ETSI TS 104 014 V1.1.1 (2024-07)

Emergency Communications (EMTEL);
PEMEA File Exchange Extension

Reference
DTS/EMTEL-00074

Keywords
application, emergency

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by ETSI Special Committee Emergency Communications (EMTEL).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

The Pan-European Mobile Emergency Application (PEMEA) architecture provides a framework to enable applications supporting emergency calling functionality to contact emergency services while roaming. PEMEA caters for a range of extension capabilities, including File Exchange which provides a channel to exchange files between the App user and the PSAP. The present document provides a specification for a File Exchange capability for PEMEA.

Introduction

Sending and receiving files is common in most modern communication Apps. These systems allow users to upload and download files in dedicated sessions by using file servers. The present document defines a File Exchange protocol for use in the Pan-European Mobile Emergency Application (PEMEA) framework.

PEMEA File Exchange extension allows PEMEA Apps and PSAPs to exchange files, enabling PSAPs to request photos or videos from the scene that the caller records during the emergency communication. It also allows PSAPs to send files with instructions when it is needed, for example during a multi-case incident where they are receiving a large number of calls.

The specification in the present document does not preclude PEMEA from being used to support and initiate other protocols or implementations.

The present document assumes a working knowledge of PEMEA and familiarity with the PEMEA specification ETSI TS 103 478 [1]. Terms common to the PEMEA specification are not redefined or explained in detail in the present document.

1 Scope

The present document defines the PEMEA File Exchange (PFE) capability, and the need for this functionality. The required entities and actors are identified along with the protocol, specifying message exchanges between entities. The message formats are specified and procedural descriptions of expected behaviours under different conditions are detailed.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 103 478: "Emergency Communications (EMTEL); Pan-European Mobile Emergency Application".

[2] IETF RFC 2617 (June 1999): "HTTP Authentication: Basic and Digest Access Authentication".

[3] IETF RFC 6750 (October 2012): "The OAuth 2.0 Authorization Framework: Bearer Token Usage".

[4] IETF RFC 6838 (January 2013): "Media Type Specifications and Registration Procedures".

[5] IETF RFC 7578 (July 2015): "Returning Values from Forms: multipart/form-data".

[6] IETF RFC 8089 (February 2017): "The "file" URI Scheme".

[7] IETF RFC 9110 (June 2022): "HTTP Semantics".

[8] IETF RFC 9112 (June 2022): "HTTP/1.1".

[9] ISO 8601-1:2019: "Date and time - Representations for information interchange - Part 1: Basic rules".

[10] WHATWG: "HTML Living Standard".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 103 756: "Emergency Communications (EMTEL); PEMEA Instant Message Extension".

[i.2] ETSI TS 103 871: "Emergency Communications (EMTEL); PEMEA Real-Time Text Extension".

[i.3] ETSI TS 103 945: "Emergency Communications (EMTEL); PEMEA Audio Video Extension".

[i.4] IETF RFC 7519 (May 2015): "JSON Web Token (JWT)".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

security: techniques and methods used to ensure:

- **authentication** of entities accessing resources or data;
- **authorization** of authenticated entities prior to accessing or obtaining resources and/or data;
- **privacy** of user data ensuring access only to authenticated and authorized entities;
- **secrecy** of information transferred between two authenticated and authorized entities;
- **trusted** is used as defined in ETSI TS 103 478 [1].

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AESGCM	Advanced Encryption Standard key used with GCM
AP	Application Provider
App	Application
CPE	Customer Premises Equipment
DHE	Diffie-Hellman key Exchange
ECDHE	Elliptic-Curve Diffie-Hellman key Exchange
EDS	Emergency Data Send (message)
GCM	Galois/Counter Mode
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JWT	JSON Web Token
MAC	Message Authentication Code
MIME	Multipurpose Internet Mail Extensions
OAuth	Open Authorization
Pa	PEMEA Application to AP interface
PEMEA	Pan-European Mobile Emergency Application
PFE	PEMEA File Exchange
PIM	PSAP Interface Module
PSAP	Public Safety Answering Point
PSP	PSAP Service Provider
RFC	Request For Comments
RSA	Rivest Shamir Adleman public key encryption algorithm
SSE	Server-Sent Events
TLS	Transport Layer Security

TS	Technical Specification
tPSP	terminating PSP
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
UTF-8	8-bit Unicode Transformation Format
WHATWG	Web Hypertext Application Technology Working Group

4 PEMEA capability extensions

4.1 Overview of extension in PEMEA

PEMEA extension capabilities are defined in ETSI TS 103 478 [1] and are implemented through the use of "reach-back" URIs. The Application Provider (AP) node advertises capabilities as part of the initial forward message through the network, the Emergency Data Send (EDS) message, and the terminating PSAP Service Provider (PSP) or PSAP responds with the subset of capabilities that it supports, thus binding the emergency session between the AP and the terminating PSP or PSAP node.

Specifically, the capabilities are sent as information elements in the apMoreInformation element of the EDS message. The information element and apMoreInformation structures are defined in clauses 10.3.11 and 10.3.12 of ETSI TS 103 478 [1]. An information element in a PEMEA EDS message identifies a capability and each capability is made up of three distinct parts:

- typeOfInfo: what function does the information element serve;
- protocol: the specific semantics for using the function;
- value: the URI through which the service is invoked.

Table 10 in ETSI TS 103 478 [1] identifies an initial set of "typeOfInfo" values used to specify a range of capability extensions for PEMEA. However, beyond the Location_Update and SIP_Request values described in Table 11 of ETSI TS 103 478 [1], protocols are left for further study and definition in subsequent specifications such as the present document. ETSI TS 103 756 [i.1] defines the concrete specification for PEMEA Instant Message protocol, ETSI TS 103 871 [i.2] defines the concrete specification for PEMEA Real-Time Text and ETSI TS 103 945 [i.3] defines the concrete specification for PEMEA Audio Video.

4.2 Service support indication and response

4.2.1 Service definition

The present document provides a concrete definition of the "File_Exchange" typeOfInfo in PEMEA through the present document of a protocol value. The definition in Table 1 shall be considered as an extension to Table 11 in ETSI TS 103 478 [1].

Table 1: Extended AP Information Type Protocol Registry

Info type Value	Protocol Token	Description
File_Exchange	PEMEA	File exchange functionality is supported using the PFE protocol

4.2.2 Service support indication

An AP needing to indicate that the Application it is serving can support file exchange using the PEMEA protocol would include the following information element in the apMoreInformation element of the EDS associated with the emergency session:

```
<information typeOfInfo="File_Exchange" protocol="PEMEA">
    https://ap.example.pemea.help/37agq1cyusbo
</information>
```

4.2.3 Service support response

A terminating node that can support the "File_Exchange" "PEMEA" capability includes this capability in the apMoreInformation element returned to the AP in the onCapSupportPost, as defined in clause 11.1.4 of ETSI TS 103 478 [1], with the value for "File_Exchange" "PEMEA" provided in the example below.

```
<apMoreInformation xmlns="urn:pemea:apps:xml:ns:pemea:base">
    <information typeOfInfo="File_Exchange" protocol="PEMEA"/>
</apMoreInformation>
```

5 Architecture

5.1 Overview

The PEMEA File Exchange (PFE) capability defines a protocol to exchange files between Apps and PSAPs with a PEMEA emergency request. In order to exchange the files, a file server is needed, and the present document defines all the interfaces and procedures that the file server shall provide. This also helps to understand explicitly what is normatively specified in the present document, what is semantic, and what is normatively referred to from the present document but normatively specified in other documents.

