
# Future Railway Mobile Communication System

# Form Fit Functional Interface Specification

| | |
|---|---|
| **Source:** | **UIC** |
| **Date:** | **12/02/2023** |
| **Reference:** | **FRMCS FFFIS-7950** |
| **Version:** | **1.0.0** |
| **No of pages:** | **59** |

ISBN 978-2-7461-3120-0

Document history

| Version | Date | Details |
|---|---|---|
| 0.0.1 | 24.02.2021 | Creation of the document |
| 0.0.2 to 0.0.8 | 08.07.2021 | Interim internal versions |
| 0.0.9 | 13.07.2021 | First version reviewed internally |
| 0.0.10 | 23.07.2021 | Add description of API features |
| 0.0.11 | 28.07.2021 | Modifications after first review with industries |
| 0.0.12 | 29.07.2021 | Modifications after additional review with industries |
| 0.1.0 | 02.08.2021 | Draft for Review (S2R Consortium) |
| 0.1.1 | 08.10.2021 | Interim version including all comments received from S2R |
| 0.1.2 | 16.11.2021 | Interim version with consolidation of content |
| 0.1.3 | 22.11.2021 | Modifications after internal review |
| 0.2.0 | 22.11.2021 | Second Draft for Review (S2R Consortium) |
| 0.2.1 | 08.12.2021 | Modifications to reflect all S2R Consortium comments |
| 0.2.2 | 16.12.2021 | Modifications after internal review |
| 0.3.0 | 17.12.2021 | Stable FFFIS draft content mainly applicable to $OB_{APP}$ for last consortium review |
| 0.3.1 | 18.01.2022 | Modifications after comments received from Kontron |
| 0.4.0 | 21.01.2022 | Final FFFIS draft with content mainly applicable to $OB_{APP.}$<br>For ERA EECT Review as official deliverable of SC3/SC4 |
| 0.4.1 | 29.03.2022 | Update to take into account EECT comments |
| 0.4.2 | 15/04/2022 | Clarification of API parameters and update to reflect EECT comments (06/04/22) |
| 0.5.0 | 06/05/2022 | Consolidated FFFIS final draft to consider EECT review comments and API parameters evolutions |
| 0.5.1 | 10/06/2022 | Consolidation of IP negotiation parameters during Session start |
| 0.6.0 | 30/06/2022 | Consolidated FFFIS final draft to consider EECT review comments (round #3) and IP negotiation evolutions |
| 0.6.1 | 2/08/2022 | Update of API parameter structure and main comments from EECT |
| 0.7.0 | 19/08/2022 | Consolidated FFFIS with parameters and API messages encoded in ASN.1 format |
| 0.7.1 | 23/09/2022 | Consolidated FFFIS following open points resolutions work frame |
| 0.8.0 | 27/09/2022 | Update to take into account EECT review comments (09/09) |
| 0.9.0 | 11/10/2022 | Amendments from last EECT review round (EECT meeting on 7/10/2022) |
| 0.10.0 | 18/10/2022 | Amendments from last EECT review round (EECT meeting on 18/10/2022) |
| 1.0.0 | 12/02/2023 | Modifications proposed by ERA through “agency consistency check on FIS and FFFIS” document and new Annex added to present the "*Interoperability requirements in EU”* coming from “Agency proposal for categorisation annexes for RMR Baseline 0” document. |

Table of Contents

# 1 List of abbreviations

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| API | Application Programming Interface |
| ATO | Automatic Train Operation |
| CCTV | Closed Circuit Television |
| CP | Control Plane |
| CS / PS | Circuit Switch / Packet Switch |
| DSD | Driver Safety Device |
| ERTMS | European Rail Traffic Management System |
| ETCS | European Train Control System |
| EUG | ERTMS Users Group |
| FFS | For Further Study |
| FRMCS | Future Railway Mobile Communication System |
| GSM-R | Global System for Mobile Communications – Railway |
| GW | Gateway |
| IEC | International Electrotechnical Commission |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| KPI | Key Performance Indicator |
| MCG | Mobile Communication Gateway |
| MCX | 3GPP Mission Critical Services |
| MOTS | Modified Off The Shelf |
| $OB_{APP}$ | On-Board Application reference point/interface |
| O&M | Operations & Maintenance |
| PKI | Public Key Infrastructure |
| PSK | Phase-Shift Keying |
| QoS | Quality of service |
| RAM | Reliability Availability Maintainability |
| RAN | Radio Access Network |
| RBC | Radio Block Centre |
| REC | Railway Emergency Call |
| RF | Radio Frequency |
| SIP | Session Initiation Protocol |
| SRS | System Requirement Specification |
| SW | Software |

| | |
|---|---|
| TCMS | Train Control and Management System |
| TCN | Train communication network |
| TLS | Transport Layer Security |
| TOBA | Telecom On-Board Architecture |
| $TS_{APP}$ | Trackside Application reference point/interface |
| TSI | Technical Specification for Interoperability |
| TSI CCS | Control Command and Signalling TSI |
| UE | User Equipment |
| UIC | Union Internationale des Chemins de Fer |
| UP | User Plane |
| URS | User Requirements Specification |
| WG | (UIC) Work Group |

# 2 List of definitions

Application

Provides functionality to the end user to cover a certain communication need necessary for current and future railway operations.

Communication services

Communication services enable two-way communication between two or more authorised service users (i.e. applications) from applications towards other applications/entities reachable through various networks.

Control Plane

The Control Plane (CP) carries signalling traffic between the network entities. Control plane and User Plane are to be considered independently of one another and can accordingly be managed separately between entities.

FRMCS Domain

A FRMCS Domain is an administrative domain which comprises a Service Domain and a Transport Domain under the control of an FRMCS Operator.

FRMCS System

Telecommunication system conforming to FRMCS specifications.

FRMCS Service client

Client that enables the use of the Communication Services and/or Complementary Services for the railway applications.

FRMCS Service server

Server that enables the use of the Communication Services and/or Complementary Services for the railway applications.

On-Board FRMCS

System enabling FRMCS communication to on-board applications. The On-Board FRMCS achieves a decoupling between On-Board Application(s) and transport service. For some applications, the decoupling is also achieved for the communication service.

Trackside FRMCS

System enabling FRMCS communication to trackside applications. The Trackside FRMCS achieves a decoupling between Trackside Application(s) and transport service. For some applications, the decoupling is also achieved for the communication service.

Interface

In this FFFIS, Interface and Reference Point describe the same notion, where Reference Point is used when discussing architecture, whereas Interface is the word used for the specification.

Low Layers

The term “low layers” corresponds to the OSI (Open Systems Interconnection) layers below the Application layer in the context of this FFFIS.

Lower Layers

The term “lower layers” originates from the UNIFE Working Group “FRMCS Lower Layers Requirements” and corresponds to the OSI layers 3 and below in the context of an on-board common bus.

Reference Point

Conceptual point applicable for interaction between functional services that enables authorised functions, e.g. in the network, to access their services. In this FFFIS, Interface and Reference Point describe the same notion, where Reference Point is used when discussing architecture, whereas Interface is the word used for the specification.

Transport service

It is a service that provides transport of user information and control signals between corresponding reference points considering the required QoS for the individual communication.

User Plane

The User Plane (UP) carries the user/application traffic. For the exchange of information between the communication partners (payload), the User Plane provides the necessary formats in order to provide the desired quality. Voice, video and data require different formats, for instance Codec to enable communication between partners. This is determined by the corresponding User Plane instance on the application side and controlled accordingly.

# 3 References

## 3.1 Applicability

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

## 3.2 List of References

| | |
|---|---|
| **[FRMCS-FRS]** | UIC, FRMCS, Functional Requirements Specification, FU-7120 |
| **[FRMCS-SRS]** | UIC, FRMCS, System Requirements Specification, FW-AT-7800 |
| **[TOBA-FRS]** | UIC**,** FRMCS, On-Board FRMCS – Functional Requirements Specification, TOBA-7510 |
| **[FRMCS-FIS]** | UIC, FRMCS, Functional Interface Specification, FIS-7970 |
| **[3GPP TS 22.280]** | 3GPP, Technical Specification Group Services and System Aspects; Mission Critical Services Common Requirements (MCCoRe) Stage 1 |
| **[3GPP TS 23.280]** | 3GPP, Technical Specification Group Services and System Aspects; Common functional architecture to support mission critical services; Stage 2 |
| **[3GPP TS 23.281]** | 3GPP, Technical Specification Group Services and System Aspects; Functional architecture and information flows to support Mission Critical Video (MCVideo); Stage 2 |
| **[3GPP TS 23.282]** | 3GPP, Technical Specification Group Services and System Aspects; Functional architecture and information flows to support Mission Critical Data (MCData); Stage 2 |
| **[3GPP TS 23.379]** | 3GPP, Technical Specification Group Services and System Aspects; Functional architecture and information flows to support Mission Critical Push To Talk (MCPTT); Stage 2 |
| **[3GPP TS 24.281]** | 3GPP, Technical Specification Group Services and System Aspects; Mission Critical Video (MCVideo) signalling control; Protocol specification |
| **[3GPP TS 24.282]** | 3GPP, Technical Specification Group Services and System Aspects; Mission Critical Data (MCData) signalling control; Protocol specification |
| **[3GPP TS 24.379]** | 3GPP, Technical Specification Group Services and System Aspects; Mission Critical Push To Talk (MCPTT) call control; Protocol specification |
| **[3GPP TS 33.180]** | 3GPP, Technical Specification Group Services and System Aspects; Security of the Mission Critical (MC) service |
| **[SUBSET-147]** | UNISIG ERTMS/ETCS and ATO over ETCS – FFFIS part: Communication Layers |

**[RFC 9113]** Hypertext Transfer Protocol Version 2 (HTTP/2) specifications, JUNE 2022

**[RFC 8259]** The JavaScript Object Notation (JSON) Data Interchange Format, December 2017

# 4 Introduction

## 4.1 Purpose of this document

4.1.1 This Form Fit Functional Interface Specification (FFFIS) specifies the following interfaces: (I)

- **$OB_{APP}$**, reference point between the On-Board Applications and the On-Board FRMCS, which is defined in **[FRMCS-SRS]**,
- and **$TS_{APP}$**, reference point between the Trackside FRMCS and the Trackside Applications, which is defined in **[FRMCS-SRS]**.

4.1.2 Figure 1 below is a simplified FRMCS architecture. It depicts the main high-level functional blocks and indicates the location of the $OB_{APP}$ and $TS_{APP}$ interfaces. (I)

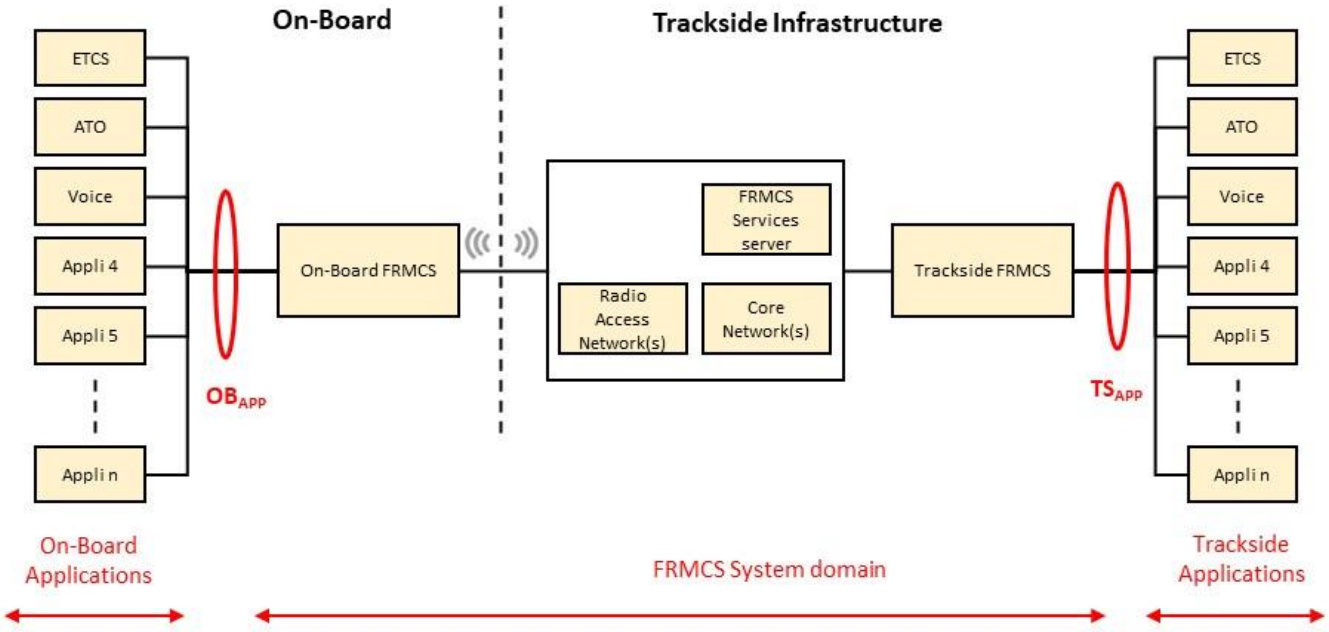


*Figure 1: Positions of $OB_{APP}$ and $TS_{APP}$ interfaces*

Note: the difference between Interface and Reference Point is given in chapter 2 (List of definitions).

