API responds to the app with `groupCallServiceError()` and the error code `ERROR_GC_UNABLE_TO_START_SERVICE`.

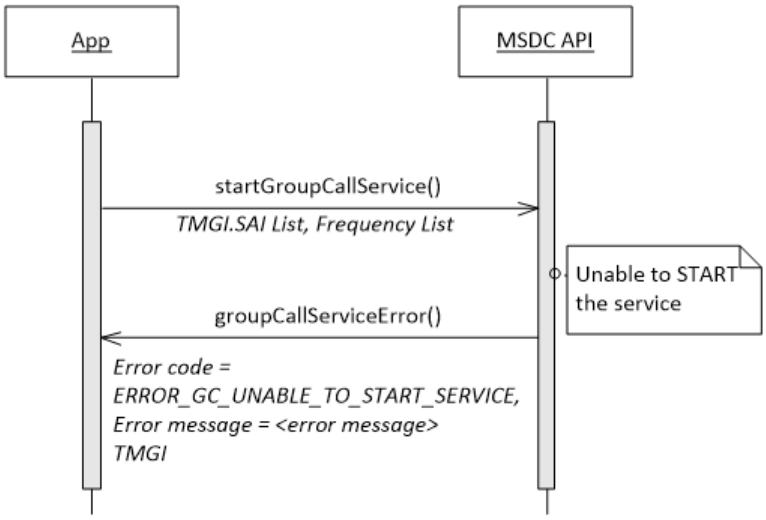


Figure 7-11 Unable to start Group Call service

7.4.3 Stop a Group Call service

7.4.3.1 Interface functions

```
void stopGroupCallService (long TMGI);
void groupCallServiceStopped (long TMGI);
void groupCallServiceError (long TMGI, int errorCode, String message);
```

7.4.3.2 Prerequisites

Group Call module connection initialization

7.4.3.3 Description

To stop a Group Call service that is in the STARTED or STALLED state, the app should use `stopGroupCallService()`. By calling this function, the app indicates the TMGI of the service to be stopped.

If stopping the Group Call service succeeds, the MSDC API responds with `groupCallServiceStopped()` to indicate that the service has moved to the STOPPED state.

7.4.3.4 Call flows

7.4.3.4.1 Stopping a Group Call service succeeds

To stop a service in the STARTED or STALLED state, the app uses the `stopGroupCallService()`. If stopping the service is successful, the MSDC API sends `groupCallServiceStopped()` to the app.

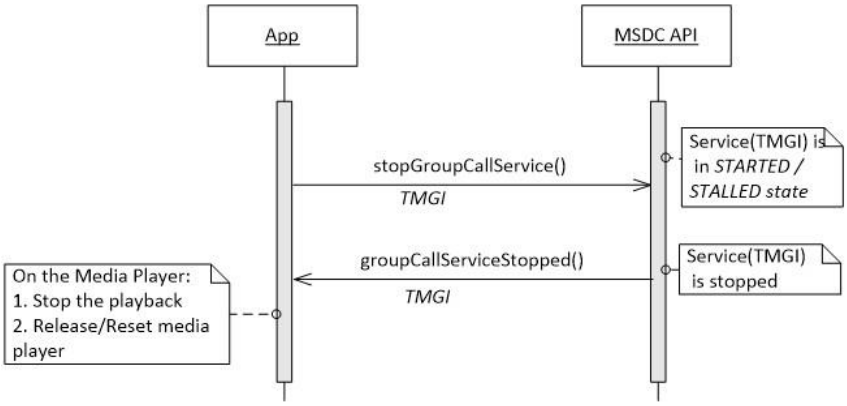


Figure 7-12 Stopping a Group Call service succeeds

7.4.3.4.2 Unable to stop Group Call service

If the MSDC cannot stop the Group Call service for some reason, the MSDC API responds to the app with `groupCallServiceError()` and the error code `ERROR_GC_UNABLE_TO_STOP_SERVICE`.

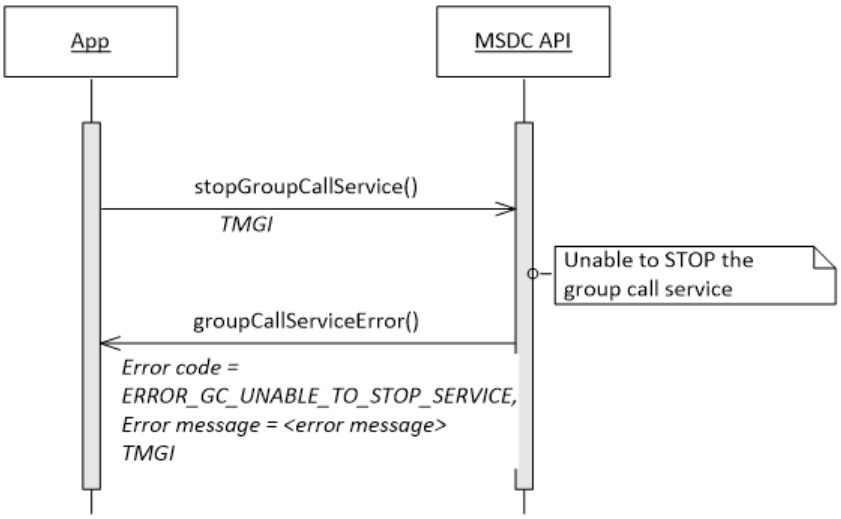


Figure 7-13 Unable to start Group Call service

7.4.4 Code

The following code snippet shows how to start/stop a Group Call service.

```
public void startOrStopGroupCallService(long tmgi) {
// Get the Current State of the Group Call Service
GroupCallServiceState state = mIMSDCGroupCallModel.getGroupCallServiceState(
    tmgi);
GroupCallService gcService = getGCServiceForTMGI(tmgi);
if (state == null){
//Start the GC Service.
 if (gcService != null) {
      mIMSDCGroupCallController.startGroupCallService(tmgi,gcService.saiList,
         gcService.freqList,
                                                        gcService.multicastIP,
                                                           gcService.
                                                           multicastPort);
 }
 else {
   // No GC Service with the given TMGI.
  }
}
else if (state == GroupCallServiceState.STATE_STARTED) {
//Stop the GC Service.
 mIMSDCGroupCallController.stopGroupCallService(tmgi);
}
else {
// GC Service in Improper or Transient State
}
}
```

7.4.5 Update a Group Call service

7.4.5.1 Interface functions

```
void updateGroupCallService (long TMGI, List <Integer> SAIList, List <Integer>
     FrequencyList);
```

7.4.5.2 Prerequisites

Group Call module connection initialization

7.4.5.3 Description

To update a Group Call service that is in the STARTED or STALLED state, the app should use the `updateGroupCallService()` function. Through this call, the app gives the TMGI of the service to be updated and the new values of SAI List and Frequency List.

The MSDC does not acknowledge updating a Group Call service already in STARTED or STALLED states.

7.4.5.4 Call flows

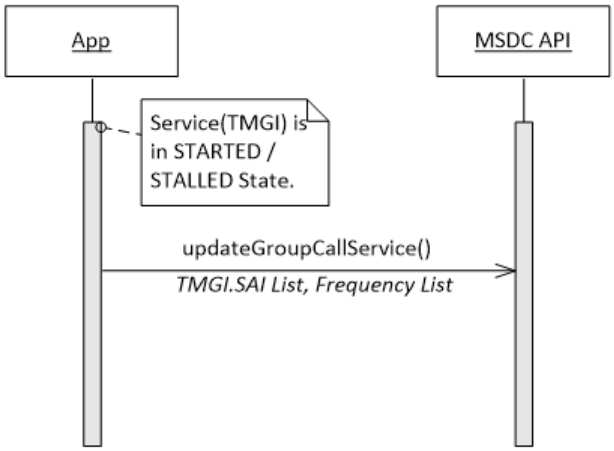


Figure 7-14 Update an active Group Call service

7.4.6 Other information notifications

7.4.6.1 Group Call service stalled

7.4.6.1.1 Interface functions

```
void groupCallServiceStalled(long TMGI);
```

7.4.6.1.2 Prerequisites

- Group Call module connection initialization
- Service is in the STARTED state

7.4.6.1.3 Description

If a service that is already in the STARTED state is temporarily unavailable, it is moved to the STALLED state. In this case, the MSDC API notifies the app with `groupCallServiceStalled()`.

If the service becomes available, the MSDC API sends `groupCallServiceStalled()` to indicate that the service has moved back to the STARTED state.

Typically, the service moves to a STALLED state due to radio control channel unavailability or adverse coverage conditions.

7.4.6.1.4 Call flows

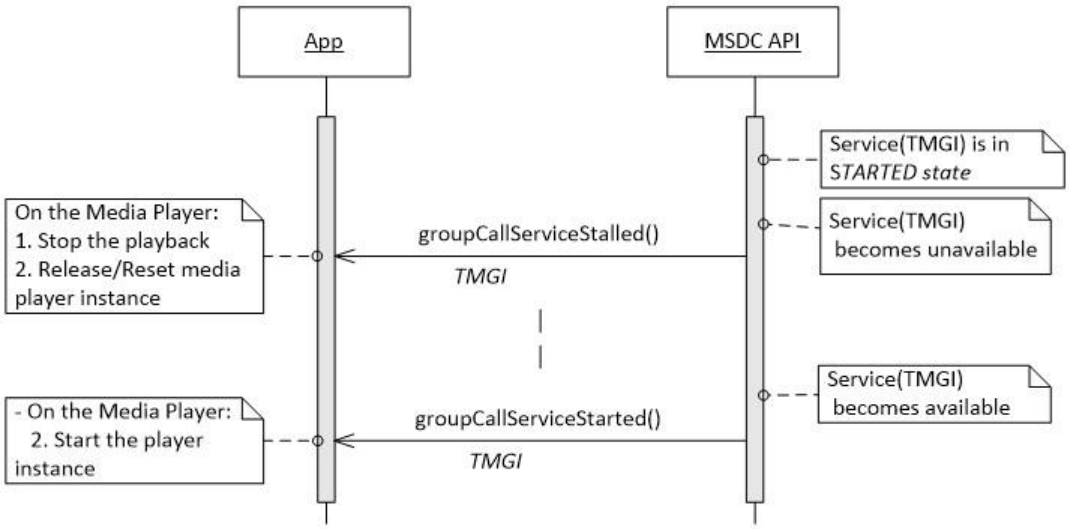


Figure 7-15 Stalled notification for Group Call service

7.4.6.2 SAI list update

7.4.6.2.1 Interface functions

```
void saiListUpdate (GroupItem currentGroup, List<GroupItem> groupList);
```

`saiListUpdate()` contains the following parameters:

- *currentGroup* – Indicates the available group of services in a camped cell.
- *groupList* – Indicates list of groups of available services in neighboring cells.

Each *GroupItem* class contains the following member variables:

```
String groupName;
List<Integer> serviceAreaIdList;
List<Integer> serviceHandleList;
```

The member variable *groupName* is the name of the camped network type. The *serviceHandleList* is the list of Group Call services which are available/valid in the Service Areas listed in the *serviceAreaIdList* of that group.

7.4.6.2.2 Prerequisites

Group Call module connection initialization

7.4.6.2.3 Description

When there is a change in the active/current Service Area ID (SAI) list, the MSDC notifies the app with `saiListUpdate()`. The app is required to refresh the available Group Call services list per the latest SAI List.

The available Group Call services list provides all Group Call services from the Group Call Client, and filtered based on the current/active SAI. Typically, the current/active SAI list changes when the device moves across Service Area boundaries.

7.4.6.2.4 Call flows

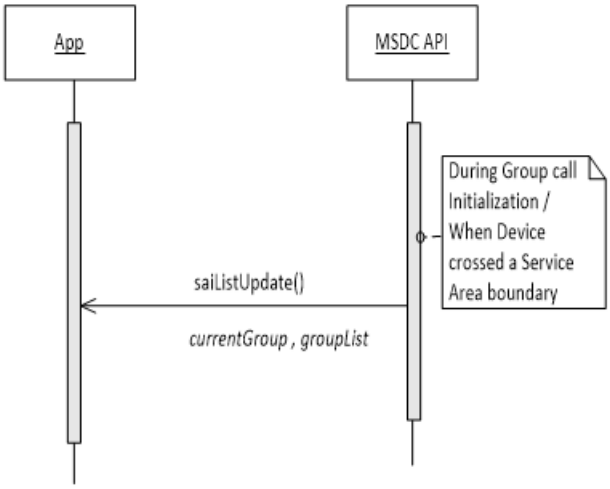


Figure 7-16 Group Call – SAI list update notification

7.4.6.3 Group Call service interface indication

7.4.6.3.1 Interface functions

```
void groupCallServiceInterfaceIndication(String interfaceName, int index);
```

7.4.6.3.2 Prerequisites

Group Call module connection initialization

7.4.6.3.3 Description

Immediately after receiving confirmation of a Group Call service initialization, the MSDC notifies the app about the data interface on which the audio packets of the Group Call service are received. Ideally, this Group Call interface does not change once it is initialized.

7.4.6.3.4 Call flows

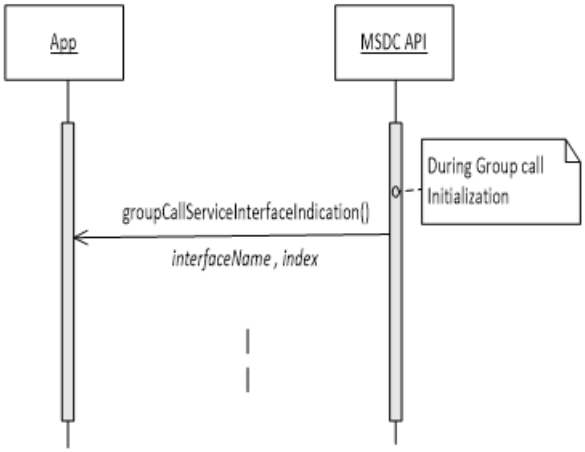


Figure 7-17 Group Call service interface indication

7.5 MSDC Manager module connection management

See Section 4.4.8.6.

7.6 App-to-MSDC connection shutdown

This section defines the calls the app uses to shut down the connection with the MSDC.

7.6.1 Close Group Call module connection

7.6.1.1 Interface functions

```
void terminateGroupCallService()
```

7.6.1.2 Prerequisites

Group Call module connection initialization

7.6.1.3 Description

To close the connection to the Network module of the MSDC, the app uses `terminateGroupCallService()`.

7.6.1.4 Call flows

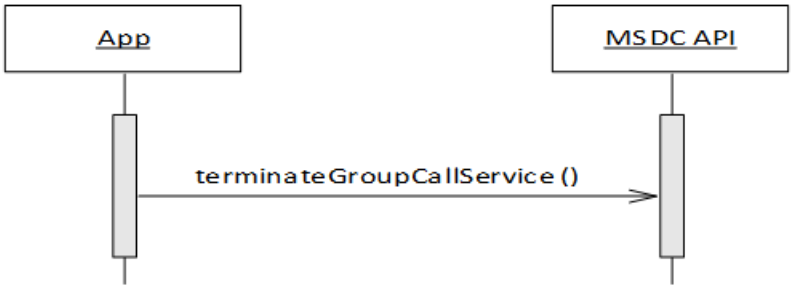


Figure 7-18 Close Group Call module connection

7.6.2 Remove Group Call module event listener

7.6.2.1 Interface functions

```
void removeGroupCallEventListener
      (IMSDCGroupCallControllerEventListener listener);
```

7.6.2.2 Prerequisites

Add Streaming module event listener

7.6.2.3 Description

To stop getting events from the Group Call module, the app must remove the event listener that it added/registered earlier (see Section 4.3.1) by using `removeGroupCallEventListener()`.

7.6.2.4 Call flows

Figure 7-19 Remove Group Callevent listener

7.6.3 Code

The following code snippet is for removing a Group Call event listener and terminating a Group Call service.

```
protected void onDestroy() {
// On destruction of the Every Activity, which Added/Registered itself as a
    listener // with the groupCallController. Remove/De-Register the Group Call
     event listener
// from controller.
mIMSDCGroupCallController.removeGroupCallEventListener(this);
// When all the activities of App are De-Registered with the Group Call
    Controller //and on Destruction of App, terminate Group Call Service.
mIMSDCGroupCallController.terminateGroupCallService();
}
```

7.6.4 Close MSDC Manager module connection

See Section 4.6.3.

7.6.5 Remove MSDC Manager module event listener

See Section 4.6.4.

8 Use Cases

Depending on what an application wants to achieve, the app can use the interface to communiate with the MSDC in different ways. This chapter describes typical use cases for applications and how they use the I-1 interface.

Figure 8-1 shows an overview of network elements typically associated with sending data to an app.

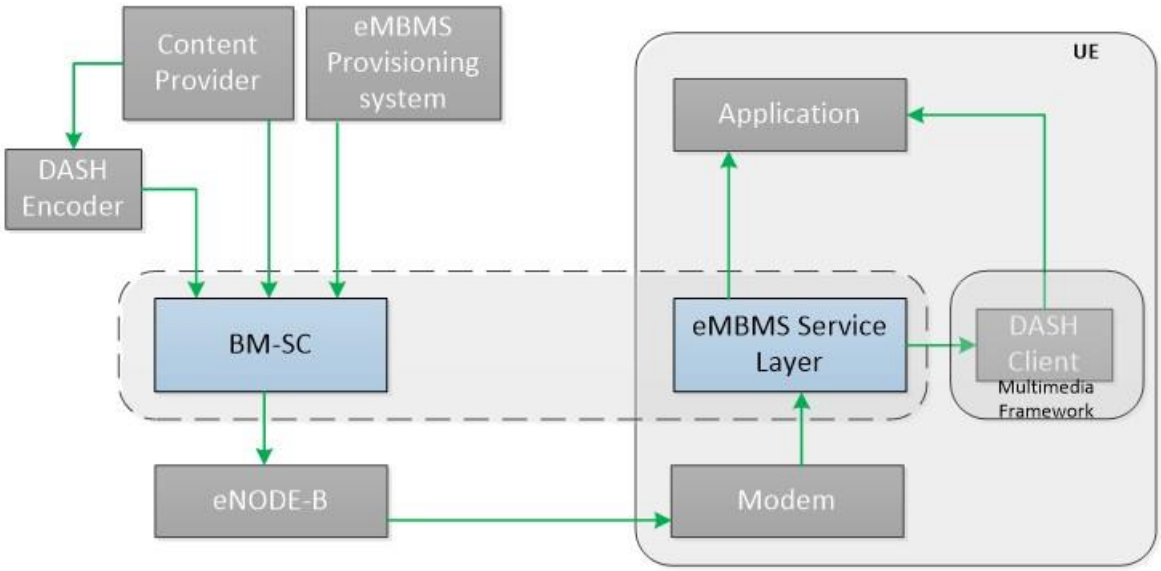


Figure 8-1 Network elements overview

8.1 Streaming application

A typical streaming application streams live videos to the user. The user sees a list of available streaming services, and based on user selection, the app streams the video to the user.