The PFE capability was realized with disability usage in mind and the usage model for this necessitates multi-party communications where the Caller and the PSAP Call-Taker represent two parties, but other third-party user may also participate in the emergency session.

Where the procedures described in clause 5 refer to the PSAP Interface Module (PIM) PEMEA node, they may be performed by a terminating PSP or by a PIM depending on architectural decisions in PSAP infrastructure according to ETSI TS 103 478 [1].

References in clause 5 to the App may be made by Apps directly or by an App server acting as a proxy. Whether there are Apps directly connected to the AP or there is an App server connected to the AP is beyond the scope of the present document and depends on the implementation details. The File Exchange Server shall accept requests from any source as long as they are made with the authentication token that was delivered to the AP node as described in clause 0 and following all security procedures described in clause 6.

5.2 Architecture and high-level flows

PEMEA is structured around the AP being the gateway between the App and the PSAP. This model requires communications to occur, first between the App and the AP over the proprietary Pa interface and then between the AP and the associated PSAP service using the protocol mechanisms defined in the specific extension capability document, such as the present document or other PEMEA extension documents, e.g. ETSI TS 103 756 [i.1], ETSI TS 103 871 [i.2], ETSI TS 103 945 [i.3] or any other related one. For most services, this approach is fine, however, exchanging files requires high bandwidth, thus this approach may need to be relaxed somewhat to ensure that the capability delivers the required functionality. Therefore, the present document does not require that all requests to the File Exchange Server are done from the AP node, Apps can make requests to the File Exchange Server directly once they have received the URI and the token from the AP.

The present document does not explicitly define the security measures that shall be taken at the File Exchange Server to detect malicious files, but it is highly recommended that all files uploaded to the File Exchange Server are analysed with malware detection software.

Figure 1 depicts the PFE service architecture. It includes a malware detection software that is not mandatory but highly recommended.

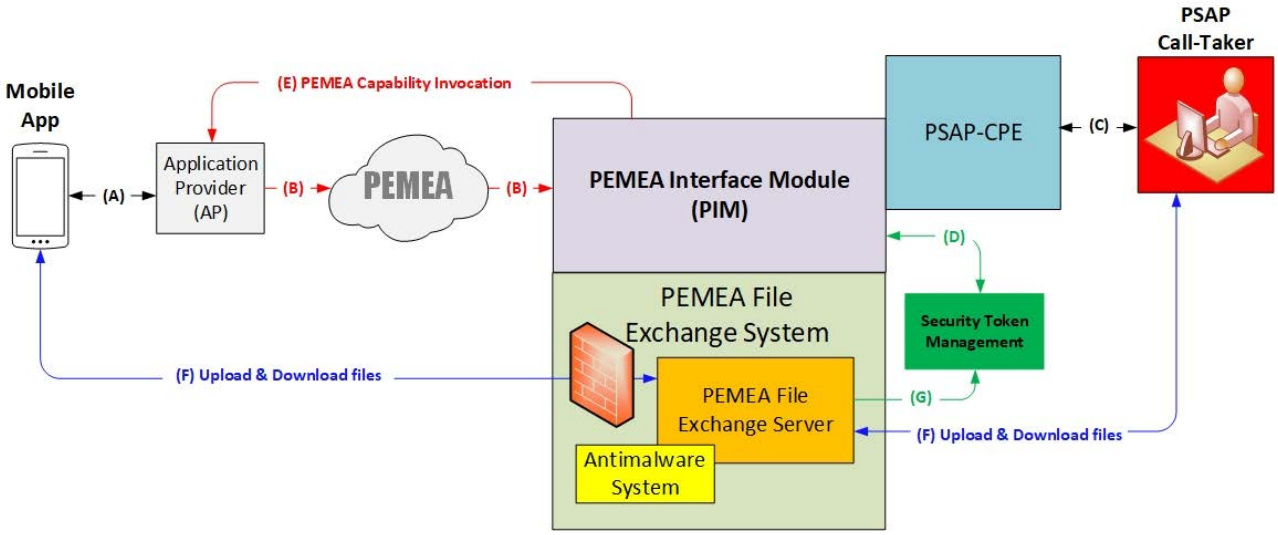

Figure 1: File Exchange architecture for PEMEA

A. Pa interface, the application makes a call to the AP indicating the PFE capability. Invocation information is returned to the App over this interface too.

B. The AP packages the information from the App into an EDS message and sends it into the PEMEA network via the Ps interface. The EDS arrives at the PIM over the Pp interface. The PIM sends an onCapSupportPost message to the AP binding the connection between the AP and the PIM.

C. The PIM notifies the PSAP-CPE which in turn notifies the PSAP Call-Taker. The call is answered and controlled by the PSAP Call-Taker over this interface also. This includes requesting the creation of a PFE session. The PSAP Call-Taker is also able to request connection credentials for additional participants over this interface.

D. On direction from the PSAP Call-Taker the PIM creates a new PFE session and requests Bearer tokens from the Security Token Management system. It shall generate at least a token for the PSAP Call-Taker and a token for the Caller.

E. The PIM invokes the PFE capability in the AP passing the URI for the PFE session as well as the Bearer token required to access it. Similar information is provided to the PSAP Call-Taker's application. The AP then passes the received information down to the App.

F. The App and the PSAP Call-Taker can make requests to the File Exchange Server using the PFE session URI and passing in their respective security tokens. Upload and download of files occurs over this interface. It is highly recommended that files received through this interface are scanned with anti-malware systems to prevent malicious files from being uploaded.

G. The File Exchange Server verifies the Bearer tokens with the Security token management system before accepting requests from the participants.

Whilst the division of the File Exchange Server in these sub-components does not need to be strictly followed, the notion of a signalling component and a media or streaming component are important for traffic path differentiation.

6 Security

6.1 Transport security

The PEMEA File Exchange (PFE) session is identified as an HTTPS URI. The requests should be made using TLS 1.3 but may be made using 1.2 and this shall not support fallback below TLS 1.2. All requests shall authenticate to the File Exchange Server using a Bearer token in the HTTP Authentication header field as defined in IETF RFC 6750 [3]. The token should be provided to the recipient when the service is invoked. The PFE session is expected to remain open while the entity is "online", that is while the EDS session remains active in the PSAP.

The lists for the TLS 1.3 and TLS 1.2 acceptable cipher suites are included in Annex B. These lists are informative and are based on best information at the time of writing. Older cipher suites not included in either of these lists shall not be used.

6.2 Security token usage

The HTTP Authorization header field is defined in IETF RFC 2617 [2] and it specifies that the usage is a scheme followed by a value, where the value may have a structure, as is the case for the digest authentication scheme.

Security token usage in the HTTP Authorization header field was originally specified for use with OAuth and is defined in IETF RFC 6750 [3]. Here the use of the OAuth "Bearer token" is specified so the scheme of the Authorization header field is Bearer, following the scheme a token is placed. The token shall not contain readable information unless it is encoded in base64.

Token usage in the PFE specification follows the Bearer scheme defined in IETF RFC 6750 [3].

Tokens issued by entities in the PFE architecture are expected also to be the validating entities, or to have ties to the validating entities, consequently, whether the tokens are opaque or follow a convention such as JSON Web Token (JWT) IETF RFC 7519 [i.4] is not considered relevant to usage and is not specified further.

IETF RFC 6750 [3] mandates the usage of TLS for use with Bearer tokens, this usage is further defined in clause 6.1 of the present document.

7 Procedures and signalling

7.1 Overview

In order to be able to exchange files using PEMEA File Exchange (PFE) capability the App, AP, PIM and PSAP have to follow the procedures and signalling specified in clause 7 of the present document.