## 4.2 Scope of this document

4.2.1 This FFFIS specifies the protocols, the messages and the format of the information exchanged over the $OB_{APP}$ and $TS_{APP}$ interfaces which enable interfacing between applications and the FRMCS System. (I)

4.2.2 This FFFIS cannot be used separately as the FRMCS specifications (**[FRMCS-FRS]**, **[FRMCS-SRS]**, **[FRMCS-FIS]** and **[TOBA-FRS]**) have to be considered as a whole. (I)

4.2.3 This FFFIS is part of the FRMCS specifications as depicted in figure below: (I)

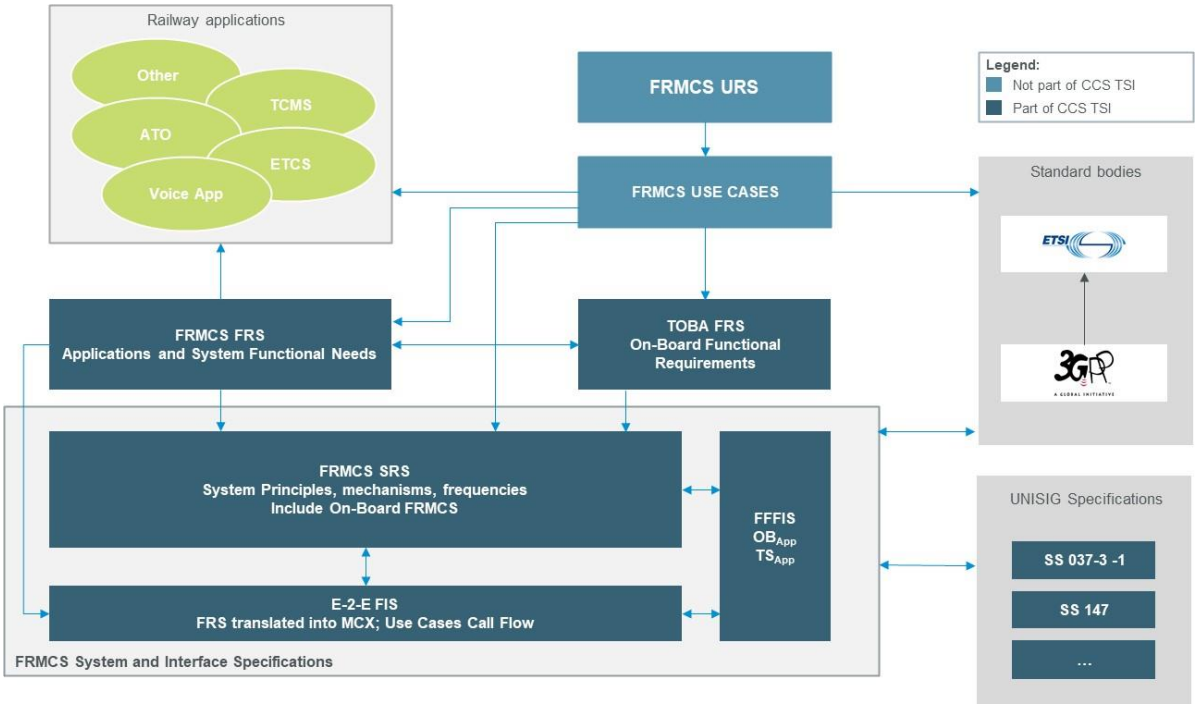


*Figure 2: FRMCS specifications*

4.2.4 The performance, availability, redundancy and security requirements applicable to $OB_{APP}$ and $TS_{APP}$ interfaces are defined in chapter 6. (I)

4.2.5 An On-Board application using On-Board FRMCS uses the communication layers defined in chapter 7. This FFFIS does not assume a train common bus in all cases, but only referring to **[SS-147]** for the case there is a common bus. (I)

4.2.6 A Trackside application using the Trackside FRMCS uses the communication layers defined in chapter 8 (I).

4.2.7 An On-Board application using On-Board FRMCS uses the API defined in chapter 9 (I).

4.2.8 A Trackside application using the Trackside FRMCS uses the API defined in chapter 10 (I).

## 4.3 Categorization of requirements

4.3.1 The requirements are categorised in (I):

- Mandatory for the System (indicated by ‘(M)’ at the end of the clause). These requirements mean a condition set out in this specification that must be met without exception in order to deliver a system ensuring the fulfilment of essential functional and system needs, compliance to relevant standards and technical integration. The mandatory requirements are identified as sentences using the keyword “shall”.
- Optional for the system (indicated by ‘(O)’ at the end of the clause). These requirements may be used based on the implementers’ choice. When an option is selected, the related requirement(s) of this specification becomes mandatory for the system. The optional requirements are identified as sentences using the keyword “should”.
- Information (indicated by “(I)” at the end of the clause). These statements provide additional information to help the reader understanding a requirement.

# 5 General principles

Note: this chapter is for information purpose only. It provides a description of the FRMCS messages going through the $OB_{APP}$ and $TS_{APP}$ leading to a better understanding of the different modes to be supported. The FRMCS end-to-end information is provided in the FRMCS Functional Interface Specifications **[FRMCS-FIS]**.

Editor's note: $TS_{APP}$ interface description is FFS. As a result, all information provided in this chapter related to the $TS_{APP}$ interface is for information only and cannot be considered as definitive.

## 5.1 $OB_{APP}$: Interface between On-Board Applications(s) and On-Board FRMCS

5.1.1 The $OB_{APP}$ corresponds to the interface between the On-Board Application(s) and the On-Board FRMCS. This interface ensures management of and access to the communication services allowing the authentication, authorisation and quality of service profile management requested by those applications. (I)

Note: information regarding the authentication and authorisation mechanisms can be found in the section 6.3 and chapter 9.

5.1.2 User Plane data from and to the application(s) is carried over the $OB_{APP}$ interface. (I)

5.1.3 Control Plane data exchange between application and On-Board FRMCS is performed over the $OB_{APP}$ interface. (I)

## 5.2 Functions supported through the $OB_{APP}$ interface

5.2.1 The $OB_{APP}$ Control Plane exposes three main functions: (I)

- Local Binding function: The Local Binding function provides functionalities to establish a secure link between an On-Board Application and the On-Board FRMCS, ensuring mutual authentication of both parties through the $OB_{APP}$ as well as the integrity and confidentiality of the information exchanges related to the $OB_{APP}$ Control Plane. The Local Binding function is spread over several mechanisms described in section 6.3 for the $OB_{APP}$ Security requirements and in chapter 9 for the API Local registration and Event stream opening features. All On-Board Applications, regardless their coupling mode (Tight or Loose), must be successfully authenticated through the Local Binding function. Note: the coupled modes are described in the section 5.6;
- Service Session function: The Service Session function provides functionalities to establish or terminate connectivity to or from a remote end point for applications operating in Loose Coupled mode. It is implemented through the API Service session features described in chapter 9;

- Auxiliary function: The Auxiliary function provides functionalities for applications to subscribe / unsubscribe to the status of the communication service feed exposed by the On-Board FRMCS and to receive notifications related to this information feed. A communication service enables communication either in a peer-to-peer user configuration or within a group of authorized users. The status of the communication service provides information about communication service availability. The recipient of this status resides in the application stratum while the sender resides in the service stratum. The status of the communication service may be available only if the FRMCS client-server association is operational. The Auxiliary function is implemented through the API Auxiliary function features described in chapter 9.

  Editor's note: Auxiliary function could provide other notifications (e.g., Positioning/Location information, FRMCS time). Definition of the status of the communication service provided by the Auxiliary function will be reviewed after V1. This is FFS.

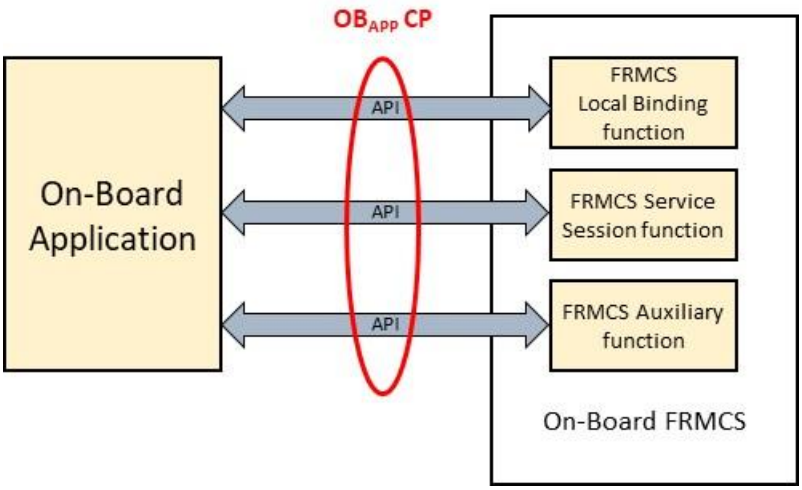


*Figure 3: API features exposed by the OB$_{APP}$ Control Plane interface*

## 5.3 TS$_{APP}$: Interface between Trackside Applications(s) and FRMCS Core Network

5.3.1 The TS$_{APP}$ corresponds to the interface between the Trackside Application(s) and the Trackside FRMCS. This interface ensures management of and access to the communication services allowing the authentication, authorisation, priority and quality of service profile management requested by those applications. (I)

## 5.4 Functions supported through the TS$_{APP}$ interface

Editor's note: It's assumed in the following that the functions supported at TS$_{APP}$ interface side will be similar to the functions supported at OB$_{APP}$ interface side. The detailed description of the TS$_{APP}$ interface is FFS.

## 5.5 $OB_{APP}$ and $TS_{APP}$ Logical End-to-End connectivity

5.5.1 The logical end-to-end User Plane connectivity (between applications) and logical Control Plane connectivity (between application and service server) flows through the FRMCS system boundaries using a FRMCS message flow compatible with the $OB_{APP}$ and $TS_{APP}$ interfaces specifications. (I)

5.5.2 Applications using the FRMCS System can be categorized in various Application regimes depending on the nature and extent of usage of the $OB_{APP}$ and $TS_{APP}$ interfaces. (I)

| Application regime | $OB_{APP}$ / $TS_{APP}$ coupling mode | FRMCS Service client in application? | FRMCS Service client in On-Board / Trackside FRMCS? |
|---|---|---|---|
| **Tight** | Tight Coupled mode | Yes | No |
| **Loose** | Loose Coupled mode | No | Yes |
| **Superloose** | Loose Coupled mode (via agent) | No | Yes |

*Table 1: Application regimes*

Editor's Note: the applicability / feasibility of the Superloose Application Regime for a trackside application is FFS.

Note: Superloose Application regime is defined as per the above table as the application being $OB_{APP}$ / $TS_{APP}$-unaware and interacting through an agent (out of FRMCS specifications) implementing $OB_{APP}$ / $TS_{APP}$ on behalf of the application. A communication via an agent is not valid for Tight coupled applications.

Note: refer to the **[FRMCS-FIS]** for more details about the end-to-end transaction flows and description of the Coupled modes.

## 5.6 FRMCS Service session in Tight Coupled mode

Note: The figure below depicts the Service session exchanges in Tight Coupled mode. The Local Binding function, Auxiliary function and SIP Core are not shown in the figure.

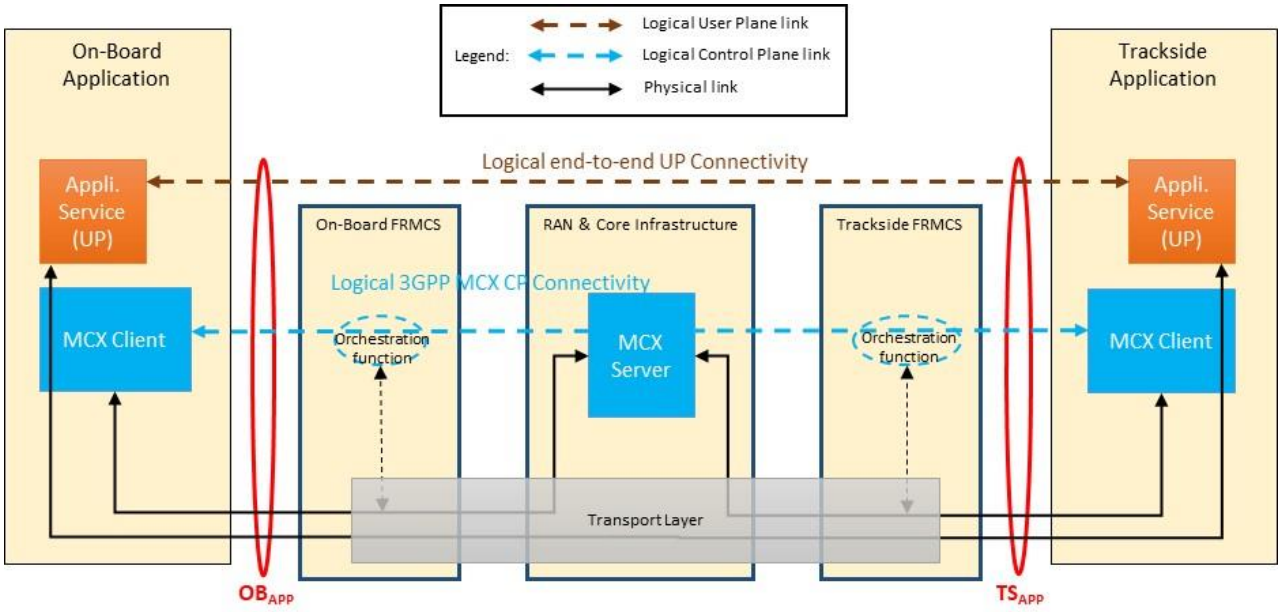


*Figure 4: End-to-End Service session for Applications in Tight coupled mode*

5.6.1 In Tight Coupled mode, after the Local Binding function (see section 5.2.1) has been successfully performed, the embedded MCX client of the application establishes the logical MCX connectivity based on 3GPP MCX protocol operations with the necessary information. All exchanges at $OB_{APP}$ and $TS_{APP}$ sides between the Application and the FRMCS System are based on standardised 3GPP MCX services over IP (see section 9.20 and 10.4). (I)

Note: The On-Board FRMCS encompasses an Orchestration function available in Tight and Loose coupled modes that enables the necessary routing and steering capability for the user plane associated with a specific service session. This is not in the scope of this FFFIS. The On-Board FRMCS Orchestration function is described in the **[FRMCS-SRS]**.

Editor's note: the Trackside FRMCS Orchestration function description is FFS.