The following use case call flow assumes that the user:

1. Opens the app.
2. Selects a streaming service from a list of services.
3. Switches to another streaming service.
4. Exits the app.

For an overview of the call flow for a typical Streaming service app, see Figure 4-1.

8.2 YouTube-like application - Top 10 videos

A typical YouTube-like application downloads the top 10 videos of the day for the user. The following use case call flow assumes that the user:

1. Opens the app.
2. Sees the top 10 videos of the day (may play a few videos).
3. Exits the app.

In this scenario, we look at how the app can avoid downloading videos over subsequent days if they have already been downloaded the previous day as part of that day's top 10 list. The unwanted older video files are also deleted from the device.

This scenario also assumes that the Top 10 video files are always broadcasted.

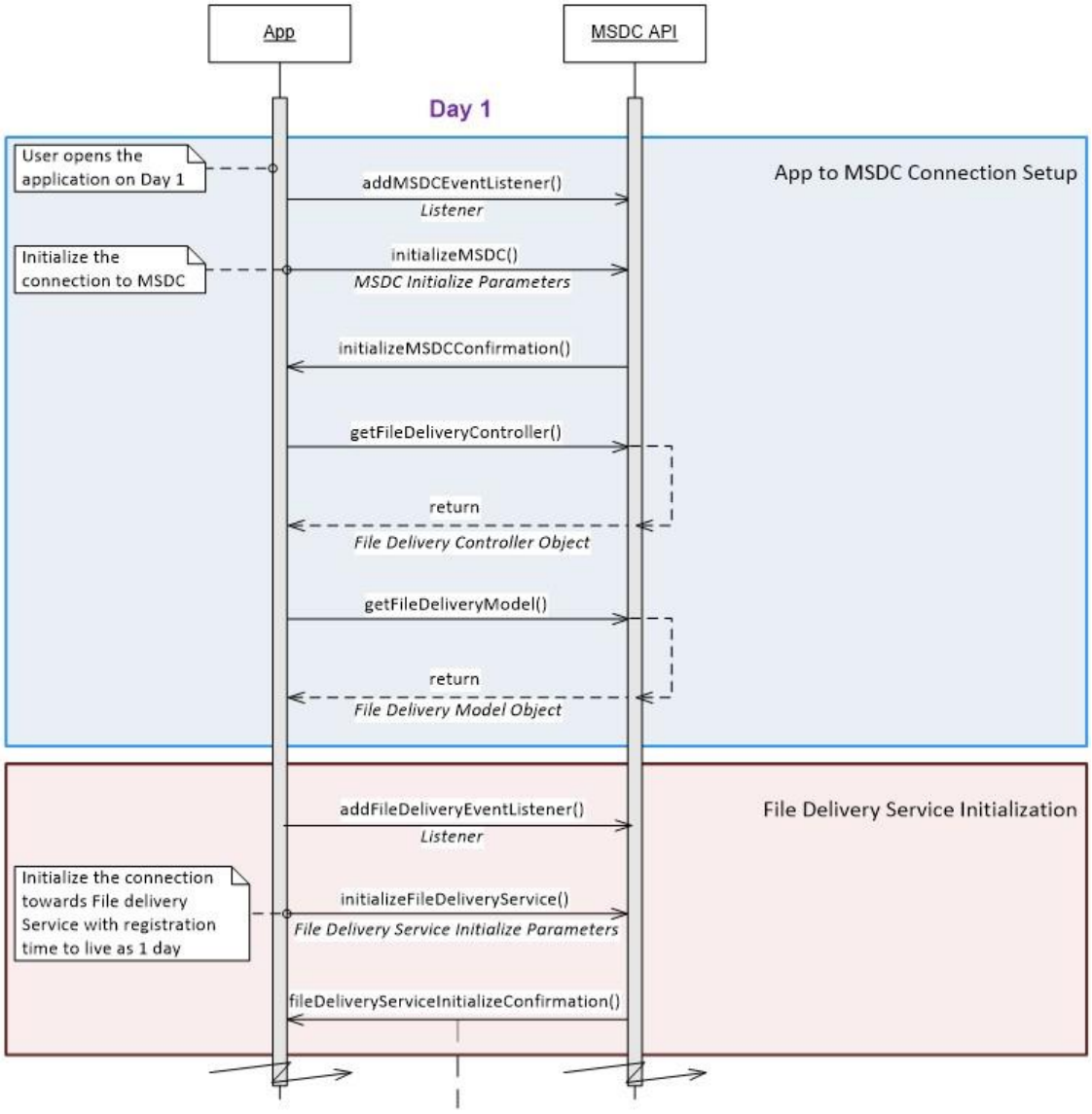


Figure 8-2 Use case - YouTube Top 10 video download (1 of 4)

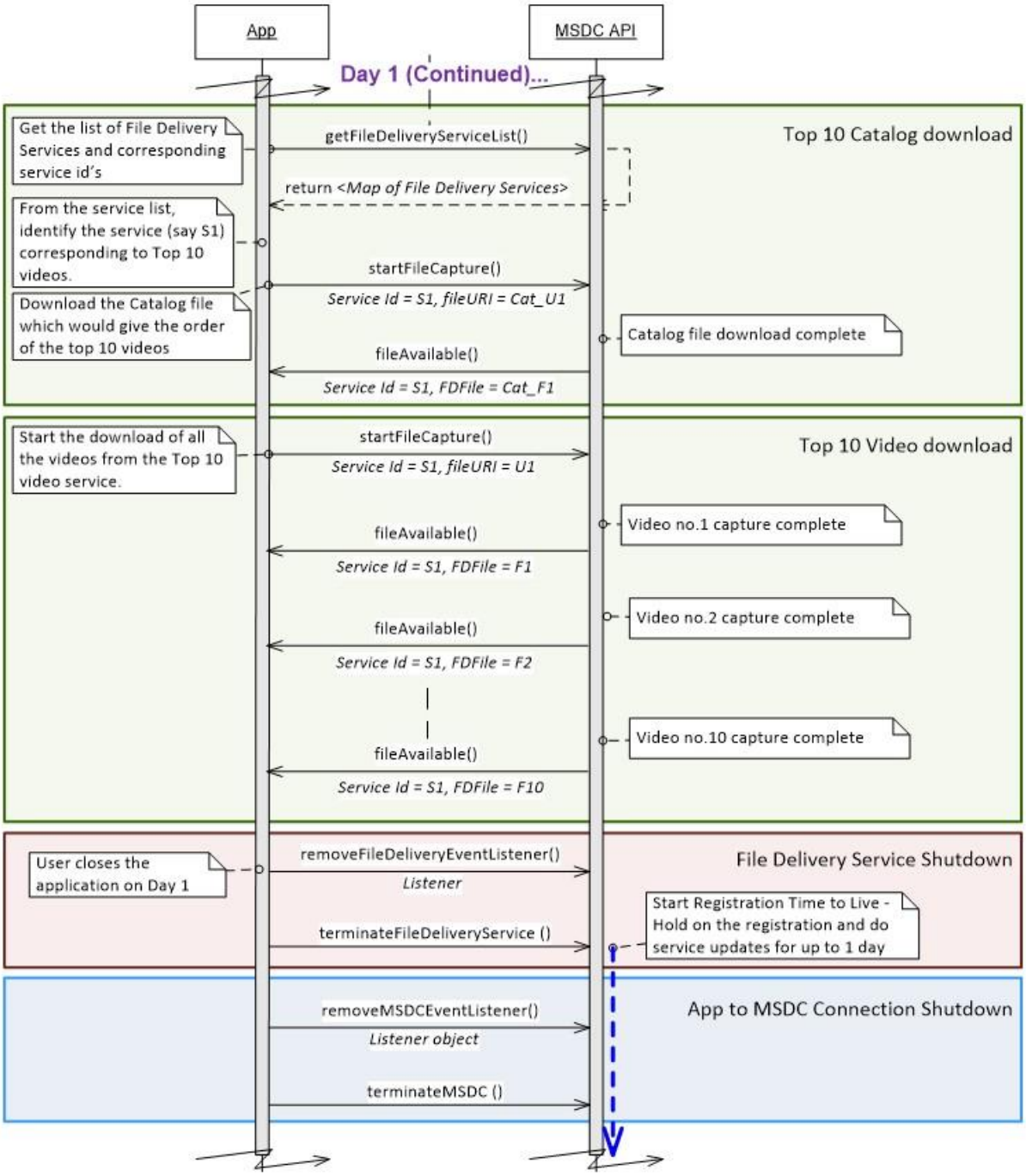


Figure 8-3 Use case - YouTube Top 10 video download (2 of 4)

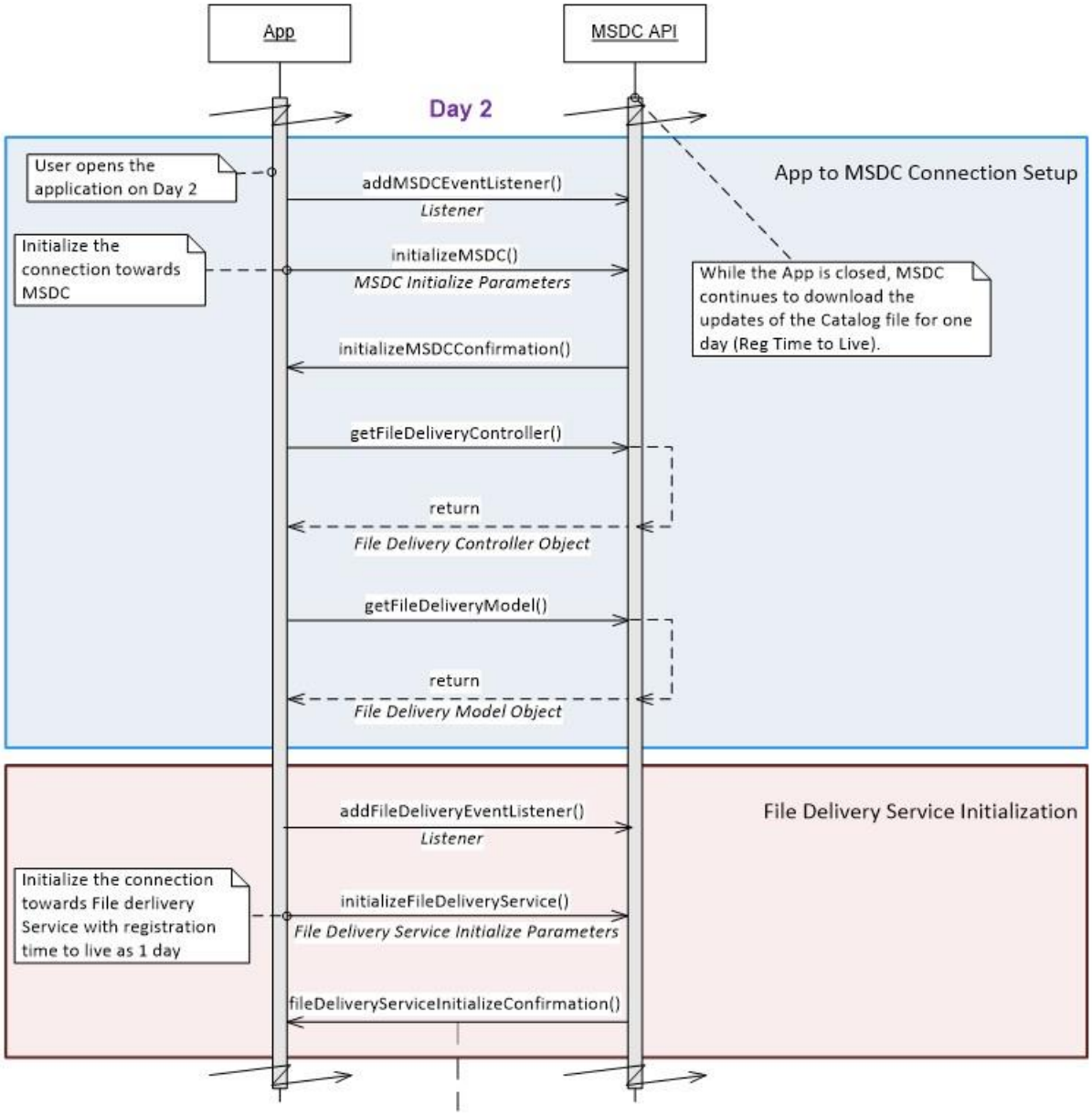


Figure 8-4 Use case - YouTube Top 10 video download (3 of 4)

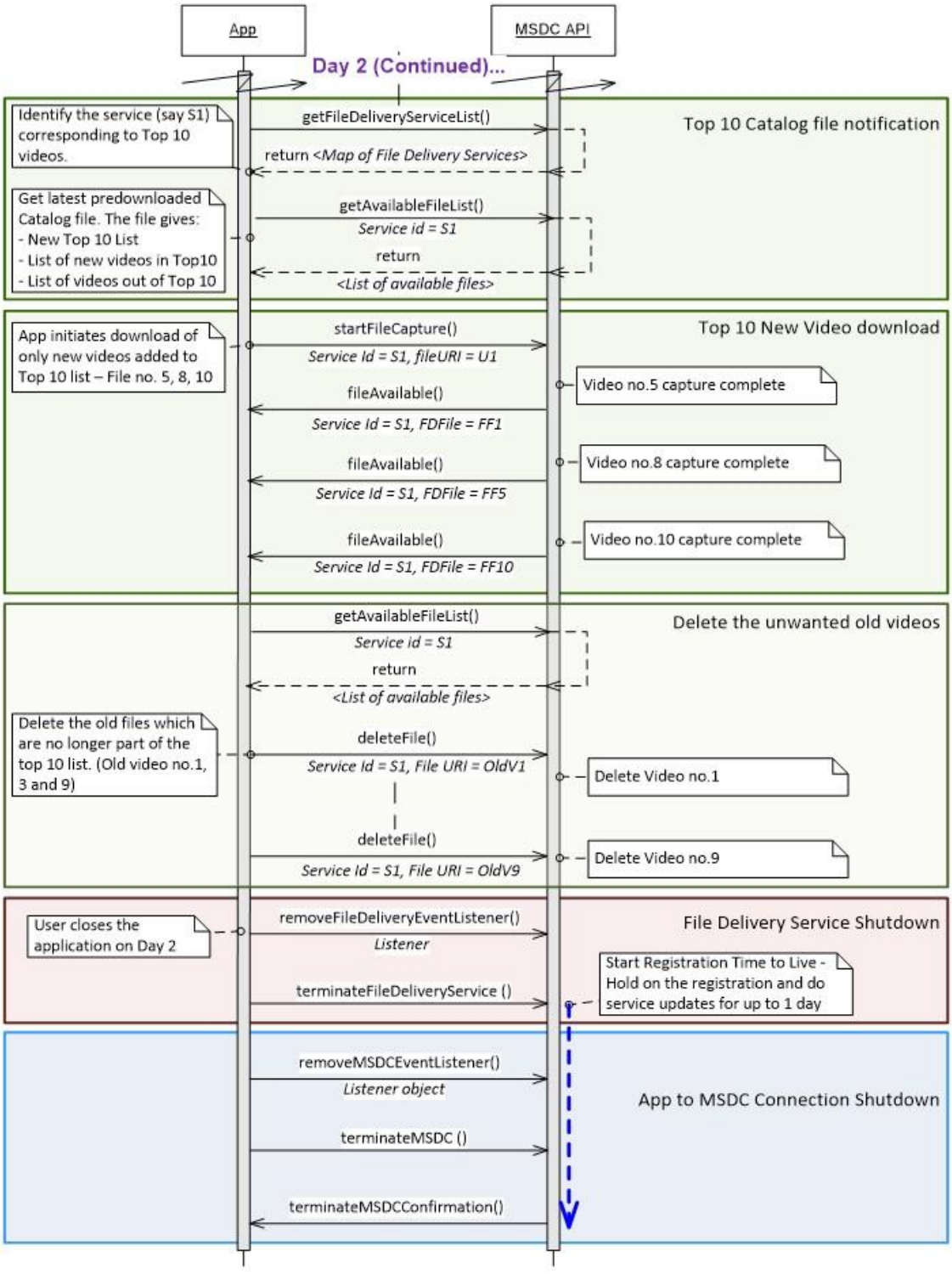


Figure 8-5 Use case - YouTube Top 10 video download (4 of 4)

8.3 Firmware update application

An application helps download the firmware update of a device over the air (FOTA) and installs it on the device. The following use case call flow assumes that the user:

1. Opens the app.
2. Checks information on the app that states whether the device firmware has any available updates to install.
3. Exits the app, or initiates installing the firmware update which makes the app terminate the connection with the MSDC.