Where the procedures defined in clause 7 refer to the PSAP Interface Module (PIM) PEMEA node, they may be performed by a terminating PSP or by a PIM depending on architectural decisions in PSAP infrastructure according to ETSI TS 103 478 [1].

References in clause 7 to the App may be made by Apps directly or by an App server acting as a proxy. The File Exchange Server shall accept requests from any source as long as they are made with the authentication token that was delivered to the AP node as described in clause 0 and following all security procedures described in clause 6.

When clause 7 refers to participants, it refers not only to the App or the PSAP Call-Taker, but also to other participants that are added to the session as indicated in clause 8.

7.2 Service invocation

7.2.1 Service invocation procedures

Once the PIM has responded to the AP that it can support the PEMEA File Exchange (PFE) service with the procedures defined in clause 4.2, then the AP shall be capable of accepting a service invocation on the provided URI at any time. The AP shall only accept a PFE service invocation from the PIM that sent the onCapSupportPost message.

Figure 2 provides the sequence diagram illustrating the flow of events between entities for the invocation procedure.

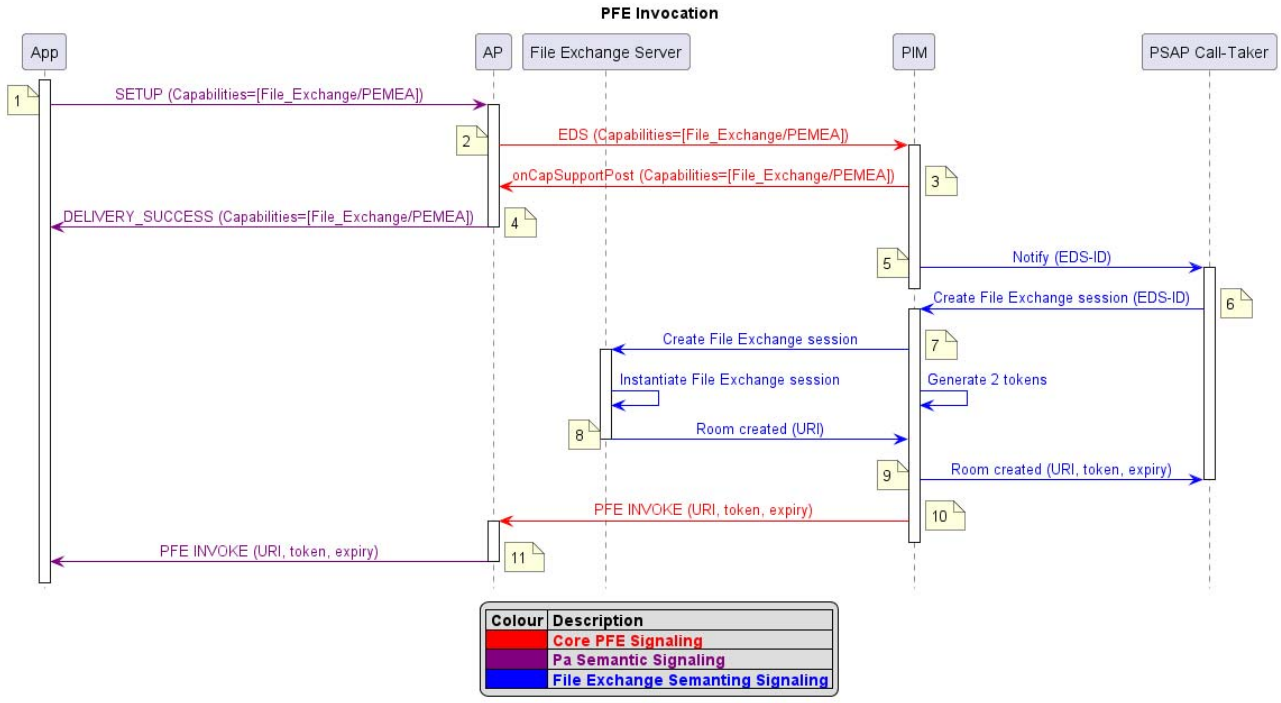

Figure 2: PFE Invocation sequence diagram

1) The App initiates an emergency session with the AP over the Pa interface indicating that it can support the PFE capability.

2) The AP creates an EDS message from the data provided by the App and includes the PFE capability. The AP sends the EDS into the PEMEA network.

3) The EDS arrives at the PIM. The PIM supports the PFE capability and includes this option in the onCapSupportPost back to the AP.

4) The AP binds the emergency session to the PIM that sent the onCapSupportPost message and then signals to the App over the Pa interface the PSAP can support the PFE functionality.

5) The PIM notifies the PSAP Call-Taker that a new EDS has arrived.

6) The PSAP Call-Taker requests the PIM to initiate the creation of a File Exchange session.

7) The PIM requests that the File Exchange Server to create a File Exchange session. The session and 2 tokens are created.

8) The File Exchange Server returns the File Exchange session URI to the PIM.

9) The PIM returns the File Exchange session URI, one token and its expiry time to the PSAP Call-Taker.

10) The PIM invokes the PFE capability in the AP using the provided reach-back URI from the EDS. The PIM includes the File Exchange session URI, token and expiry time in the body of the HTTP POST to the reach-back URI.

11) The AP signals to the App over the Pa interface that the PSAP has invoked the PFE capability and provides the File Exchange session URI, token and expiry.

Once the PFE service is invoked, the operations specified in clause 7.3 can be performed using the PFE session URI and the security token.

7.2.2 Service invocation object

The PIM invokes the PEMEA File Exchange (PFE) service in the AP by posting to the URI provided in the File_Exchange information element included in the apMoreInformation contained in the EDS. The POST message includes a body containing a JSON object. The JSON object provides the File_Exchange session URI as well as a security token and corresponding expiry time.

The JSON schema for the PFE service invocation message is provided in Annex A.

Table 2: PFE service invocation message

Property	Type	Description
url	String	The URI of the File Exchange session.
token	String	A security token used to authenticate the App to the File Exchange session. The App shall include the token in the HTTP Authorization header using the Bearer token scheme. The App shall use the token in all requests to the File_Exchange session for the duration of the App emergency session.
expiry	Integer	Specifies the expiry time of the security token. It is an integer specifying the number of second since UTC epoch, 00:00:00 1st of January 1970.

Invocation example:

```
{
    "url":"https://file_exchange_server.example.com/session/534wafds21s21fdf",
    "token":"PPtzs5zzG5Pkf61KPz51",
    "expiry":1574092280231
}
```

7.2.3 File Exchange session creation and deletion

The File Exchange session is created by the File Exchange Server under direction of the PSAP Call-Taker via the PIM as defined in clause 7.2.

Once the File Exchange session is created it remains active as long as the PIM maintains a context for the EDS.

7.3 File Exchange operations

7.3.1 Overview

The File Exchange session provides a RESTful interface to perform the operations defined in clause 7.3.

All the operations shall be authenticated as defined in clause 6.

7.3.2 List files

7.3.2.1 Description

Participants can list the available files in the File Exchange session. This allows to check what files had been uploaded to the File Exchange session.

To list the available files the participants shall make an HTTP GET request to the URI of the File Exchange session.

The HTTP request shall have the following headers:

- Accept: The value shall be "application/json".

- Authorization: The token to access the File Exchange session as defined in clause 6.2. This is obtained as defined in clause 7.2.

The HTTP request body shall be empty.

The HTTP response shall have the following header:

- Content-Type: The value shall be "application/json; charset=utf-8".

