Functional Analysis - Attachments

Proposal for HL7 Belgium WG Infrastructure & Security

Version management

Version	Date	Description	Author
1.0	21/09/2023	Initial analysis	Maxime Caucheteur (Smals - Vitalink)

1. Introduction

Not all useful information can be shared textually as part of standardized FHIR resources. Attachments like images, videos, PDF's can be useful to visualize or add details to the medical resource or Communication resources. Therefore, we need to find a way to store and serve back attachments that are sent along with FHIR resources.

2. Solutions in FHIR

The FHIR specification provides 1 data type and 3 resource types to handle this use case:

- Data Type:
  - **Attachment**: *"This type is for containing or referencing attachments - additional data content defined in other formats. The most common use of this type is to include images or reports in some report format such as PDF. However, it can be used for any data that has a MIME type."*
- Resource Type:
  - **Binary**: *"A resource that represents the data of a single raw artifact as digital content accessible in its native format. A Binary resource can contain any content, whether text, image, pdf, zip archive, etc."*
  - **Media**: *"A photo, video, or audio recording acquired or used in healthcare. The actual content may be inline or provided by direct reference."*
  - **DocumentReference**: *"A reference to a document of any kind for any purpose. Provides metadata about the document so that the document can be discovered and managed. The scope of a document is any serialized object with a mime-type, so includes formal patient centric documents (CDA), clinical notes, scanned paper, and non-patient specific documents like policy text."*

As the clients are interacting with us through a REST API, and as the FHIR specification recommends it, the file attachments will be sent to us as an embedded base64 string in the resource data itself.

3. Pros and Cons

Pros and Cons of Binary

PROS	CONS
- **Uniform way of using attachments/files**: client always needs to use the resource type Binary. The Binary will be linked in the medical resource. - **File is loosely bounded to the medical resource**: an update of the medical resource does not require a new upload of the file (only if there is a new file of course). - **A Binary resource can contain any content** which allows for more flexibility. - **Easier to plugin on the attachments coming in** to offload it to external file storage/service/db ... ⟶ abstraction layer in the vault. - **The FHIR Binary resource is designed to efficiently store and manage binary data within a FHIR server**. It does not impose strict size limitations on the data it can represent, which means it can handle files of substantial size without causing performance issues or requiring complex workarounds. - FHIR servers and applications that support the Binary resource **can implement chunking and streaming mechanisms to transmit, store, or retrieve large binary files**.	- **Specific implementation will be required to cover the security on the Binary resource**: a Binary can only be requested if the access is granted on the medical resource (securityContext). - **We need to avoid that the vault will be used as file storage**: the endpoint of Binary can only be used if there is a medical resource linked to it where the client has create/update access for. - Large binary files may need to be retained and archived for extended periods for compliance and legal reasons. **Implementing long-term storage and archival solutions is essential**. - **When retrieving the medical resource the Binary should be retrieved in a separate call**. At least for a GET by ID (read). For searches we could allow the include search to include the attachments in the search result, BUT we need to be careful with the size the Bundle will become. - The Binary resource **does not provide specific semantics or metadata related to the content of the binary data**.

Pros and Cons of Media

PROS	CONS
– **It provides a standardized way to capture rich metadata associated with media content**. This metadata can include information about the patient, the context in which the media was created, and technical details. - **The Media resource makes it easier to search for and retrieve media content within a FHIR**	- **Narrow focus on media content**. It may not be suitable for handling binary data that doesn't fit the typical definition of "media" such as pdf, zip, proprietary formats, ... - For simple use cases where basic binary data storage is sufficient, the **Media resource may**

ecosystem. You can query and filter media records based on attributes like modality, patient, or date, facilitating efficient content retrieval. - **It aligns with FHIR's principles of standardization**, making it easier for healthcare organizations to adopt and implement a consistent approach to managing media data.	**introduce unnecessary complexity due to its rich metadata and specialized structure**. - Media resource can handle binary data, but **it may not be as efficient as the FHIR Binary resource when dealing with large files**.

Pros and Cons of DocumentReference

PROS	CONS
- DocumentReference resource is **designed specifically for structured documents**, such as clinical notes, reports, and other healthcare documents. **It provides a standardized way to represent document content**, including information about who/what/when, ... - It **allows to link to external documents or binary resources** using references, making it suitable for referencing media files or binary data associated with documents. - **The DocumentReference resource can capture clinical context**, making it valuable when documents need to be associated with specific patient encounters or episodes of care.	- While DocumentReference can be used to reference media files or binary data associated with documents, **it is not as specialized for media content as the Media resource**.