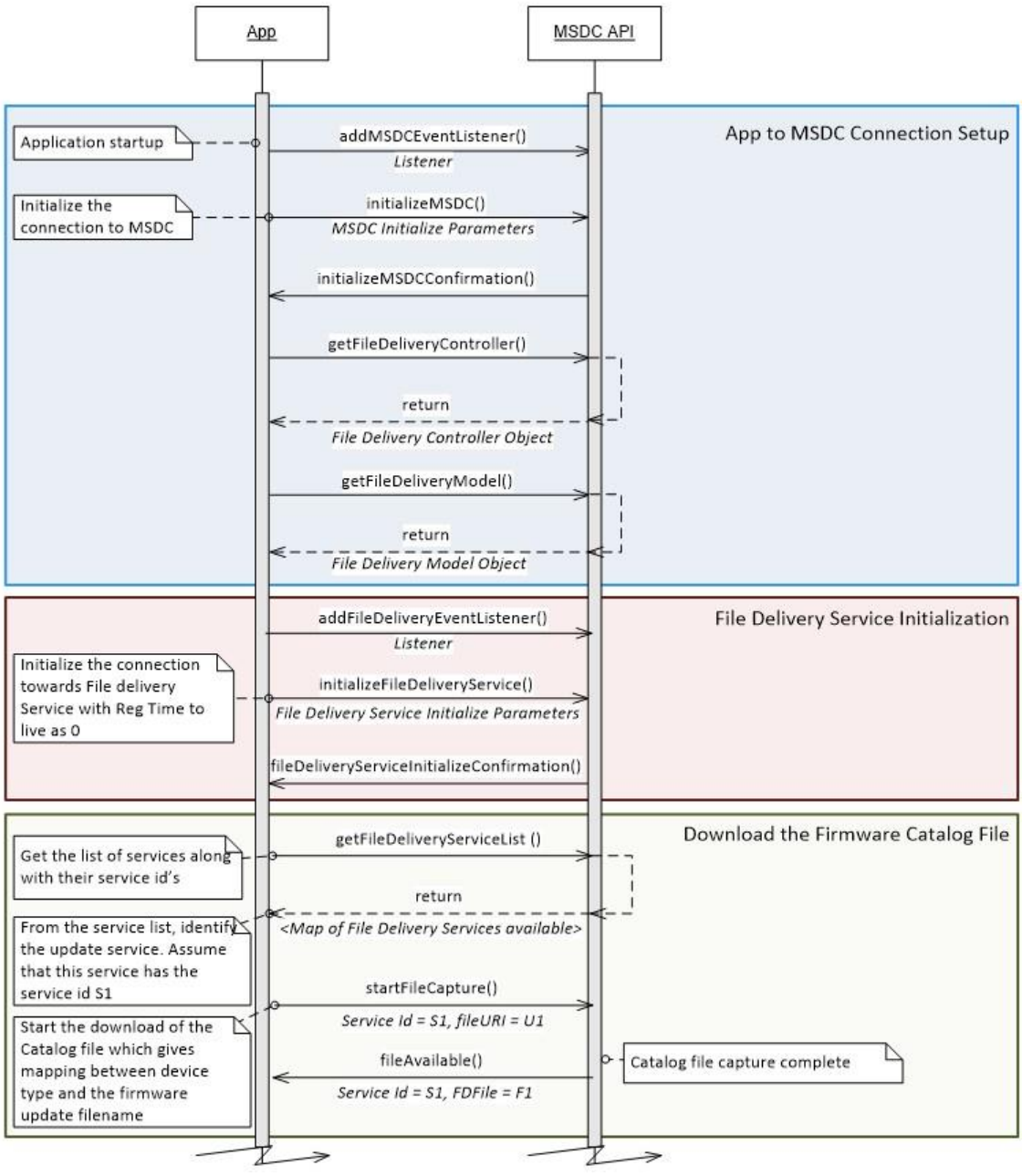


Figure 8-6 Use case - FOTA update application (1 of 2)

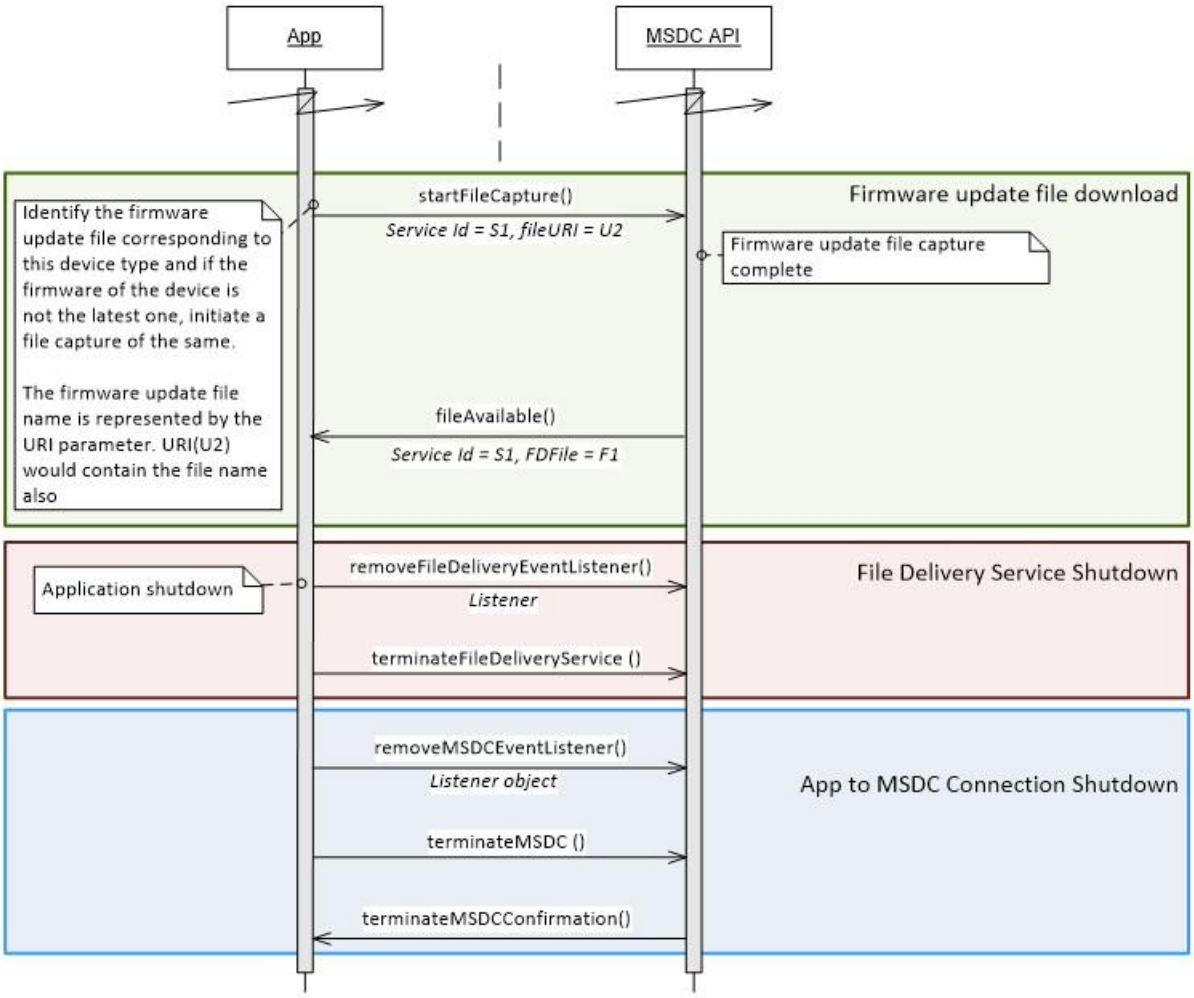


Figure 8-7 Use case - FOTA update application (2 of 2)

8.4 Weekly magazine application

An application delivers a weekly magazine to the user. This scenario assumes the following:

- The magazine app always runs in the background.
- The latest edition of the magazine is available only on a single day of the week. Subsequent editions are available at a 7-day interval thereafter.
- The app shows only one edition of the magazine at a time to the user.

The following call flows show two editions of the magazine.

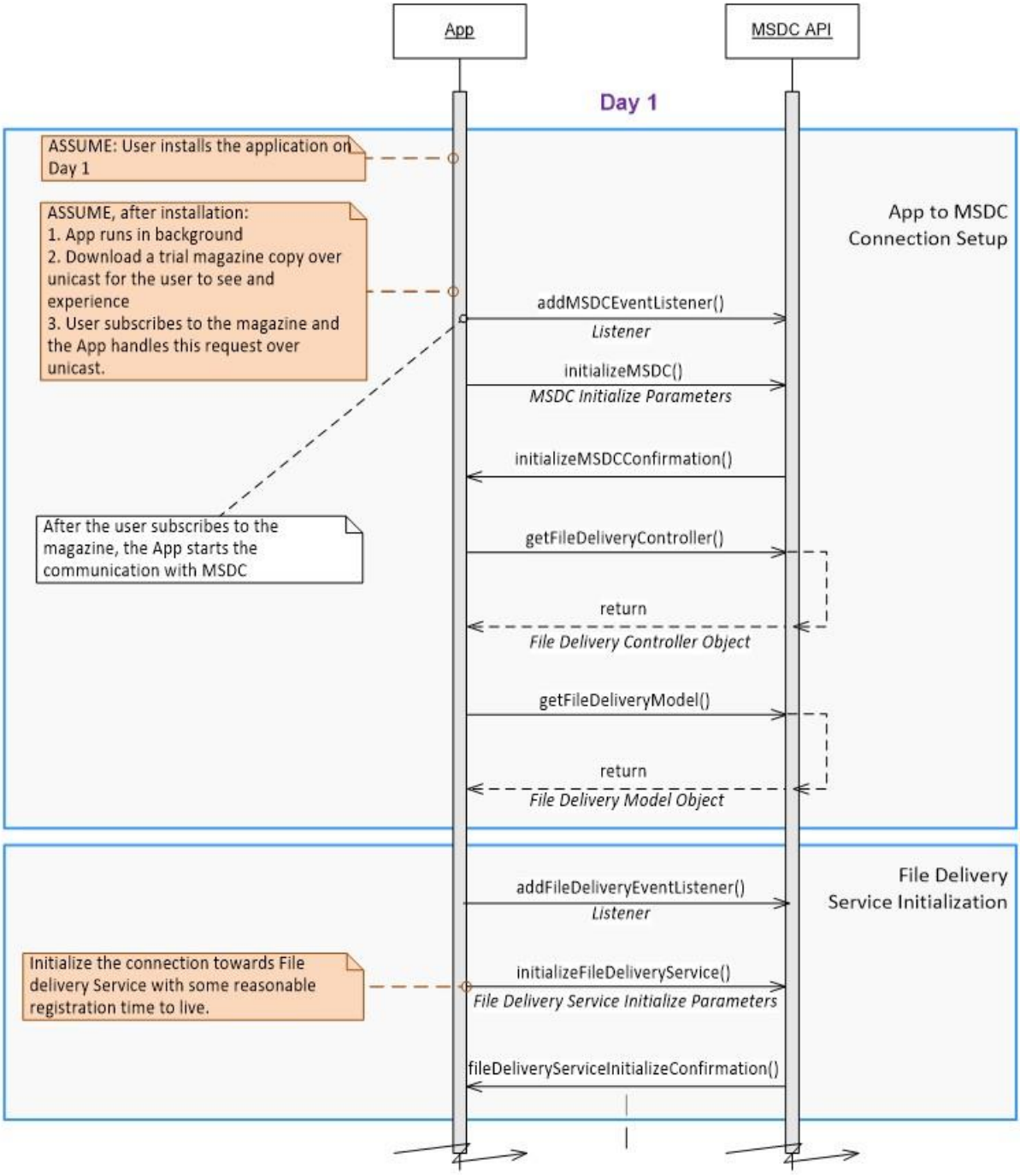


Figure 8-8 Use case - Weekly magazine application (1 of 3)

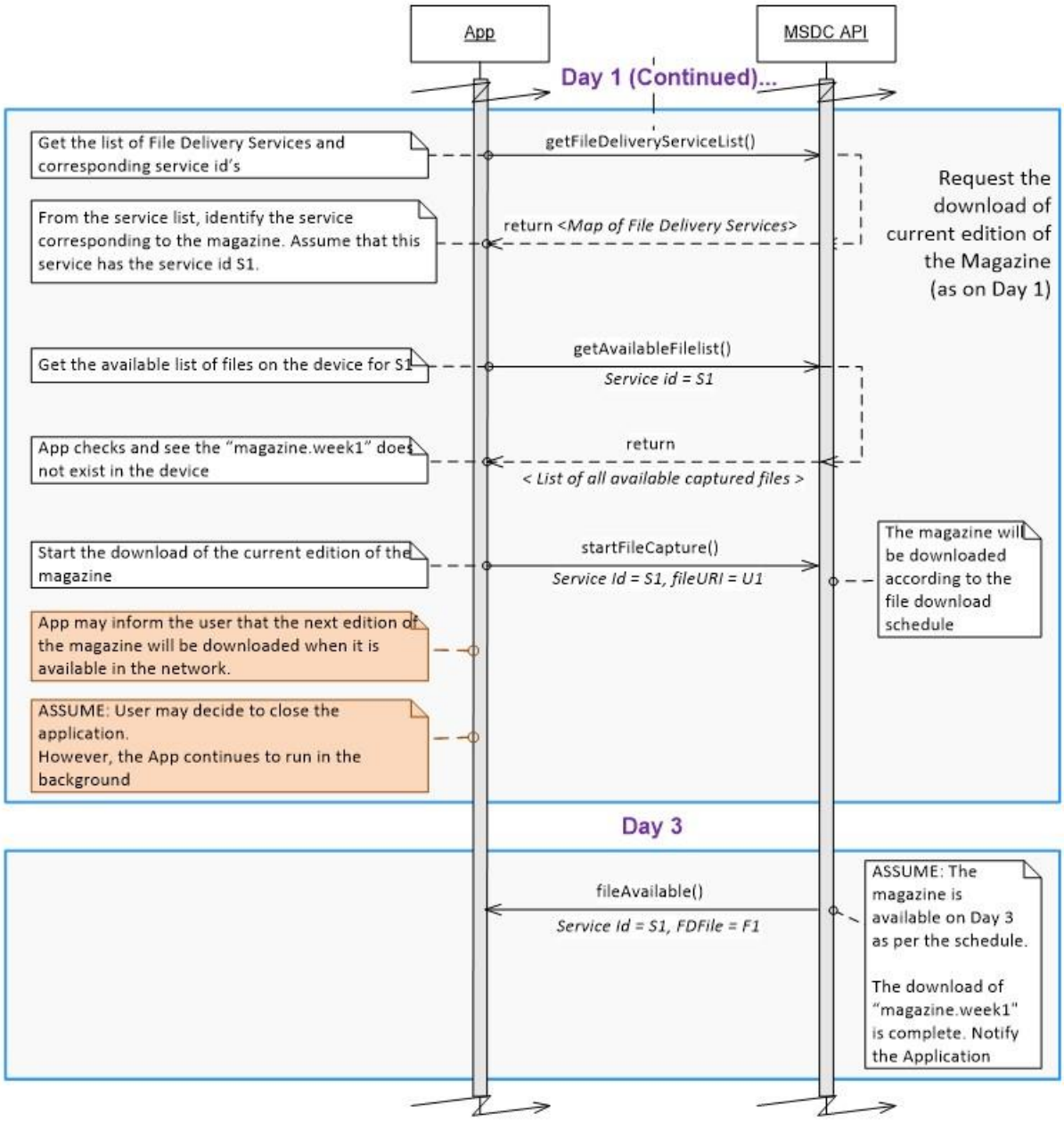


Figure 8-9 Use case - Weekly magazine application (2 of 3)

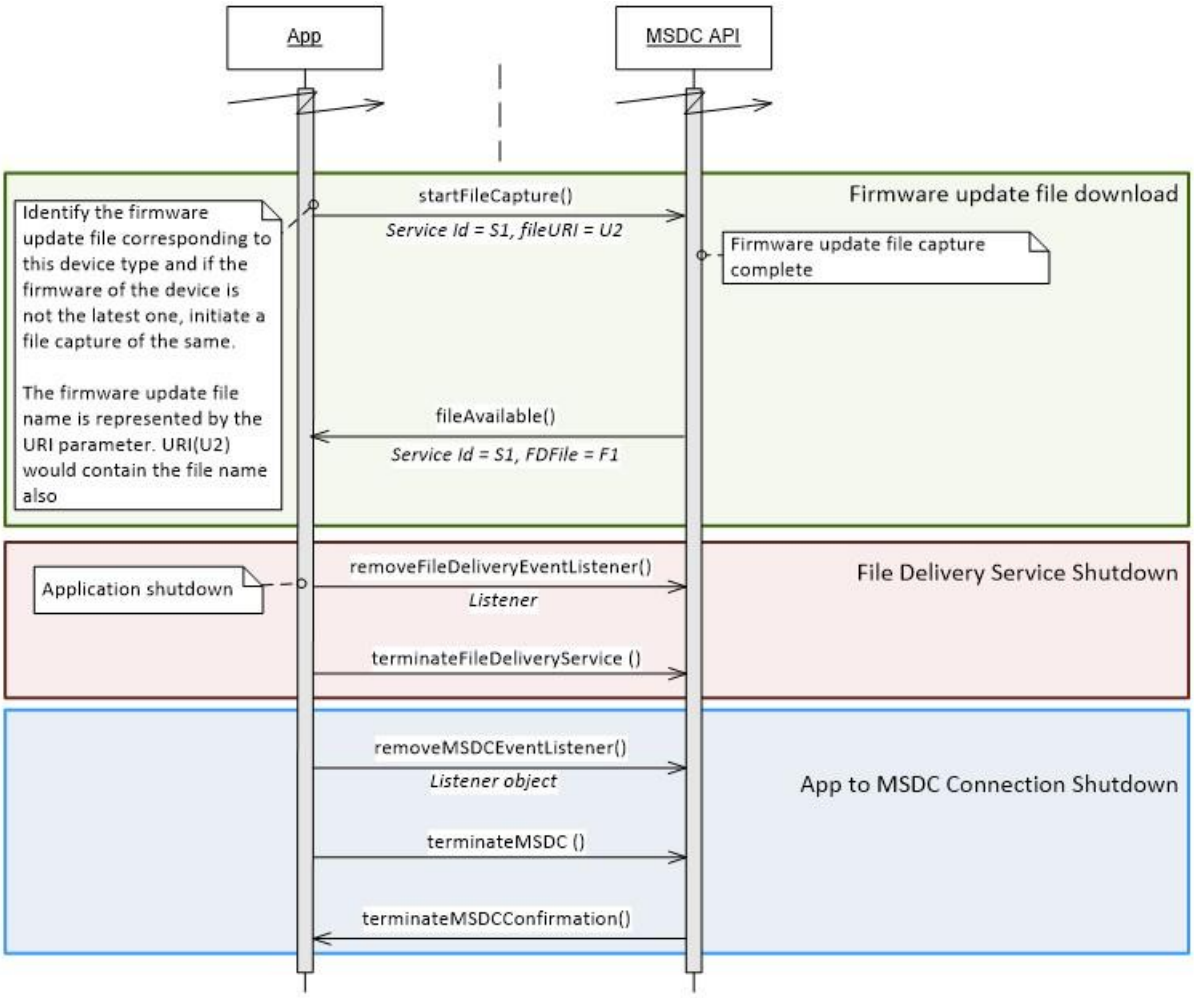


Figure 8-10 Use case - Weekly magazine application (3 of 3)

8.5 Enterprise Group application

A typical Enterprise Group Call application allows the user to be part of Group Call Services and communicate with the members of the Group.