The HTTP response shall have a 200 OK status code.

The HTTP response body shall contain the list of files represented as a JSON array of FileMetadata elements which structure is defined in clause 10.2.1.

Figure 3 provides the sequence diagram illustrating the flow of events between entities for the list files operation.

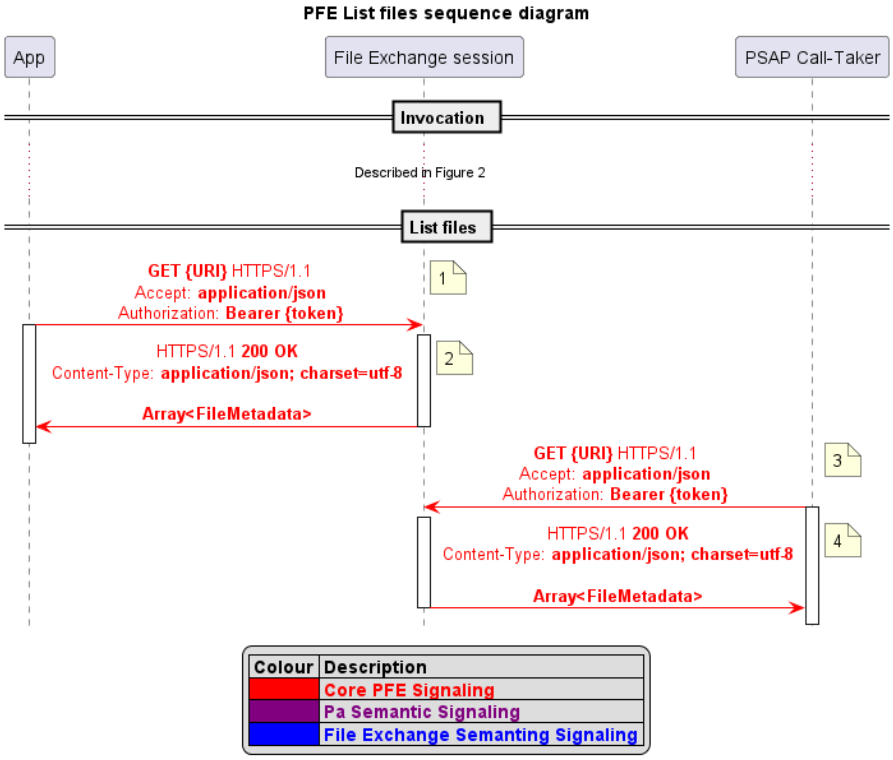

Figure 3: PFE List files sequence diagram

1) To obtain the list of files of the File Exchange session the App makes an HTTP GET request to the File Exchange session URI. The request contains the Accept header with value "application/json" and the Authorization header with the "Bearer token" as defined in clause 6.2.

2) The File Exchange Server validates that the token is not expired and is valid for the requested File Exchange session. The file Exchange Server answers the HTTP GET request from the App with a 200 OK status code, with the Content-Type header with value "application/json; charset=utf-8" and with the body containing the list of files in the File Exchange session represented as a JSON array of FileMetadata elements.

3) To obtain the list of files of the File Exchange session the PSAP Call-Taker makes an HTTP GET request to the File Exchange session URI. The request contains the Accept header with value "application/json" and the Authorization header with the "Bearer token" as defined in clause 6.2.

4) The File Exchange Server validates that the token is not expired and is valid for the requested File Exchange session. The file Exchange Server answers the HTTP GET request from the PSAP Call-Taker with a 200 OK status code, with the Content-Type header with value "application/json; charset=utf-8" and with the body containing the list of files in the File Exchange session represented as a JSON array of FileMetadata elements.

7.3.2.2 Example

HTTP request:

```
GET /session/534wafds21s21fdf HTTPS/1.1
Host: file_exchange_server.example.com
Accept: application/json
Authorization: Bearer PPtzs5zzG5Pkf61KPz51
```

HTTP response:

```
HTTPS/1.1 200 OK
Content-Type: application/json; charset=utf-8

[
    {
        "name":"example.png",
        "size":328049,
        "timestamp":1574092280231,
        "type":"image/png",
        "url":"https://file_exchange_server.example.com/session/534wafds21s21fdf/example.png"
    }
]
```

7.3.3 Upload file

7.3.3.1 Description

Participants can upload a file to the File Exchange session. This operation adds a new file to the files of the File Exchange session.

To upload a file the participants shall make an HTTP PUT request to the URI of the File Exchange session.

The HTTP request shall have the following headers:

- Accept: The value shall be "application/json".
- Authorization: The token to access the File Exchange session as defined in clause 6.2. This is obtained as defined in clause 7.2.
- Content-Type: The value shall be multipart/form-data with the boundary that delimits the end of the file as defined in clause 4 of IETF RFC 7578 [5].

The HTTP request body shall be a multipart/form-data body as defined in IETF RFC 7578 [5] with the following considerations:

- It shall have only 1 part that starts and ends with a boundary according to IETF RFC 7578 [5].
- The part shall have the Content-Disposition header field. The value shall have the following format: "form-data; name="attachment"; filename="example.png"". The name of the field shall be "attachment" and the file name shall be placed in the filename field like "example.png" is in the example.
- The part shall have the Content-Type header field. The value shall be a valid MIME type as defined in clause 4.5 of IETF RFC 6838 [4].

If the file is too big to fit in one single HTTP request, the file shall be split in multiple HTTP requests with the same structure and the limit of the file shall be delimited by the boundary property as defined in clause 4.1 of IETF RFC 7578 [5].

If the File Exchange session receives multiple files with the same filename, the files shall be named by the File Exchange session according to the following rule:

- If there is a file in the File Exchange session with the same filename provided in the upload file operation, the filename for the new file shall have "(1)" appended to it.
- Each time that a file is uploaded with the same filename, the number shall be increased by 1. That is, the second file shall have "(2)" appended to it, the third file shall have "(3)" appended to it, etc.

The HTTP response shall have the following headers:

- Content-Type: The value shall be "application/json; charset=utf-8".
- Location: The value shall be the URI to download the uploaded file as defined in clause 10.2.2 of IETF RFC 9110 [7].

The HTTP response shall have a 201 Created status code.

The HTTP response body shall contain the file represented as a FileMetadata element which structure is defined in clause 10.2.1.

Figure 4 provides the sequence diagram illustrating the flow of events between entities for the upload file operation.

