# Handbuch M2ModuleService Android™* 9/

# Manual for M2ModuleService Android™* 9

**Grundlegende Information zur Verwendung des M2Module Service/**
**Basic information on the use of the M2ModuleService**

Version: 1.20

# Inhaltsübersicht/Content

## Deutsch



## English

# 1 Übersicht

## 1.1 Umfang dieser Dokumentation

Dieses Dokument richtet sich an Entwickler, die mithilfe des M2ModuleService, eine Kommunikation zwischen Apps und Aufsteckmodulen herstellen möchten. Dabei werden sowohl die Protokolle (ADIP, Service-Messages) als auch die Hauptbestandteile (ForegroundAPK, M2ModuleService, BackgroundAPK, Aufsteckmodul) genauer erläutert.

## 1.2 Zugehörige Dokumente

Um den M2ModuleService richtig nutzen zu können sind entsprechende Templates vorhanden. Diese beschreiben, wie der M2ModuleService richtig in die eigene App eingebunden werden kann.
Die folgenden Templates stehen im ACD Kundenportal unter folgendem Link zum Download bereit https://www.acd-gruppe.de/kundenportal/:

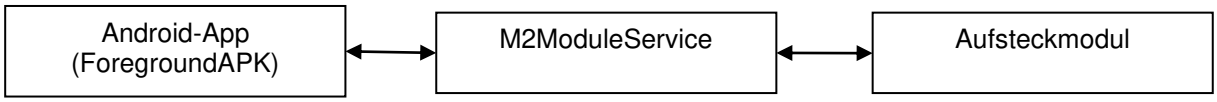
- Template_ForegroundAPK.java
- Template_BackgroundAPK.java

## 1.3 Beschreibung und Einsatz des Service

Zur einfacheren Kommunikation zwischen Aufsteckmodulen und Android-Apps wurde ein Service in das Betriebssystem eingepflegt. Dieser Service handelt einerseits die Verwaltung und Kommunikation zu den Aufsteckmodulen ab. Andererseits stellt er einen Messanger bereit, welcher von Android-Apps verwendet werden kann. Damit ist es möglich, Befehle an Aufsteckmodule zu senden, sowie Daten zu erhalten.

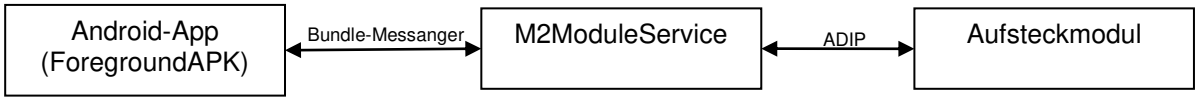
## 1.4 Aufbau des Service

Der M2ModuleService stellt die Verbindungsbrücke zwischen Android-Apps und Aufsteckmodulen dar.




Zur Kommunikation werden zwei unterschiedliche Protokolle verwendet:

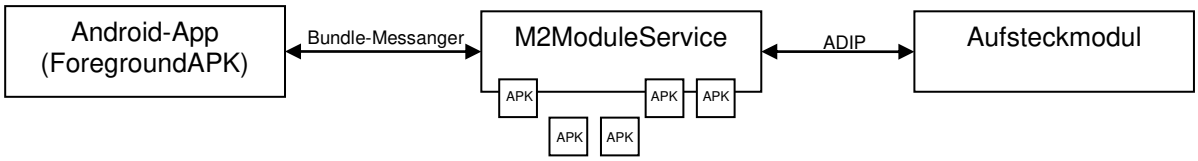
- Zwischen Android-Apps und M2ModuleService besteht ein Messanger. Die Informationen werden dabei in Bundles gepackt (weitere Informationen im Kapitel 4.1 und Kapitel 5.1).
- Zur Kommunikation zwischen M2ModuleService und Aufsteckmodulen wird das „ACD Interconnect Protocol“ ADIP verwendet (weitere Informationen im Kapitel 3.1).




Um dem modularen Konzept des M2Smart®SE gerecht zu werden, wird der ausgeführte Programmcode zu Laufzeit angepasst. Jedes Aufsteckmodul benötigt einen speziellen Programmcode, welcher die Informationen interpretiert. Diese Interpretationen sind in BackgroundAPKs ausgelagert (siehe hierzu Kapitel 4 BackgroundAPK). Dieser kompilierte Programmcode wird zur Laufzeit nachgeladen, sobald ein entsprechendes Aufsteckmodul angeschlossen wurde.