Upon starting the app, a user typically sees a list of available Group Call services at their present location. Based on their selection, the app activates the selected Group Call for the user, i.e., the user becomes part of the selected Group Call.

The following use case call flow assumes that the user:

1. Opens the app.
2. Selects a Group Call service from a list of available services.
3. Participates in the Group Call service.
   (a) Consumes the downlink portion of the selected Group Call service via eMBMS broadcast.
   (b) Uplink portion of the selected Group Call service is via Unicast.
4. Exits the app.

Call flow

A typical Enterprise Group Call application takes the following sequence:

- Figure 7-1 describes the call flows observed during the initialization phase of the eMBMS Broadcast module.
- Figure 7-2 describes the call flows observed when a selected group call service is activated to consume the downlink of the Group Call service via eMBMS broadcast.
- Figure 7-3 describes the call flows observed when the app is closed.

8.6 Public Safety Group Call application

A typical Public Safety Group Call app allows the user to be part of Emergency Group Call Services and communicate with members of the group to participate and coordinate in rescue/relief activities during various emergencies.

Upon starting the app, a user typically sees a list of available, on-going emergency group calls at their present location. Based on their selection, the app activates the selected emergency group call for the user, i.e., the user becomes part of the selected ongoing group call and he can then start communicating with the members of the group calls in Unicast mode. Typically, an emergency group call starts as Unicast group calls for both the uplink and downlink. The network may broadcast the downlink portion of the group call if/when needed.

A Public Safety Group Call app periodically gets an SAI List from the modem, which it then reports to the Group Call Server in the network via Unicast mode. Based on the location information of the group call participants, the Group Call Server may transition the call to broadcast and inform the app/user of the downlink availability for the selected emergency group call on a TMGI via broadcast within a list of SAIs.

If the device is within the SAIs where the group call is broadcast, the app/user activates the TMGI and it receives the selected emergency group call downlink on eMBMS.

The following use case call flow assumes that the user:

1. Opens the app.
2. Selects an emergency group call, joins the call, and actively participates in the group call via Unicast mode.
3. App periodically gets the SAI List from the modem and reports it to the Group Call Server via Unicast mode.
4. Group Call Server informs the app about downlink availability for the selected emergency group call in a list of SAIs and frequencies on a TMGI.
5. If there is downlink availability, the user/app activates the TMGI for consuming the downlink of selected emergency group call via eMBMS Broadcast.
6. Exits the app.

Call flow

A typical Public Safety Group Call application takes the following sequence:

- Figure 7-1 describes the call flow for the initialization phase of the app. After initialization, the app starts receiving an SAI List from the modem, either periodically or whenever there is a change in the cell due to mobility/contents of the SAI List.
- Figure 7-2 describes the call flow for TMGI activation in the app. After activation, the app presents the downlink of the emergency group call for user consumption.
- Figure 7-3 describes the call flow for the termination of the emergency group call service in the app.

9 Class Documentation

9.1 FileInfo

9.1.1 Class Documentation

9.1.1.1 class com::qualcomm::ltebc::aidl::FileInfo

Data fields

Type	Parameter	Description
String	uri	URI of the file.
String	downloaded-Location	The physical location of the downloaded file on the SD card.
String	downloadHttp-Url	The HTTP URL of the downloaded file.
String	contentType	Multipurpose Internet Mail Extension (MIME) file type.
String	mode	For future use, not supported at this time.
Time	cacheControl-Expires	cacheControlExpires is a cacheControl-related parameter. The application uses Cache directives to manage the length of time to retain files. This parameter defines the expected expiration time for a specific file (or set of files) so it can be deleted at that time. **Note:** cacheControlExpires object is NULL if cacheControlMaxStale or cacheControlNoCache is TRUE.
boolean	cacheControl-MaxStale	cacheControlMaxStale is a cacheControl-related parameter. The application uses Cache directives to manage the length of time to retain files. When cacheControlMaxStale is set to TRUE, the file should be cached for an indefinite period of time, if possible. The file has no expiration date.
boolean	cacheControl-NoCache	cacheControlNoCache is a cacheControl-related parameter. The application uses Cache directives to manage the length of time to retain files. When cacheControlNoCache is set to TRUE, the file (or set of files) will not be cached. This can be useful if file changes occur often, or if the file will only be used once by the receiver application.
long	fileSize	Size of the file in bytes.
String	md5	MD5 checksum of the downloaded file.
Time	fileDeleteTime	Time at which file is deleted.
Uri	contentUri	Content provider URI of the file.

9.2 ServiceInfo

9.2.1 Class Documentation

9.2.1.1 class com::qualcomm::ltebc::aidl::ServiceInfo

Data fields

Type	Parameter	Description
List< Service-Name >	serviceName-List	List of language and name. The list is user-displayable info.
String	serviceClass	serviceClass is a URN and identifies a set of services for a given application.
String	serviceId	serviceId is a URN and is unique to the service.
String	service-Language	serviceLanguage shows language of service.
int	serviceHandle	serviceHandle is an integer and unique to the service.
int	service-Availability	serviceAvailability shows whether or not a service is available: • STREAMING_SERVICE_AVAILABLE = 0 • STREAMING_SERVICE_NOT_AVAILABLE_IN_BC = 1 • STREAMING_SERVICE_NOT_AVAILABLE = 2
String	mpdUri	MPD URI for DASH (only for streaming services).
List< String >	fileUriList	Contains File URIs list of File Delivery Service.
List< Integer >	service-Announcement-ServiceAreaId-List	Contains list of service announcement service areas.
Date	sessionStart-Time	Contains session start time.
Date	sessionEnd-Time	Contains session end time.

9.2.1.2 class com::qualcomm::ltebc::aidl::ServiceInfo::ServiceName

Data fields

Type	Parameter	Description
String	name	Name of the service.
String	lang	Service language.

9.3 GroupItem

9.3.1 Class Documentation

9.3.1.1 class com::qualcomm::ltebc::aidl::GroupItem

Public Member Functions

- String getGroupName ()
- List< Integer > getServiceAreaIdList ()
- List< Integer > getServiceHandleList ()

9.3.1.1.1 Member Function Documentation

9.3.1.1.1. 9 com.qualcomm.ltebc.aidl.GroupItem.getGroupName ( ) String

Returns group name of GroupItem.

9.3.1.1.1. 9 com.qualcomm.ltebc.aidl.GroupItem.getServiceArealdList ( ) List<Integer>

Returns Service Area ID List from GroupItem.

Returns

List<Integer>.

9.3.1.1.1. 9 com.qualcomm.ltebc.aidl.GroupItem.getServiceHandleList ( ) List<Integer>

Returns Service Handle List from GroupItem.

Returns

List<Integer>.

9.4 App Constants

This section describes the status message, error, and warning codes of the application.

9.4.1 Class Documentation

9.4.1.1 class com::qualcomm::msdc::AppConstants

This class contains all the constants used in the application.

Static Public Attributes

- static final int ERROR_MSDC_MINIMUM_VALUE = 10000
- static final int ERROR_MSDC_UNABLE_TO_INITIALIZE = 10000
- static final int ERROR_MSDC_SERVICE_UNAVAILABLE = 10001
- static final int ERROR_MSDC_VERSION_INCOMPATIBLE = 10002
- static final int ERROR_MSDC_UNKNOWN = 10003
- static final int ERROR_MSDC_UNABLE_TO_LOCATE_PROVISIONING = 10004
- static final int ERROR_MSDC_PROVISIONING_FILE_INCOMPATIBILITY = 10005
- static final int ERROR_MSDC_UNABLE_TO_PARSE_PROVISIONING_FILE = 10006
- static final int ERROR_MSDC_UNABLE_TO_ALLOCATE_MEMORY = 10007
- static final int ERROR_MSDC_UNABLE_TO_BIND_PORT = 10008
- static final int ERROR_MSDC_TIME_SYNC = 10009
- static final int ERROR_MSDC_MODEM_VERSION_INCOMPATIBLE = 10010
- static final int ERROR_MSDC_BOOTSTRAP_CONNECTION_FAIL = 10011
- static final int ERROR_MSDC_UNABLE_TO_PARSE_BOOTSTRAP = 10012
- static final int ERROR_MSDC_NOT_CONNECTED_TO_HOMECARRIER_LTE = 10016
- static final int ERROR_MSDC_FAILED_BOOTSTRAP_SERVER_DISCOVERY = 10017
- static final int ERROR_MSDC_SIM_READ = 10018
- static final int ERROR_MSDC_UNABLE_TO_ENABLE_MODEM = 10019
- static final int ERROR_MSDC_MIDDLEWARE_NOT_INSTALLED = 10020
- static final int ERROR_MSDC_CARRIER_CHANGE_NOT_ALLOWED = 10021
- static final int ERROR_MSDC_APP_PERMISSIONS_NOT_GRANTED = 10022
- static final int ERROR_MSDC_EMBMS_SERVICE_NOT_AVAILABLE = 10023
- static final int ERROR_MSDC_MAXIMUM_VALUE = 15000
- static final int WARNING_MSDC_MINIMUM_VALUE = 15001
- static final int WARNING_MSDC_NETWORK_CHANGE_DETECTED = 15001

- static final int WARNING_MSDC_REDUCED_FEATURE_SET = 15003
- static final int WARNING_MSDC_MAXIMUM_VALUE = 19999
- static final int ERROR_NW_MINIMUM_VALUE = 20000
- static final int ERROR_NW_UNABLE_TO_INITIALIZE = 20000
- static final int ERROR_NW_SERVICE_UNAVAILABLE = 20001
- static final int ERROR_NW_UNABLE_TO_ENABLE_SIGNAL_LEVEL = 20002
- static final int ERROR_NW_UNABLE_TO_DISABLE_SIGNAL_LEVEL = 20003
- static final int ERROR_NW_MAXIMUM_VALUE = 25000
- static final int WARNING_NW_MINIMUM_VALUE = 25001
- static final int WARNING_NW_MAXIMUM_VALUE = 29999
- static final int ERROR_S_MINIMUM_VALUE = 30000
- static final int ERROR_S_UNABLE_TO_INITIALIZE = 30000
- static final int ERROR_S_SERVICE_UNAVAILABLE = 30001
- static final int ERROR_S_UNABLE_TO_START_SERVICE = 30002
- static final int ERROR_S_UNABLE_TO_STOP_SERVICE = 30003
- static final int ERROR_S_SERVICE_RESET = 30004
- static final int ERROR_S_FREQUENCY_CONFLICT = 30005
- static final int ERROR_S_CONCURRENT_SERVICE_LIMIT_REACHED = 30006
- static final int ERROR_S_SERVICE_CLASS_INITIALIZATION_FAILED = 30007
- static final int ERROR_S_INVALID_SERVICE_ID = 30008
- static final int ERROR_S_BEARER_UNAVAILABLE_AT_INITIATION = 30009
- static final int ERROR_S_END_OF_SESSION = 30010
- static final int ERROR_S_MAXIMUM_VALUE = 35000
- static final int WARNING_S_MINIMUM_VALUE = 35001
- static final int WARNING_S_MAXIMUM_VALUE = 39999
- static final int ERROR_FD_MINIMUM_VALUE = 40000
- static final int ERROR_FD_UNABLE_TO_INITIALIZE = 40000
- static final int ERROR_FD_SERVICE_UNAVAILABLE = 40001
- static final int ERROR_FD_UNABLE_TO_DELETE_FILE = 40002
- static final int ERROR_FD_UNABLE_TO_DELETE_ALL_FILES = 40003
- static final int ERROR_FD_SERVICE_ALREADY_USED_BY_ANOTHER_APP = 40004
- static final int ERROR_FD_INVALID_SERVICE_ID = 40005
- static final int ERROR_FD_SERVICE_RESET = 40006

- static final int ERROR_FD_SERVICE_CLASS_INITIALIZATION_FAILED = 40007
- static final int ERROR_FD_UNABLE_TO_START_SERVICE = 40008
- static final int ERROR_FD_UNABLE_TO_STOP_SERVICE = 40009
- static final int ERROR_FD_MAXIMUM_VALUE = 45000
- static final int WARNING_FD_MINIMUM_VALUE = 45001
- static final int WARNING_FD_FREQUENCY_CONFLICT = 45002
- static final int WARNING_FD_STALLED = 45003
- static final int WARNING_FD_STORAGE_LOCATION_COPY_FAILED = 45004
- static final int WARNING_FD_MAXIMUM_VALUE = 50000
- static final int ERROR_GC_UNABLE_TO_INITIALIZE = 51001
- static final int ERROR_GC_UNABLE_TO_START_SERVICE = 51002
- static final int ERROR_GC_UNABLE_TO_STOP_SERVICE = 51003
- static final int IN_COVERAGE = 0
- static final int OUT_OF_COVERAGE = 1
- static final int E911_OUT = 0
- static final int E911_IN = 1
- static final int IN_NETWORK = 1
- static final int OUT_OF_NETWORK = 2
- static final int MSDC_NONE = 0
- static final int MSDC_NO_WIFI = 1
- static final int MSDC_REMOTE_AVAILABLE = 2
- static final int MSDC_LOCAL = 1
- static final int MSDC_REMOTE = 2
- static final int STREAMING_SERVICE_AVAILABLE = 0
- static final int STREAMING_SERVICE_NOT_AVAILABLE_IN_BC = 1
- static final int STREAMING_SERVICE_NOT_AVAILABLE = 2

9.4.1.1.1 Member Data Documentation

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_MINIMUM_VALUE = 10000

Minimum value of MSDC error code range.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_UNABLE_TO_INITIALIZE = 10000**

Unable to initialize MSDC.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_SERVICE_UNAVAILABLE = 10001**

MSDC service is unavailable.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_VERSION_INCOMPATIBLE = 10002**

MSDC version is incompatible.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_UNKNOWN = 10003**

MSDC is unknown.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_UNABLE_TO_LOCATE_PROVI- SIONING = 10004**

MSDC is unable to locate provisioning file.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_PROVISIONING_FILE_INCOM-PATIBILITY = 10005**

MSDC provisioning file is incompatible.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_UNABLE_TO_PARSE_PROVIS-IONING_FILE = 10006**

MSDC is unable to parse provisioning file.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_UNABLE_TO_ALLOCATE_ME- MORY = 10007**

MSDC is unable to allocate memory to the service.

9.4.1.1.1. **9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_UNABLE_TO_BIND_PORT = 10008**

MSDC is unable to bind the port.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_TIME_SYNC = 10009

MSDC time sync error.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_MODEM_VERSION_INCOMP- ATIBLE = 10010

MSDC modem version is incompatible.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_BOOTSTRAP_CONNECTION- _FAIL = 10011

MSDC bootstrap connection failed.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_UNABLE_TO_PARSE_BOOT- STRAP = 10012

MSDC is unable to parse bootstrap file.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_NOT_CONNECTED_TO_HOM- ECARRIER_LTE = 10016

UE not connected to home carrier LTE network.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_FAILED_BOOTSTRAP_SERV- ER_DISCOVERY = 10017

MSDC ailed to discover bootstrap server.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_SIM_READ = 10018

MSDC failed to get SIM carrier PLMN information.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_UNABLE_TO_ENABLE_MOD- EM = 10019

MSDC is unable to enable modem.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_MIDDLEWARE_NOT_INSTAL- LED = 10020

MSDC middleware is not installed.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_CARRIER_CHANGE_NOT_AL- LOWED = 10021

MSDC carrier change is not allowed.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_APP_PERMISSIONS_NOT_G- RANTED = 10022

MSDC app permissions are not granted.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_EMBMS_SERVICE_NOT_AVA- ILABLE = 10023

MSDC eMBMS service is not available.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_MSDC_MAXIMUM_VALUE = 15000

Maximum value of MSDC error code range.

If an error is not within this range, it is not an MSDC error.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.WARNING_MSDC_MINIMUM_VALUE = 15001

Minimum value of MSDC warning code range.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.WARNING_MSDC_NETWORK_CHANGE_DET- ECTED = 15001

Warning: MSDC network change detected.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.WARNING_MSDC_REDUCED_FEATURE_SET = 15003

Warning: MSDC reduced feature set.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.WARNING_MSDC_MAXIMUM_VALUE = 19999

Maximum value of MSDC warning code range.

If a warning is not within this range, it is not an MSDC warning.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_NW_MINIMUM_VALUE = 20000

Minimum value of Network service error range.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_NW_UNABLE_TO_INITIALIZE = 20000

Unable to initialize Network service.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_NW_SERVICE_UNAVAILABLE = 20001

Network service is unavailable .

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_NW_UNABLE_TO_ENABLE_SIGNAL-_LEVEL = 20002

Network service unable to enable signal level.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_NW_UNABLE_TO_DISABLE_SIGNA- L_LEVEL = 20003

Network service unable to disable signal level.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_NW_MAXIMUM_VALUE = 25000

Maximum value of Network service error code range.

If an error is not within this range, it is not a Network error.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.WARNING_NW_MINIMUM_VALUE = 25001

Minimum value of Network service warning code range.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.WARNING_NW_MAXIMUM_VALUE = 29999

Maximum value of Network service warning code range.

If a warning is not within this range, it is not a Network warning.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_S_MINIMUM_VALUE = 30000

Minimum value of Streaming service error code range.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_S_UNABLE_TO_INITIALIZE = 30000

Unable to initialize Streaming service.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_SERVICE_UNAVAILABLE = 30001

Streaming service is unavailable.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_UNABLE_TO_START_SERVICE = 30002

Streaming service is unable to start.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_UNABLE_TO_STOP_SERVICE = 30003

Streaming service is unable to stop.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_SERVICE_RESET = 30004

Error in Streaming service reset.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_FREQUENCY_CONFLICT = 30005

Streaming frequency conflicts with another Streaming service.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_CONCURRENT_SERVICE_LIMIT_- REACHED = 30006

Concurrent Streaming service limit reached.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_SERVICE_CLASS_INITIALIZATIO- N_FAILED = 30007

Streaming service class initialization failed.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_INVALID_SERVICE_ID = 30008

Invalid Streaming service ID.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_S_BEARER_UNAVAILABLE_AT_INIT- IATION = 30009

Streaming bearer is unavailable at initiation.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_S_END_OF_SESSION = 30010

End of Streaming session.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_S_MAXIMUM_VALUE = 35000

Maximum value of Streaming error code range.