5.6.2 In Tight Coupled mode, the Application User Plane is carried out over $OB_{APP}$ and $TS_{APP}$ through the embedded MCX client of the Application. Refer to section 9.20 and section 10.4. The Application User Plane over $OB_{APP}$ and $TS_{APP}$ is secured depending on the type of application. Refer to section 6.3 and section 6.6 for the Security requirements. (I)

## 5.7 FRMCS Service session in Loose Coupled mode

Note: The figure below depicts the Service session exchanges in Loose Coupled mode The Local Binding function, Auxiliary function and SIP Core are not shown in this figure.

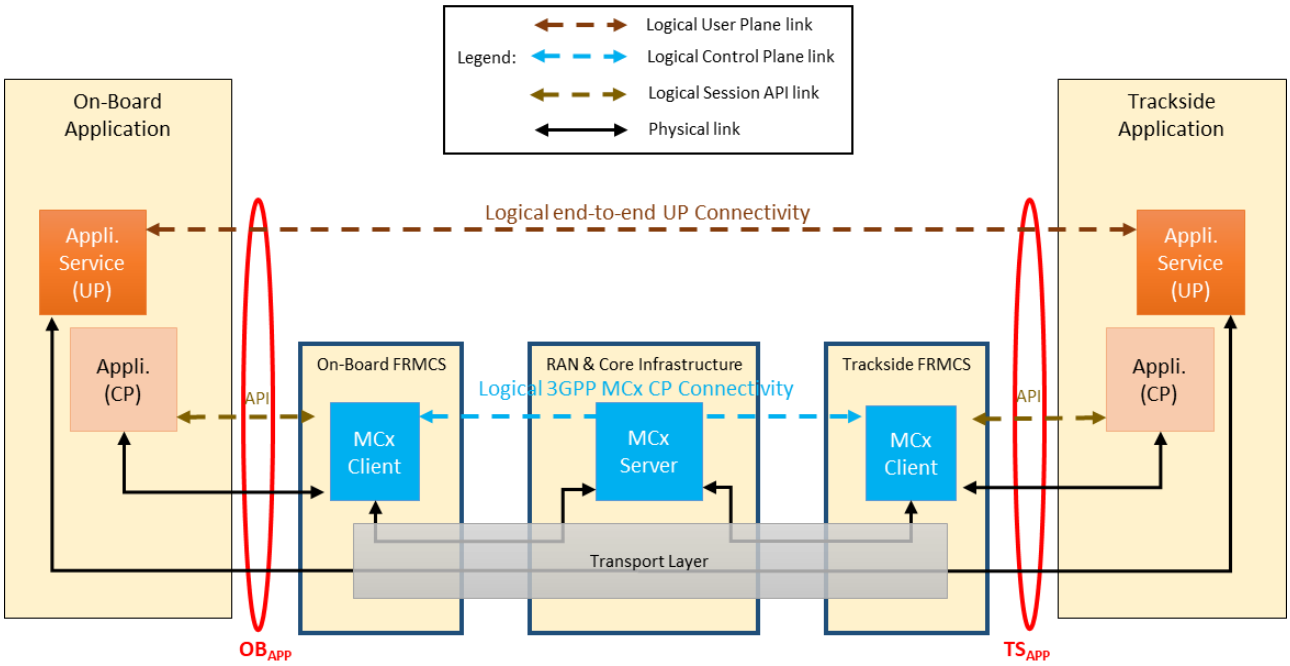


*Figure 5: End-to-End Service session for Applications in Loose coupled mode*

5.7.1 In Loose Coupled mode, the On-Board FRMCS initiates a FRMCS Service registration between the FRMCS Service client (MCX Client in the figure) and the FRMCS Service server (MCX Server in the figure). After the Local Binding function (see section 5.2.1) has been successfully performed, the Application requests the FRMCS System to establish a logical Application Control Plane based on 3GPP MCX on its behalf. It does it by calling a dedicated application interface (API) exposed by the On-Board FRMCS or by the Trackside FRMCS. The features supported by this API are described in the Functional Services chapters (refer to chapter 9 and section 10.1). The On-Board FRMCS and Trackside FRMCS are in charge to translate these API calls into the relevant calls to standardised 3GPP MCX protocol operations with the necessary information. (I)

5.7.2 In Loose Coupled mode, the logical Application User Plane connectivity managed by the application is carried out through the $OB_{APP}$ and $TS_{APP}$ over IP. This User Plane dataflow is then managed by the FRMCS Service client (MCX Client in the figure) located in the On-Board FRMCS and Trackside FRMCS. The $OB_{APP}$ and $TS_{APP}$ User Plane should be secured depending on the type of application. Refer to section 6.3 and section 6.6 for the security requirements. (I)

# 6 Performance, Availability, Redundancy and Security

This chapter provides the requirements in terms of performance, availability, redundancy, and security for both $OB_{APP}$ and $TS_{APP}$.

## 6.1 $OB_{APP}$ Performance requirements

6.1.1 The physical layer of the $OB_{APP}$ interface at On-Board FRMCS side shall support a minimum gross data rate of 100 Mbit/s. (M)

## 6.2 $OB_{APP}$ Availability / Redundancy requirements

6.2.1 The $OB_{APP}$ interface is designed to achieve the availability and redundancy requirements in accordance with the principles defined in **[SUBSET-147]**. (I)

Editor's note: the $OB_{APP}$ requirement about the possibility for the application to use one or more On-Board FRMCSs for data transmission is FFS.

## 6.3 $OB_{APP}$ Security requirements

6.3.1 In the case of a connection to be established between an On-Board Application and an On-Board FRMCS, and when they are connected to a train network compliant with **[SUBSET-147]**, their interface shall comply with the authentication mechanisms specified in **[SUBSET-147]**. (M)

6.3.2 For application authentication on the $OB_{APP}$ Control Plane, mutual authentication based on client and server certificates shall be performed between the application and the On-Board FRMCS using the Transport Layer Security (TLS) protocol. During the TLS handshake, client (application) and server (On-Board FRMCS) send their certificate and authenticate themselves. (M)

6.3.3 The integrity and confidentiality protection of the $OB_{APP}$ Control Plane implemented through the API features shall rely on the Transport Layer Security (TLS) protocol. (M)

Editor's note: The exact requirements regarding the TLS and its associated version (1.2, 1.3 or higher) applicable to the $OB_{APP}$ Control Plane implemented through the API features are FFS. Several on-going open points (e.g. backward compatibility, Certificate Authority management, Identifiers, mutual authentication with PKI based key management, monitoring of traffic flow, etc.) related to the applicability of the $OB_{APP}$ CP security requirements are FFS.

6.3.4 When integrity and/or confidentiality protection is implemented at end-to-end level, no additional protection of the $OB_{APP}$ User Plane of the local link between the application and the On-Board FRMCS is required. (I)

Editor's note: In the first version of FRMCS, only applications which have end-to-end protection will be implemented. For applications that do not have an end-to-end protection (integrity and/or confidentiality) mechanism, the need for an appropriate protection mechanism of the $OB_{APP}$ User Plane of the local link is FFS. IPsec could be an option, but other solutions could be also considered.

## 6.4 $TS_{APP}$ Performance requirements

Editor's note: $TS_{APP}$ interface Performance requirements are FFS.

## 6.5 $TS_{APP}$ Availability/Redundancy requirements

Editor's note: $TS_{APP}$ interface Availability and redundancy requirements are FFS.

## 6.6 $TS_{APP}$ Security requirements

Editor's note: $TS_{APP}$ interface Security requirements are FFS.

# 7 $OB_{APP}$ Low layers specifications and protocol stacks

## 7.1 $OB_{APP}$ Connectivity

7.1.1 The On-Board Applications need to have connectivity to use the On-Board FRMCS. This connectivity can be established according to different technical choices depending on which device/entity the application is installed, e.g., commercial off-the-shelf (COTS) computer, proprietary fixed equipment within a train. (I)

7.1.2 The figure below presents the two possibilities to logically connect an Application to the On-Board FRMCS. (I)

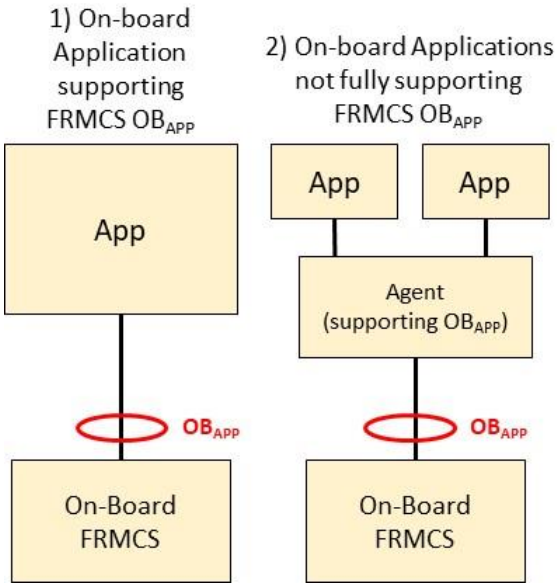


*Figure 6: Logical implementation options for applications requiring access to On-Board FRMCS*

7.1.3 In case the application does not support $OB_{APP}$ requirements (physical and/or logical), an agent supporting $OB_{APP}$ is used in between to connect to the On-Board FRMCS. The physical and logical interface specifications between the application and agent are outside the scope of the FRMCS specifications. (I)

Note: refer to the **[FRMCS-FIS]** for more details about the end-to-end transaction flows.

## 7.2 $OB_{APP}$ Physical interface

7.2.1 The physical interface of the $OB_{APP}$ at On-Board FRMCS side is made of common off-the-shelf technologies based on Ethernet (IEEE 802.3). (I)

7.2.2 When connected to a train network compliant with **[SUBSET-147]**, the physical interface of the $OB_{APP}$ at On-Board FRMCS side shall be made in accordance with **[SUBSET-147]**. (M)

Editor's note: other physical interface possibilities such as Wi-Fi are FFS. On-Board hardware platform development initiatives could be considered as soon as they are mature enough for consideration.

7.2.3 The $OB_{APP}$ interface should support the following physical interface requirements: (O)

- Support links over copper twisted-pair cable or over fiber-optical cable
- Use standardized physical connectors which are compliant with environmental requirements of railways, for instance M12 in case of twisted-pair cable or 10GBASE-SR connector in case of fiber-optical cable.
- Use cabling that is prepared for 10GBit/s link speeds

Editor's note: the possibility for the $OB_{APP}$ interface at On-Board FRMCS side to share the physical interface with others IP flows is FFS.

## 7.3 $OB_{APP}$ Internet Protocol versions

7.3.1 All messages exchanged over $OB_{APP}$ interface shall be based on Internet Protocol IPv6. (M)

Editor's note: the support of IPv4, in addition to IPv6 for backward compatibility will be investigated for next version of this specification.

## 7.4 $OB_{APP}$ local IP allocation scheme

7.4.1 At the $OB_{APP}$ interface side, the On-Board FRMCS is seen as a host in the train network and hence it shall be configured in accordance with the IP plan of the train network. The On-Board FRMCS, when in a train equipped with a Common Bus, benefits from services offered by the Common Bus as defined in **[SUBSET-147]**. (I)

7.4.2 The local IP address of the On-Board FRMCS API Control Plane can be based on a predefined configuration, set as a configuration parameter through the operation and maintenance interface of the On-Board FRMCS. (I)

Editor's note: improvement of the local IP address allocation of the On-Board FRMCS API Control Plane and clarification of the path to request the service via HTTP are FFS.

## 7.5 $OB_{APP}$ Protocol stacks

7.5.1 The different protocol stacks that shall be used over the $OB_{APP}$ interface between the On-Board Application and the On-Board FRMCS are presented in the table below: (M)

| Role | $OB_{APP}$ Message flow | Protocol stack to be used |
|---|---|---|
| **Local Binding Control Plane** | API features. Refer to Chapter 9 | HTTP/2 over TLS. Refer to standard RFC 9113. |
| **Service session User Plane** | Transparent applications data UP in Loose Coupled or Tight Coupled mode | IP Protocol |
| **Service session Control Plane** | API features for applications in Loose Coupled mode. Refer to Chapter 9 | HTTP/2 over TLS. Refer to standard RFC 9113. |
| | Transparent MCX message flow over IP for applications in Tight Coupled mode: | MCX framework protocol over IP. See reference 3GPP TS 22.280, TS 23.280 and TS 33.180. Applicable version number is provided in the Reference chapter |
| **Auxiliary function Control Plane** | API features. Refer to Chapter 9 | HTTP/2 over TLS. Refer to standard RFC 9113. |

*Table 2: List of protocol stacks used over the $OB_{APP}$ interface*

Note: refer to section 9.3.3 and following for more details about the HTTP usage for the API features.

7.5.2 The data format of the $OB_{APP}$ API shall be realized following JSON specifications **[RFC 8259]**. (M)

# 8 $TS_{APP}$ Low layers specifications and protocol stacks

## 8.1 $TS_{APP}$ Connectivity

8.1.1 The Trackside Applications need to have connectivity to use the Trackside FRMCS. This connectivity can be established according to different technical choices depending on which device/entity the application is installed, e.g. commercial off-the-shelf (COTS) computer, proprietary fixed equipment. It depends also on the location of the physical Application entities and Trackside FRMCS. (I)

8.1.2 The communication network architecture and distance between the Trackside Application and the Trackside FRMCS are fully dependant on implementation choice of the Railway infrastructure manager. This is outside the scope of this FFFIS. (I)

8.1.3 In case the application does not support $TS_{APP}$ requirements (physical and/or logical), an agent supporting $TS_{APP}$ is used in between to connect to the Trackside FRMCS. The physical and logical interface specifications between the application and agent are outside the scope of the FRMCS specifications. (I)

Note: refer to the **[FRMCS-FIS]** for more details about the end-to-end transaction flows.

## 8.2 $TS_{APP}$ Physical interface

8.2.1 The physical interface of the $TS_{APP}$ at Trackside FRMCS side is made of common off-the-shelf technologies based on Ethernet (IEEE 802.3). (I)

8.2.2 The $TS_{APP}$ interface should support the following physical interface requirements: (O)

- Support links over copper twisted-pair cable or over fiber-optical cable
- Use standardized physical connectors, for instance RJ45 or M12 in case of twisted-pair cable or 10GBASE-SR or LR connector in case of fiber-optical cable.