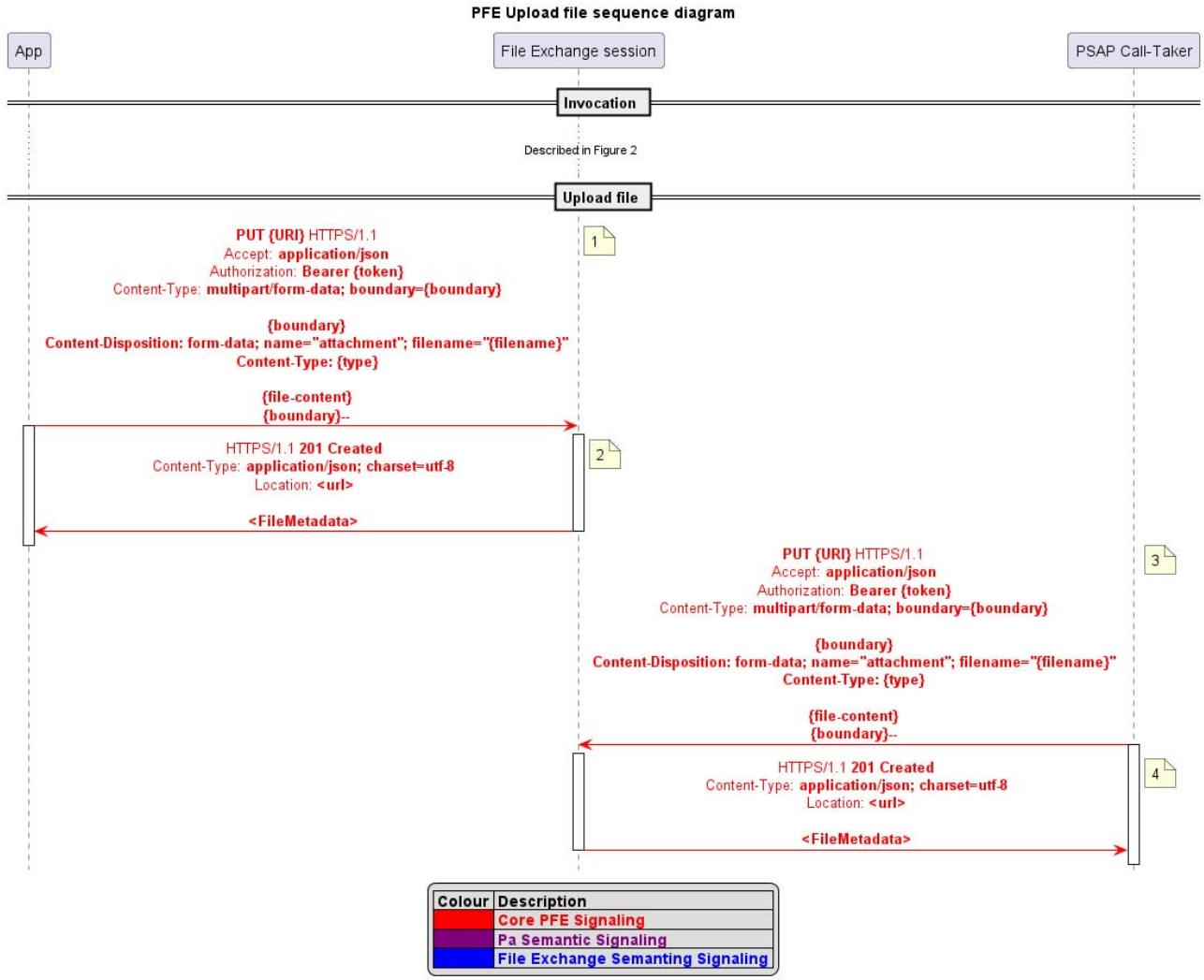

Figure 4: PFE Upload file sequence diagram

1) To upload a file to the File Exchange session the App makes an HTTP PUT request to the File Exchange session URI. The request contains the Accept header with value "application/json", the Authorization header with the "Bearer token" as defined in clause 6.2 and the Content-Type header with value "multipart/form-data" with the boundary that delimits the end of the file as defined in IETF RFC 7578 [5].

2) The File Exchange Server validates that the token is not expired and is valid for the requested File Exchange session. The file Exchange Server answers the HTTP PUT request from the App with a 201 Created status code, with the Content-Type header with value "application/json; charset=utf-8" and with the body containing the uploaded file represented as a FileMetadata data structure.

3) To upload a file to the File Exchange session the PSAP Call-Taker makes an HTTP PUT request to the File Exchange session URI. The request contains the Accept header with value "application/json", the Authorization header with the "Bearer token" as defined in clause 6.2 and the Content-Type header with value "multipart/form-data" with the boundary that delimits the end of the file as defined in IETF RFC 7578 [5].

4) The File Exchange Server validates that the token is not expired and is valid for the requested File Exchange session. The file Exchange Server answers the HTTP PUT request from the PSAP Call-Taker with a 201 Created status code, with the Content-Type header with value "application/json; charset=utf-8" and with the body containing the uploaded file represented as a FileMetadata data structure.

7.3.3.2 Example

HTTP request:

```
PUT /session/534wafds21s21fdf HTTPS/1.1
Host: file_exchange_server.example.com
Accept: application/json
Authorization: Bearer PPtzs5zzG5Pkf61KPz51
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryDRKRUlAriyYJ28VG

------WebKitFormBoundaryDRKRUlAriyYJ28VG
Content-Disposition: form-data; name="attachment"; filename="example.png"
Content-Type: image/png

<binary>
------WebKitFormBoundaryDRKRUlAriyYJ28VG--
```

HTTP response:

```
HTTPS/1.1 201 Created
Content-Type: application/json; charset=utf-8
Location: https://file_exchange_server.example.com/session/534wafds21s21fdf/example%20(1).png

{
    "name":"example (1).png",
    "size":328049,
    "type":"image/png",
    "timestamp":1715858118715,
    "url":"https://file_exchange_server.example.com/session/534wafds21s21fdf/example%20(1).png"
}
```

7.3.4 Download file

7.3.4.1 Description

Participants can download a file from the File Exchange session.

To download a file the participants shall make an HTTP GET request to the URL of the file they want to download. The URL is obtained using the list files or the upload file operations defined in clauses 7.3.2 and 7.3.3.

The HTTP request shall have the following header:

- Authorization: The token to access the File Exchange session as defined in clause 6.2. This is obtained as defined in clause 7.2.

The HTTP request body shall be empty.

The HTTP response shall have the following headers:

- Content-Disposition: The value shall have the following format: "attachment; filename="example (1).png"" where example (1).png is the name of the file that is being downloaded.

- Content-Type: The value shall be a valid MIME type as defined in clause 4.5 of IETF RFC 6838 [4].

The HTTP response shall have a 200 OK status code.

The HTTP response body shall contain the downloaded file. If the file is too big to fit in one single HTTP response, the file shall be chunked as defined in IETF RFC 9110 [7] and IETF RFC 9112 [8].

Figure 5 provides the sequence diagram illustrating the flow of events between entities for the download file operation.

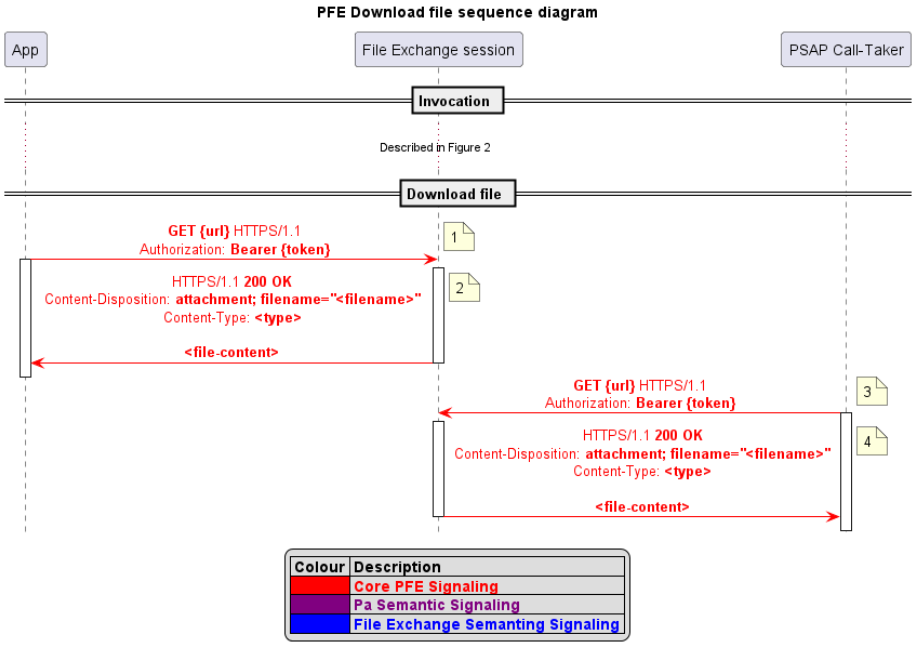

Figure 5: PFE Download file sequence diagram

1) To download a file from the File Exchange session the App makes an HTTP GET request to the URL of the file. The request contains the Authorization header with the "Bearer token" as defined in clause 6.2.