Somit besteht der M2ModuleService aus zwei Komponenten:

1. Einem statischen, fest im Android verankerten, Service (M2ModuleService)
2. Einer/mehreren modularen APKs, welche den Programmcode aller Module enthalten (BackgroundAPK)




# 2 Funktionen des M2ModuleService

Der statische Service ist das Herzstück des M2ModuleService. Er ist verantwortlich für die Verwaltung der Aufsteckmodule, außerdem dient er als Kommunikationsbrücke zwischen Aufsteckmodulen und Android-Apps.
Nachfolgend werden kurz die wichtigsten Funktionen des M2ModuleService beschrieben. Eine detaillierte Erläuterung ist im Kapitel 4.1 und Kapitel 5.1 zu finden.

Möchte sich eine gebindete App für den Messanger registrieren, muss sie eine „WHAT_HELLO (Value=2) Nachricht an den Service senden.
Wird ein Aufsteckmodul abgezogen oder aufgesteckt, versendet der Service an alle gebindeten Anwendungen eine Message mit der Kennung „WHAT_NEW_MODULE_PLUGGED (Value=5)“.
Mit der Nachricht „WHAT_ALL_MODULES_PLUGGED (Value=6)“ können alle gesteckten Module abgefragt werden.
Sollen Informationen an ein Aufsteckmodul gesendet bzw. werden Daten vom Aufsteckmodul versendet, so werden diese mit der „WHAT_MODULE_MESSAGE (Value=4)“ gekennzeichnet.
Über die Nachricht „WHAT_MODULESERVICE_INFO (Value=7)“ können allgemeine Informationen über den Service abgefragt werden.

Der statische Service verwendet eine/mehrere APKs, welche den Programmcode für Aufsteckmodule enthalten (BackgroundAPK). Die BackgroundAPK beinhaltet den eigentlichen Programmcode der Module und kann zur Laufzeit ausgetauscht werden. Seine Aufgabe ist es, aus den überlieferten Daten die Informationen zu extrahieren. Dabei können Befehle von der ForegroundAPK an Aufsteckmodule gesendet, sowie Daten der Aufsteckmodule interpretiert werden.

# 3 Kommunikationsprotokolle

## 3.1 ADIP

Das „ACD Interconnect Protocol“ ADIP ist ein Protokoll, welches von ACD M2-Geräten implementiert wird. Beim ADIP handelt es sich um ein sehr leicht zu verstehendes Protokoll. Der statische M2ModuleService implementiert dieses Protokoll ebenfalls und kommuniziert so mit den Aufsteckmodulen. Die relevanten Felder sind dabei receiver ID/sender ID, message ID sowie payload und werden nachfolgend kurz erläutert.

- **receiver ID/sender ID**
  Diese ID wird für jedes Aufsteckmodul dynamisch vergeben. Dabei bezeichnet die receiver ID, wohin das Datenpacket gehen soll. Die sender ID beschreibt, woher das Datenpacket stammt.
  Der M2ModuleService mit seinen BackgroundAPKs und ForegroundAPKs besitzt die ID „Uplink“ mit der Adresse 0x02.
  Für jedes Aufsteckmodul wird diese ID dynamisch generiert (MODULE_ADDRESS). Für die Kommunikationen zwischen ForegroundAPK, M2ModuleService und BackgroundAPK wird ein lesbarer dynamischer String verwendet (MODULE_POSITION).
  Beim **Senden** von Befehlen an ein Aufsteckmodul, muss zwingend eine receiver ID (MODULE_POSITION) angegeben werden. Die sender ID wird vom M2ModuleService selbst generiert.

Beim **Empfangen** von Informationen durch das Aufsteckmodul, werden sender ID und receiver ID an die BackgroundAPK sowie die ForegroundAPK übermittelt (MODULE_POSITION). Die Verarbeitung der Nachricht ist vom Steckplatz (receiver/sender) unabhängig.

- **message ID**
  Die message ID definiert eine spezielle Nachricht. Sie wird von ACD vergeben und ist eindeutig zuzuordnen.
- **payload**
  Die payload ist optional und kann weitere Informationen enthalten.

Zur Erkennung und Verarbeitung einer Nachricht wird die message ID verwendet. Der Steckplatz des Aufsteckmodules ist dabei nicht für die Verarbeitung der Nachricht relevant, Format und Bedeutung einer Nachricht sind vom Steckplatz unabhängig.