## 8.3 $TS_{APP}$ Internet Protocol versions

8.3.1 All messages exchanged over $TS_{APP}$ interface shall be based on Internet Protocol IPv6. (M)

Editor's note: the support of IPv4, in addition to IPv6 for backward compatibility will be investigated for next revision of this specification.

## 8.4 $TS_{APP}$ local IP allocation scheme

8.4.1 The Trackside FRMCS shall expose on $TS_{APP}$ an IP interface with an IP gateway address that can be used by the Trackside Applications to send/receive User Plane and Control Plane data to/from the remote Applications. (M)

## 8.5 $TS_{APP}$ Protocol stacks

8.5.1 The different protocol stacks that shall be used over the $TS_{APP}$ interface between the Trackside Applications and Trackside FRMCS are presented in the table below: (M)

| Role | Message flow | Protocol stack to be used |
|---|---|---|
| **Local Binding Control Plane*** | API features | HTTP/2 over TLS. Refer to standard RFC 9113. |
| **Service session User Plane** | Transparent applications data UP in Loose Coupled or Tight Coupled mode | IP Protocol |
| **Service session Control Plane** | API features for applications in Loose Coupled mode | HTTP/2 over TLS. Refer to standard RFC 9113. |
| | Transparent MCX message flow over IP for applications in Tight Coupled mode: | MCX framework protocol over IP. See reference 3GPP TS 22.280, TS 23.280 and TS 33.180. Applicable version number is provided in the Reference chapter |
| **Auxiliary function Control Plane*** | API features | HTTP/2 over TLS. Refer to standard RFC 9113. |

Editor's note (*): Local Binding and Auxiliary functions on $TS_{APP}$ are FFS. See section 5.4.

*Table 3: List of protocol stacks used over the $TS_{APP}$ interface*

8.5.2 The data format of the $TS_{APP}$ API shall be realized following JSON specifications **[RFC 8259]**. (M)

# 9 $OB_{APP}$ Functional Services Messages and Dataflow

## 9.1 Overview of $OB_{APP}$ API features

$OB_{APP}$ enables the following features between an application and the On-Board FRMCS:

9.1.1 **Event stream opening feature**: This feature shall be used to request the creation of the event stream enabling the On-Board FRMCS to send notifications to the On-Board application. (M)

9.1.2 **Local registration feature**: This feature shall be used to perform the Local registration between an On-Board application and the On-Board FRMCS. The local registration to the On-Board FRMCS shall be carried out only once after the start of the application. (M)

9.1.3 **Session start feature**: This feature shall be used to establish a communication session between an On-Board application and a remote (Trackside or On-Board) application. (M)

Editor's note: The analysis of the potential impact on the Session start feature of a Host-to-Network session establishment is FFS.

9.1.4 **Session status feature**: This feature should be used to get a list of all sessions that may be opened between an On-Board application and a remote (Trackside or On-Board) application, and that are still open. (O)

9.1.5 **Auxiliary function subscription feature**: This feature should be used to subscribe to a set of FRMCS information notification. (O)

Note: the auxiliary function notification subscription is linked to the successful local registration.

9.1.6 **Auxiliary function notification feature**: In case an On-Board application has subscribed to an Auxiliary function information, this feature should be used to notify the On-Board application about the subscribed Auxiliary function information. (O)

9.1.7 **Auxiliary function query feature**: This feature should be used to request the current status of FRMCS information notification provided by the Auxiliary function. (O)

9.1.8 **Auxiliary function unsubscription feature**: In case an On-Board application has subscribed to an Auxiliary function information, this feature should be used to unsubscribe to Auxiliary function information notification. (O)

9.1.9 **Session end feature**: This feature shall be used to release a communication session between an On-Board application and a remote (Trackside or On-Board) application. (M)

9.1.10 **Incoming session start feature**: This feature shall be used to inform an On-Board application of an incoming session start requested by a remote (Trackside or On-Board) application. (M)

9.1.11 **Incoming session end feature**: This feature shall be used to inform the On-Board application of an incoming session end requested by a remote (Trackside or On-Board) application. The On-Board FRMCS can use this feature in case a session is ended because, for instance, there is a breakdown of the communication session detected at On-Board FRMCS side. (M)

9.1.12 **Local deregistration feature**: This feature shall be used to request a local de-registration of the On-Board application from the On-Board FRMCS. (M)

9.1.13 **Event stream closing feature**: This feature shall be used to close the event stream following the deregistration. (M)

Note: in the context of the API, the term application refers to the application instance, which is a concrete running software occurrence of an application of a specific type.

Note: If the optional Auxiliary function subscription feature is selected, the corresponding optional features (Auxiliary function notification and Auxiliary function unsubscription) must be also selected.

9.1.14 The Local Binding function shall consist of: (M)

(i) A first step in which an application and the On-Board FRMCS shall mutually authenticate using TLS, which is not part of the API. See section 6.3 for more details.

(ii) A second step in which the application, through the API Event stream opening and Local registration features, shall request the local registration to the On-Board FRMCS, including the transmission of the supported version(s) of $OB_{APP}$.

(iii) And a third step in which the On-Board FRMCS, through the API Local registration feature, shall accept or reject the local registration, including for incompatible $OB_{APP}$ versions, and then notify the application of the decision and of the chosen $OB_{APP}$ version, if applicable.

9.1.15 In case an agent represents multiple applications, the agent is in charge to register for each application. In such case, there is one Local binding phase per represented application. These represented applications are considered independently at On-Board FRMCS side. (I)

Note: an agent is an entity (as described in **[FRMCS-SRS]**) that implements the API for applications that do not have this capability.

9.1.16 Any API features beside local registration feature and event stream opening feature shall be conditioned on the successful execution of the Local Binding steps. (M)

## 9.2 Terminology in $OB_{APP}$ API features

9.2.1 In this section, the following definitions apply to the different type of API messages: (I)

- An application request is defined as a message sent from an application to the On-Board FRMCS.
- An On-Board FRMCS answer is defined as a message sent from the On-Board FRMCS to an application in response to an application request.
- An On-Board FRMCS notification is defined as a message sent from the On-Board FRMCS to an application without being triggered by an application request.
- An On-Board FRMCS request is defined as a message sent from the On-Board FRMCS to an application.
- An application answer is defined as a message sent from an application to the On-Board FRMCS in response to an On-Board FRMCS request.

9.2.2 The following table summarizes the different types of API messages: (I)

| Name | From | To | Following |
|---|---|---|---|
| **Application request** | Application | On-Board FRMCS | - |
| **On-Board FRMCS answer** | On-Board FRMCS | Application | Application request |
| **On-Board FRMCS notification** | On-Board FRMCS | Application | - |
| **On-Board FRMCS request** | On-Board FRMCS | Application | - |
| **Application answer** | Application | On-Board FRMCS | On-Board FRMCS request |

*Table 4: Summary of the different type of API messages.*

Note: The fourth column ("Following") presents the triggers to some messages. The prerequisites for the messages are described in section 9.3.2.

## 9.3 Summary of the API features and corresponding message names:

9.3.1 The API features shall respect the message names presented in Table 5 below. The table highlights the source and destination of the message through the message type, and the type of coupling mode. The "Coupling" column indicates which application is concerned by the message: "T" for Tight coupled applications and "L" for Loose Coupled applications. (M)

| Feature | Message name | Message type | Coupling |
|---|---|---|---|
| **Event stream opening** | FRMCS_EVENT_STREAM_OPENING_APPLICATION_REQUEST | Application request | T, L |
| **Event stream opening** | FRMCS_EVENT_STREAM_OPENING_ON-BOARD_FRMCS_ANSWER | On-Board FRMCS answer | T, L |
| **Local registration** | FRMCS_LOCAL_REGISTRATION_APPLICATION_REQUEST | Application request | T, L |
| **Local registration** | FRMCS_LOCAL_REGISTRATION_ON-BOARD_FRMCS_ANSWER | On-Board FRMCS answer | T, L |
| **Session start** | FRMCS_SESSION_START_APPLICATION_REQUEST | Application request | L |
| **Session start** | FRMCS_SESSION_START_ON-BOARD_FRMCS_FIRST_ANSWER | On-Board FRMCS answer | L |
| **Session start** | FRMCS_SESSION_START_ON-BOARD_FRMCS_FINAL_ANSWER | On-Board FRMCS answer | L |
| **Session status** | FRMCS_SESSION_STATUS_APPLICATION_REQUEST | Application request | L |
| **Session status** | FRMCS_SESSION_STATUS_ON-BOARD_FRMCS_ANSWER | On-Board FRMCS answer | L |
| **Auxiliary function subscription** | FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_APPLICATION_REQUEST | Application request | T, L |

| Feature | Message name | Message type | Coupling |
|---|---|---|---|
| **Auxiliary function subscription** | FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_ON-BOARD_FRMCS_ANSWER | On-Board FRMCS answer | T, L |
| **Auxiliary function notification** | FRMCS_AUXILIARY_FUNCTION_ON-BOARD_FRMCS_NOTIFICATION | On-Board FRMCS notification | T, L |
| **Auxiliary function query** | FRMCS_AUXILIARY_FUNCTION_QUERY_APPLICATION_REQUEST | Application request | T, L |
| **Auxiliary function query** | FRMCS_AUXILIARY_FUNCTION_QUERY_ON-BOARD_FRMCS_ANSWER | On-Board FRMCS answer | T, L |
| **Auxiliary function unsubscription** | FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_APPLICATION_REQUEST | Application request | T, L |
| **Auxiliary function unsubscription** | FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_ON-BOARD_FRMCS_ANSWER | On-Board FRMCS answer | T, L |
| **Session end** | FRMCS_SESSION_END_APPLICATION_REQUEST | Application request | L |
| **Session end** | FRMCS_SESSION_END_ON-BOARD_FRMCS_ANSWER | On-Board FRMCS answer | L |
| **Incoming session start** | FRMCS_INCOMING_SESSION_START_ON-BOARD_FRMCS_REQUEST | On-Board FRMCS request | L |
| **Incoming session start** | FRMCS_INCOMING_SESSION_START_APPLICATION_ANSWER | Application answer | L |
| **Incoming session end** | FRMCS_INCOMING_SESSION_END_ON-BOARD_FRMCS_NOTIFICATION | On-Board FRMCS notification | L |
| **Local deregistration** | FRMCS_LOCAL_DEREGISTRATION_APPLICATION_REQUEST | Application request | T, L |
| **Local deregistration** | FRMCS_LOCAL_DEREGISTRATION_ON-BOARD_FRMCS_ANSWER | On-Board FRMCS answer | T, L |
| **Event Stream closing** | FRMCS_EVENT_STREAM_CLOSING_ON-BOARD_FRMCS_NOTIFICATION | On-Board FRMCS answer | T, L |

*Table 5: Summary of the API features and the corresponding message names.*

Note: the usage of session related features by Tight Coupled mode applications is not envisaged in the current version of this FFFIS.

9.3.2 The table below summarizes the request / answer relationships between the messages and their prerequisites. (I)

| Message name | Prerequisite | Answer to |
|---|---|---|
| **FRMCS_EVENT_STREAM_OPENING_APPLICATION_REQUEST** | | |
| **FRMCS_EVENT_STREAM_OPENING_ON-BOARD_FRMCS_ANSWER** | | FRMCS_EVENT_STREAM_OPENING_APPLICATION_REQUEST |
| **FRMCS_LOCAL_REGISTRATION_APPLICATION_REQUEST** | Event stream opening | FRMCS_EVENT_STREAM_OPENING_ON-BOARD_FRMCS_ANSWER |

| Message name | Prerequisite | Answer to |
|---|---|---|
| **FRMCS_LOCAL_REGISTRATION_ON-BOARD_FRMCS_ANSWER** | Event stream opening | FRMCS_LOCAL_REGISTRATION_APPLICATION_REQUEST |
| **FRMCS_SESSION_START_APPLICATION_REQUEST** | Local registration | |
| **FRMCS_SESSION_START_ON-BOARD_FRMCS_FIRST_ANSWER** | Local registration | FRMCS_SESSION_START_APPLICATION_REQUEST |
| **FRMCS_SESSION_START_ON-BOARD_FRMCS_FINAL_ANSWER** | Local registration | FRMCS_SESSION_START_APPLICATION_REQUEST |
| **FRMCS_SESSION_STATUS_APPLICATION_REQUEST** | Local registration | |
| **FRMCS_SESSION_STATUS_ON-BOARD_FRMCS_ANSWER** | Local registration | FRMCS_SESSION_STATUS_APPLICATION_REQUEST |
| **FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_APPLICATION_REQUEST** | Local registration | |
| **FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_ON-BOARD_FRMCS_ANSWER** | Local registration | FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_APPLICATION_REQUEST |
| **FRMCS_AUXILIARY_FUNCTION_ON-BOARD_FRMCS_NOTIFICATION** | Auxiliary function subscription | |
| **FRMCS_AUXILIARY_FUNCTION_QUERY_APPLICATION_REQUEST** | Local registration | |
| **FRMCS_AUXILIARY_FUNCTION_QUERY_ON-BOARD_FRMCS_ANSWER** | Local registration | FRMCS_AUXILIARY_FUNCTION_QUERY_APPLICATION_REQUEST |
| **FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_APPLICATION_REQUEST** | Auxiliary function subscription | |
| **FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_ON-BOARD_FRMCS_ANSWER** | Auxiliary function subscription | FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_APPLICATION_REQUEST |
| **FRMCS_SESSION_END_APPLICATION_REQUEST** | Session start | |
| **FRMCS_SESSION_END_ON-BOARD_FRMCS_ANSWER** | Session start | FRMCS_SESSION_END_APPLICATION_REQUEST |
| **FRMCS_INCOMING_SESSION_START_ON-BOARD_FRMCS_REQUEST** | Local registration | |
| **FRMCS_INCOMING_SESSION_START_APPLICATION_ANSWER** | Local registration | FRMCS_INCOMING_SESSION_START_ON-BOARD_FRMCS_REQUEST |
| **FRMCS_INCOMING_SESSION_END_ON-BOARD_FRMCS_NOTIFICATION** | Local registration | |
| **FRMCS_LOCAL_DEREGISTRATION_APPLICATION_REQUEST** | Local registration | |
| **FRMCS_LOCAL_DEREGISTRATION_ON-BOARD_FRMCS_ANSWER** | Local registration | FRMCS_LOCAL_DEREGISTRATION_APPLICATION_REQUEST |
| **FRMCS_EVENT_STREAM_CLOSING_ON-BOARD_FRMCS_NOTIFICATION** | Local registration | FRMCS_LOCAL_DEREGISTRATION_APPLICATION_REQUEST |