If an error is not within this range, it is not a Streaming error

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.WARNING_S_MINIMUM_VALUE = 35001

Minimum value of Streaming service warning code range.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.WARNING_S_MAXIMUM_VALUE = 39999

Maximum value of Streaming service warning code range.

If a warning is not within this range, it is not Streaming warning.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_MINIMUM_VALUE = 40000

Minimum value of File Delivery service error code range.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_UNABLE_TO_INITIALIZE = 40000

Unable to initialize File Delivery service.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_SERVICE_UNAVAILABLE = 40001

File Delivery service is unavailable.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_UNABLE_TO_DELETE_FILE = 40002

File Delivery service is unable to delete a single file.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_UNABLE_TO_DELETE_ALL_FIL- ES = 40003

File Delivery service is unable to delete all files.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_SERVICE_ALREADY_USED_BY_- ANOTHER_APP = 40004

File Delivery service is in use by another application.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_INVALID_SERVICE_ID = 40005

Invalid File Delivery service ID.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_SERVICE_RESET = 40006

Error in File Delivery service reset.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_SERVICE_CLASS_INITIALIZATI-ON_FAILED = 40007

File Delivery service class initialization failed.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_UNABLE_TO_START_SERVICE = 40008

Unable to start File Delivery service.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_UNABLE_TO_STOP_SERVICE = 40009

File Delivery service is unable to stop.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.ERROR_FD_MAXIMUM_VALUE = 45000

Maximum value of File Delivery error code range.

If an error is not within this range, it is not a File Delivery error.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.WARNING_FD_MINIMUM_VALUE = 45001

Minimum value of File Delivery service warning code range.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.WARNING_FD_FREQUENCY_CONFLICT = 45002

Warning: Frequency conflict with another file download service.

9.4.1.1.1. 9 final int com.qualcomm.msdc.AppConstants.WARNING_FD_STALLED = 45003

Warning: File Delivery service stalled.

9.4.1.1.1. 9 final int

com.qualcomm.msdc.AppConstants.WARNING_FD_STORAGE_LOCATION_COPY-_FAILED = 45004

Warning: File Delivery service failed to copy file in storage location path.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.WARNING_FD_MAXIMUM_VALUE = 50000

Maximum value of File Delivery service warning code range.

If a warning is not within this range, it is not File Delivery warning.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_GC_UNABLE_TO_INITIALIZE = 51001

Unable to initialize Group Call service.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_GC_UNABLE_TO_START_SERVICE = 51002

Unable to start Group Call service.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.ERROR_GC_UNABLE_TO_STOP_SERVICE = 51003

Unable to stop Group Call service.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.IN_COVERAGE = 0

Broadcast service is in coverage.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.OUT_OF_COVERAGE = 1

Broadcast service is out of coverage.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.E911_OUT = 0

Broadcast service is supported (normal scenario).

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.E911_IN = 1

Emergency call is taking place.

Broadcast service is not supported.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.IN_NETWORK = 1

Broadcast service is supported.

9.4.1.1.1.9 final int com.qualcomm.msdc.AppConstants.OUT_OF_NETWORK = 2

Broadcast service is not supported.

9.4.1.1.1.1.9 final int com.qualcomm.msdc.AppConstants.MSDC_NONE = 0

No MSDC middleware found.

9.4.1.1.1.1.9 final int com.qualcomm.msdc.AppConstants.MSDC_NO_WIFI = 1

Remote MSDC middleware is not connected to Wi-Fi.

9.4.1.1.1.1.9 final int com.qualcomm.msdc.AppConstants.MSDC_REMOTE_AVAILABLE = 2

Remote MSDC middleware is available.

9.4.1.1.1.1.9 final int com.qualcomm.msdc.AppConstants.MSDC_LOCAL = 1

UI application is connected to local MSDC middleware found on the same device as the UI.

9.4.1.1.1.1.9 final int com.qualcomm.msdc.AppConstants.MSDC_REMOTE = 2

UI application is connected to remote MSDC middleware found on mobile broadband product.

9.4.1.1.1.1.9 final int com.qualcomm.msdc.AppConstants.STREAMING_SERVICE_AVAILABLE = 0

Streaming service is available via broadcast and unicast.

9.4.1.1.1.1.9 final int com.qualcomm.msdc.AppConstants.STREAMING_SERVICE_NOT_AVAILABLE_IN-_BC = 1

Streaming service is not available via broadcast, but may be available via unicast.

9.4.1.1.1.1.9 final int com.qualcomm.msdc.AppConstants.STREAMING_SERVICE_NOT_AVAILABLE = 2

Streaming service is not available via broadcast and unicast.

9.5 IMSDCAppManager

9.5.1 Class Documentation

9.5.1.1 interface com::qualcomm::msdc::IMSDCAppManager

Public Member Functions

- void initializeMSDC (MSDCAppManagerInitParams params)
- void terminateMSDC ()
- IMSDCFileDeliveryController getFileDeliveryController ()
- IMSDCStreamingController getStreamingController ()
- IMSDCNetworkController getNetworkController ()
- IMSDCGroupCallController getGroupCallController ()
- IMSDCStreamingModel getStreamingModel ()
- IMSDCFileDeliveryModel getFileDeliveryModel ()
- IMSDCNetworkModel getNetworkModel ()
- IMSDCGroupCallModel getGroupCallModel ()
- void addMSDCEventListener (IMSDCAppManagerEventListener listener)
- void removeMSDCEventListener (IMSDCAppManagerEventListener listener)
- boolean isMSDCInitialized ()
- String getAPIVersion ()
- String getServerAPIVersion ()

9.5.1.1.1 Member Function Documentation

9.5.1.1.1.9 void com.qualcomm.msdc.IMSDCAppManager.initializeMSDC (MSDCAppManagerInit- Params *params*)

Initializes the MSDC.

Parameters

params	As defined in MSDCAppManagerInitParams.

9.5.1.1.1.9 void com.qualcomm.msdc.IMSDCAppManager.terminateMSDC ( )

Terminates the MSDC component.

9.5.1.1.1. 9 IMSDCFileDeliveryController com.qualcomm.msdc.IMSDCAppManager.getFileDelivery- Controller ( )

Provides a File Delivery Controller instance.

Returns

IMSDCFileDeliveryController File Delivery Controller instance.

9.5.1.1.1. 9 IMSDCStreamingController com.qualcomm.msdc.IMSDCAppManager.getStreaming- Controller ( )

Provides a Streaming Controller instance.

Returns

IMSDCStreamingController Streaming controller Instance.

9.5.1.1.1. 9 IMSDCNetworkController com.qualcomm.msdc.IMSDCAppManager.getNetworkController ( )

Provides a Network Controller instance.

Returns

IMSDCNetworkController Network Controller instance.

9.5.1.1.1. 9 IMSDCGroupCallController com.qualcomm.msdc.IMSDCAppManager.getGroupCall- Controller ( )

Provides a Group Call Controller instance.

Returns

IMSDCGroupCallController Group Call Controller instance.

9.5.1.1.1. 9 IMSDCStreamingModel com.qualcomm.msdc.IMSDCAppManager.getStreamingModel ( )

Provides a Streaming instance.

Returns

IMSDCStreamingModel Streaming Model Instance.

9.5.1.1.1.9 IMSDCFileDeliveryModel com.qualcomm.msdc.IMSDCAppManager.getFileDeliveryModel ( )

Provides a File Delivery Model instance.

Returns

IMSDCFileDeliveryModel File Delivery Model Instance.

9.5.1.1.1.9 IMSDCNetworkModel com.qualcomm.msdc.IMSDCAppManager.getNetworkModel ( )

Provides a Network Model instance.

Returns

IMSDCNetworkModel Network Model Instance.

9.5.1.1.1.9 IMSDCGroupCallModel com.qualcomm.msdc.IMSDCAppManager.getGroupCallModel ( )

Provides a Group Call Model instance.

Returns

IMSDCGroupCallModel Group Call Model Instance.

9.5.1.1.1.9 void com.qualcomm.msdc.IMSDCAppManager.addMSDCEventListener (IMSDCApp- ManagerEventListener *listener*)

Adds MSDC controller event listener to listen for MSDC controller events.

Parameters

listener	The event listener.

9.5.1.1.1.9 void com.qualcomm.msdc.IMSDCAppManager.removeMSDCEventListener (IMSDCApp- ManagerEventListener *listener*)

Removes MSDC controller event listener from listening for MSDC controller events.

Parameters

listener	The event listener.

9.5.1.1.1.9 com.qualcomm.msdc.IMSDCAppManager.isMSDCInitialized() boolean

Indicates whether MSDC has been initialized or not.

Returns

TRUE if MSDC is initialized.

9.5.1.1.1.9 com.qualcomm.msdc.IMSDCAppManager.getAPIVersion() String

Returns MSDC version.

Returns

String MSDC API version.

9.5.1.1.1.9 com.qualcomm.msdc.IMSDCAppManager.getServerAPIVersion() String

Returns server version.

Returns

String server API version.

9.6 MSDCAppManager

9.6.1 Function Documentation

9.6.1.1 static IMSDCAppManager com.qualcomm.msdc.MSDCAppManager.getInstance ( )

Returns instance for MSDCAppManager.

Returns

IMSDCAppManager

9.7 MSDCApplication

9.7.1 Class Documentation

9.7.1.1 class com::qualcomm::msdc::MSDCApplication

MSDCApplication is an initializer class for the MSDC API library that provides context and the global state for the API library.

An application's context is passed to MSDCApplication by onCreate() of an overridden application or base activity. This instantiates MSDCApplication with valid context when the process for your application/package is created.

Public Member Functions

- MSDCApplication (Context context)

Static Public Member Functions

- static Context getAppContext ()

9.7.1.1.1 Constructor & Destructor Documentation

9.7.1.1.1. 9 com.qualcomm.msdc.MSDCApplication.MSDCApplication (Context *context*)

Instantiates a new MSDC application.

Parameters

context	The context of the new application.

9.7.1.1.2 Member Function Documentation

9.7.1.1.2. 9 com.qualcomm.msdc.MSDCApplication.getAppContext ( ) static Context

Any component in the application can get the application context from this application class.

Returns

Application context.

9.8 IMSDCAppManagerEventListener

9.8.1 Class Documentation

9.8.1.1 interface com::qualcomm::msdc::controller::IMSDCAppManagerEventListener

Public Member Functions

- void initializeMSDCConfirmation ()
- void msdcError (int errorCode, String message)
- void msdcWarning (int warningCode, String message)
- void terminateMSDCConfirmation ()
- void e911Indication (int state)
- void msdcUnavailableNotification (int reason)
- void msdcAvailableNotification (int which, Object obj)

9.8.1.1.1 Member Function Documentation

9.8.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCAppManagerEventListener.initializeMSDC-Confirmation ( )

Notifies the UI once MSDC initialization is complete.

9.8.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCAppManagerEventListener.msdcError (int *errorCode*, String *message*)

Notifies that an error has occurred.

Parameters

errorCode	Error codes specified in AppConstants documentation.
message	String error message.

9.8.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCAppManagerEventListener.msdcWarning (int *warningCode*, String *message*)

This method notifies that a warning has occurred.

Parameters

warningCode	Warning codes specified in AppConstants documentation.
message	String warning message.

9.8.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCAppManagerEventListener.terminateMSDC-Confirmation ( )

Notifies the UI once MSDC termination is complete.

9.8.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCAppManagerEventListener.e911Indication (int *state*)

Notifies the UI about the state of E911.

Parameters

state	E911_OUT is 0; E911_IN is 1.

9.8.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCAppManagerEventListener.msdcUnavailable-Notification (int *reason*)

Roaming notification.

Parameters

reason	• 0 = MSDC_NONE • 1 = MSDC_NO_WIFI • 2 = MSDC_REMOTE_AVAILABLE

9.8.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCAppManagerEventListener.msdcAvailable-Notification (int *which,* Object *obj*)

Roaming notification.

Parameters

which	• 1 = MSDC_LOCAL • 2 = MSDC_REMOTE
obj	NULL - Reserved for future use.

9.9 IMSDCStreamingController

9.9.1 Class Documentation

9.9.1.1 interface com::qualcomm::msdc::controller::IMSDCStreamingController

Public Member Functions

- void initializeStreamingService (StreamingInitParams params)
- void terminateStreamingService ()
- void startStreamingService (int serviceId)
- void switchStreamingService (int fromServiceId, int toServiceId)
- void stopStreamingService (int serviceId)
- void addStreamingEventListener (IMSDCStreamingControllerEventListener listener)
- void removeStreamingEventListener (IMSDCStreamingControllerEventListener listener)

9.9.1.1.1 Member Function Documentation

9.9.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCStreamingController.initializeStreamingService (StreamingInitParams *params*)

Initializes the MSDC for Streaming services.

Parameters

params	As defined in StreamingInitParams.

9.9.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCStreamingController.terminateStreamingService ( )

Terminates Streaming service-related communication with the MSDC.

9.9.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCStreamingController.startStreamingService (int *serviceId*)

Starts Streaming service.

Parameters

serviceId	ID of the Streaming service to be started

9.9.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCStreamingController.switchStreamingService (int *fromServiceId,* int *toServiceId*)

Starts Streaming service with toServiceId and stops Streaming service with fromServiceId.

Parameters

fromServiceId	ID of the Streaming service to be stopped.
toServiceId	ID of the Streaming service to be started.

9.9.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCStreamingController.stopStreamingService (int *serviceId*)

Stops Streaming service.

Parameters

serviceId	ID of the Streaming service to be stopped.

9.9.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCStreamingController.addStreamingEventListener (IMSDCStreamingControllerEventListener *listener*)

Adds MSDC controller event listener to listen for MSDC controller events.

Parameters

listener	The event listener.

9.9.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCStreamingController.removeStreamingEvent-Listener (IMSDCStreamingControllerEventListener *listener*)

Removes MSDC controller event listener from listening for MSDC controller events.

Parameters

listener	The event listener.

9.10 IMSDCStreamingControllerEventListener

9.10.1 Class Documentation

9.10.1.1 interface com::qualcomm::msdc::controller::IMSDCStreamingControllerEventListener

This interface must be implemented by VIEW.

It provides a list of events generated from the MSDC Streaming controller based on the actions sent by VIEW. All these events are an asynchronous response to the corresponding actions listed in MSDCStreamingController.

Public Member Functions

- void streamingServiceInitializeConfirmation ()
- void streamingServiceListUpdate ()
- void streamingServiceStarted (int serviceId)
- void streamingServiceStopped (int serviceId)
- void streamingServiceStalled (int serviceId)
- void mpdUpdated (int serviceId)
- void streamingServiceError (int errorCode, String message, Integer serviceId)
- void streamingServiceWarning (int warningCode, String warningMsg, Integer serviceId)
- void resetStreamingNotification ()

9.10.1.1.1 Member Function Documentation

9.10.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.streaming-ServiceInitializeConfirmation ( )

Notifies the event listener that the MSDC controller is initialized for Streaming services.

9.10.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.streaming-ServiceListUpdate ( )

Notifies that the updated Streaming service list is available.

9.10.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.streaming-ServiceStarted (int *serviceId*)

Notifies that the Streaming service has started.

Parameters

serviceId	Service ID of the started service.

9.10.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.streaming-ServiceStopped (int *serviceId*)

Notifies that the Streaming service has stopped.

Parameters

serviceId	Service ID of the stopped service.

9.10.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.streaming-ServiceStalled (int *serviceId*)

Notifies that the Streaming service is stalled.

Parameters

serviceId	Service ID of the stalled service

9.10.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.mpdUpdated (int *serviceId*)

Notifies that MPD is updated.

Parameters

serviceId	Service ID of the service whose MPD is updated.

9.10.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.streaming-ServiceError (int *errorCode*, String *message*, Integer *serviceId*)

Notifies that an error has occurred.

Parameters

errorCode	Error codes specified in AppConstants documentation
message	String error message.
serviceId	Service ID of the Streaming service.

9.10.1.1.1.9 void

com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.-streamingServiceWarning (int *warningCode*, String *warningMsg*, Integer *serviceId*)

Notifies that a warning has occurred.

Parameters

warningCode	Warning codes specified in AppConstants documentation.
warningMsg	String warning message.
serviceId	Service ID of the Streaming service.

9.10.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCStreamingControllerEventListener.reset-StreamingNotification ( )

Resets notification to UI if there is any corruption in Streaming service information in the MSDC.

If the UI application cached the serviceHandles, it must reset them after receiving this notification.

9.11 IMSDCFileDeliveryController

9.11.1 Class Documentation

9.11.1.1 interface com::qualcomm::msdc::controller::IMSDCFileDeliveryController

Public Member Functions

- void initializeFileDeliveryService (FileDeliveryInitParams params)
- void terminateFileDeliveryService (FileDeliveryTerminateParams params)
- void startFileCapture (int serviceId)
- void startFileCapture (int serviceId, String fileURI)
- void startFileCapture (FileCaptureParams params)
- void stopFileCapture (int serviceId)
- void stopFileCapture (int serviceId, String fileURI)
- void deleteFile (int serviceId, String fileURI)
- void deleteAllCapturedFiles (int serviceId)
- void setStorageLocation (String path)
- void addFileDeliveryEventListener (IMSDCFileDeliveryControllerEventListener listener)
- void removeFileDeliveryEventListener (IMSDCFileDeliveryControllerEventListener listener)

9.11.1.1.1 Member Function Documentation

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.initializeFileDelivery-Service (FileDeliveryInitParams *params*)

Initializes the MSDC for File Delivery services.

Parameters

params	As defined in FileDeliveryInitParams.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.terminateFileDelivery-Service (FileDeliveryTerminateParams *params*)

Terminates File Delivery service-related communication with the MSDC.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.startFileCapture (int *serviceId*)

Starts the passed File Delivery service.