# 4 BackgroundAPK

Der statische Service verwendet eine/mehrere APKs, welche den Programmcode für Aufsteckmodule enthalten (BackgroundAPK). Die BackgroundAPK beinhaltet den eigentlichen Programmcode der Module und kann zur Laufzeit ausgetauscht werden. Seine Aufgabe ist es, aus den überlieferten Daten die Informationen zu extrahieren. Dabei können Befehle von der ForegroundAPK an Aufsteckmodule gesendet, sowie Daten der Aufsteckmodule interpretiert werden.
Es wird zwischen zwei Arten von BackgroundAPK unterschieden.

1. **ACD-spezifische:**
   Diese APKs werden von ACD zur Verfügung gestellt und sind fest im Betriebssystem verankert. Diese Moduldateien werden von ACD-Aufsteckmodulen verwendet.
2. **Kundenspezifische:**
   Diese APKs sind unter ` /sdcard/ACD/Add-On-Modules` abgelegt. Hier können Kunden ihre eigenen modularen BackgroundAPKs ablegen.
3. **ACD-Updatepakete:**
   BackgroundAPKs welche unter `/sdcard/ACD/Add-On-Modules` abgelegt sind, werden bevorzugt behandelt. Somit ist es möglich ACD-spezifische Moduldateien updaten zu können.

Damit eine BackgroundAPK geladen werden kann, müssen mehrere Sicherheitsparameter erfüllt werden. Diese werden nachfolgend erläutert:

1. Der Klassenname muss in einem bestimmten Format vorliegen:
   Jedes Aufsteckmodul besitzt ein Feld mit allen zur Verfügung stehenden Funktionalitäten. Dieses Feld ist von ACD definiert und im MCon des Aufsteckmoduls verankert. Damit eine Klasse geladen wird, muss sie einen dieser Funktionen anbieten und im Klassennamen veröffentlichen. Das Format lautet: `modulehandler_ ` + Funktionalität.
   Beispiel: `modulehandler_temperaturACD`
2. Die APK muss in einem bestimmten Package abgelegt sein:
   Damit eine Klasse aus der BackgroundAPK geladen werden kann, muss sie im Package `de.acdgruppe.templatem2smartmodule` abgelegt sein.
3. Es gibt fest definierte Funktionen, die in jeder Klasse vorhanden sein müssen:
   - public void InitClass(Context context)
   - private boolean knownMessageID(byte[] messageID)
   - private boolean knownCommand(String command)
   - private Bundle prepareData(String receiver, String sender, byte[] message, byte[] payload)
   - private Bundle prepareCommand(Sring receiver, String sender, Bundle command)
   - public void UnInitClass()
4. message ID und Command dürfen jeweils nur einmal im kompletten Verzeichnis vorkommen. Werden message ID oder Command mehrfach gefunden, sind sie nichtmehr eindeutig zuzuordnen und werden verworfen.
5. Der Programmcode muss lauffähig kompiliert sein und als APK unter `/sdcard/ACD/Add-On-Modules` abgelegt werden. Dabei können mehrere APKs im Verzeichnis liegen. Die Namensgebung der APK spielt keine Rolle.

Innerhalb der BackgroundAPK wird der eigentliche Programmcode für ein Aufsteckmodul abgearbeitet. Das Hauptziel ist es, soviel Programmcode wie möglich in die BackgroundAPK auszulagern und somit redundanten Code zu minimieren.
Es können jegliche Standard Java-Rechenoperationen ausgeführt werden. Ebenfalls können ACD-Dienste (z. B. ACD ScanService) verwendet werden. Auch können weitere Methoden und Variablen verwendet werden.

Ein Link zum Template, wie eine BackgroundAPK auszusehen hat, finden Sie in Kapitel 1.2.