*Table 6: Summary of the request/answer relationships between messages.*

9.3.3 The following API messages shall use the HTTP messages as presented in the table below. (M)

| Feature | API Message name | HTTP message |
|---|---|---|
| **Event stream opening** | FRMCS_EVENT_STREAM_OPENING_APPLICATION_REQUEST | HTTP request method=GET |
| **Event stream opening** | FRMCS_EVENT_STREAM_OPENING_ON-BOARD_FRMCS_ANSWER | HTTP response |
| **Local registration** | FRMCS_LOCAL_REGISTRATION_APPLICATION_REQUEST | HTTP request method=POST |
| **Local registration** | FRMCS_LOCAL_REGISTRATION_ON-BOARD_FRMCS_ANSWER | HTTP response |
| **Session start** | FRMCS_SESSION_START_APPLICATION_REQUEST | HTTP request method=POST |

| Feature | API Message name | HTTP message |
|---|---|---|
| **Session start** | FRMCS_SESSION_START_ON-BOARD_FRMCS_FIRST_ANSWER | HTTP response |
| **Session start** | FRMCS_SESSION_START_ON-BOARD_FRMCS_FINAL_ANSWER | SSE |
| **Session status** | FRMCS_SESSION_STATUS_APPLICATION_REQUEST | HTTP request method=POST |
| **Session status** | FRMCS_SESSION_STATUS_ON-BOARD_FRMCS_ANSWER | HTTP response |
| **Auxiliary function subscription** | FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_APPLICATION_REQUEST | HTTP request method=POST |
| **Auxiliary function subscription** | FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_ON-BOARD_FRMCS_ANSWER | HTTP response |
| **Auxiliary function notification** | FRMCS_AUXILIARY_FUNCTION_ON-BOARD_FRMCS_NOTIFICATION | SSE |
| **Auxiliary function query** | FRMCS_AUXILIARY_FUNCTION_QUERY_APPLICATION_REQUEST | HTTP request method=POST |
| **Auxiliary function query** | FRMCS_AUXILIARY_FUNCTION_QUERY_ON-BOARD_FRMCS_ANSWER | HTTP response |
| **Auxiliary function unsubscription** | FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_APPLICATION_REQUEST | HTTP request method=POST |
| **Auxiliary function unsubscription** | FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_ON-BOARD_FRMCS_ANSWER | HTTP response |
| **Session end** | FRMCS_SESSION_END_APPLICATION_REQUEST | HTTP request method=POST |
| **Session end** | FRMCS_SESSION_END_ON-BOARD_FRMCS_ANSWER | HTTP response |
| **Incoming session start** | FRMCS_INCOMING_SESSION_START_ON-BOARD_FRMCS_REQUEST | SSE |
| **Incoming session start** | FRMCS_INCOMING_SESSION_START_APPLICATION_ANSWER | HTTP request method=POST (*) |
| **Incoming session end** | FRMCS_INCOMING_SESSION_END_ON-BOARD_FRMCS_NOTIFICATION | SSE |
| **Local deregistration** | FRMCS_LOCAL_DEREGISTRATION_APPLICATION_REQUEST | HTTP request method=POST |
| **Local deregistration** | FRMCS_LOCAL_DEREGISTRATION_ON-BOARD_FRMCS_ANSWER | HTTP response |
| **Event stream closing** | FRMCS_EVENT_STREAM_CLOSING_ON-BOARD_FRMCS_NOTIFICATION | HTTP response |

*Table 7: HTTP functionality to be used by the API messages*

Note (*): The HTTP response to the FRMCS_INCOMING_SESSION_START_APPLICATION_ANSWER message (carried by a HTTP request) is not presented in the Table 7 as no API message is associated to it.

9.3.4 The API shall use HTTP request with method = POST to carry every API message REQUEST from the On-Board Application to the On-Board FRMCS, except for the creation of the event stream where it shall use HTTP request with method = GET as specified in **[RFC 9113]**. (M)

9.3.5 The API shall use HTTP response to carry every API message ANSWER from the On-Board FRMCS to the On-Board Application. (M)

9.3.6 The API shall use the HTTP status code field (refer to **[RFC 2616]**) contained in the HTTP response in order to get the status of the HTTP request. (M)

9.3.7 The API shall use Server-Sent Events (SSE) based on EventSource interface to carry every API message NOTIFICATION using the opened event stream from the On-Board FRMCS to the On-Board Application. (M)

Note: The EventSource interface is web content's interface to server-sent events. An EventSource instance opens a persistent connection to the HTTP server (On-Board FRMCS) thanks to the Event stream opening feature (refer to section 9.1.1), which allows to send events in text/event-stream format from On-Board FRMCS to the application. Refer for instance to: https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events.

Editor's note: the management of the "keep alive" of the opened event stream during the overall connection between the On-Board Application and the On-Board FRMCS is FFS.

9.3.8 The API messages contained in the body of HTTP request and HTTP response shall be encoded in JSON as specified in **[RFC 8259]**. (M)

## 9.4 Definition of the parameters used in the API features:

9.4.1 The parameter types that are used by the API features shall respect the format presented in Table 8 below. (M)

9.4.2 The content of the parameters in Table 8 that are defined with the ASN.1 UTF8String type shall be encoded based on the Normalization Form KC (NFKC) as specified in the Unicode® Standard Annex #15 (see https://unicode.org/reports/tr15/). (M)

Editor's note: About the coding of the UTF8String content, an analysis will be done after V1 to check if it is possible to rely only to JSON (RFC 8259) and HTTP/2 (RFC 9113) definitions without referring to the NFKC. This is FFS.

Note: All parameters and messages are respectively presented in ASN.1 format and ASN.1 notation to structure them for a better understanding. As mentioned previously, the API messages contained in the body of HTTP request and HTTP response will be encoded in JSON. The ASN.1 notation of all $OB_{APP}$ API parameters and messages is presented in the Annex A.

Editor's note: The JSON encoding of API parameters and messages will be presented later (after V1) in an Annex.

| # | Parameter name | Details | Description in ASN.1 format |
|---|---|---|---|
| 1 | Application category | Provides the category of the application instance (ETCS, ATO, CabRadio, etc.). When applicable, this parameter is used by the On-Board FRMCS to manage the registration to the necessary 3GPP MCX service(s). | `ApplicationCategory ::= ENUMERATED {etcs, ato, cabRadio}` |
| 2 | Static identifier of an application | Unique identifier of an application instance<br>Refer to sections 9.4.5 | `ApplicationStaticId ::= UTF8String (SIZE(3..256))` |
| 3 | Version of $OB_{APP}$ | Version of the interface: [major.minor] or empty if unsuccessful | `OBAppVersion ::=  UTF8String (SIZE(0..5))` |
| 4 | Coupling mode of the application | Among Tight Coupled mode and Loose Coupled mode | `CouplingMode ::= ENUMERATED {tight, loose}` |
| 5 | Application On-Board identifier | Identifier of the application instance dynamically assigned at the On-Board FRMCS, unique in the scope of the On-Board FRMCS. The format of this parameter is based on UUID (see parameter 7). This parameter is empty if request is unsuccessful | `appOBId Uuid` |

| # | Parameter name | Details | Description in ASN.1 format |
|---|---|---|---|
| 6 | Remote address | Remote address of an application in the scope of session exchange messages.<br>Refer to section 9.4.6 | `RemoteAddress ::= UTF8String (SIZE(3..256))` |
| 7 | UUID | UUID (Universally Unique Identifier, refer to RFC 4122) format used for following parameters:<br>- 5, Application On-Board Identifier<br>- 9, Identifier of a session | `Uuid ::= UTF8String (SIZE(36))` |
| 8 | deleted | deleted | `Deleted` |
| 9 | Identifier of a session | Identifier of the session, unique in the scope of the On-Board FRMCS. The format of this parameter is based on UUID (see parameter 7). This parameter is empty if request is unsuccessful | `sessionId Uuid` |
| 10 | Category of Auxiliary function information | Category of Auxiliary function information e.g. status of the communication service. Editor's note: Only "status of the communication service" is supported in FRMCS V1. Other information such as location update, FRMCS time, observed QoS, etc. are FFS. | `AuxiliaryFunctionCategory ::= ENUMERATED{communicationStatus}` |
| 11 | Update period for the auxiliary function | Requested update period for the auxiliary function information in seconds. 0 means "no periodic update", only status changes are sent to the Application. If periodic update value is greater or equal to 1 then status changes are not sent spontaneously. The information is transmitted to the Application periodically. Maximum value is 120 seconds. | `-- update period in seconds. 0 = on change event information --`<br>`AuxiliaryFunctionUpdatePeriod ::= INTEGER(0..120)` |
| 12 | Value of auxiliary function information | Value of auxiliary function information sent by the On-Board FRMCS to the Application.<br>In V1, only the value of the "status of the communication service" can be obtained. Refer to parameter 17, Status of the communication service. Other values such as location, time, etc. are FFS. | `AuxFunctionValue ::= CHOICE{`<br>`  commStatValue  CommStatValue,`<br>`  ffs NULL`<br>`}` |
| 13 | Status of the request | Additional description of the HTTP RESPONSE status code placed in the Answer of a Request message.<br>The content of this status depends on which API Answer message it is used in. There are 4 parameters associated to this status.<br>Refer to parameters 22, 23, 24, 25 | `reqStatus GenericReqStatus`<br>`reqStatus LocRegReqStatus`<br>`reqStatus SessionStartReqFirstStatus`<br>`reqStatus SessionStartReqFinalStatus` |

| # | Parameter name | Details | Description in ASN.1 format |
|---|---|---|---|
| 14 | Communication category | This parameter reflects the different categories of communication (session) that can be established over the FRMCS according to the [FRMCS FRS]. This parameter is used by the On-Board FRMCS to initiate an end-to-end session. Based on this parameter, the On-Board FRMCS assigns the right communication profile including the QoS level to the session.<br>Refer to sections 9.4.7.<br>The possible modes applicable to the data and video communications are respectively defined in parameters 26 (DataComm) and 27 (VideoComm) | CommunicationCategory::= CHOICE{<br>dataComm DataComm,<br>videoComm VideoComm<br>} |
| 15 | Session acceptance decision | Decision of the application regarding an incoming session.<br>The content of this parameter is based on the Generic status of the request. Refer to parameter 22 | sessionStartDecision GenericReqStatus |
| 16 | IP address | IPv6 address to be used by the application as destination address for the user plane data.<br>Editor's note: Supporting both IPv4 and IPv6 is FFS. The corresponding JSON schema reflecting the IP address in the API messages is also FFS. | IPAddress::= UTF8String (SIZE(1..43)) |
| 17 | Status of the communication service | Value of the "Status of the communication service" provided by the Auxiliary function. See parameter 12 | CommStatValue ::= ENUMERATED{available, notAvailable} |
| 18 | Session status | Provides the status of a given session as requested in the Session status request message | SessionStatusReqStatus ::= ENUMERATED{established, inProgress, networkNotReady, notRegistered, rejected} |
| 19 | Origin of the session establishment | Provides the origin of the session establishment: local application or incoming remote application | SessionEstablishmentOrigin ::= ENUMERATED{localApplication, remoteApplication} |
| 20 | Status of an Auxiliary function subscription | Provides the Auxiliary function subscription status | AuxFunctionSubStatus ::= ENUMERATED{active, inactive} |
| 21 | Status of an Auxiliary function unsubscription | Provides the Auxiliary function unsubscription status | AuxFunctionUnsubStatus ::= ENUMERATED{rejectedNotSubscribed, successfullyUnsubscribed, alreadyUnsubscribed} |
| 22 | Generic status of the request | Provides the status of a request used in several API Answer messages. The possible values are "accepted" or "rejected". If the status of the request is rejected, there is an additional field (256 characters) to provide more details | GenericReqStatus ::= CHOICE{<br>accepted NULL,<br>rejected UTF8String (SIZE(0..256))<br>} |
| 23 | Status of the Local registration request | Provides the status of a Local registration request. This parameter is used in the Local registration answer message. If the status of the request is "not registered", there is an additional field (256 characters) to provide more details | LocRegReqStatus ::= CHOICE{<br>registered NULL,<br>notRegistered UTF8String (SIZE(0..256))<br>} |

| # | Parameter name | Details | Description in ASN.1 format |
|---|---|---|---|
| 24 | Status of the Session start request (first) | Provides the first status of a Session start request. This parameter is used in the Session start first answer message. | `SessionStartReqFirstStatus ::= ENUMERATED{inProgress, networkNotReady, notRegistered, rejected}` |
| 25 | Status of the Session start request (final) | Provides the final status of a Session start request. This parameter is used in the Session start final answer message. | `SessionStartReqFinalStatus ::= ENUMERATED{established, rejected}` |
| 26 | Data communication mode | The data communication mode is used by the Communication category parameter. It reflects the possible values applicable to this mode: basic or critical. | `DataComm ::= ENUMERATED{basic, critical}` |
| 27 | Video communication mode | The video communication mode is used by the Communication category parameter. It reflects the possible values applicable to this mode: basic or critical. | `VideoComm ::= ENUMERATED{basic, critical}` |

*Table 8: Definition of the parameter types that are used in the API features.*

9.4.3 The $OB_{APP}$ interface to the On-Board FRMCS will have different **versions** as new features will be introduced. Supported versions are communicated over the $OB_{APP}$. For each interface version, a change log is maintained, and changes are categorised into Major and Minor categories. (I)

9.4.4 The $OB_{APP}$ version number of this FFFIS shall be V1.0. Where “1” is the major version number and “0” the minor version number of the current version. (M)

9.4.5 The Static identifier of the application shall be unique in the scope of all FRMCS application instances. The structure of FRMCS System identities that are used to set up the relevant FRMCS services and communication link(s) with other FRMCS users shall fulfil the requirements as specified in the **[FRMCS-SRS]**. (M)

9.4.6 The remote address of an application in the scope of $OB_{APP}$ session exchange messages shall fulfil the requirements as specified in the **[FRMCS-SRS]**. (M)

9.4.7 The communication profiles, containing the list of allowed QoS profiles for each Communication category, are set as a configuration file through the operation and maintenance interface of the On-Board FRMCS. During a session establishment, the applicable communication profile is determined according to the value of the Communication category provided by the application. (I)

Editor’s note: error handling within the API features and the related state diagrams, are FFS.