Parameters

serviceId	ID of the FD service to be started

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.startFileCapture (int *serviceId*, String *fileURI*)

Starts file capture.

Parameters

serviceId	The service ID of the file to capture.
fileURI	The file URI.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.startFileCapture (FileCaptureParams *params*)

Starts file capture.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.stopFileCapture (int *serviceId*)

Stops the passed File Delivery service.

Parameters

serviceId	ID of the File Delivery service to be stopped.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.stopFileCapture (int *serviceId*, String *fileURI*)

Stops file capture.

Parameters

serviceId	The service ID of the file.
fileURI	The file URI.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.deleteFile (int *serviceId*, String *fileURI*)

Deletes the file.

Parameters

serviceId	The service ID of the file to be deleted.
fileURI	The file URI.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.deleteAllCapturedFiles (int *serviceId*)

Deletes all captured files.

Parameters

serviceId	The service ID of all captured files.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.setStorageLocation (String *path*)

Sets storage location (File Download Location)

Parameters

path	Storage location path.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.addFileDeliveryEvent-Listener (IMSDCFileDeliveryControllerEventListener *listener*)

Adds MSDC File Delivery controller event listener to listen for MSDC controller events.

Parameters

listener	The event listener.

9.11.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryController.removeFileDelivery-EventListener (IMSDCFileDeliveryControllerEventListener *listener*)

Removes MSDC File Delivery controller event listener from listening for MSDC controller events.

Parameters

listener	The event listener.

9.12 IMSDCFileDeliveryControllerEventListener

9.12.1 Class Documentation

9.12.1.1 interface com::qualcomm::msdc::controller::IMSDCFileDeliveryController-EventListener

Public Member Functions

- void fileDeliveryServiceInitializeConfirmation ()
- void fileDeliveryServiceError (int errorCode, String message, Integer serviceId)
- void fileDeliveryServiceWarning (int warningCode, String message, Integer serviceId)
- void fileDeliveryServiceListUpdate ()
- void fileAvailable (int serviceId, FDFile file)
- void resetFileDeliveryNotification ()
- void fileDownloadFailure (int serviceHandle, String uri)
- void insufficientStorage (int serviceHandle, String uri, String path, Long totalStorageNeeded)
- void inaccessibleLocation (String msg, String path)
- void fileListAvailable (int serviceHandle)
- void activeFileDownloadStateInfoListUpdate ()
- void fileDownloadProgress (int serviceHandle, String fileUri, Long receivedBytes, Long receivedBytesTarget, Long decodedBytes, Long decodedBytesTarget, int receptionType)
- void fileDownloadProgressSuspended (int serviceHandle, String uri)

9.12.1.1.1 Member Function Documentation

9.12.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.file-DeliveryServiceInitializeConfirmation ( )

Notifies the application that the MSDC controller is initialized for File Delivery services.

9.12.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.-fileDeliveryServiceError (int *errorCode*, String *message*, Integer *serviceId*)

Notifies that an error has occurred.

Parameters

errorCode	Error codes specified in AppConstants documentation.
message	String error message.
serviceId	Service ID of the File Download service.

9.12.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.fileDeliveryServiceWarning (int *warningCode*, String *message*, Integer *serviceId*)

Notifies that a warning has occurred.

Parameters

warningCode	Warning code specified in AppConstants documentation.
message	String error message.
serviceId	Service ID of the File Download service.

9.12.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.fileDeliveryServiceListUpdate ( )

Notifies that the updated File Delivery service list is available.

9.12.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.fileAvailable (int *serviceId*, FDFile *file*)

Notifies that file download is complete for the File Delivery service.

Parameters

serviceId	Service ID of downloaded file.
file	Information about the downloaded file.

9.12.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.resetFileDeliveryNotification ( )

Resets notification to UI if there is any corruption in File Delivery service information in the MSDC.

If the UI application cached the serviceHandles, it must reset them after receiving this notification.

9.12.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.fileDownloadFailure (int *serviceHandle*, String *uri*)

Notifies that file download failed.

Parameters

serviceHandle	Service handle of the File Download service.
uri	URI of the file that failed to download.

9.12.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.-insufficientStorage (int *serviceHandle*, String *uri*, String *path*, Long *totalStorageNeeded*)

Notifies that storage is insufficient.

Parameters

serviceHandle	Service handle of File Download service.
uri	URI of the file being downloaded.
path	Path of insufficient storage
totalStorageNeeded	Total storage space needed.

9.12.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.-inaccessibleLocation (String *msg*, String *path*)

Notifies that storage location is inaccessible.

Parameters

msg	Inaccessible location message.
path	Path of the inaccessible location.

9.12.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.fileList-Available (int *serviceHandle*)

Indicates that MSDC has list of files for the service handle.

Parameters

serviceHandle	Service handle of the File Download service.

9.12.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.active-FileDownloadStateInfoListUpdate ( )

Indicates that MSDC has anupdated list of active file downloads.

9.12.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.file-DownloadProgress (int *serviceHandle*, String *fileUri*, Long *receivedBytes*, Long *receivedBytesTarget*, Long *decodedBytes*, Long *decodedBytesTarget*, int *receptionType*)

Notifies the status of file download progress for each file.

Parameters

serviceHandle	Service handle of the File Download service.

fileUri	URI of the file.
receivedBytes	Number of received bytes.
receivedBytesTarget	Total/maximum number of received bytes.
decodedBytes	Number of decoded bytes.
decodedBytesTarget	Total/maximum number of decoded bytes.
receptionType	Type of reception: • 0 = Broadcast • 1 = Unicast

9.12.1.1.1.9 void com.qualcomm.msdc.controller.IMSDCFileDeliveryControllerEventListener.fileDownloadProgressSuspended (int *serviceHandle*, String *uri*)

Notifies the file download progress is suspended.

Parameters

serviceHandle	Service handle of the File Download service.
uri	URI of the suspended file.

9.13 IMSDCNetworkController

9.13.1 Class Documentation

9.13.1.1 interface com::qualcomm::msdc::controller::IMSDCNetworkController

Public Member Functions

- void initializeNetworkService ()
- void terminateNetworkService ()
- void addNetworkEventListener (IMSDCNetworkControllerEventListener listener)
- void removeNetworkEventListener (IMSDCNetworkControllerEventListener listener)
- void enableSignalLevelMonitoring (int periodicity)
- void disableSignalLevelMonitoring ()

9.13.1.1.1 Member Function Documentation

9.13.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkController.initializeNetworkService ( )

Initializes the network service.

9.13.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkController.terminateNetworkService ( )

Terminates the network service.

9.13.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkController.addNetworkEventListener (IMSDCNetworkControllerEventListener *listener*)

Adds Event Listener to Network Controller.

Parameters

listener	The listener

9.13.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkController.removeNetworkEvent-Listener (IMSDCNetworkControllerEventListener *listener*)

Removes Event Listener from Network Controller.

Parameters

listener	The listener

9.13.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkController.enableSignalLevel-Monitoring (int *periodicity*)

Enables signal Level.

Parameters

periodicity	Periodicity

9.13.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkController.disableSignalLevel-Monitoring ( )

Disables/stops signal-level notifications.

9.14 IMSDCNetworkControllerEventListener

9.14.1 Class Documentation

9.14.1.1 interface com::qualcomm::msdc::controller::IMSDCNetworkControllerEventListener

Public Member Functions

- void networkServiceInitializeConfirmation ()
- void broadcastCoverageNotification (int state)
- void signalLevelNotification (int signalLevel)
- void networkServiceError (int errorCode, String message)
- void roamingNotification (int state)

9.14.1.1.1 Member Function Documentation

9.14.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkControllerEventListener.network-ServiceInitializeConfirmation ( )

Network service initialization confirmation.

9.14.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkControllerEventListener.broadcast-CoverageNotification (int *state*)

Broadcast coverage notification.

Parameters

state	State of broadcast coverage (in or out).

9.14.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkControllerEventListener.signalLevel-Notification (int *signalLevel*)

Signal level notification from the MSDC.

Parameters

signalLevel	Indicates broadcast signal strength.

9.14.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkControllerEventListener.network-ServiceError (int *errorCode*, String *message*)

Network service error notification.

Parameters

errorCode	Error codes specified in AppConstants documentation.
message	String error message.

9.14.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCNetworkControllerEventListener.roaming-Notification (int *state*)

Roaming notification.

Parameters

state	• 1 = IN_NETWORK; • 2 = OUT_OF_NETWORK

9.15 IMSDCGroupCallController

9.15.1 Class Documentation

9.15.1.1 interface com::qualcomm::msdc::controller::IMSDCGroupCallController

Public Member Functions

- void initializeGroupCallService ()
- void terminateGroupCallService ()
- void startGroupCallService (long tmgi, List< Integer > saiList, List< Integer > freqList)
- void startGroupCallService (long tmgi, List< Integer > saiList, List< Integer > freqList, String multicastIP, int multicastPort)
- void updateGroupCallService (long tmgi, List< Integer > saiList, List< Integer > freqList)
- void stopGroupCallService (long tmgi)
- void addGroupCallEventListener (IMSDCGroupCallControllerEventListener listener)
- void removeGroupCallEventListener (IMSDCGroupCallControllerEventListener listener)

9.15.1.1.1 Member Function Documentation

9.15.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallController.initializeGroupCallService ( )

Initializes the MSDC for Group Call services.

9.15.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallController.terminateGroupCall-Service ( )

Terminates Group Call service-related communication with the MSDC.

9.15.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallController.startGroupCallService (long *tmgi,* List< Integer > *saiList,* List< Integer > *freqList*)

Starts Group Call service.

This API is deprecated.

Parameters

tmgi	TMGI of the Group Call service.
saiList	Service Area ID list of the Group Call service.
freqList	Frequency list of the Group Call service.

9.15.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallController.startGroupCallService (long *tmgi,* List< Integer > *saiList,* List< Integer > *freqList,* String *multicastIP,* int *multicastPort*)

Starts Group Call service.

Parameters

tmgi	TMGI of the Group Call service.
saiList	Service Area ID list of the Group Call service.
freqList	Frequency list of the Group Call service.
multicastIP	Multicast IP of the Group Call service.
multicastPort	Multicast port of the Group Call service.

9.15.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallController.updateGroupCallService (long *tmgi,* List< Integer > *saiList,* List< Integer > *freqList*)

Updates Group Call.

Parameters

tmgi	TMGI of the Group Call service.
saiList	Service Area ID list of the Group Call service.
freqList	Frequency list of the Group Call service.

9.15.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallController.stopGroupCallService (long *tmgi*)

Stops Group Call.

Parameters

tmgi	TMGI of the Group Call service.

9.15.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallController.addGroupCallEvent-Listener (IMSDCGroupCallControllerEventListener *listener*)

Adds MSDC controller event listener to listen for MSDC controller events.

Parameters

listener	The event listener.

9.15.1.1.1. 9 void

com.qualcomm.msdc.controller.IMSDCGroupCallController.removeGroupCallEvent-Listener (IMSDCGroupCallControllerEventListener *listener*)

Removes MSDC controller event listener from listening for MSDC controller events.

Parameters

listener	The event listener.

9.16 IMSDCGroupCallControllerEventListener

9.16.1 Class Documentation

9.16.1.1 interface com::qualcomm::msdc::controller::IMSDCGroupCallControllerEventListener

Public Member Functions

- void groupCallServiceInitializeConfirmation ()
- void saiListUpdate (GroupItem currentGroup, List< GroupItem > groupList)
- void groupCallServiceStarted (long tmgi)
- void groupCallServiceStarted (long tmgi, String server, int port)
- void groupCallServiceStopped (long tmgi)
- void groupCallServiceStalled (long tmgi, int errorCode, String errorMessage)
- void groupCallServiceError (long tmgi, int errorCode, String message)
- void groupCallServiceInterfaceIndication (String interfaceName, int index)

9.16.1.1.1 Member Function Documentation

9.16.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallControllerEventListener.groupCall-ServiceInitializeConfirmation ( )

Notifies that the MSDC controller is initialized for Group Call services.

9.16.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallControllerEventListener.saiList- Update (GroupItem *currentGroup,* List< GroupItem > *groupList*)

Notifies the updated SAI List.

Parameters

currentGroup	Current group of the Service Area ID.
groupList	Neighbor groups of the service Area ID.

9.16.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallControllerEventListener.groupCall-ServiceStarted (long *tmgi*)

Notifies that the Group Call service has started.

Parameters

tmgi	Service ID of the started service.

9.16.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallControllerEventListener.groupCallServiceStarted (long *tmgi*, String *server*, int *port*)

Notifies that the Group Call service has started successfully.

Parameters

tmgi	Service ID of the started service.
server	Server that gets UDP data.
port	Port to be read to get UDP data.

9.16.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallControllerEventListener.groupCallServiceStopped (long *tmgi*)

Notifies that the Group Call service has stopped.

Parameters

tmgi	Service ID of the stopped service.

9.16.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallControllerEventListener.groupCallServiceStalled (long *tmgi*, int *errorCode*, String *errorMessage*)

Notifies that the Group Call service is stalled.

Parameters

tmgi	Service ID of the stalled service.
errorCode	Error codes specified in AppConstants documentation.
errorMessage	String error message.

9.16.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallControllerEventListener.groupCallServiceError (long *tmgi*, int *errorCode*, String *message*)

Notifies that an error has occurred.

Parameters

tmgi	TMGI of the Group Call service.
errorCode	Error codes specified in AppConstants documentation.
message	String error message.

9.16.1.1.1. 9 void com.qualcomm.msdc.controller.IMSDCGroupCallControllerEventListener.groupCall-

ServiceInterfaceIndication (String *interfaceName,* int *index*)

Notifies the Group Call interface name and index.

Parameters

interfaceName	Group Call interface name.
index	Group Call interface index.

9.17 IMSDCStreamingModel

9.17.1 Class Documentation

9.17.1.1 interface com::qualcomm::msdc::model::IMSDCStreamingModel

This interface defines the APIs to retrieve and update Streaming service-related information.

Public Member Functions

- Map< Integer, StreamingService > getStreamingServiceList ()
- List< Integer > getRunningStreamingServices ()
- String getPlaybackUrl (int serviceId)
- StreamingServiceState getStreamingServiceState (int serviceId)
- List< GroupItem > getStreamingGroupList ()
- GroupItem getCampedGroup ()
- Map< Integer, StreamingService > getStreamingServiceListByGroup (String groupName)

9.17.1.1.1 Member Function Documentation

9.17.1.1.1.9 Map<Integer, StreamingService> com.qualcomm.msdc.model.IMSDCStreamingModel.get-StreamingServiceList ( )

Gets a map of all Streaming services.

Note: In the ServiceInfo object inside the StreamingService, serviceId is a String, while serviceHandle is actually the numerical integer ID that is used as the integer key for this map and to pass into other API calls, such as startStreamingService(int serviceId).

Returns

Map of Streaming services ID < key > and Streaming service object < value >.

9.17.1.1.1.9 List<Integer> com.qualcomm.msdc.model.IMSDCStreamingModel.getRunningStreaming-Services ( )

Gets a list of currently running service IDs.

Returns

The running Streaming services and an array of service IDs of all currently running services.

9.17.1.1.1.9 String com.qualcomm.msdc.model.IMSDCStreamingModel.getPlaybackUrl (int *serviceId*)

Gets the MPD URL of a Streaming service.

Parameters

serviceId	ID of the Streaming service.

Returns

MPD URL of Streaming service.

9.17.1.1.1.9 StreamingServiceState com.qualcomm.msdc.model.IMSDCStreamingModel.getStreaming-ServiceState (int *serviceId*)

Gets the current service state.

Parameters

serviceId	ID of the Streaming service.

Returns

Current state of Streaming service.

9.17.1.1.1.9List<GroupItem> com.qualcomm.msdc.model.IMSDCStreamingModel.getStreamingGroup-List ( )

Returns list of available groups.

Returns

List<GroupItem>

9.17.1.1.1.9 GroupItem com.qualcomm.msdc.model.IMSDCStreamingModel.getCampedGroup ( )

Returns the current camped group name.

Returns

GroupItem

9.17.1.1.1.9 Map<Integer,StreamingService> com.qualcomm.msdc.model.IMSDCStreamingModel.get-StreamingServiceListByGroup (String *groupName*)

Returns list of Streaming services by group name.

Parameters

groupName	Group name of the Streaming services.

Returns

Map<Integer,StreamingService>

9.18 IMSDCFileDeliveryModel

9.18.1 Class Documentation

9.18.1.1 interface com::qualcomm::msdc::model::IMSDCFileDeliveryModel

This interface defines the APIs to retrieve and update File Delivery service-related information.

Public Member Functions

- Map< Integer, FDService > getFileDeliveryServiceList ()
- List< FDFile > getAvailableFileList (int serviceId)
- FDServiceState getFileDeliveryServiceState (int serviceId)
- List< Integer > getRunningFileDeliveryServices ()
- List< GroupItem > getFileDeliveryGroupList ()
- GroupItem getCampedGroup ()
- Map< Integer, FDService > getFileDeliveryServiceListByGroup (String groupName)
- String getStorageLocation ()
- Map< String, Enum < ActiveFileDownloadState > > getActiveFileDownloadInfoList (int serviceHandle, String fileUri)

9.18.1.1.1 Member Function Documentation

9.18.1.1.1. 9 Map<Integer, FDService> com.qualcomm.msdc.model.IMSDCFileDeliveryModel.getFile- DeliveryServiceList ( )

Gets a map of all File Delivery services.

Note: In the ServiceInfo object inside FDService, serviceId is a String, while serviceHandle is actually the numerical integer ID that is used as the integer key for this map and to pass into other API calls, such as startFileCapture(int serviceId).

Returns

Map of File Delivery services ID < key > and File Delivery service object < value >.

9.18.1.1.1. 9 List<FDFile> com.qualcomm.msdc.model.IMSDCFileDeliveryModel.getAvailableFileList (int *serviceId*)

Gets a list of available (downloaded) files for the respective File Delivery service.

Parameters

serviceId	ID of the File Delivery service.

Returns

List of FDFiles downloaded for passed File Delivery service.

9.18.1.1.1. 9 FDServiceState com.qualcomm.msdc.model.IMSDCFileDeliveryModel.getFileDelivery- ServiceState (int *serviceId*)

Returns the current service state.