## 4.1 Bundle-Messanger der BackgroundAPK

Der Bundle-Messanger dient zur Kommunikation zwischen der **ForegroundAPK** und der **BackgroundAPK**.
Die Grundlegende Funktion von Bundles ist hier beschrieben:
https://developer.android.com/reference/android/os/Bundle

In der **BackgroundAPK** gibt es einen Bundle-Messanger, mit den folgenden Extras:

### 4.1.1 prepareCommand

In dieser Funktion werden Bundles bearbeitet, welche von der **ForegroundAPK** an die **BackgroundAPK** gesendet werden.
Der prepareCommand enthält folgendes:

- Command → Eindeutiges Kommando zum Ausführen einer Aktion
- Receiver → Empfängermodul (bspw. top)
- Message-Keys → Eindeutige Parameter zur Erweiterung des Commands

Aus dem empfangenen Foreground-Bundle wird in prepareCommand ein neues Bundle generiert. Dieses muss folgendes enthalten:

- accepted → Das empfangene Kommando konnte verarbeitet werden
- Receiver → Empfängermodul (bspw. top) wird von ForegroundAPK durchgereicht
- Sender → Senderstation (ForegroundAPK)
- Message → Enthält die eindeutige Message-ID (wird von ACD vergeben)
- Payload → Enthält Informationen für das Empfängermodul

Das neu erstellte Bundle wird an den Empfänger gesendet (bspw. Modul)

### 4.1.2 prepareData

In dieser Funktion werden Bundles bearbeitet, welche vom Empfänger (bspw. Modul) an die **BackgroundAPK** gesendet werden.
Der prepareData enthält folgendes:

- Receiver → Empfangsstation (ForegroundAPK)
- Sender → Aufsteckmodul
- Message → Enthält die eindeutige Message-ID (wird von ACD vergeben)
- Payload → Enthält Informationen von dem Aufsteckmodul

Aus dem empfangenen (Modul-) Bundle wird in prepareData ein neues Bundle generiert. Dieses muss folgendes enthalten:

- accepted → Das empfangene Kommando konnte verarbeitet werden
- Sender → Aufsteckmodul (bspw. top)
- Command → Eindeutiges Kommando was ausgeführt wurde
- Message-Key → Eindeutige Parameter, enthält Informationen des Aufsteckmoduls

Das neu erstellte Bundle wird an die **ForegroundAPK** gesendet.

# 5 ForegroundAPK

Die ForegroundAPK stellt die letztendliche Android-App dar. Damit eine Android-App die Funktionalität des M2ModuleService verwenden kann, muss sie sich auf ihn binden. Der ComponentName lautet hierzu `componentName("de.acdgruppe.acdm2smartmoduleservice", „de.acdgruppe.acdm2smartmoduleservice.moduleservice.AcdModuleService")`
Die Kommunikation erfolgt über Messages, welche Bundle versendet. Das vollständige Protokoll wird im Kapitel 5.1 beschrieben.
Damit zwischen ForegroundAPK und BackgroundAPK kommuniziert werden kann, werden spezielle Command Bundle verwendet. Diese beinhalten einen Command und zugehörige Parameter. Der Command muss eindeutig einer BackgroundAPK zuzuordnen sein und muss von ACD spezifiziert werden. Ist der Command nicht eindeutig zuzuordnen, werden Anfragen direkt vom M2ModuleService verworfen.

Ein Link zum Template, wie eine ForegroundAPK auszusehen hat, finden Sie in Kapitel 1.2.

## 5.1 Bundle-Messanger der ForegroundAPK

Der Bundle-Messanger dient zur Kommunikation zwischen der **ForegroundAPK** und der **BackgroundAPK**.
Die Grundlegende Funktion von Bundles ist hier beschrieben:
https://developer.android.com/reference/android/os/Bundle

In der **ForegroundAPK** gibt es einen Bundle-Messanger, mit den folgenden Extras:

Ein Bundle das gesendet werden soll, muss folgende Extras enthalten:
- Receiver → Empfängermodul (bspw. top)
- Command → Eindeutiges Kommando zum Ausführen einer Aktion
- Message-Keys → Eindeutige Parameter zur Erweiterung des Commands

Ein zurückkommendes Bundle wird im Messangerhandler (Beispiel Template: moduleReceiver) empfangen.
Das empfangene Bundle enthält folgende Extras:
- accepted → Gibt an ob beim Verarbeiten ein Fehler aufgetreten ist
- Sender → Aufsteckmodul (bspw. top)
- Command → Eindeutiges Kommando, welches ausgeführt wurde
- Message-Key → Eindeutige Parameter, enthält Informationen des Aufsteckmoduls

# 6 Handlungsanweisung

## 6.1 Verwenden von bereitgestellten Aufsteckmodulen

Bereitgestellte Aufsteckmodule von ACD, haben eine BackgroundAPK von ACD.
Diese wird vom M2ModuleService geladen und das Modul kann verwendet werden.
Hierzu kann eine eigene ForegroundAPK geschrieben werden oder eine offiziell bestehende ForegroundAPK von ACD verwendet werden.