2) The File Exchange Server validates that the token is not expired and is valid for the requested File Exchange session. The file Exchange Server answers the HTTP GET request from the App with a 200 OK status code, with the Content-disposition header with value "attachment; filename="<filename>"" and the header Content-Type with the MIME type of the file.

3) To download a file from the File Exchange session the PSAP Call-Taker makes an HTTP GET request to the URL of the file. The request contains the Authorization header with the "Bearer token" as defined in clause 6.2.

4) The File Exchange Server validates that the token is not expired and is valid for the requested File Exchange session. The file Exchange Server answers the HTTP GET request from the PSAP Call-Taker with a 200 OK status code, with the Content-disposition header with value "attachment; filename="<filename>"" and the header Content-Type with the MIME type of the file.

7.3.4.2 Example

HTTP request:

```
GET /session/534wafds21s21fdf/example%20(1).png HTTPS/1.1
Host: file_exchange_server.example.com
Authorization: Bearer PPtzs5zzG5Pkf61KPz51
```

HTTP response:

```
HTTPS/1.1 200 OK
Content-Disposition: attachment; filename="captura-de-pantalla-2024-03-11-143604 (1).png"
Content-Type: image/png

<binary>
```

7.4 Notifications channel

7.4.1 Overview

The operations defined in clause 7.3 allow users to upload files, download files and get the list of uploaded files. With these operations a participant can only know when another participant has uploaded a new file by polling with the list files operation, i.e. by making periodic HTTP requests at a certain time interval.

This polling implies many extra requests during the emergency session. This is not efficient and could saturate the File Exchange Server if the requests are done with a high frequency or if there are many parallel sessions. To solve this issue a notifications channel that allows participants to be notified of new uploaded files is required.

There are many subscription mechanisms available, but as PEMEA can be used by many different types of applications, each with its own architecture and needs, Server-Sent Events (SSE) is the selected mechanism to be used in the present document. As specified in clause 9.2 of HTML Living Standard [10], the Server-Sent Events mechanism allows for efficient real-time communication between servers and clients.

7.4.2 Subscribe to the File Exchange session

7.4.2.1 Description

Participants can subscribe to events produced in a File Exchange session by opening a Server-Sent Event (SSE) connection to the File Exchange session URI as defined in HTML Living Standard [10].

To open the SSE connection a participant shall make an HTTP GET request to the URI of the File Exchange session.

The HTTP request shall have the following headers:

- Accept: The value shall be "text/event-stream".
- Authorization: The token to access the File Exchange session as defined in clause 6.2. This is obtained as defined in clause 7.2.

The HTTP request body shall be empty.

The HTTP response shall have the following header:

- Content-Type: The value shall be "text/event-stream".

The HTTP response shall have a 200 OK status code.

The HTTP connection shall remain opened as described in HTML Living Standard 10.

In order to avoid the connection to be closed by proxies, the ping pong mechanism of SSE shall be used, this is done by sending a line starting with the ":" character. This message shall be sent periodically from the file sharing session to the connected participants every 60 seconds.

Figure 6 provides the sequence diagram illustrating the flow of events between entities for the download file operation.

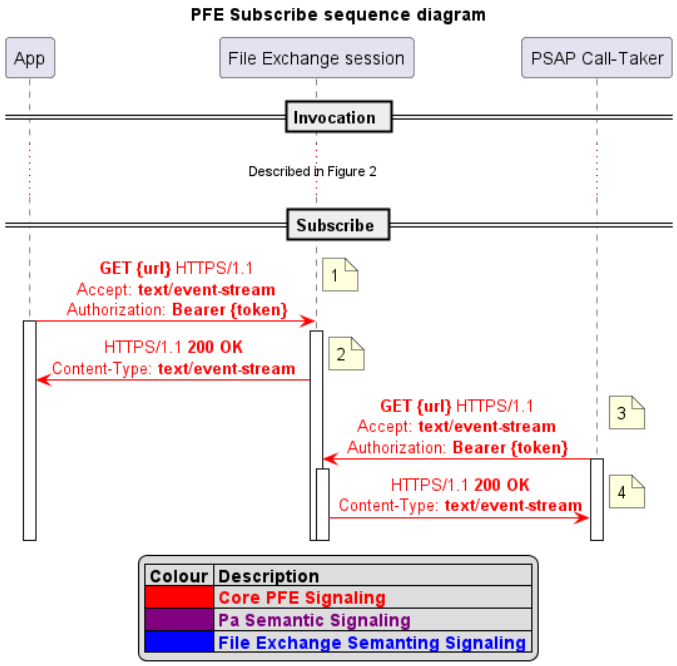

Figure 6: PFE Subscribe sequence diagram

1) To subscribe to the File Exchange session events the App makes an HTTP GET request to the File Exchange session URI. The request contains the Accept header with value "text/event-stream" and the Authorization header with the "Bearer token" as defined in clause 6.2.

2) The File Exchange Server validates that the token is not expired and is valid for the requested File Exchange session. The File Exchange Server answers the HTTP GET request from the App with a 200 OK status code, with the Content-Type header with value "text/event-stream" which indicates that the SSE connection has been successfully established. The File Exchange Server shall maintain the connection opened while the File Exchange session is open, and shall send events through the SSE as defined in clause 7.4.3.

3) To subscribe to the File Exchange session events the PSAP Call-Taker makes an HTTP GET request to the File Exchange session URI. The request contains the Accept header with value "text/event-stream" and the Authorization header with the "Bearer token" as defined in clause 6.2.

4) The File Exchange Server validates that the token is not expired and is valid for the requested File Exchange session. The File Exchange Server answers the HTTP GET request from the PSAP Call-Taker with a 200 OK status code, with the Content-Type header with value "text/event-stream" which indicates that the SSE connection has been successfully established. The File Exchange Server shall maintain the connection opened while the File Exchange session is open, and shall send events through the SSE as defined in clause 7.4.3.

7.4.2.2 Example

HTTP request:

```
GET /session/534wafds21s21fdf HTTPS/1.1
Host: file_exchange_server.example.com
Accept: text/event-stream
Authorization: Bearer PPtzs5zzG5Pkf61KPz51
```

HTTP response:

```
HTTPS/1.1 200 OK
Content-Type: text/event-stream
```

7.4.3 Receive notification events

7.4.3.1 Overview

Once the SSE connection is established as defined in clause 7.4.2 the File Exchange Server can send messages over the opened HTTP connection as it is defined in clause 9.2 of HTML Living Standard [10].

The data transmitted shall be encoded as UTF-8.

The data shall be separated by lines, and in each line an event may be sent. The events shall be transmitted with the following format:

```
data:{event1}

data:{event2}
```

Where {event1} and {event2} are 2 consecutive events. The present document defines an event to notify when new files are successfully uploaded in clause 7.4.3.2 and another event to notify when the File Exchange session has been closed in clause 7.4.3.3.