Note: in the following clauses, the parameter types are referenced by number and descriptive name as <#, string>.

## 9.5 Event stream opening feature:

9.5.1 The **FRMCS_EVENT_STREAM_OPENING_APPLICATION_REQUEST** shall be an empty GET message with content-type “text/event stream”. (M)

Note: there is no ASN.1 notation for FRMCS_EVENT_STREAM_OPENING_APPLICATION_REQUEST as it is a GET message with no content.

9.5.2 The **FRMCS_EVENT_STREAM_OPENING_ON-BOARD_FRMCS_ANSWER** shall contain the following information: (M)

- Identifier of the application dynamically assigned at the On-Board FRMCS <5, application on-board identifier>.

9.5.3 ASN.1 notation for **FRMCS_EVENT_STREAM_OPENING_ON-BOARD_FRMCS_ANSWER** is: (I)

```
EventStreamOpeningOBAnswer ::= SEQUENCE{
 appOBId Uuid
}
```

Editor’s note: assessment of the use of the event stream opening feature is FFS .

## 9.6 Local registration feature:

9.6.1 The **FRMCS_LOCAL_REGISTRATION_APPLICATION_REQUEST** shall contain the following information: (M)

- Category of the registering application instance <1, application category>;
- Static identifier of the registering application instance <2, static identifier of an application>;
- Identifier of the application dynamically provided by the On-Board FRMCS at the event stream opening <5, application on-board identifier>.
- List of supported versions of $OB_{APP}$ <list of <3, version of OBAPP>>.

9.6.2 In case an agent is connected to the On-Board FRMCS, the category of the application instance shall be the category of the represented application instance. (M)

9.6.3 In case an agent is connected to the On-Board FRMCS, the static identifier of the application instance shall be the static identifier of the represented application instance. (M)

Note: an agent is an entity (as described in **[FRMCS-SRS]**) that implements the API for applications that do not have this capability.

9.6.4 The **FRMCS_LOCAL_REGISTRATION_APPLICATION_REQUEST** should contain the following information: (O)

- Coupling mode of the registering application instance <4, coupling mode of the application>. In the case this value is not provided, the per default value is “Loose coupled mode”.

9.6.5 ASN.1 notation for **FRMCS_LOCAL_REGISTRATION_APPLICATION_REQUEST** is: (I)

```
LocalRegAppReq ::= SEQUENCE{
  appCategory ApplicationCategory,
  staticId ApplicationStaticId,
  appOBId Uuid,
  obAppVersionList OBAppVersionList,
  couplingMode CouplingMode DEFAULT loose
}
-- Where: --
OBAppVersionList ::= SET OF OBAppVersion
```

9.6.6 The **FRMCS_LOCAL_REGISTRATION_ON-BOARD_FRMCS_ANSWER** shall contain the following information: (M)

- Status of the request <13, request status>;
- Chosen version of $OB_{APP}$ in case of successful registration <3, version of OBAPP>.

Note: the status of the request may include the rationale in case of failure.

9.6.7 ASN.1 notation for **FRMCS_LOCAL_REGISTRATION_ON-BOARD_FRMCS_ANSWER** is: (I)

```
LocalRegFRMCSAnswer ::=SEQUENCE{
  reqStatus LocRegReqStatus,
  selectedObAppVer OBAppVersion
}
```

## 9.7 Session start feature:

9.7.1 The **FRMCS_SESSION_START_APPLICATION_REQUEST** shall contain the following information: (M)

- On-board identifier of the requesting application <5, application on-board identifier>;
- Local Application IP address to be used by the On-Board FRMCS as destination address for the User Plane data in case of successful session establishment <16, IP address >;
- List of recipients of the communication with the following information for each recipient <list of>:
  - Remote address of the recipient of the communication <6, remote address>;
  - Category of communication applied to the session <14, communication category>;

Editor's note: FRMCS Multipath invoked at Application level will be considered after FRMCS V1. This point is FFS.

Editor's note: the handling of group communication is FFS:

1. In the case of groupcast, the session is addressed to one group identifier and the list contains one item.
2. In the case of ad-hoc group communication in which the service client (e.g. MC client) creates the group identifier after receiving the session start request, the list is filled with the members of the group.
3. In the case of no dedicated service for group communication, the session for a group of session can be broken down in multiple individual calls of session start request.

9.7.2 ASN.1 notation for **FRMCS_SESSION_START_APPLICATION_REQUEST** is: (I)

```
FRMCSSessionStartAppReq ::= SEQUENCE{
  appOBId Uuid,
  localAppIPAddress IPAddress,
  recipientList RecipientList
}
-- Where: --
RecipientList ::= SET OF Recipient

Recipient ::= SEQUENCE{
  remoteAddress RemoteAddress,
  communicationCategory CommunicationCategory
}
```

9.7.3 The **FRMCS_SESSION_START_ON-BOARD_FRMCS_FIRST_ANSWER** is sent directly after the session start request by the On-Board FRMCS and shall contain the following information: (M)

- Status of the request (in progress, network not ready, not registered or rejected) <13, request status>;

- Identifier of the session <9, identifier of a session>. This parameter is empty if status of the request is different from “in progress”.

9.7.4 ASN.1 notation for **FRMCS_SESSION_START_ON-BOARD_FRMCS_FIRST_ANSWER** is: (I)

```
FRMCSSessionStartFirstAns ::= SEQUENCE{
  reqStatus SessionStartReqFirstStatus,
  sessionId Uuid OPTIONAL
}
```

9.7.5 The **FRMCS_SESSION_START_ON-BOARD_FRMCS_FINAL_ANSWER** is sent by the On-Board FRMCS when it has resolved all addresses and shall contain the following information: (M)

- Status of the request (established or rejected) <13, request status>;
- Identifier of the session <9, identifier of a session>;
- Local On-Board FRMCS IP address to be used by the On-Board Application as destination address for the User Plane data in case of successful session establishment <16, IP address >.

9.7.6 The **FRMCS_SESSION_START_ON-BOARD_FRMCS_FINAL_ANSWER** shall be sent by the On-Board FRMCS only if the status of the request placed in the FRMCS_SESSION_START_ON-BOARD_FRMCS_FIRST_ANSWER is “in progress” (M)

9.7.7 ASN.1 notation for **FRMCS_SESSION_START_ON-BOARD_FRMCS_FINAL_ANSWER** is: (I)

```
FRMCSSessionStartFRMCSFinalAns ::= SEQUENCE{
  reqStatus SessionStartReqFinalStatus,
  sessionId Uuid,
  -- next field is present only in case reqStatus is established --
  localDestFRMCSIPAddress IPAddress OPTIONAL
}
```

## 9.8 Session status feature

9.8.1 In case the optional Session status feature is selected (refer to 9.1.4), the **FRMCS_SESSION_STATUS_APPLICATION_REQUEST** shall contain the following information: (M)

- On-board identifier of the requesting application <5, application on-board identifier>.

9.8.2 The **FRMCS_SESSION_STATUS_APPLICATION_REQUEST** should contain the following information: (O)

- A list of sessions for which the status is requested with the following information <list of>:
  - Identifier of the session <9, identifier of a session>

Note: if this parameter is not provided, the gateway assumes that all active sessions are required.

9.8.3 ASN.1 notation for **FRMCS_SESSION_STATUS_APPLICATION_REQUEST** is: (I)

```
FRMCSSessionStatAppReq::=SEQUENCE{
  appOBId Uuid,
  sessionIdList SessionIdList OPTIONAL
}
-- Where: --
SessionIdList ::= SET OF Uuid
```

9.8.4 In case the optional Session status feature is selected (refer to 9.1.4), the **FRMCS_SESSION_STATUS_ON-BOARD_FRMCS_ANSWER** shall contain: (M)

- Status of the request <13, request status>;
- A list of active sessions with the following information <list of>:
  - Identifier of the session <9, identifier of a session>
  - Status of the session (pending or established) <18, session status>
  - Origin of the session start request (application or incoming) <19, origin of the session establishment>
  - Category of communication applied to the session <14, communication category>
  - Local On-Board FRMCS IP address to be used by the On-Board application as destination address for the User Plane data of this session <16, IP address>
  - Local Application IP address to be used by the On-Board FRMCS as destination address for the User Plane data <16, IP address >;
  - List of recipients with the following information <list of>:
    - Remote address of a participant to the communication<6, remote address>

Note: This On-Board FRMCS answer has two behaviours depending on the optional parameter section 9.8.2. In the case it is not provided, all active sessions are returned. If it is provided, the returned list is filtered with the requested sessions.

Note: only the existing active sessions are returned. If the list provided in section 9.8.2 contains a non-existing or inactive session, no error is returned.

9.8.5 ASN.1 notation for **FRMCS_SESSION_STATUS_ON-BOARD_FRMCS_ANSWER** is: (I)

```
FRMCSSessionStatAns::=SEQUENCE{
 reqStatus GenericReqStatus,
 activeSessionList ActiveSessionList
}
-- Where: --
ActiveSessionList ::= SET OF ActiveSession

 ActiveSession::=SEQUENCE{
   sessionId Uuid,
   sessionStatus SessionStatusReqStatus,
   sessionOriginator SessionEstablishmentOrigin,
   communicationCategory CommunicationCategory,
   localDestFRMCSIPAddress IPAddress,
   localAppIPAddress IPAddress,
   remoteAddressList RemoteAddressList
```

```
 }

RemoteAddressList ::= SET OF RemoteAddress
```

## 9.9 Auxiliary function subscription feature

9.9.1 In case the optional Auxiliary function subscription feature is selected (refer to 9.1.5), the **FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_APPLICATION_REQUEST** shall contain the following information: (M)

- On-board identifier of the requesting application <5, application on-board identifier>;
- List of information to which the application requests the subscription with the following parameters <list of>:
  - name of the information <10, auxiliary function information category>
  - requested period of the subscription <11, update period for the auxiliary function>.

9.9.2 ASN.1 notation for **FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_APPLICATION_REQUEST** is: (I)

```
AuxiliaryFunctionSubReq ::= SEQUENCE{
 appOBId Uuid,
 auxFunctionSubList AuxFunctionSubList
}
-- Where: --
AuxFunctionSubList ::= SET OF AuxFunctionSubDef

  AuxFunctionSubDef::= SEQUENCE{
   auxiliaryFunctionCategory AuxiliaryFunctionCategory,
   auxiliaryFunctionUpdatePeriod AuxiliaryFunctionUpdatePeriod
 }
```

9.9.3 In case the optional Auxiliary function subscription feature is selected (refer to 9.1.5), tthe **FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_ON-BOARD_FRMCS_ANSWER** shall contain the following information: (M)

- Status of the request <13, request status>;
- List of subscription status <list of>:
  - name of the information <10, auxiliary function information category>
  - Status of an Auxiliary function subscription <20, status of a subscription>>.

9.9.4 ASN.1 notation for **FRMCS_AUXILIARY_FUNCTION_SUBSCRIPTION_ON-BOARD_FRMCS_ANSWER** is: (I)

```
AuxiliaryFunctionSubAns ::= SEQUENCE{
 reqStatus GenericReqStatus,
 auxFunctionStatList AuxFunctionStatList
}
```

```
-- Where: --
AuxFunctionStatList ::= SET OF AuxFunctionStat

 AuxFunctionStat ::= SEQUENCE{
   auxiliaryFunctionCategory AuxiliaryFunctionCategory,
   auxFunctionSubStatus AuxFunctionSubStatus
 }
```

## 9.10 Auxiliary function notification feature

9.10.1 In case the optional Auxiliary function notification feature is selected (refer to 9.1.6), the **FRMCS_AUXILIARY_FUNCTION_ON-BOARD_FRMCS_NOTIFICATION** shall contain the following information: (M)

- Name of the information corresponding to the value <10, auxiliary function information category>.
- Value of one of the information to which the application has previously subscribed <12, value of auxiliary function information>;

Note: The Auxiliary function notification can be sent to the application spontaneously in case of status change (e.g. the status of the communication service has changed from “not available” to “available”) or periodically based on the requested period parameter even if there is no status change. Refer to parameter 11 in Table 8.

9.10.2 ASN.1 notation for **FRMCS_AUXILIARY_FUNCTION_ON-BOARD_FRMCS_NOTIFICATION** is: (I)

```
AuxiliaryFunctionNotification ::= SEQUENCE{
  auxFunctionName AuxiliaryFunctionCategory,
  auxFunctionValue AuxFunctionValue
}
```

## 9.11 Auxiliary function query feature

9.11.1 In case the optional Auxiliary function query feature is selected (refer to 9.1.7), the **FRMCS_AUXILIARY_FUNCTION_QUERY_APPLICATION_REQUEST** shall contain the following information: (M)

- On-board identifier of the requesting application <5, application on-board identifier> ;
- List of information for which a status update is requested <10, auxiliary function information category>>.