## 6.2 Verwenden von eigenentwickelten Aufsteckmodulen

Eigenentwickelt und produzierte Aufsteckmodule brauchen eine BackgroundAPK, sowie eine ForegroundAPK.
Um die BackgroundAPK und ForegroundAPK ordnungsgemäß aufzubauen, ist Kapitel 4. BackgroundAPK und Kapitel 5. ForegroundAPK zu beachten.

# 7 Overview

## 7.1 Scope of this documentation

This document is intended for developers who want to establish communication between apps and plug-in modules using the M2ModuleService. Both the protocols (ADIP, Service-Messages) and the main components (ForegroundAPK, M2ModuleService, BackgroundAPK, plug-in module) are explained in more detail.

## 7.2 Related documents

In order to use the M2ModuleService correctly, appropriate templates are available. They describe how the M2ModuleService can be correctly integrated into your own app.
The following templates are available in the ACD Customer Portal for download under the following link https://www.acd-gruppe.de/en/customerportal/:

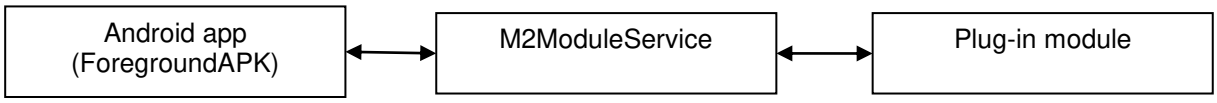
- Template_ForegroundAPK.java
- Template_BackgroundAPK.java

## 7.3 Description and use of the service

For easier communication between plug-in modules and Android apps, a service has been added to the operating system. On the one hand, this service handles the administration and communication to the plug-in modules. On the other hand it provides a messanger which can be used by Android apps. This makes it possible to send commands to plug-in modules and to receive data.

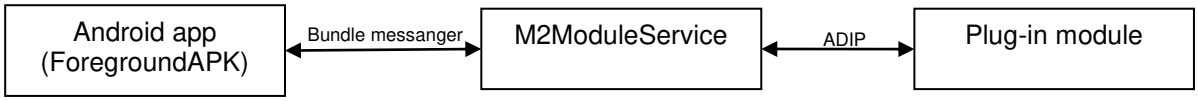
## 7.4 Structure of the service

The M2ModuleService is the connecting bridge between Android apps and plug-in modules.




Two different protocols are used for communication:

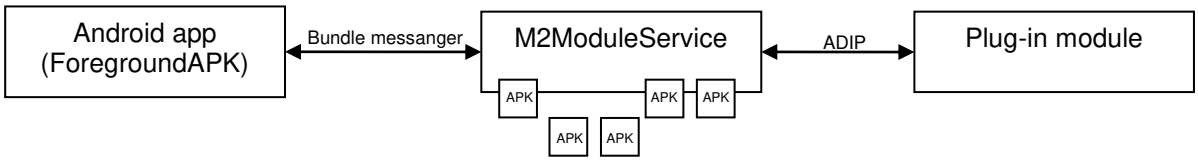
- There is a messanger between Android apps and M2ModuleService. The information is packed into bundles (for more information see section 10.1 and section 11.1).
- The "ACD Interconnect Protocol" ADIP is used for the communication between M2ModuleService and plug-in modules (for more information see section 9.1).




The executed program code is adapted during runtime to meet the requirements of the modular concept of the M2Smart®SE. Each plug-in module requires a special program code that interprets the information. These interpretations are stored in BackgroundAPKs (see section 10 BackgroundAPK). This compiled program code is reloaded at runtime as soon as an appropriate plug-in module is connected.

Thus the M2ModuleService consists of two components:

3. A static service, firmly anchored in Android (M2ModuleService)
4. One or more modular APKs, which contain the program code of all modules (BackgroundAPK)




# 8 Functions of the M2ModuleService

The static service is the core of the M2ModuleService. It is responsible for managing the plug-in modules and also serves as a communication bridge between plug-in modules and Android apps.
The most important functions of the M2ModuleService are briefly described below. A detailed explanation can be found under 10.1/11.1.

