**3GPPSA4#120-e S4-220959**

**E-meeting, 17 – 26 August 2022**

|  |
| --- |
| *CR-Form-v12.0* |
| **PSEUDO CHANGE REQUEST** |
|  |
|  | **26**.**857** | **CR** | pseudo | **rev** | **-** | **Current version:** | **0.3.1** |  |
|  |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network |  | Core Network |  |

|  |
| --- |
|  |
| ***Title:***  | **[FS\_5G\_MSE] MSE framework proposal and comparison** |
|  |  |
| ***Source to WG:*** | Qualcomm Incorporated |
| ***Source to TSG:*** |  |
|  |  |
| ***Work item code:*** | FS\_5G\_MSE |  | ***Date:*** | 11/08/2022 |
|  |  |  |  |  |
| ***Category:*** | **B** |  | ***Release:*** | Rel-18  |
|  | *Use one of the following categories:****F*** *(correction)****A*** *(mirror corresponding to a change in an earlier release)****B*** *(addition of feature),* ***C*** *(functional modification of feature)****D*** *(editorial modification)*Detailed explanations of the above categories canbe found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | *Use one of the following releases:Rel-10 (Release 10)Rel-11 (Release 11)Rel-12 (Release 12)**Rel-13 (Release 13)Rel-14 (Release 14)Rel-15 (Release 15)Rel-16 (Release 16)* *Rel-17 (Release 17)* *Rel-18 (Release 18)* |
|  |  |
| ***Reason for change:*** |  |
|  |  |
| ***Summary of change:*** |  |
|  |  |
| ***Consequences if not approved:*** |  |
|  |  |
| ***Clauses affected:*** |  |
|  |  |
|  | **Y** | **N** |  |  |
| ***Other specs*** |  |  |  Other core specifications  | TS/TR ... CR ...  |
| ***affected:*** |  |  |  Test specifications | TS/TR ... CR ...  |
| ***(show related CRs)*** |  |  |  O&M Specifications | TS/TR ... CR ...  |
|  |  |
| ***Other comments:*** | This pCR assumes that S4-220954, 955, 956, 957 and 958 are agreed |
|  |  |
| ***This CR's revision history:*** |  |

**===== CHANGE =====**

## 5.3 MSE framework proposal #2

### 5.3.1 Overview

The proposal follows existing practices in 3GPP for device and network APIs, as introduced in clause 4.2, and extends the documentation with best practices identified in other organizations as introduced in clause 4.3.

### 5.3.2 Architecture

The basic concept of the Media Service Enabler is to support third-party media delivery over the 5G System. Figures 5.3.2-1 and 5.3.2-2 provide an overview for an application that is deployed on top of a 5G System. In this case, the Application Provider is operating an external DN and connects to the 5G System using N6 for data delivery and possibly N33 to use specific 5G network services. A UE-resident application makes use of device functions (for example hardware and software exposed through APIs) and connects to the Application Provider.



Figure 5.3.2-1: Third-party application on top of 5G System – 5G System Architecture



Figure 5.3.2-2: Third-party application on top of 5G System – Interfaces

Figure 5.3.2-3 now extends the above basic architecture to provide to the Application Provider a set of 3GPP-specified functions, possibly both on UE and network side, in order to simplify operations. These functions are bundled as a Media Service Enabler (MSE) and offered to the Application Provider as follows:

- The service may be provisioned on the network side using an MSE Application Function. The provisioning reference point is summarized as MSE-1.

- User plane data may be exchanged with the Application Provider using an Ingest/Egest interface, MSE-2. Generally, this is a generic IP-based interface that directly uses N6 and the UPF. However, the MSE may offer specific Application Server functions at MSE-2.

- On the UE side, the functions of an MSE Client are accessed through a well-defined client API, MSE-6, that is aligned with other device APIs. The MSE Client may make use of other device functions that are expected to be accessible via existing device APIs.

- The MSE Client may be decomposed into C*ore Functions* defined in the relevant Media Service Enabler specification, and *External Device Reference Functions* that are accessed through well-defined APIs MSE-7.

- The MSE Client connects to the 5G network and may make use of Application Functions associated with this Media Service Enabler. Those functions are exposed through MSE-5.

- User data is exchanged with the MSE Application Server (if any) through MSE-4, which may define specific requirements on the usage of protocols, codecs, formats etc.



Figure 5.3.2-2: Addition of MSE to 5G-based media delivery

Providing an Media Service Enabler in this form has several benefits:

- The Application Providerhas a set of functions that can be easily accessed in the same way that device functions are accessed today, namely through well-defined device APIs. The Application Provider can also use regular IP connectivity to operate its application.