7.4.3.2 File uploaded to the File Exchange session

7.4.3.2.1 Description

Whenever a new file is successfully uploaded to a File Exchange session the File Exchange Server shall send a notification to all participants subscribed to the File Exchange session. It is done by sending a FILE_UPLOADED event which is a JSON object which structure is defined in clause 10.3.

Figure 7 provides the sequence diagram illustrating the flow of events between entities to receive notifications from the File Exchange session.

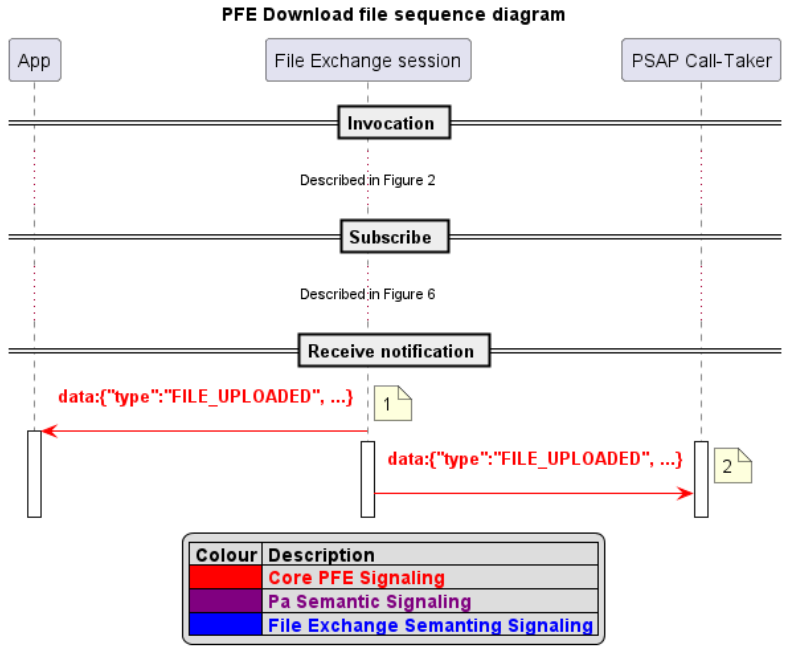

Figure 7: PFE Download file sequence diagram

1) After establishing the SSE channel subscription as described in clause 7.4.2 the File Exchange Server will send to the App the events produced in the File Exchange session. When a new file is successfully uploaded, the File Exchange Server will send a "FILE_UPLOADED" event to the App through the SSE.

2) After establishing the SSE channel subscription as described in clause 7.4.2 the File Exchange Server will send to the PSAP Call-Taker the events produced in the File Exchange session. When a new file is successfully uploaded, the File Exchange Server will send a "FILE_UPLOADED" event to the PSAP Call-Taker through the SSE.

7.4.3.2.2 Example

Subscription is done as defined in clause 7.4.2, so the HTTP request would be opened and only the SSE messages would be sent, but for clarification the HTTP request is written again in this example.

HTTP request:

```
GET /session/534wafds21s21fdf HTTPS/1.1
Host: file_exchange_server.example.com
Accept: text/event-stream
Authorization: Bearer PPtzs5zzG5Pkf61KPz51
```

HTTP response:

```
HTTPS/1.1 200 OK
Content-Type: text/event-stream

data:{"type":"FILE_UPLOADED","file":{"name":"example%20(1).png","size":328049,"type":"image/png","ti
mestamp":1715858118715,"url":"https://file_exchange_server.example.com/session/534wafds21s21fdf/exam
ple%20(1).png"}}

data:{"type":"FILE_UPLOADED","file":{"name":"example.png","size":328049,"type":"image/png","timestam
p":1574092280231,"url":"https://file_exchange_server.example.com/session/534wafds21s21fdf/example.pn
g"}}
```

7.4.3.3 File Exchange session closed by the File Exchange Server

7.4.3.3.1 Description

Whenever a File Exchange session is closed as described in clause 9 the File Exchange Server shall send a notification to all participants subscribed to the File Exchange session before closing all the subscriptions. It is done by sending a FOLDER_CLOSED event which is a JSON object which structure is defined in clause 10.3.

7.4.3.3.2 Example

Subscription is done as defined in clause 7.4.2, so the HTTP request would be opened and only the SSE messages would be sent, but for clarification the HTTP request is written again in this example.

HTTP request:

```
GET /session/534wafds21s21fdf HTTPS/1.1
Host: file_exchange_server.example.com
Accept: text/event-stream
Authorization: Bearer PPtzs5zzG5Pkf61KPz51
```

HTTP response:

```
HTTPS/1.1 200 OK
Content-Type: text/event-stream

data:{"type":"FOLDER_CLOSED","reason":"Folder was closed"}
```

7.4.4 Unsubscribe from the File Exchange session

To end the subscription the HTTP connection that established the SSE channel shall be closed. This can be done either by the participants or by the File Exchange Server.

The File Exchange Server shall only close the SSE connections when a File Exchange session is closed as described in clause 9 and the File Exchange Server shall send a File Exchange session closed notification before closing the connections as defined in clause 7.4.3.3.

7.5 Errors

7.5.1 Description

The File Exchange Server may encounter various situations where a request results in an error, these errors may occur because the schemas and operations defined in the present document are not followed or by circumstances that are out of control of the File Exchange Server. When the File Exchange Server shall return an error to a received request, the response shall be an HTTP answer with JSON formatted body with the structure defined in clause 7.5.

7.5.2 Example

HTTP request:

```
GET /session/534wafds21s21fdf/unknown.png HTTPS/1.1
Host: file_exchange_server.example.com
Authorization: PPtzs5zzG5Pkf61KPz51
```

HTTP response:

```
HTTPS/1.1 404 Not Found
Content-Type: application/json; charset=utf-8

{
    "error":"Not Found",
    "message":"File Not Found",
    "path":"/session/534wafds21s21fdf/unknown.png",
    "status":404,
    "timestamp":"2024-05-13T13:30:37.418Z"
}
```

8 Add participants to the File Exchange session

The ability to add participants into the emergency session is important as it brings additional expertise to the emergency situation that is able to assist in providing a successful outcome. The general concept for linking in third-parties is that the PSAP Call-Taker instructs the PIM to obtain additional security access tokens for the File Exchange session and that the PSAP Call-Taker provides these and the File Exchange session URI, to the relevant third-parties. The subsequent operations defined in clause 7.3 are clearly normatively defined in the present document.

Acquisition and dissemination of the File Exchange session URI and additional security access tokens may vary widely in implementation based on the type of third-party and their Call-Taker equipment and whether policy, such as language or disability, is also applied at the third-party receiver to link in the correct personnel. Consequently, the present document does not go into further details on how this occurs, only that the system shall support the ability of the PSAP Call-Taker to request the additional tokens and provide a facility for the dissemination of this contact information to the relevant third-parties.

9 File Exchange session closure

The File Exchange session can be closed though the tPSP/PIM by the PSAP-CPE. The present document does not instruct when the tPSP/PIM should close the File Exchange session and this decision shall be made by the PSAP-CPE.

In most of the cases it can be done when the PSAP Call-Taker considers the emergency session to be terminated or when the App instructs the AP that the Caller wants to terminate the call, which will result in the AP invalidating the reach-back URI as defined in clause 14.1.3 of ETSI TS 103 478 [1], but with the confirmation of the PSAP-CPE.

When the File Exchange session is closed by the tPSP/PIM under instruction of the PSAP-CPE, the URI to access the File Exchange session shall be invalidated and the File Exchange Server shall not accept more requests to the File Exchange session URI. Files uploaded to the File Exchange session can be stored for later analysis.