If a bound app wants to register for the messanger, it must send a "WHAT_HELLO (Value=2)" message to the service.
If a plug-in module is removed or attached, the service sends a message with the identification "WHAT_NEW_MODULE_PLUGGED (Value=5)" to all bound applications.
All attached modules can be queried with the message "WHAT_ALL_MODULES_PLUGGED (Value=6)".
If information is to be sent to a plug-in module or data is to be sent from the plug-in module, this is marked with the "WHAT_MODULE_MESSAGE (Value=4)".
The message "WHAT_MODULESERVICE_INFO (Value=7)" can be used to request general information about the service.

The static service uses one or more APKs that contain the program code for plug-in modules (BackgroundAPK). The BackgroundAPK contains the actual program code of the modules and can be exchanged at runtime. Its task is to extract the information from the transmitted data. Commands can be sent from ForegroundAPK to plug-in modules and data from the plug-in modules can be interpreted.

# 9 Communication protocols

## 9.1 ADIP

The "ACD Interconnect Protocol" ADIP is a protocol implemented by ACD M2 devices. ADIP is a very easy to understand protocol. The static M2ModuleService also implements this protocol and uses it to communicate with the plug-in modules. The relevant fields – receiver ID/sender ID, message ID and payload – are explained briefly below.

- **receiver ID/sender ID**
  This ID is assigned dynamically for each plug-in module. The receiver ID indicates where the data packet should go. The sender ID describes where the data packet comes from.
  The M2ModuleService with its BackgroundAPKs and ForegroundAPKs has the ID "Uplink" with the address 0x02.
  This ID is generated dynamically for each plug-in module (MODULE_ADDRESS). A readable dynamic string is used for communication between ForegroundAPK, M2ModuleService and BackgroundAPK (MODULE_POSITION).
  When **sending** commands to a plug-in module, a receiver ID (MODULE_POSITION) must always be specified. The sender ID is generated by the M2ModuleService itself.
  When the plug-in module **receives** information, the sender ID and receiver ID are transmitted to the BackgroundAPK and the ForegroundAPK (MODULE_POSITION). The processing of the message is independent of the slot (receiver/sender).

- **message ID**
  The message ID defines a special message. It is defined by ACD and must be assigned uniquely.
- **payload**
  The payload is optional and may contain further information.

The message ID is used to recognize and process a message. The slot of the plug-in module is not relevant for the processing of the message. The format and meaning of a message are independent of the slot.

# 10 BackgroundAPK

The static service uses one or more APKs that contain the program code for plug-in modules (BackgroundAPK). The BackgroundAPK contains the actual program code of the modules and can be exchanged at runtime. Its task is to extract the information from the transmitted data. Commands can be sent from the ForegroundAPK to plug-in modules and data from the plug-in modules can be interpreted.
There are two different types of BackgroundAPKs.

1. **ACD-specific:**
   These APKs are provided by ACD and are firmly anchored in the operating system. These module files are used by ACD plug-in modules.
2. **Customized:**
   These APKs are stored under '/sdcard/ACD/Add-On-Modules'. Customers can store their own modular BackgroundAPKs here.
3. **ACD update packages:**
   BackgroundAPKs which are stored under '/sdcard/ACD/Add-On-Modules' are given priority. It is therefore possible to update ACD-specific module files.

Several security parameters must be met before a BackgroundAPK can be loaded. These are explained below:

1. The class name must be in a specific format:
   Each plug-in module has a field with all available functionalities. This field is defined by ACD and anchored in the MCon of the plug-in module. For a class to be loaded, it must offer one of these functions and publish it in the class name. The format is: 'modulehandler_ ' + functionality.
   Example: 'modulehandler_temperatureACD'
2. The APK must be stored in a specific package:
   For a class to be loaded from the BackgroundAPK, it must be stored in the package 'de.acdgruppe.templatem2smartmodule'.
3. There are predefined functions that must be available in every class:
   - public void InitClass(Context context)
   - private boolean knownMessageID(byte[] messageID)
   - private boolean knownCommand(String command)
   - private Bundle prepareData(String receiver, String sender, byte[] message, byte[] payload)
   - private Bundle prepareCommand(Sring receiver, String sender, Bundle command)
   - public void UnInitClass()
4. message ID and Command may only appear once in the complete directory. If message ID or Command are found more than once, they are no longer uniquely assignable and are discarded.
5. The program code must be compiled so it is executable and stored as an APK under '/sdcard/ACD/Add-On-Modules'. A directory can contain multiple APKs. The choice of name of the APK makes no difference.