- For the MSE developer, the focus is on providing a well-defined set of functions that are exposed to the application through MSE-1 and MSE-2 on the network side, and via MSE-6 on the UE device side.

- The MSE developer may provide the MSE Application Function and Application Server as well as the MSE Client. In this case, the primary interoperability aspects are at reference points MSE-1 and MSE-6.

In another case, the network functions for MSE may be provided by a 5G System operator. In this case the MSE Client and MSE AF are expected to also implement the functions and interoperability defined at reference points MSE-4 and MSE-5.

### 5.3.3 Functions and reference points

The following functions are defined:

- *Application:*

- *MSE Client*: A UE-internal function dedicated to a specific Media Service Enabler. The MSE Client is a logical function and its subfunctions may be distributed within the UE according to implementation choice. For example, it may define new core functions as well as referencing existing functions that are required to complete the expected functions.

- *MSE Application Function*: An Application Function similar to that defined in clause 6.2.10 of TS 23.501 [2], dedicated to a specific Media Service Enabler.

- *MSE Application Server*: An Application Server dedicated to a specific Media Service Enabler.

The following reference points, interfaces and APIs are defined:

- *MSE-1 (MSE Provisioning API):* External API, exposed by the MSE AF, which enables the Application Provider to provision the usage of the MSE.

- *MSE-2: (MSE Ingest/Egest API):* Optional external API exposed to the Application Provider by the MSE AS and used when the MSE AS in the trusted DN is selected to process content for the MSE.

NOTE: MSE-3 may be used for communication between MSE AS and MSE AF, but is not considered relevant.

- *MSE-4: (MSE User Plane interface):* Interface used by an MSE Client to exchange user data with an MSE AS.

- *MSE-5: (MSE Control API):* APIs exposed by an MSE AF to the MSE Client to configure and control MSE functions.

- *MSE-6: (MSE Client APIs):* APIs exposed by the MSE to the Application for client-internal communication to make use of MSE functions

- *MSE-7: (External Device API):* APIs exposed by the UE device to the MSE to make use of resident client functions such as rendering, playback, etc.

- *MSE-8: (Application APIs):* Interface used for information exchange between the Application and the Application Provider.

### 5.3.4 Specification

Media Service Enabler specifications do not attempt to define an entire service, but only a subset of small defined functions. Hence, it is essential to understand that whatever is not defined to complete a service does not need to be documented. An MSE specification is a bottom-up specification: it specifies what is needed and does not address what is not needed.

An MSE specification is proposed to include the following information:

1. *Pre-requisites and Assumptions* (Highly recommended): Pre-requisites document what is expected to be available either from the 5G System (i.e. certain functionalities of the 5G System) or from implementation (for example functions available on the device). These pre-requisites may be considered to be part of the specification (as reference to an external specification), but it is important to identify this separately in order to clearly demarcate the boundaries of the MSE with respect to other functions. Example for pre-requisites include, but are not limited to:

a) Existing and required device functions and the corresponding APIs defined as MSE-7.

b) Existing and required 5G System functions.

2. *Overall specification of the function, including a specific architecture* (Highly recommended). This includes:

a) Instantiations of the MSE reference points and functions.

b) A typical call flow.

3. *Specification of the MSE Client functions and the corresponding MSE-6 APIs* (Highly recommended). This typically includes functionalities such as configuration, settings, notifications, events, data and status query as well as functional methods. It includes:

a) Definition of the internal functions itself.

b) Definition of how to use existing and required device functions.

c) Strict definition of the API methods with details such as name, abstracted code language, functions. As a common language IDL or C is proposed to be used.

4. *Control Plane API and network/MSE Application Function* (Highly recommended)

a) Definition of the internal functions of the AF, using common practices of a RESTful API

b) Alignment with 5G Media Streaming functionalities as defined at reference point M5 of TS 26.501 [?] and TS 26.511 [?], using OpenAPI/YAML.

5. *User plane reference point and network/MSE Application Server* (Optional but recommended)

a) Definition of internal functions of the Application Server, based on common Internet protocols, preferably by reference to external specifications (IETF, MPEG, etc.)

### 5.3.5 Implementation support beyond specification

Beyond the specification, it is proposed to document guidelines and additional support material for developers. The following aspects are considered:

1. *Guidelines for application developers* (Highly Recommended)

- Providing guidance on how an application developer can make use of the Media Service Enabler.

- This is preferably done by providing examples and implementation hints.

2. *Guidelines for MSE implementers* (Optional):

- Providing guidance to an implementer of an MSE Client and/or AF in order to support implementation. Such guidelines may also be provided in line with the specification text.

- If provided, the guidelines are preferably separated in style and form from the main specification text.