9.11.2 ASN.1 notation for **FRMCS_AUXILIARY_FUNCTION_QUERY_APPLICATION_REQUEST** is: (I)

```
AuxFunctionQueryAppReq ::=SEQUENCE{
  appOBId Uuid,
  auxFunctionNameList AuxiliaryFunctionCategoryList
}
-- Where: --
AuxiliaryFunctionCategoryList ::=SET OF AuxiliaryFunctionCategory
```

9.11.3 In case the optional Auxiliary function query feature is selected (refer to 9.1.7), the **FRMCS_AUXILIARY_FUNCTION_QUERY_ON-BOARD_FRMCS_ANSWER** shall contain the following information: (M)

- Status of the request <13, request status>;
- List of last up-to-date statuses of the information to which the application requested a status update <list of>:
  - Name of the information corresponding to the value <10, auxiliary function information category>;
  - Value <12, value of auxiliary function information>;

9.11.4 ASN.1 notation for **FRMCS_AUXILIARY_FUNCTION_QUERY_ON-BOARD_FRMCS_ANSWER** is: (I)

```
AuxFunctionQueryAns ::= SEQUENCE{
 reqStatus GenericReqStatus,
 auxFunctionNotificationList AuxFunctionNotificationList
}
-- Where: --
AuxFunctionNotificationList ::= SET OF AuxiliaryFunctionNotification

 AuxiliaryFunctionNotification ::= SEQUENCE{
   auxFunctionName AuxiliaryFunctionCategory,
   auxFunctionValue AuxFunctionValue
 }
```

## 9.12 Auxiliary function unsubscription feature

9.12.1 In case the optional Auxiliary function unsubscription feature is selected (refer to 9.1.8), the **FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_APPLICATION_REQUEST** shall contain the following information: (M)

- On-board identifier of the requesting application <5, application on-board identifier> ;
- List of information to which the application requests the unsubscription <list of <10, auxiliary function information category>>. If there is no list, all subscriptions are removed.

9.12.2 ASN.1 notation for **FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_APPLICATION_REQUEST** is: (I)

```
AuxiliaryFunctionUnsubReq ::= SEQUENCE{
 appOBId Uuid,
 auxFunctionUnsubList AuxiliaryFunctionCategoryList OPTIONAL
}
-- Where: --
AuxiliaryFunctionCategoryList ::= SET OF AuxiliaryFunctionCategory
```

9.12.3 In case the optional Auxiliary function unsubscription feature is selected (refer to 9.1.8), the **FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_ON-BOARD_FRMCS_ANSWER** shall contain the following information: (M)

- Status of the request <13, request status>;
- List of unsubscriptions status with the following information:
  - Name of the information <10, auxiliary function information category>
  - Status of the unsubscription (successful, failed because not subscribed to this information, already unsubscribed) <list of <21, status of an unsubscription>>.

9.12.4 ASN.1 notation for **FRMCS_AUXILIARY_FUNCTION_UNSUBSCRIPTION_ON-BOARD_FRMCS_ANSWER** is: (I)

```
AuxiliaryFunctionUnsubAns ::= SEQUENCE{
 reqStatus GenericReqStatus,
 auxFunctionUnsubStatList  AuxFunctionUnsubStatList
}
-- Where: --
AuxFunctionUnsubStatList ::= SET OF AuxFunctionUnsubStat

  AuxFunctionUnsubStat ::= SEQUENCE{
    auxiliaryFunctionCategory AuxiliaryFunctionCategory,
    auxFunctionUnsubStatus AuxFunctionUnsubStatus
  }
```

## 9.13 Session end feature

9.13.1 The **FRMCS_SESSION_END_APPLICATION_REQUEST** shall contain the following information: (M)

- On-board identifier of the requesting application <5, application on-board identifier> ;
- Identifier of the session to be ended <9, identifier of a session>.

9.13.2 ASN.1 notation for **FRMCS_SESSION_END_APPLICATION_REQUEST** is: (I)

```
FRMCSSessionEndReq::= SEQUENCE{
  appOBId Uuid,
  sessionId Uuid
}
```

9.13.3 The **FRMCS_SESSION_END_ON-BOARD_FRMCS_ANSWER** shall contain the following information: (M)

- Status of the request <13, request status>.

9.13.4 ASN.1 notation for **FRMCS_SESSION_END_ON-BOARD_FRMCS_ANSWER** is: (I)

```
FRMCSSessionEndAns ::=SEQUENCE{
  reqStatus GenericReqStatus
}
```

## 9.14 Incoming session start feature

9.14.1 The **FRMCS_INCOMING_SESSION_START_ON-BOARD_FRMCS_REQUEST** shall contain the following information: (M)

- Remote address of the initiator of the communication <6, remote address>;
- Category of communication applied to the session <14, communication category>;
- Session identifier of the incoming communication <9, identifier of a session>;
- Local On-Board FRMCS IP address to be used by the On-Board application as destination address for the User Plane data of this session <16, IP address>.

9.14.2 ASN.1 notation for **FRMCS_INCOMING_SESSION_START_ON-BOARD_FRMCS_REQUEST** is: (I)

```
IncomingSessionStartReq ::=SEQUENCE{
  remoteAddress RemoteAddress,
  communicationCategory CommunicationCategory,
  sessionId Uuid,
  localDestFRMCSIPAddress IPAddress
}
```

9.14.3 The **FRMCS_INCOMING_SESSION_START_APPLICATION_ANSWER** shall contain the following information: (M)

- Session identifier of the incoming communication <9, identifier of a session>;
- Session start acceptance decision <15, session acceptance decision>.
- Local Application IP address to be used by the On-Board FRMCS as destination address for the User Plane data in case of successful session establishment <16, IP address >. This parameter is empty if status of the request is rejected;

9.14.4 ASN.1 notation for **FRMCS_INCOMING_SESSION_START_APPLICATION_ANSWER** is: (I)

```
IncomingSessionStartAppAns ::= SEQUENCE{
  sessionId Uuid,
  sessionStartDecision  GenericReqStatus,
  localAppIPAddress IPAddress OPTIONAL
}
```

## 9.15 Incoming session end feature

9.15.1 The **FRMCS_INCOMING_SESSION_END_ON-BOARD_FRMCS_NOTIFICATION** shall contain the following information: (M)

- Session identifier of the incoming communication being terminated <9, identifier of a session>.

9.15.2 ASN.1 notation for **FRMCS_INCOMING_SESSION_END_ON-BOARD_FRMCS_NOTIFICATION** is: (I)

```
IncomingSessionEndNotif ::=SEQUENCE{
```

```
  sessionId Uuid
}
```

## 9.16 Local deregistration feature

9.16.1 The **FRMCS_LOCAL_DEREGISTRATION_APPLICATION_REQUEST** shall contain the following information: (M)

- On-board identifier of the requesting application <5, application on-board identifier>;

9.16.2 ASN.1 notation for **FRMCS_LOCAL_DEREGISTRATION_APPLICATION_REQUEST** is: (I)

```
LocalDeregAppReq::=SEQUENCE{
 appOBId Uuid
}
```

9.16.3 The **FRMCS_LOCAL_DEREGISTRATION_ON-BOARD_FRMCS_ANSWER** shall contain the following information: (M)

- Status of the request <13, request status>.

9.16.4 ASN.1 notation for **FRMCS_LOCAL_DEREGISTRATION_ON-BOARD_FRMCS_ANSWER** is: (I)

```
LocalDeregAppans::=SEQUENCE{
 reqStatus GenericReqStatus
}
```

## 9.17 Event Stream closing feature

9.17.1 Following the FRMCS_LOCAL_DEREGISTRATION_ON-BOARD_FRMCS_ANSWER, the On-Board FRMCS shall send **FRMCS_EVENT_STREAM_CLOSING_ON-BOARD_FRMCS_NOTIFICATION** to the event stream with no content (HTTP Response Status 204). (M)

Note: there is no ASN.1 notation for FRMCS_EVENT_STREAM_CLOSING_ON-BOARD_FRMCS_NOTIFICATION as there is no content.

## 9.18 $OB_{APP}$ API Abnormal Cases

Editor's note: $OB_{APP}$ API abnormal cases must be consolidated. Further analysis in terms of security implications is also required. This is FFS. The following statements are provided for information but will be reviewed in a next version.

Note: in following requirements, a failure is defined as any event that requires the application to restart.

9.18.1 In case of failure of the application, the On-Board FRMCS should expect a new local registration of the application. (I)

9.18.2 After receiving a local registration request of an already registered application, e.g. following a failure, the On-Board FRMCS recovers the dynamic identifier of the

previous registration and return it to the Application without performing another registration. (I)

9.18.3 After receiving a session start request of an application already having open session(s) following a failure, the On-Board FRMCS attempts to recover the session identifier of the previous session(s) and return it to the application. (I)

9.18.4 After receiving a subscription to the Auxiliary function notification of an application already having subscriptions, e.g., following a failure, the On-Board FRMCS returns the list of existing subscriptions (name of the subscription and update period) to the application. (I)

## 9.19 $OB_{APP}$ API Dataflows

9.19.1 Following figure presents an example of API dataflows with the optional Auxiliary function activated for a Tight Coupled application. (I)

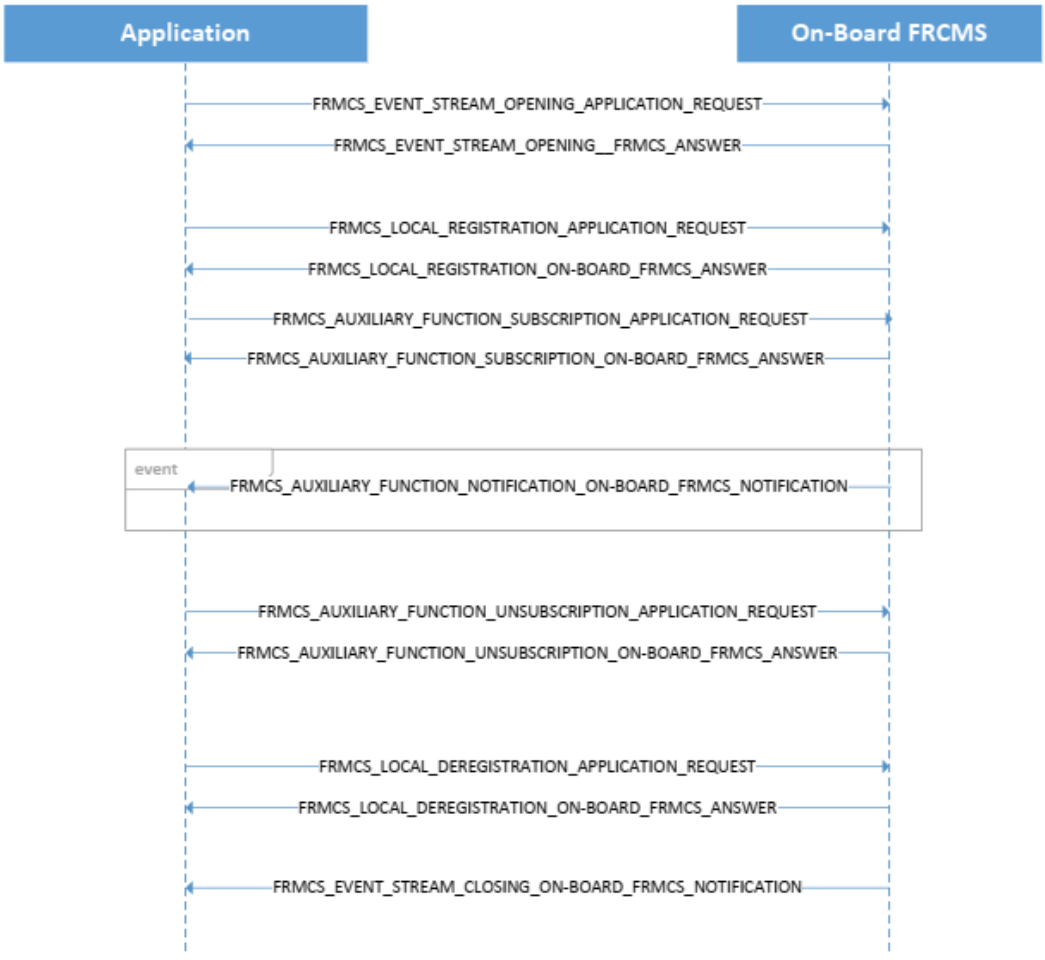


*Figure 7: Dataflow example for Tight coupled application*

9.19.2 Following figure presents an example of API dataflows with the optional Auxiliary function and Session status features activated for a Loose Coupled application. (I)