The actual program code for a plug-in module is processed within the BackgroundAPK. The main goal is to swap out as much program code as possible to the BackgroundAPK and thereby minimize redundant code. Any standard Java computing operations can be executed. ACD services (such as ACD ScanService) can also be used. Other methods and variables can be used as well.

A link to the template, which shows what a BackgroundAPK should look like, can be found in section 7.2.

## 10.1 Bundle-Messanger der BackgroundAPK

The Bundle Messenger is used for communication between the **ForegroundAPK** and the **BackgroundAPK**.
The fundamental function of Bundles is described here:
https://developer.android.com/reference/android/os/Bundle

The **BackgroundAPK** contains a Bundle Messenger with the following tools:

### 10.1.1 prepareCommand

This function processes Bundles that are sent from the **ForegroundAPK** to the **BackgroundAPK**.
prepareCommand contains the following:

- Command → Unique command for performing an action
- Receiver → Receiver module (e.g. top)
- Message-Keys → Unique parameters for expanding the command

A new Bundle is generated in prepareCommand from the Foreground Bundle that is received. It must contain the following:

- accepted → The received command was processed
- Receiver → receiver module (e.g. top) is forwarded by ForegroundAPK
- Sender → Transmitter station (ForegroundAPK)
- Message → Contains the unique Message-ID (assigned by ACD)
- Payload → Contains information for the receiver module

The newly created Bundle is sent to the receiver (e.g. module)

### 10.1.2 prepareData

This function processes Bundles that are sent from the receiver (e.g. module) to the **BackgroundAPK**.
prepareData contains the following:

- Receiver → Receiving station (ForegroundAPK)
- Sender → Attachable module
- Message → Contains the unique Message ID (as by ACD)
- Payload → Contains information from the attachable module

A new Bundle is generated in prepareData from the Bundle (module) that is received. It must contain the following:

- accepted → The received command was processed
- Sender → Attachable module (e.g. top)
- Command → Unique command that was performed
- Message-Key → Unique parameter containing information about the attachable module

The newly created Bundle is sent to the **ForegroundAPK**.

# 11 ForegroundAPK

The ForegroundAPK is the final Android app. An Android app must be bound to an M2ModuleService to be able to use its functionality.
The ComponentName for this purpose is 'componentName("de.acdgruppe.acdm2smartmoduleservice", "de.acdgruppe.acdm2smartmoduleservice.moduleservice.AcdModuleService")'
The communication takes place via messages, which are sent in bundles. The complete protocol is described in section 11.1.
Special command bundles are used to enable communication between ForegroundAPK and BackgroundAPK. They contain a command and associated parameters. The command must be uniquely assignable to a BackgroundAPK and must be specified by ACD. If the command cannot be clearly assigned, requests are rejected directly by the M2ModuleService.

A link to the template, which shows what a ForegroundAPK should look like, can be found in section 7.2.

## 11.1 Bundle messenger of the ForegroundAPK

The Bundle Messenger is used for communication between the **ForegroundAPK** and the **BackgroundAPK**.
The fundamental function of Bundles is described here:
https://developer.android.com/reference/android/os/Bundle

The **ForegroundAPK** contains a Bundle Messenger with the following tools:

For a Bundle to be sent, it must contain the following tools:

- Receiver → Receiver module (e.g. top)
- Command → Unique command for performing an action
- Message-Keys → Unique parameters for expanding the command

A Bundle that is returning is received in the Messenger Handler (sample template: moduleReceiver).
The received Bundle contains the following tools:

- accepted → Indicates whether an error occurred during processing
- Sender → Attachable module (e.g. top)
- Command → Unique command that was performed
- Message-Key → Unique parameter containing information about the attachable module

# 12 Instructions

## 12.1 Using the plug-in modules provided

The plug-in modules provided by ACD have a BackgroundAPK from ACD.
This is loaded by the M2ModuleService and the module can be used.
You can write your own ForegroundAPK or use an official ForegroundAPK available from ACD.

## 12.2 Using plug-in modules developed in-house

Plug-in modules developed and produced in-house need a BackgroundAPK as well as a ForegroundAPK.
To set up the BackgroundAPK and ForegroundAPK properly, see section 10 BackgroundAPK and section 11 ForegroundAPK must be observed.