10 PEMEA File Exchange type definitions

10.1 Overview

The PEMEA File Exchange (PFE) capability use JSON format for File Exchange Server answers to the different operations defined in the present document.

The JSON specifications for the responses are provided in Annex A, they are also maintained in a repository outside of the present document and are available for download from ETSI Forge:

- https://forge.etsi.org/rep/emtel/ts-104-014/-/tree/1.1.1.

The subsequent clause 10.2 of the present document defines each of the PFE data types. Operations and procedures are specified in clause 7.3.

10.2 Data types

10.2.1 FileMetadata

Defines a file in a File Exchange session. It is returned from the File Exchange server when making the list file operation defined in clause 7.3.2.

Table 3: FileMetadata properties

Property	Type	Description
name	String	The name of the file.
size	Number	The size of the file in Bytes.
type	String	The media type of the file. A valid media type shall be defined in MIME types.
url	String	The URL to download the file. A valid IETF RFC 8089 [6] URI.
timestamp	Number	The time at which the file is uploaded. A valid UTC time in milliseconds since January 1, 1970.

10.2.2 EventType

The EventType enumerable defines the "type" of events being sent over the SSE channel.

Table 4: PEMEA File Exchange event type values

Property	Description
FILE_UPLOADED	Event sent from the File Exchange session when a file is successfully uploaded.
FOLDER_CLOSED	Event sent from the File Exchange session when the File Exchange session is terminated.

10.2.3 Error

Defines an error in a File Exchange session. It is returned from the File Exchange server when there is an error with one of the operations defined in clause 7.3.2.

Table 5: Error properties

Property	Type	Description
status	Number	The HTTP status code as defined in clause 9.3 of IETF RFC 9110 [7].
error	String	The name of the HTTP status code as defined in clause 9.3 of IETF RFC 9110 [7].
message	String	The reason why the error happened.
path	String	The path that handled the request.
timestamp	String	The time at which the error is sent. A date in the simplified format according to ISO 8601-1 [9].

10.3 File uploaded event

The FILE_UPLOAD event is the event sent from the File Exchange Server to the participants that are subscribed to a File Exchange session via the SSE channel when a new file is successfully uploaded to the File Exchange session as defined in clause 7.3.3.

Table 6: File uploaded event

Property	Type	Description
type	String	Type of the event. The value is "FILE_UPLOADED". Refer to Table 4.
file	FileMetadata	The uploaded file. Refer to Table 3.

10.4 File Exchange session closed event

The FOLDER_CLOSED event is the event sent from the File Exchange Server to the participants that are subscribed to a File Exchange session via the SSE channel when the File Exchange session is closed as described in clause 9.

Table 7: File Exchange session closed event

Property	Type	Description
type	String	Type of the event. The value is "FOLDER_CLOSED". Refer to Table 4.
reason	String	The reason of the closure of the File Exchange session.

Annex A (normative): PEMEA File Exchange JSON schema

A.1 General

This normative annex includes all of the JSON schema necessary to implement the present document.

A.2 File Exchange invocation schema

This schema is used by the tPSP/PIM to invoke the PEMEA File Exchange capability in the AP.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "object",
    "title": "File_Exchange invocation schema",
    "properties": {
        "token": {
            "type": "string"
        },
        "uri": {
            "type": "string",
            "format": "uri"
        },
        "expiry": {
            "type": "number"
        }
    },
    "required": ["token", "uri", "expiry"]
}
```

A.3 File List schema

This schema specifies the structure returned by the File Exchange Server when the list of files is requested.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "title": "File_Exchange file list schema",
    "items": {
        "properties": {
            "name": {
                "type": "string"
            },
            "type": {
                "type": "string"
            },
            "size": {
                "type": "integer"
            },
            "url": {
                "type": "string",
                "format": "uri"
            },
            "timestamp": {
                "type": "integer"
            }
        },
        "required": [
            "name",
            "type",
            "size",
            "url",
            "timestamp"
        ],
        "type": "object"
    },
    "type": "array"
```

```
}
```

A.4 File schema

This schema specifies the structure returned by the File Exchange Server when a file is successfully uploaded.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "title": "File_Exchange file schema",
    "type": "object",
    "properties": {
        "name": {
            "type": "string"
        },
        "type": {
            "type": "string"
        },
        "size": {
            "type": "integer"
        },
        "url": {
            "type": "string",
            "format": "uri"
        },
        "timestamp": {
            "type": "integer"
        }
    },
    "required": ["name", "type", "size", "url", "timestamp"]
}
```

A.5 Error schema

This schema specifies the body of error responses.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "object",
    "title": "File_Exchange error schema",
    "properties": {
        "error": {
            "type": "string"
        },
        "message": {
            "type": "string"
        },
        "status": {
            "type": "integer"
        },
        "timestamp": {
            "type": "string",
            "format": "date-time"
        }
    },
    "required": ["error", "message", "status", "timestamp"]
}
```

A.6 File uploaded event schema

This schema specifies the event sent from the File Exchange session through SSE channels when a new file is uploaded to the File Exchange session.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "object",
    "title": "File_Exchange file uploaded event schema",
    "properties": {
        "file": {
```

```
            "type": "object",
            "properties": {
                "name": {
                    "type": "string"
                },
                "type": {
                    "type": "string"
                },
                "size": {
                    "type": "integer"
                },
                "url": {
                    "type": "string",
                    "format": "uri"
                },
                "timestamp": {
                    "type": "integer"
                }
            },
            "required": ["name", "type", "size", "url", "timestamp"]
        },
        "type": {
            "const": "FILE_UPLOADED"
        }
    },
    "required": ["file", "type"]
}
```

A.7 File Exchange session closed event schema

This schema specifies the event sent from the File Exchange session through SSE channels when the File Exchange session is terminated.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "object",
    "title": "File_Exchange session closed event schema",
    "properties": {
        "reason": {
            "type": "string"
        },
        "type": {
            "const": "FOLDER_CLOSED"
        }
    },
    "required": ["reason", "type"]
}
```

Annex B (informative): Recommended TLS cipher suits

This annex provides a recommended set of cipher suites for use with this protocol.

Table B.1: Recommended TLS 1.3 cipher suites

Cipher	TLS version	Encryption	MAC
TLS_AES_128_GCM_SHA256	1.3	AESGCM(128)	AEAD
TLS_AES_256_GCM_SHA384	1.3	AESGCM(256)	AEAD
TLS_CHACHA20_POLY1305_SHA256	1.3	CHACHA20/POLY1305(256)	AEAD

Table B.2: Acceptable TLS 1.2 cipher suites

Cipher	TLS version	Encryption	MAC
ECDHE-ECDSA-AES128-GCM-SHA256	1.2	AESGCM(128)	AEAD
ECDHE-RSA-AES128-GCM-SHA256	1.2	AESGCM(128)	AEAD
ECDHE-ECDSA-AES256-GCM-SHA384	1.2	AESGCM(256)	AEAD
ECDHE-RSA-AES256-GCM-SHA384	1.2	AESGCM(256)	AEAD
ECDHE-ECDSA-CHACHA20-POLY1305	1.2	CHACHA20/POLY1305(256)	AEAD
ECDHE-RSA-CHACHA20-POLY1305	1.2	CHACHA20/POLY1305(256)	AEAD
DHE-RSA-AES128-GCM-SHA256	1.2	AESGCM(128)	AEAD
DHE-RSA-AES256-GCM-SHA384	1.2	AESGCM(256)	AEAD

History

Document history		
V1.1.1	July 2024	Publication