Parameters

serviceId	ID of the File Delivery service.

Returns

Current state of passed File Delivery service.

9.18.1.1.1. 9 List<Integer> com.qualcomm.msdc.model.IMSDCFileDeliveryModel.getRunningFile- DeliveryServices ( )

Gets a list of currently running service IDs.

Returns

The running File Delivery services, and an array of service IDs of all currently running services

9.18.1.1.1. 9 List<GroupItem> com.qualcomm.msdc.model.IMSDCFileDeliveryModel.getFileDelivery- GroupList ( )

Gets list of File Delivery groups.

Returns

List<GroupItem>

9.18.1.1.1. 9 GroupItem com.qualcomm.msdc.model.IMSDCFileDeliveryModel.getCampedGroup ( )

Gets current camped group name.

Returns

GroupItem

9.18.1.1.1. 9 Map<Integer,FDService> com.qualcomm.msdc.model.IMSDCFileDeliveryModel.getFile- DeliveryServiceListByGroup (String *groupName*)

Gets list of File Delivery services based on group name.

Parameters

groupName	Group name of the File Delivery services

Returns

Map<Integer,FDService>

9.18.1.1.1. 9 com.qualcomm.msdc.model.IMSDCFileDeliveryModel.getStorageLocation ( ) String

Gets storage location (file download location).

Returns

Storage location path.

9.18.1.1.1. 9 Map<String,Enum<ActiveFileDownloadState>> com.qualcomm.msdc.model.IMSDC-FileDeliveryModel.getActiveFileDownloadInfoList (int *serviceHandle,* String *fileUri*)

9.19 IMSDCNetworkModel

9.19.1 Class Documentation

9.19.1.1 interface com::qualcomm::msdc::model::IMSDCNetworkModel

Public Member Functions

- int getSignalLevel ()
- int getBroadcastCoverage ()
- int getRoamingState ()

9.19.1.1.1 Member Function Documentation

9.19.1.1.1. 9 int com.qualcomm.msdc.model.IMSDCNetworkModel.getSignalLevel ( )

Returns signal level.

Returns

Integer of signal level.

9.19.1.1.1. 9 int com.qualcomm.msdc.model.IMSDCNetworkModel.getBroadcastCoverage ( )

Gets Broadcast coverage.

Returns

int

- 0 - IN_COVERAGE
- 1 - OUT_OF_COVERAGE

9.19.1.1.1. 9 int com.qualcomm.msdc.model.IMSDCNetworkModel.getRoamingState ( )

Gets roaming state.

Returns

int

- 0 - UNKNOWN_NETWORK_STATE (default)
- 1 - IN_NETWORK
- 2 - OUT_OF_NETWORK

9.20 IMSDCGroupCallModel

9.20.1 Class Documentation

9.20.1.1 interface com::qualcomm::msdc::model::IMSDCGroupCallModel

Public Member Functions

- GroupCallServiceState getGroupCallServiceState (long tmgi)

9.20.1.1.1 Member Function Documentation

9.20.1.1.1.1 GroupCallServiceState com.qualcomm.msdc.model.IMSDCGroupCallModel.getGroupCallServiceState (long *tmgi*)

Gets Group Call state for the given TMGI.

Parameters

tmgi	TMGI of the Group Call service.

9.21 ActiveFileDownloadState

9.21.1 Class Documentation

9.21.1.1 enum com::qualcomm::msdc::object::ActiveFileDownloadState

Active file download states.

Data fields

Type	Parameter	Description
	IN_PROGRES-S	Active file download is in progress.

9.22 FDFile

9.22.1 Class Documentation

9.22.1.1 class com::qualcomm::msdc::object::FDFile

This class defines a file object that will be downloaded for the respective File Delivery service.

Public Member Functions

- FDFile (int fileId, FileInfo fileInfo, int serviceHandle)
- int getFileId ()
- FileInfo getFileInfo ()
- int getServiceHandle ()

9.22.1.1.1 Constructor & Destructor Documentation

9.22.1.1.1. 9 com.qualcomm.msdc.object.FDFile.FDFile (int *fileId,* FileInfo *fileInfo,* int *serviceHandle*)

Instantiates a new File Delivery file.

Parameters

fileId	The file ID of the new file.
fileInfo	The file information of the new file.
serviceHandle	The service handle of the new file.

9.22.1.1.2 Member Function Documentation

9.22.1.1.2. 9 com.qualcomm.msdc.object.FDFile.getFileId ( ) int

Gets the file ID.

Returns

int The file ID

9.22.1.1.2. 9 com.qualcomm.msdc.object.FDFile.getFileInfo ( ) FileInfo

Gets file information.

Returns

fileInfo object

9.22.1.1.2. 9 com.qualcomm.msdc.object.FDFile.getServiceHandle( ) int

Gets the service handle.

Returns

serviceHandle

9.23 FDService

9.23.1 Class Documentation

9.23.1.1 class com::qualcomm::msdc::object::FDService

This class defines the File Delivery service object.

Public Member Functions

- FDService (ServiceInfo serviceInfo, FDServiceState state, List< FDFile > files)
- ServiceInfo getServiceInfo ()
- FDServiceState getState ()
- void setState (FDServiceState state)
- List< FDFile > getFiles ()
- void setFiles (List< FDFile > files)
- void addFile (FDFile file)
- void deleteFile (FDFile file)
- List< RunningFdServiceInfo > getRunningFdServiceInfo ()
- void setRunningFdServiceInfo (List< RunningFdServiceInfo > runningFdServiceInfo)

9.23.1.1.1 Constructor & Destructor Documentation

9.23.1.1.1. 9 com.qualcomm.msdc.object.FDService.FDService (ServiceInfo *serviceInfo,* FDService- State *state,* List< FDFile > *files*)

This constructor creates the File Delivery service object.

Parameters

serviceInfo	serviceInfo object of File Delivery service
state	State of File Delivery service.
files	List of files downloaded for this File Delivery service.

9.23.1.1.2 Member Function Documentation

9.23.1.1.2. 9 com.qualcomm.msdc.object.FDService.getServiceInfo ( ) ServiceInfo

Gets service information.

Returns

The service information object.

9.23.1.1.2.9 com.qualcomm.msdc.object.FDService.getState ( ) FDServiceState

Gets the current state.

Returns

States for File Delivery service.

9.23.1.1.2.9 void com.qualcomm.msdc.object.FDService.setState (FDServiceState *state*)

Sets the state.

Parameters

state	Sets the state(s) for File Delivery service

9.23.1.1.2.9 com.qualcomm.msdc.object.FDService.getFiles ( ) List<FDFile>

Gets the downloaded files list.

Returns

List of FDFile object.

See also

FDFile

9.23.1.1.2.9 void com.qualcomm.msdc.object.FDService.setFiles (List< FDFile > *files*)

Sets the files list.

Parameters

files	List of FDFile objects.

See also

FDFile

9.23.1.1.2.9 void com.qualcomm.msdc.object.FDService.addFile (FDFile *file*)

Adds the file to the list.

Parameters

file	File to be added in the list.

9.23.1.1.2. 9 void com.qualcomm.msdc.object.FDService.deleteFile (FDFile *file*)

Deletes the file from the list.

Parameters

file	File to be deleted from the list.

9.23.1.1.2. 9 List<RunningFdServiceInfo> com.qualcomm.msdc.object.FDService.getRunningFd- ServiceInfo ( )

Gets running File Delivery service information.

Returns

List of RunningFdServiceInfo objects.

See also

RunningFdServiceInfo

9.23.1.1.2. 9 void com.qualcomm.msdc.object.FDService.setRunningFdServiceInfo (List< RunningFdServiceInfo > *runningFdServiceInfo*)

Sets Running File Delivery service information.

Parameters

runningFdService-Info	List of RunningFdServiceInfo objects.

9.24 FDServiceState

9.24.1 Class Documentation

9.24.1.1 enum com::qualcomm::msdc::object::FDServiceState

File Delivery service states.

Data fields

Type	Parameter	Description
	STATE_RUN-NING	File Delivery service is running.
	STATE_STOP-PED	File Delivery service is stopped.

9.25 FileCaptureParams

9.25.1 Class Documentation

9.25.1.1 class com::qualcomm::msdc::object::FileCaptureParams

FileCaptureParams contains the options required for File Delivery Service Start Capture.

Parameters

serviceId	The Service ID of the File Delivery service that must be captured.
fileURI	The URI of the File Delivery service that must be captured. If empty, all files will be captured.
reportDownload-Progress	If TRUE, the app will get File Download Progress notifications.

Public Member Functions

- FileCaptureParams (int serviceId, String fileURI, boolean reportDownloadProgress)

Public Attributes

- int serviceId
- String fileURI = ""
- boolean reportDownloadProgress = false

9.25.1.1.1 Constructor & Destructor Documentation

9.25.1.1.1.9 com.qualcomm.msdc.object.FileCaptureParams.FileCaptureParams (int *serviceId*, String *fileURI*, boolean *reportDownloadProgress*)

Constructor of FileCaptureParams.

9.25.1.1.2 Member Data Documentation

9.25.1.1.2.9 int com.qualcomm.msdc.object.FileCaptureParams.serviceId

The Service ID of the File Delivery service that must be captured.

9.25.1.1.2.9 String com.qualcomm.msdc.object.FileCaptureParams.fileURI = ""

The URI of the File Delivery service that must be captured.

If empty, all files will be captured.

9.25.1.1.2. 9 boolean com.qualcomm.msdc.object.FileCaptureParams.reportDownloadProgress = false

Boolean for enabling File Download Progress notifications.

9.26 FileDeliveryInitParams

9.26.1 Class Documentation

9.26.1.1 class com::qualcomm::msdc::object::FileDeliveryInitParams

FileDeliveryInitParams contains the options required for the initialization of a File Delivery service.

FileDeliveryInitParams starts the "LTE BC Service".

Parameters

serviceClassNames	List of service classes. It is NULL by default.
regTimeToLive	The expected time the app returns to get downloaded files. It is NULL by default.
cancelCapture-History	If TRUE, the app wants the MSDC to clear all file captures initiated during a previous session It is NULL be default.
enableWakeup-Notification	If TRUE, the MSDC broadcasts the wake-up notification when a background file download is complete. It is NULL by default.
storageLocation	Location to store the files. All files will be stored in an emulated location (/storage/emulated/0) by default.

Public Member Functions

- FileDeliveryInitParams ()

Public Attributes

- List< String > serviceClassNames = null
- Integer regTimeToLive = null
- Boolean cancelCaptureHistory = null
- Boolean enableWakeupNotification = null
- String storageLocation = null
- Boolean copyDownloadedFiles = null

9.26.1.1.1 Constructor & Destructor Documentation

9.26.1.1.1. 9 com.qualcomm.msdc.object.FileDeliveryInitParams.FileDeliveryInitParams ( )

Constructor of FileDeliveryInitParams().

9.26.1.1.2 Member Data Documentation

9.26.1.1.2. 9 List<String> com.qualcomm.msdc.object.FileDeliveryInitParams.serviceClassNames = null

Holds File Delivery service class names.

9.26.1.1.2. 9 Integer com.qualcomm.msdc.object.FileDeliveryInitParams.regTimeToLive = null

Holds Registration Time to Live value.

9.26.1.1.2. 9 Boolean com.qualcomm.msdc.object.FileDeliveryInitParams.cancelCaptureHistory = null

Holds cancelCaptureHistory TRUE/FALSE.

9.26.1.1.2. 9 Boolean com.qualcomm.msdc.object.FileDeliveryInitParams.enableWakeupNotification = null

When enableWakeupNotification is TRUE, it enables sending a wake-up notification to the deregistered application that is interested in file download completion during Time to Live.

9.26.1.1.2. 9 String com.qualcomm.msdc.object.FileDeliveryInitParams.storageLocation = null

Holds location to store the files.

9.26.1.1.2. 9 Boolean com.qualcomm.msdc.object.FileDeliveryInitParams.copyDownloadedFiles = null

Specifies whether to copy downloaded file(s) locally.

9.27 FileDeliveryTerminateParams

9.27.1 Class Documentation

9.27.1.1 class com::qualcomm::msdc::object::FileDeliveryTerminateParams

FdTerminateParams contains the options required for the termination of a File Delivery service.

FdTerminateParams starts "LTE BC Service".

Parameters

regTimeToLive	The expected time the application returns to get downloaded files. It is NULL by default.

Public Member Functions

- FileDeliveryTerminateParams()

Public Attributes

- Integer regTimeToLive = new Integer(0)

9.27.1.1.1 Constructor & Destructor Documentation

9.27.1.1.1.9 com.qualcomm.msdc.object.FileDeliveryTerminateParams.FileDeliveryTerminateParams ( )

Contains the option required during termination of the File Download service.

9.27.1.1.2 Member Data Documentation

9.27.1.1.2.9 Integer com.qualcomm.msdc.object.FileDeliveryTerminateParams.regTimeToLive = new Integer(0)

Holds Registation Time to Live value.

9.28 GroupCallService

9.28.1 Class Documentation

9.28.1.1 class com::qualcomm::msdc::object::GroupCallService

This class defines the service object for a Group Call service.

Data fields

Type	Parameter	Description
long	tmgi	TMGI of the Group Call.
int	serviceHandle	Service handle of the Group Call.
List< Integer >	saiList	Service Area ID list of the Group Call.
List< Integer >	freqList	Frequency list of the Group Call.
GroupCall-ServiceState	state	The state of the Group Call service.
String	multicastIP	Multicast IP of the Group Call.
int	multicastPort	Multicast port of the Group Call.
int	clientPort	Client port of the Group Call.

9.29 GroupCallServiceState

9.29.1 Class Documentation

9.29.1.1 enum com::qualcomm::msdc::object::GroupCallServiceState

Group Call service states.

Data fields

Type	Parameter	Description
	STATE_STOPPED	Group Call service is stopped.
	STATE_SENT_START_SERVICE	Group Call service is sent to start.
	STATE_SENT_UPDATE_SERVICE	Group Call service is sent for an update.
	STATE_STARTED	Group Call service is running.
	STATE_SENT_STOP_SERVICE	Group Call service sent to stop.
	STATE_STALLED	Group Call service stalled.

9.30 MSDCAppManagerInitParams

9.30.1 Class Documentation

9.30.1.1 class com::qualcomm::msdc::object::MSDCAppManagerInitParams

MSDCAppManagerInitParams contains the options required for the initialization of MSDCAppManager.

Parameters

optIn	If TRUE, the app is participating in reception reporting. optIn is NULL by default.

Data fields

Type	Parameter	Description
Boolean	reception-ReportingOptIn	To enable OptInOptOut feature value.
String	appId	Application instance ID that uniquely identifies the MSDC API Client.
String	middleware-PackageName	Middleware package name that identifies targeted middleware.
MSDC-Connection-Type	middleware-Connection-Mode	Transport Connection mode.
long	remoteRetry-TimerMS	Time after which the Remote MSDC will try to connected again (if disconnected) in MSDC. This is applicable if the MSDCConnectionType is REMOTE or REMOTE_PREFERRED. If the MSDCConnectionType is LOCAL, this will be ignored.

9.31 ServiceInitializationState

9.31.1 Class Documentation

9.31.1.1 enum com::qualcomm::msdc::object::ServiceInitializationState

Service initialization states for all services.

Data fields

Type	Parameter	Description
	UNINITIALIZ-ED	Service is uninitialized.
	INITIALIZAT-ION_REQUE-STED	Initialization is requested for service.
	INITIALIZED	Service is initialized.

9.32 StreamingInitParams

9.32.1 Class Documentation

9.32.1.1 class com::qualcomm::msdc::object::StreamingInitParams

StreamingInitParams contains the options required during initialization of the Streaming service.

StreamingInitParams starts "LTE BC Service".

Parameters

serviceClassNames	List of service classes. It is NULL by default.

Data fields

Type	Parameter	Description
List< String >	serviceClass-Names	List of service classes.

9.33 StreamingService

9.33.1 Class Documentation

9.33.1.1 class com::qualcomm::msdc::object::StreamingService

This class defines the service object for a Streaming service.

Public Member Functions

- StreamingService (ServiceInfo serviceInfo, StreamingServiceState state)
- ServiceInfo getServiceInfo ()
- StreamingServiceState getState ()
- void setState (StreamingServiceState state)

9.33.1.1.1 Constructor & Destructor Documentation

9.33.1.1.1. 9 com.qualcomm.msdc.object.StreamingService.StreamingService (ServiceInfo *serviceInfo,* StreamingServiceState *state*)

This constructor is used to create the DASH service object.

Parameters

serviceInfo	serviceInfo object of DASH service.
state	State of the DASH service.

9.33.1.1.2 Member Function Documentation

9.33.1.1.2. 9 ServiceInfo com.qualcomm.msdc.object.StreamingService.getServiceInfo ( )

Gets the service information.

Returns

ServiceInfo object.

9.33.1.1.2. 9 StreamingServiceState com.qualcomm.msdc.object.StreamingService.getState ( )

Gets the state of the Streaming service.

Returns

Streaming service state.

9.33.1.1.2. 9 void com.qualcomm.msdc.object.StreamingService.setState (StreamingServiceState *state*)

Sets the Streaming service state.

Parameters

state	The Streaming service state to set.

9.34 StreamingServiceState

9.34.1 Class Documentation

9.34.1.1 enum com::qualcomm::msdc::object::StreamingServiceState

Streaming service states.

Data fields

Type	Parameter	Description
	STATE_STOPPED	Service initialized.
	STATE_SENT_START_SERVICE	Service sent to start.
	STATE_STARTED	Service running.
	STATE_SENT_STOP_SERVICE	Service sent to be stopped.
	STATE_STALLED	Service stalled.

9.35 MSDCConnectionType

9.35.1 Class Documentation

9.35.1.1 enum com::qualcomm::msdc::transport::interfaces::MSDCConnectionType

MSDC connection types.

Data fields

Type	Parameter	Description
	UNKNOWN	Connection is unknown.
	LOCAL	Connection is local.
	REMOTE	Connection is remote.
	REMOTE_PR-EFERRED	A remote connections is preferred.

A Using the MSDC API

This chapter provides code snippets and examples of how to use the MSDC interface.

For more detailed information on classes and functions, see Chapter 9.