3. *Considerations on device API implementations* (Recommended)

- The device APIs MSE-6 and MSE-7 are typically only documented on a conceptual level.

- Considerations on the specifics for implementing the APIs, for example in Android as RESTful APIs in devices, is relevant.

4. *Considerations of a Conformance Test Suite* (Optional, but expected to be at least considered):

- A Conformance Test Suite is a collection of tests covering the breadth of the MSE functions. The tests include the definition of test cases, the definition of test assets as well as the success criteria to pass the tests. A typical figure for a test application to test the implementation of the MSE Client is shown in Figure 5.3.5-1.

- The considerations documented are expected to allow third parties to implement a full Conformance Test Suite in order to test the 3GPP-defined APIs and conformance for correct implementation. Follow-up such as adopter programs may be considered.

- The Conformance Test Suites and adopter program may be provided by external organizations, for example 3GPP market representation partners (MRPs).



Figure 5.3.5-1: Test Framework for MSE Client Implementation

### 5.3.6 Style and documentation guidelines

The primary goal is to achieve consistency across the API, as well as across all specifications. Consistency makes it easier for developers, editors, reviewers, and users of the documentation to understand and modify it. While each organization and specification may and should have its own look and feel, it is considered appropriate to establish a style guide convention. The Style Guide of the OpenXR Documentation has been branched from the Vulkan documentation and is hence considered a broadly adopted and established convention. In addition, 3GPP uses OpenAPI for the API definition towards the network.

Hence, it is proposed to align with the style guide and documentation conventions from OpenXR as well as OpenAPI as follows:

1. Develop APIs for the relevant reference points in a Github- or gitlab-based environment and only port agreements or full specifications to 3GPP specifications. The development of the formal APIs is done in a git-based environment.

2. For device-internal API definitions, align with the OpenXR style guide https://registry.khronos.org/OpenXR/specs/1.0/styleguide.html as follows:

- Use Asciidoc <http://www.asciidoctor.org/> to the extent possible to define formal APIs.

NOTE: References to the Asciidoctor User Manual are to sections in the document at <http://asciidoctor.org/docs/user-manual/>

- For API naming conventions, it is proposed that the rules defined in <https://registry.khronos.org/OpenXR/specs/1.0/styleguide.html#naming> apply with the following adaptation:

- Each MSE is assigned a prefix (for example MSE). This prefix is used as XR is used in the description above.

- Prefixes are used in the API to denote specific semantic meaning of MSE names, or as a label to avoid name clashes, and are explained here:

* + - * MSE/Mse/mse
			* All types, commands, enumerates and C macro definitions in the specification are prefixed with these characters, according to the rules defined above.

- For the markup style, it is proposed that the ETSI/3GPP documentation rules as well as the rules defined in https://registry.khronos.org/OpenXR/specs/1.0/styleguide.html#markup apply. In particular, section 5.7 on writing reference pages is expected to apply: <https://registry.khronos.org/OpenXR/specs/1.0/styleguide.html#writing-refpages>

- Provide reference pages for the MSE according to the OpenXR principle https://registry.khronos.org/OpenXR/specs/1.0/man/html/openxr.html

3. For the network-based APIs and reference points, define RESTful APIs and use the conventional OpenAPI rules as defined by 3GPP in TS 29.501 [29.501].

4. For regular data communication reference to existing protocols and formats.

Editor’s Note: The W3C process is not yet documented and some aspects of the W3C process may also be considered https://www.w3.org/developers/tools/

### 5.3.7 Examples

#### 5.3.7.1 Example 1: MBMS Client

Based on the specification template in clause 5.3.5 and the style guidelines in clause 5.3.6, table 5.3.7-1 provides a potential mapping of the MBMS Client function, as introduced in clause 4.2.1, to the MSE concept.