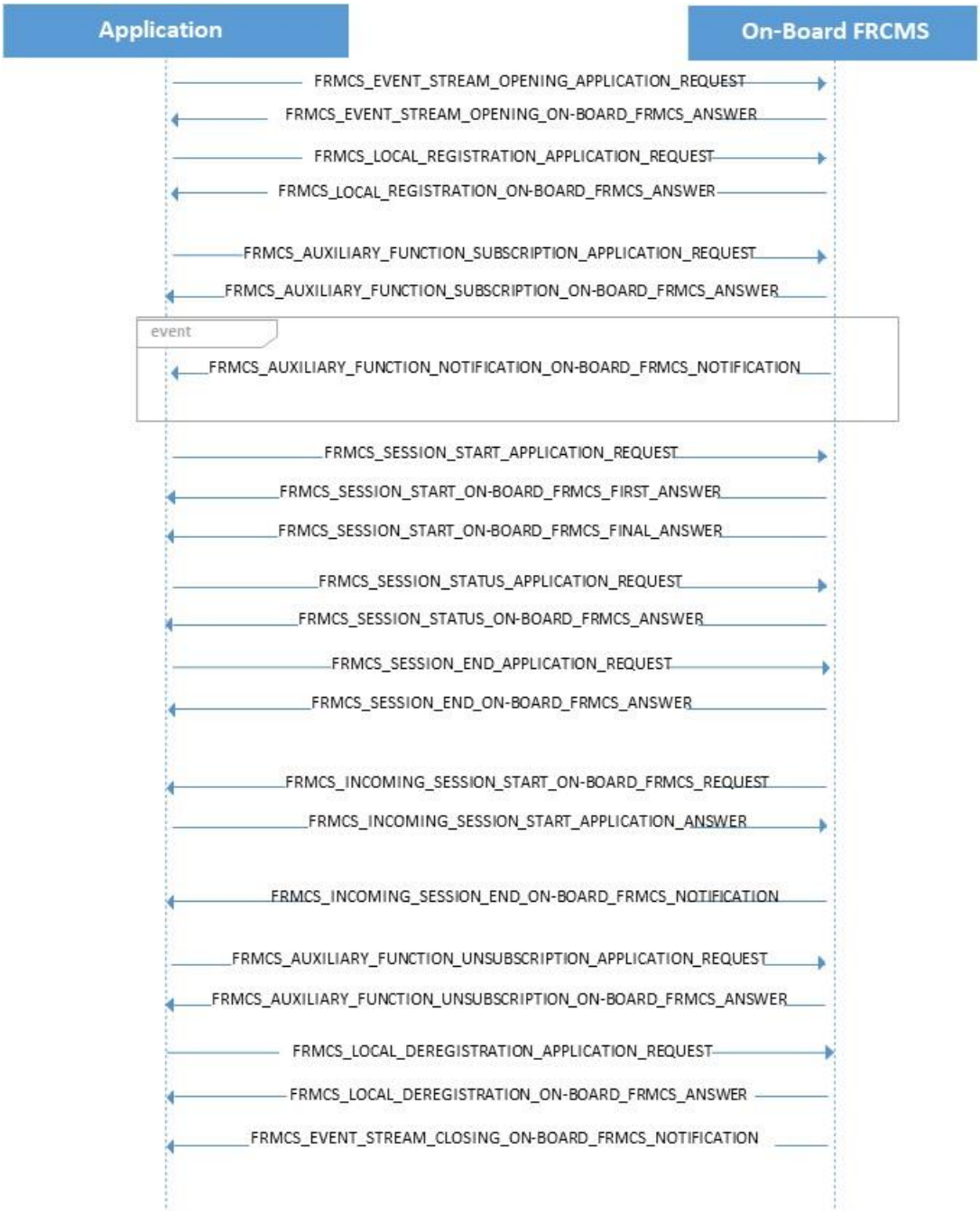


*Figure 8: Dataflow example for a Loose Coupled application.*

## 9.20 3GPP MCX Services at $OB_{APP}$ interface

9.20.1 3GPP MCX functions exchanged through the $OB_{APP}$ interface is based on 3GPP TS 22.280 and TS 23.280 technical specifications. Release 17 is the basis. (I)

9.20.2 The list of 3GPP MCPTT functions exchanged through the $OB_{APP}$ interface is based on 3GPP TS 23.379 and 24.379 Rel. 17, and for MCVideo on 3GPP TS 23.281 and TS24.281 Rel 17 technical specifications. (I)

Note: MCVideo is not expected in the first FRMCS version since there is not clear definition of any function making use of it.

9.20.3 The list of 3GPP MCData functions exchanged through the $OB_{APP}$ interface is based on the 3GPP TS 23.282 and TS 24.282 Rel. 17 technical specifications. (I)

Note: At $OB_{APP}$ interface side, the MCX message flow applicable to the Tight Coupled mode applications is transparent to $OB_{APP}$. Refer to section 7.5. The **[FRMCS-FIS]** defines the end-to-end transaction flows and covers the communication applications based on the use of the 3GPP MCX services. The applicable 3GPP MCX references are listed and maintained in **[FRMCS-SRS]**.

## 9.21 $OB_{APP}$ Communication attributes exchanges (QoS mechanism)

Refer to section 9.4.6

# 10 $TS_{APP}$ Functional Services message and dataflow

## 10.1 Description of $TS_{APP}$ session API features

Editor's note: $TS_{APP}$ session API features requirements are FFS.

## 10.2 $TS_{APP}$ API Abnormal Cases

Editor's note: $TS_{APP}$ API abnormal cases requirements are FFS.

## 10.3 $TS_{APP}$ API Dataflows

Editor's note: $TS_{APP}$ session API Dataflows are FFS.

## 10.4 3GPP MCX Services at $TS_{APP}$ interface

10.4.1 3GPP MCX functions exchanged through the $TS_{APP}$ interface is based on 3GPP TS 22.280 and TS 23.280 technical specifications. Release 17 is the basis. (I)

10.4.2 The list of 3GPP MCPTT functions exchanged through the $TS_{APP}$ interface is based on 3GPP TS 23.379 and 24.379 Rel. 17, and for MCVideo on 3GPP TS 23.281 and TS24.281 Rel 17 technical specifications. (I)

Note: MCVideo is not expected in the first FRMCS version since there is not clear definition of any function making use of it.

10.4.3 The list of 3GPP MCData functions exchanged through the $TS_{APP}$ interface is based on the 3GPP TS 23.282 and TS 24.282 Rel. 17 technical specifications. (I)

Note: At $TS_{APP}$ interface side, the MCX message flow applicable to the Tight Coupled mode applications is transparent to $TS_{APP}$. Refer to section 8.5. The **[FRMCS-FIS]** defines the end-to-end transactions flows and covers the communication applications based on the use of the 3GPP MCX services. The applicable 3GPP MCX references are listed in **[FRMCS-SRS]**.

## 10.5 $TS_{APP}$ Communication attributes exchanges (QoS mechanism)

Editor's note: $TS_{APP}$ Communication attributes exchanges requirements are FFS.

# 11 Annex A: ASN.1 notation of $OB_{APP}$ API parameters and messages

```
--<ASN1.HugeInteger World-Schema.FRMCS.FFFIS>--
World-Schema DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

-------------------------------------------------------------
--                 OBapp PARAMETERS                        --
-------------------------------------------------------------
 ActiveSession::=SEQUENCE{
   sessionId Uuid,
   sessionStatus SessionStatusReqStatus,
   sessionOriginator SessionEstablishmentOrigin,
   communicationCategory CommunicationCategory,
   localDestFRMCSIPAddress IPAddress,
   localAppIPAddress IPAddress,
   remoteAddressList RemoteAddressList
}

 ActiveSessionList ::= SET OF ActiveSession

 ApplicationCategory ::= ENUMERATED {etcs, ato, cabRadio}

 ApplicationStaticId::= UTF8String (SIZE(3..256))

 AuxFunctionNotificationList ::= SET OF AuxiliaryFunctionNotification

 AuxFunctionStat ::= SEQUENCE{
   auxiliaryFunctionCategory AuxiliaryFunctionCategory,
   auxFunctionSubStatus AuxFunctionSubStatus
  }

 AuxFunctionStatList ::= SET OF AuxFunctionStat

  AuxFunctionSubDef::= SEQUENCE{
    auxiliaryFunctionCategory AuxiliaryFunctionCategory,
    auxiliaryFunctionUpdatePeriod AuxiliaryFunctionUpdatePeriod
  }

  AuxFunctionSubList ::= SET OF AuxFunctionSubDef

  AuxFunctionSubStatus ::= ENUMERATED{active, inactive}

  AuxFunctionUnsubStat ::= SEQUENCE{
    auxiliaryFunctionCategory AuxiliaryFunctionCategory,
    auxFunctionUnsubStatus AuxFunctionUnsubStatus
   }

  AuxFunctionUnsubStatList ::= SET OF AuxFunctionUnsubStat
  AuxFunctionUnsubStatus ::= ENUMERATED{rejectedNotSubscribed,
successfullyUnsubscribed, alreadyUnsubscribed}
```

```
AuxFunctionValue ::= CHOICE{
  commStatValue  CommStatValue,
  ffs NULL
}

CommStatValue ::= ENUMERATED{available, notAvailable}

AuxiliaryFunctionCategory ::= ENUMERATED{communicationStatus}

AuxiliaryFunctionCategoryList ::=SET OF AuxiliaryFunctionCategory

-- update period in seconds. 0 = on change event information --
AuxiliaryFunctionUpdatePeriod ::= INTEGER(0..120)

AuxiliaryFunctionValue ::= UTF8String (SIZE(1..1024))

CommunicationCategory::= CHOICE{
  dataComm  DataComm,
  videoComm VideoComm
}

CouplingMode ::= ENUMERATED {tight, loose}

DataComm ::= ENUMERATED{basic, critical}

 GenericReqStatus ::= CHOICE{
  accepted NULL,
  rejected UTF8String (SIZE(0..256))
}

IPAddress::= UTF8String (SIZE(1..40))

LocRegReqStatus ::= CHOICE{
  registered NULL,
  notRegistered UTF8String (SIZE(0..256))
}


OBAppVersion ::=  UTF8String (SIZE(0..5))

OBAppVersionList ::= SET OF OBAppVersion

Recipient ::= SEQUENCE{
  remoteAddress RemoteAddress,
  communicationCategory CommunicationCategory
}

RecipientList ::= SET OF Recipient

RemoteAddress ::= UTF8String (SIZE(3..256))

RemoteAddressList ::= SET OF RemoteAddress
```

```
  SessionEstablishmentOrigin ::= ENUMERATED{localApplication,
remoteApplication}

  SessionIdList ::= SET OF Uuid

  SessionStartReqFinalStatus ::= ENUMERATED{established, rejected}

  SessionStartReqFirstStatus ::= ENUMERATED{inProgress, networkNotReady,
notRegistered, rejected}

  SessionStatusReqStatus ::= ENUMERATED{established, inProgress,
networkNotReady, notRegistered, rejected}

  VideoComm ::= ENUMERATED{basic, critical}

  Uuid ::= UTF8String (SIZE(36))


------------------------------------------------------------
--                   OBapp MESSAGES                       --
------------------------------------------------------------
EventStreamOpeningOBAnswer ::= SEQUENCE{
 appOBId Uuid
}

LocalRegAppReq ::= SEQUENCE{
  appCategory ApplicationCategory,
  staticId ApplicationStaticId,
  appOBId Uuid,
  obAppVersionList OBAppVersionList,
  couplingMode CouplingMode DEFAULT loose
}

LocalRegFRMCSAnswer ::=SEQUENCE{
  reqStatus LocRegReqStatus,
  selectedObAppVer OBAppVersion
}

FRMCSSessionStartAppReq ::= SEQUENCE{
  appOBId Uuid,
  localAppIPAddress IPAddress,
  recipientList RecipientList
}

FRMCSSessionStartFirstAns ::= SEQUENCE{
  reqStatus SessionStartReqFirstStatus,
  sessionId Uuid OPTIONAL
}

FRMCSSessionStartFRMCSFinalAns ::= SEQUENCE{
  reqStatus SessionStartReqFinalStatus,
  sessionId Uuid,
  -- next field present only in case reqStatus is established --
localDestFRMCSIPAddress IPAddress OPTIONAL
```

```
}

FRMCSSessionStatAppReq ::=SEQUENCE{
  appOBId Uuid,
  sessionIdList SessionIdList OPTIONAL
}

FRMCSSessionStatAns ::=SEQUENCE{
 reqStatus GenericReqStatus,
 activeSessionList ActiveSessionList
}

AuxiliaryFunctionSubReq ::= SEQUENCE{
 appOBId Uuid,
 auxFunctionSubList AuxFunctionSubList
}

AuxiliaryFunctionSubAns ::= SEQUENCE{
 reqStatus GenericReqStatus,
 auxFunctionStatList AuxFunctionStatList
}

AuxiliaryFunctionNotification ::= SEQUENCE{
  auxFunctionName AuxiliaryFunctionCategory,
  auxFunctionValue AuxFunctionValue
}

AuxFunctionQueryAppReq ::=SEQUENCE{
  appOBId Uuid,
  auxFunctionNameList AuxiliaryFunctionCategoryList
}

AuxFunctionQueryAns ::= SEQUENCE{
 reqStatus GenericReqStatus,
 auxFunctionNotificationList AuxFunctionNotificationList
}

AuxiliaryFunctionUnsubReq ::= SEQUENCE{
 appOBId Uuid,
 auxFunctionUnsubList AuxiliaryFunctionCategoryList OPTIONAL
}

AuxiliaryFunctionUnsubAns ::= SEQUENCE{
 reqStatus GenericReqStatus,
 auxFunctionUnsubStatList  AuxFunctionUnsubStatList
}

FRMCSSessionEndReq::= SEQUENCE{
  appOBId Uuid,
  sessionId Uuid
}

FRMCSSessionEndAns ::=SEQUENCE{
  reqStatus GenericReqStatus
```

```
}

IncomingSessionStartReq ::=SEQUENCE{
  remoteAddress RemoteAddress,
  communicationCategory CommunicationCategory,
  sessionId Uuid,
  localDestFRMCSIPAddress IPAddress
}

IncomingSessionStartAppAns ::= SEQUENCE{
  sessionId Uuid,
  sessionStartDecision  GenericReqStatus,
  localAppIPAddress IPAddress OPTIONAL
}

IncomingSessionEndNotif ::=SEQUENCE{
  sessionId Uuid
}

LocalDeregAppReq::=SEQUENCE{
 appOBId Uuid
}

LocalDeregAppans::=SEQUENCE{
 reqStatus GenericReqStatus
}

END
```

# 12 Annex B: Interoperability requirements in EU

This annex is the placeholder for identifying the requirements relevant for interoperability in the European Union, i.e. the requirements, with respect to the authorisation in the EU according to the TSI, that are considered in the European Directives to be relevant for interoperability as fulfilling the essential requirements for the Control-Command and Signalling (CCS) subsystem related to safety and technical compatibility which must be met by the rail system, the subsystems, and the interoperability constituents, including interfaces according to the corresponding conditions set out in Directive (EU) 2016/797. It is mandatory that each railway subsystem in the EU meets these requirements on lines under the scope of the Directive and the CCS TSI to ensure technical compatibility between Member States and safe integration between train and track.

At this stage, the version of this specification is not considered complete for the purpose of tendering On-Board FRMCS equipment, and the identification of all requirements relevant for interoperability is for further study.

This annex is therefore only informative.

Printed by

International Union of Railways

16, rue Jean Rey 75015 Paris - France

February 2023

Legal deposit: March 2023

ISBN 978-2-7461-3120-0