A.1 Android permissions

The following Android permissions are required in the Android Manifest file to use the MSDC:

```
<uses-permission android:name = "android.permission.INTERNET"/>
<uses-permission android:name = "android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
</application>
```

A.2 Initialization

A.2.1 Create an instance of the MSDCApplication class

At startup, we recommend creating an instance of the MSDCApplication class inside the `onCreate()` method of the extended application base class. For example:

```
public class  MSDCUIApplication extends Application {
MSDCApplication msdcApplication;

@Override
public void onCreate() {
super.onCreate();
msdcApplication = new MSDCApplication(getApplicationContext());
}
}
```

This gives the application's context to the library as soon as it is loaded. This can be done in the first base Activity's `onCreate()` method as well.

A.2.2 Initialize MSDC

If the application is interested in communicating with the MSDC, the app must initialize the connection to the MSDC using `initializeMSDC()`. This function call makes `MSDCAppManagerInitParams` an argument. For example:

```
MSDCAppManagerInitParams mSDCAppManagerInitParams = new
    MSDCAppManagerInitParams();
// Add values to initialize params object
mSDCAppManagerInitParams.receptionReportingOptIn = MSDCUIApplication.
    optInValue;
MSDCAppManager.getInstance().initializeMSDC(mSDCAppManagerInitParams);
@Override
public void msdcError(int errorCode, String errorMessage) {
// If initialization failure occurs, MSDC sends msdcError notification to
    application. // It will be good to handle errors like client server version
     compatibility here.
}
// In initializeMSDCConfirmation,it is recommended to call initialize on any
    desired
// service like streaming, file delivery or network.

@Override
public void initializeMSDCConfirmation() {
//call initialize on desired services i.e. streaming, FD and network
StreamingInitParams params = new StreamingInitParams();
params.serviceClassNames = serviceClassNames; mMSDCStreamingController.
    initializeStreamingService(params);
FileDeliveryInitParams fdParams = new FileDeliveryInitParams();
fdParams.regTimeToLive = regTimeToLiveInit;
fdParams.serviceClassNames = sServiceClassList;
fdParams.enableWakeupNotification = enableWakeUpNotificationInit;
mMSDCFdController.initializeFileDeliveryService(fdParams);
mIMSDCNetworkController.initializeNetworkService();

mMSDCGroupCallController.initializeGroupCallService();
}
@Override
public void msdcWarning(int warningCode, final String warningMessage) {
//This callback gets invoked when there are warnings during the initialization
     connection.
}
@Override
public void terminateMSDCConfirmation() {
//This is the confirmation for terminate request
}
```

If the application must request a specific MSDC middleware, its package name should be set during MSDC initialization.

```
MSDCAppManagerInitParams mSDCAppManagerInitParams = new
    MSDCAppManagerInitParams();
//Add targeted middleware package name to initialize params object
mSDCAppManagerInitParams.middlewarePackageName = "com.example.myCarrier";
MSDCAppManager.getInstance().initializeMSDC(mSDCAppManagerInitParams);
```

A.2.3 Get Controller and Model instances

To get the Controller instances of all MSDC (Streaming, File Delivery, Network, and Group Call) modules, the app can perform the following:

```
IMSDCStreamingController mMSDCStreamingController =
MSDCAppManager.getInstance().getStreamingController();

IMSDCFileDeliveryController mMSDCFdController =
MSDCAppManager.getInstance().getFileDeliveryController();

IMSDCNetworkController mMSDCNetworkController =
MSDCAppManager.getInstance().getNetworkController();

IMSDCGroupCallController mIMSDCGroupCallController =
MSDCAppManager.getInstance().getGroupCallController();
```

To get the Model instances of all MSDC modules, the app can perform the following:

```
IMSDCStreamingModel mStreamingServiceModel =
MSDCAppManager.getInstance().getStreamingModel();

IMSDCFileDeliveryModel mFdServiceModel =
MSDCAppManager.getInstance().getFileDeliveryModel();

IMSDCNetworkModel mNetworkServiceModel =
MSDCAppManager.getInstance().getNetworkModel();

IMSDCGroupCallModel mIMSDCGroupCallModel =
MSDCAppManager.getInstance().getGroupCallModel();
```

These `getInstance()` functions return a single instance of the model and controller. These singleton instances can be defined early and as many times as desired. However, we recommend defining once early on and maintaining a reference to it.

A.2.4 Event listener

If the app wants to receive events from various MSDC modules, it must implement related event listeners. For example:

```
public class ViewActivity extends Activity implements
IMSDCAppManagerEventListener,
IMSDCStreamingControllerEventListener,
IMSDCFileDeliveryControllerEventListener,
IMSDCNetworkControllerEventListener,IMSDCGroupCallControllerEventListener
```

For improved efficiency, the app may want to add back or remove a listener during `onResume()` or `onPause()` (or `onDestroy()`, depending on usage). For example:

```
public void onResume () {
super.onResume();
MSDCAppManager.getInstance().addMSDCEventListener(this);
    mMSDCStreamingController.addStreamingEventListener(this);
    mMSDCFdController.addFileDeliveryEventListener(this);
mMSDCNetworkController.addNetworkEventListener(this);
    mIMSDCGroupCallController.addGroupCallEventListener(this);
}
```

```
public void onPause() {
super.onPause();
MSDCAppManager.getInstance().removeMSDCEventListener(this);
    mMSDCStreamingController.removeStreamingEventListener(this);
    mMSDCFdController.removeFileDeliveryEventListener(this);
mMSDCNetworkController.removeNetworkEventListener(this);
    mIMSDCGroupCallController.removeGroupCallEventListener(this);
}
```

A.2.5 Define the Service class during initialization

To specify the Service class in which the app is interested, the app must define the class in the Service class list before sending it to the MSDC. The following example shows how to define the class for streaming services:

```
List<String> serviceClassNames= new ArrayList<String>();
serviceClassNames.add("serviceClassName1");
serviceClassNames.add("serviceClassName2");
serviceClassNames.add("serviceClassName3");

StreamingInitParams params = new StreamingInitParams();
params.serviceClassNames = serviceClassNames;
mMSDCStreamingController.initializeStreamingService(params);
```

A.2.6 Define the file download storage location

To specify the storage location in which the app is interested, the app can define the absolute path of the storage location in the `storageLocation` before sending it to the MSDC during file delivery initialization. The storage location can also be updated after initialization. For example:

```
//getExternalFilesDirs - Android API: Returns absolute paths to application-
    specific //directories on all shared/external storage devices where the
    application can place //persistent files it owns.
//This method will return external storage locations (including emulated / SD
    Card paths)

File[] storageLocationArray = getApplicationContext().getExternalFilesDirs(
    null);

//UI chooses the appropriate storage path for its need:
String selectedLocation  = storageLocationArray[1]

FileDeliveryInitParams fdParams = new FileDeliveryInitParams();
fdParams.storageLocation = selectedLocation;
mMSDCFdController.initializeFileDeliveryService(fdParams);

//Updating the storage path after initialization

MSDCController mMSDCController = MSDCControllerImpl.getInstance();
mMSDCFdController.setStorageLocation(selectedLocation);
```

A.3 Streaming module

A.3.1 Get the Streaming service list

The app can get the following services:

List of all Streaming services

```
Map<Integer, StreamingService> mServiceMap =
mStreamingServiceModel.getStreamingServiceList();
```

Camped group

For more information on camped groups, see Section 4.4.8.5.

```
GroupItem campedGroupItem = mStreamingServiceModel.getCampedGroup();
```

Streaming services associated with a specific group

```
Map<Integer, StreamingService> serviceMap =
mStreamingServiceModel.getStreamingServiceListByGroup(groupName);
```

A.3.2 onPause and onResume of Activity/Fragments for Streaming service

For an `onPause()` event of a streaming Activity/Fragment, this call flow is recommended:

```
@Override
public void onPause() {
mMSDCStreamingController.removeStreamingEventListener(this);
mIMSDCNetworkController.removeNetworkEventListener(this);

if (mMediaPlayer.isPlaying()) {
mMediaPlayer.stop();
}

mMediaPlayer.reset();
mMediaPlayer.release();
mMediaPlayer = null;
mMSDCStreamingController.stopStreamingService(mStreamingServiceList.get(index)
    );
super.onPause();
}
```

For an `onResume()` event of a streaming Activity/Fragment this call flow is recommended:

```
public void onResume() {
super.onResume();
mMSDCStreamingController.startStreamingService(mStreamingServiceList.get(index
    ));
mMSDCStreamingController.addStreamingEventListener(this);
mIMSDCNetworkController.addNetworkEventListener(this);
}
```

A.3.3 Switch Streaming services

Use the following API to switch Streaming services:

```
mMSDCStreamingController.switchStreamingService(mStreamingServiceList.get(
    currentServiceIndex),mStreamingServiceList.get(toBePlayedServiceIndex));
```

A.3.4 Implement Streaming service callback

To minimize the perceived service switching time and the black screen flash during service switching, we recommend using MediaPlayer with SurfaceView instead of "Video View."

We also recommend reusing the mediaplayer object that showed the last frame of the previous service until the next mediaplayer object is ready.

```
// Following call flow is recommended in streamingServiceStarted call back.

@Override
public void streamingServiceStarted(int serviceId){

// stop old stream
if (mMediaPlayer.isPlaying()){
mMediaPlayer.stop();
}

mMediaPlayer.reset();
mMediaPlayer.release();
mMediaPlayer = null;
//start new stream
mMediaPlayer = new MediaPlayer();

mMediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {

public void onPrepared(MediaPlayer mediaPlayer) {
mediaPlayer.start();
}
});

mMediaPlayer.setOnErrorListener(new OnErrorListener() {

@Override
public boolean onError(MediaPlayer mp, int what, int extra) {
//handle media player error
}
});
```

```
mMediaPlayer.setDataSource("<THE URL FOR THIS SERVICE ID>");
mMediaPlayer.setDisplay(mSurfaceHolder);
mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mMediaPlayer.setScreenOnWhilePlaying(true);
mMediaPlayer.prepareAsync();
}

// On receiving streamingServiceStopped call back, you can handle it for
   better user // experience in application.
public void streamingServiceStopped(int serviceId) {
// You can show a splash screen or something similar
}

// On receiving mpdUpdated notification, check playback url got updated and if
    yes
// then do the same process as indicated in streamingServiceStarted call back.
public void mpdUpdated (int serviceId) {
// Check if the url is updated for the service you are playing
// and then basically do the same process as streamingServiceStarted
// if that is the case.
}

// On receiving streamingServiceListUpdate notification, it is recommended to
   refresh
// UI and display new streaming service data as services may be removed or
   added.
@Override
public void streamingServiceListUpdate() {
//refresh UI to display new streaming services data
}

// If application has any functionality which is dependent on streaming
// initialization, it is recommended to add that functionality here.
@Override
public void streamingServiceInitializeConfirmation() {
}
// It is recommended to handle frequency conflict error in
// streamingServiceError call back.
@Override
public void streamingServiceError(int errorCode, String message, Integer
   serviceId) {
//handle frequency conflict error
}

// Following call flow is recommended on receiving stalled notification.
@Override
public void streamingServiceStalled(int serviceId) {

if (mMediaPlayer.isPlaying()) {
mMediaPlayer.stop();
}

mMediaPlayer.reset();
mMediaPlayer.release();
mMediaPlayer = null;
}
```

A.4 File Delivery module

A.4.1 Get File Delivery service list

The app can get the following services:

List of all File Delivery services

```
Map<Integer, FDService> fdServiceMap = mFDServiceModel.
    getFileDeliveryServiceList();
```

Camped group

For more information on camped groups, see Section 5.4.6.4.

```
GroupItem campedGroupItem = mFDServiceModel.getCampedGroup();
```

File Delivery services associated with a specific group

```
Map<Integer, FDService> fdserviceMap =
mFDServiceModel.getFileDeliveryServiceListByGroup(groupName);
```

A.4.2 Implement File Delivery callback

```
// On receiving fileDeliveryServiceListUpdate, it is recommended to refresh UI
     and
// display new file delivery services data as services may be added or removed
    .

@Override
public void fileDeliveryServiceListUpdate() {
//Refresh UI to display new file delivery services data
}

// If application has any functionality which is dependent on file delivery
// initialization it is recommended to add that functionality here.
@Override
public void fileDeliveryServiceInitializeConfirmation() {
}

// It is recommended to handle ERROR_FD_SERVICE_ALREADY_USED_BY_ANOTHER_APP
    here.
@Override
public void fileDeliveryServiceError(int arg0, String arg1, Integer arg2) {
// handle ERROR_FD_SERVICE_ALREADY_USED_BY_ANOTHER_APP
}
```

A.4.3 Delete file

You must pass the file URI (**not** the download location) when deleting the file. For example:

```
// You get the FDFile object in FileAvailable notification
FDFile myFdFile;
String uri= myFdFile.getFileInfo().uri;
MSDCController mMSDCController = MSDCControllerImpl.getInstance();

//You can print the UIR and serviceHandle value here for debugging

mMSDCController.deleteFile(myFdFile.getServiceHandle(), uri);
```

A.5 Network module

A.5.1 Implement Network service callback

```
// It is recommended to handle network service error in this callback to
   display some
// message to user.

@Override
public void networkServiceError(int errorCode, String errorMessage) {
//handle network service error
}

// It is recommended to add here, operations which are dependent on network
   service
// initialization. Example is shown below:
@Override
public void networkServiceInitializeConfirmation() {
//call enable/disable signal strength monitoring
mIMSDCNetworkController.enableSignalLevelMonitoring(period);
mIMSDCNetworkController.disableSignalLevelMonitoring();
}

// SignalLevelNotification provides latest signal strength value. It is
   recommended to
// add code to display that value here if app wants to always update UI to
   show latest
// signal strength value.
@Override
public void signalLevelNotification(int signalLevelNotification) {
//update signal strength value
}
```

```
// It is recommended to implement broadcast coverage notification in streaming
    App
// as follows

@Override
public void broadcastCoverageNotification(int state) {

if (state == AppConstants.OUT_OF_COVERAGE) {

if (mMediaPlayer != null && mMediaPlayer.isPlaying()) {
mMediaPlayer.pause();
}
} else if (state == AppConstants.IN_COVERAGE) {
if (mMediaPlayer != null && (!mMediaPlayer.isPlaying())) {
mMediaPlayer.start();
}
}
}
```

A.6 Group Call module

A.6.1 Get Group Call service state

The app can get the state of a specific Group Call service by using its TMGI.

```
GroupCallServiceState mServiceState =
mGroupCallServiceModel.getGroupCallServiceState(TMGI);
```

A.6.2 Start Group Call service

The app can start a Group Call service using the service tmgi, service area ID list, and frequency list.

```
   public void startGroupCallService(long tmgi,
                                     List<Integer> saiList,
                                     List<Integer> freqList
                                     String multicastIP,
                                     int multicastPort
                                     ) {

      mIMSDCGroupCallController.startGroupCallService(tmgi, saiList, freqList,
          multi
                    castIP, multicastPort);
   }
```

A.6.3 Stop Group Call service

The app can stop the Group Call service by using tmgi.

```
public void stopGroupCallService(long tmgi) {
   mIMSDCGroupCallController.stopGroupCallService.(tmgi);
}
```

A.6.4 Update Group Call service

The app can stop the Group Call service by using tmgi.

```
public void stopGroupCallService(long tmgi) {
   mIMSDCGroupCallController.stopGroupCallService.(tmgi);
}
```

A.6.5 Implement Group Call service callback

A.6.5.1 Group Call service interface indication

Upon successful registration of a Group Call service, the app will get `groupCallServiceInterfaceIndication(String interfaceName, int index)` as a callback. The `interfaceName` is a data interface on which the audio packets of the Group Call service are received. Ideally, this group call interface will not change once it is initialized.

A.6.5.2 Group Call service started

On the request of `startgroupcallservice(...)`, the app will get `groupcallservicestarted(tmgi, server, port)` as a callback if the service is started successfully.

The app can start playing media using any player. The following code snippet shown uses a VLC player as an example. The mediaURI shown below is an RTP URL formed with the multicast IP address and port. The multicast IP address and port are provided by the Group Call Client (see Section 7.1.1).

```
    public void groupCallServiceStarted(long tmgi, String server, int port) {

        /*if(mSocket == null){
      //open a multicast socket
      mSocket = new MulticastSocket(PORT);

      //Create SocketAddress
      SocketAddress socketAddress =
                              new InetSocketAddress(MULTICAST_GROUP, PORT);

      //Get the NetworkInterface from the interfaceName
      NetworkInterface networkInterface =
                              NetworkInterface.getByName(interfaceName);

      //join the multicast group on given interface name
      mSocket.joinGroup(socketAddress, networkInterface);
  }*/
The above section is deprecated from version 4.3.03.01.0

  if(mLibVLC == null){
       mLibVLC = new LibVLC(this,null);
  }

  // To play with LibVLC, we need a media player object.
  if(mMediaPlayer == null){
       mMediaPlayer = new MediaPlayer(mLibVLC);
  }
```

```
//Forming the rtp url using multicast group and port
String mediaURI = "rtp://" + server + ":" + port;

// Create a new Media object for the file.
if(media == null){
      media = new Media(mLibVLC, Uri.parse(mediaURI));
}

// Tell the media player to play the new Media.
mMediaPlayer.setMedia(media);

// Finally, play it!
mMediaPlayer.play();


   }
```

A.6.5.3 Group Call service stopped

After the UI triggers `stopgroupcallservice(tmgi)`, it gets `groupcallservicestopped(tmgi)` notification if the service is stopped successfully by the middleware. Upon receiving this indication, the media player should stop.

A.6.5.4 Group call service stalled

The app gets a `groupCallServiceStalled()` notification when tmgi is not available or active. Upon receiving this indication, the media player may be stopped.

B References

B.1 Acronyms and Terms

Acronym or term	Definition
BM-SC	Multicast Service Center
DASH	Dynamic Adaptive Streaming over HTTP
E911	Enhanced 911
eMBMS	Enhanced Multimedia Broadcast Mlticast Services
FOTA	Firmware Over The Air
LTE	Long Term Evolution
MIME	Multipurpose Internet Mail Extension
MPD	Media Presentation Description
MSDC	Multicast Service Device Client
MVC	Model-View-Controller
SAI	Service Area ID
TMGI	Temporary Mobile Group ID
UE	User equipment