Table 5.3.7.1-1 Mapping of MBMS Client to MSE concept

|  |  |  |
| --- | --- | --- |
| MSE Specification | Specification | Comments |
| Pre-requisites and assumptions | TS 26.347, clause 6.1: Background | Some high-level aspects are discussed, but a detailed listing of pre-requisites and assumptions is not available. |
| Overall specification of the function including a specific architecture | TS 26.347, clause 5: Reference Client Architecture | A reference architecture is provided, but no high-level call flows. |
| Specification of the MSE Client functions and the corresponding MSE-6 APIs | TS 26.347, clause 6.2, 6.3 and 6.4 for different APIs | Detailed specification of the APIs. However, improved documentation using well-defined types, structures, highlighting and so on could be applied. |
| Control Plane API and network/MSE Application Function | TS 26.346, clause 6 and 9 on user services and referenced in TS 26.347 | The control plane API is not defined explicitly, but as a general protocol. |
| User Plane reference point and network/MSE Application Server | TS 26.346, clause 7, 8, 8A, and 8B and referenced in TS 26.347 | The user plane reference point is defined explicitly, but as a general protocol. |
| Guidelines for application developers | TS 26.347, annex E | Some high-level implementation guidelines are provided. More detailed call flows would be needed. |
| Guidelines for MSE developers | TS 26.347, clause 6.2.2, 6.2.3 and 6.2.4, MBMS Client State Model | A detailed set of basic implementation ideas for the internal handling of an MBMS Client is provided as part of the description. |
| Considerations on device API implementations | TS 26.347, Annex B, Interface Definition Language for MBMS-APIs | A full IDL-based interface definition is provided, but it is informative. It is also not provided as “code”, but as text in the document. |
| Considerations for a Conformance Test Suite | (Not existing) | Nothing is documented on this matter. However, as seen in clause 4.2.1, Android APIs exist |
| Style and documentation  | TS 26.347, annex A | Style and documentation is weak. Annex A introduces the usage of IDL, but is lacking compared to clause 5.3.6:- No git based approach- No usage of ASCIIDOC- No consistent API naming conventions are applied- No markup or reference pages are generated- No OpenAPI-based network protocols are defined. |

In summary, the MBMS Client, as currently specified by 3GPP, quite closely follows the definition of an MSE. Because the MBMS Client and the APIs were developed in stages, the documentation is not consistent in one specification, but rather is spread over several documents. However, most of the considered information is present. An improved overall documentation process, more style guidelines and so on would be needed.

#### 5.3.7.2 Example 2: DASH Player

Based on the specification template in clause 5.3.5 and the style guidelines in clause 5.3.6, table 5.3.7.2-1 provides a potential mapping of the DASH Player function, as introduced in clause 4.2.2, to the MSE concept.

Table 5.3.7.2-1 Mapping of DASH Player to MSE concept

|  |  |  |
| --- | --- | --- |
| MSE Specification | Specification | Comments |
| Pre-requisites and assumptions | TS 26.512, clause 13.2. The Media Playback and Content Decryption Platform external APIs. | Reference to TS 26.511 which in itself expects availability of playback based on CMAF playback requirements.  |
| Overall specification of the function including a specific architecture | TS 26.501, clause 5.4, DASH Streaming and TS 26.512, clause 13.2. | Provides call flows and architecture. |
| Specification of the MSE Client functions and the corresponding MSE-6 APIs | Corresponds to the M7d APIs as defined TS 26.501 and TS 26.512, clause 13.2. | Definition of internal functions, methods, notification and error events, and status information in a formal manner. Reference to dash.js documentation. |
| Control Plane API and network/MSE Application Function | Not defined (unless MPD is considered control plane) |  |
| User plane reference point and network/MSE Application Server | Defined as M4d in TS 26.501 and TS 26.512, clause 10. | References TS 26.247 and ISO/IEC 23009-1. |
| Guidelines for application developers | Examples are provided by reference to dash.js |  |
| Guidelines for MSE developers | Partially provided in TS26.247 and reference to ISO/IEC 23009-1, Annex A |  |
| Considerations on device API implementations | Nothing available |  |
| Considerations of a Conformance Test Suite | Nothing referenced | DASH-IF defines reference player. |
| Style and documentation  | TS 26.347, annex A | Style and documentation is weak. Annex A introduces the usage of IDL, but is lacking compared to clause 5.3.6:- No git based approach- No usage of ASCIIDOC- No consistent API naming conventions are applied- No markup or reference pages are generated- No OpenAPI-based network protocols are defined. |

**===== CHANGE =====**

## 5.4 Discussion on different MSE framework proposals

Two different approaches for an MSE framework are provided in clause 5. The approaches share many similarities, in particular:

* Functional definitions of the Media Service Enabler.
* Definition of device-internal APIs and network interfaces.

However, there are also differences in the configuration and API definition:

* Approach 1, as proposed in clause 5.2, focusses on a workflow-based approach using a workflow document, the MSE Description Document (MDD). The API is restricted in the exchange of this document.
* Approach 2, as proposed in clause 5.3, provides a set of APIs that beyond of the definition of configuration allows additional methods and functions to be supported.

Based on the analysis, it is considered that the approach in clause 5.2 may be a specific instantiation of an MSE API as documented in the second approach, for example in case the API is supported as a microservice in the network using a RESTful network-based interface communication. Hence, it is suggested to follow approach 2, and to define the MSE based on approach 2, but possibly consider approach 1 as one possible instantiation of the API implementation.