4. Proposal

The choice between Binary, DocumentReference and Media depends on business needs. Seeing the properties of the three different resources types: Binary is the most simple and straight forward, DocumentReference is more focused on documents and Media on images/videos.

As we will need to support different files format such as MP4, MOV, MKV, Dicom, g3fax, gif, jpeg, png, tiff or pdf, Binary seems more appropriate as Media and DocumentReference will not be flexible enough.

FHIR Specification: https://hl7.org/fhir/R4/binary.html

There are different options to send the attachments to the server: in a Transaction, as a contained resource or as a 'standalone' resource.

4.1 Transaction

Below are presented the different flows for each CRUD operation when the Binary is sent in a Transaction:

CRUD Operations Flow

Create:

If a Binary is created, it has to have a reference to the medical resource (in securityContext). The medical resource must also have a reference to the binary.

1. Medical resource & binary are created in a Transaction: We check that the resource has a reference to the Binary and vice-versa. Internal references can be used.
2. Medical resource already exists: If there is a create of a Binary, it has to be in a Transaction and contain an update of the medical resource adding a reference to the Binary.

All other scenarii will fail.

Read:

Basic behavior, the access will be checked based on the securityContext of the Binary.

Update:

We won't support the update of Binary.

Delete:

We need to preserve referential integrity.

1. The Binary & the medical resource are deleted in a Transaction: We check that the two resources are linked and delete both resources.
2. The Binary is deleted: Check that the transaction contains an update of the medical resource where we remove the reference to the binary.
3. The medical resource is deleted: If there is a Binary linked to this resource (need to check properties looking for a reference to a Binary), the resource cannot be deleted if there is no delete of that binary in the Transaction.

Search:

No Search supported on Binary in the FHIR specifications.

4.2 Contained resources

For Contained resources, there are two different options. We can either keep the contained resource inside the resource and store the resource as is in the database, or we extract the Binary from the contained resource and store it in a separate database as a resource with its own ID (The medical resource would have a reference to this created Binary).

Below are presented the different flows for each CRUD operation when the Binary is sent as a contained resource and is extracted:

CRUD Operations Flow – Contained resource is extracted

Create:

The Binary is added as a contained resource inside the FHIR resource. The server checks for the presence of a Binary in contained resource. The Binary is extracted and stored in a separate database. The medical resource holds a reference to the attachment.

Read:

When reading the medical resource, the response is smaller as we only have a reference to the Binary and if we want to get that Binary, we can do a GET on the reference.

Update:

In case of an update on a medical resource (without changing the Binary), the reference to the Binary stays.

In case of an update with a change to the Binary (new contained resource), the server will need to detect there is a new contained resource, extract it and create a new Binary and store it. Then, what happens to the previous Binary ? Keep it as if we read the history of the resource, we might still need it ?

Delete:

Delete of the Binary as it is created as a resource on its own. But if we want to ensure referential integrity, it can only be deleted if the medical resource referring to it, is deleted or has a new version without a reference to the Binary.

Search:

No Search supported on Binary in the FHIR specifications.

Search is done for the medical resource and then we can do a GET on the Binary referenced.

CRUD Operations Flow – Contained resource kept as is

Create:

The medical resource (with the contained resource) is stored as is in the database.

Read:

As the contained resource is in the medical resource, the server always returns the attachment with the resource (in case of large files, this can be a problem).

Update:

As the contained resource is in the medical resource, updates can be done in a classic way.

Delete:

Delete the medical resource and by this the attachment is deleted together with it.

Search:

Search is not supported on Binary as it does not have any searchable properties, the search happens on the medical resource and the attachment is retrieved in the response.

4.3 ‘Standalone’ resource

By ‘Standalone’ resource is meant the idea of having an endpoint for Binary.

Below are presented the different flows for each CRUD operation when the Binary is sent on the Binary endpoint:

CRUD Operations Flow

Create:

The Binary is sent to the /Binary endpoint with a reference to the medical resource in the securityContext (on the condition the medical resource ID is created first). There is one thing the server can’t enforce and that is that the resource does not reference back to the Binary. If a client is not updating the resource we have an orphan Binary resource. In case the client does send the update it is ok. The disadvantage of this setup is 3 separate requests need to take place for the creation of the resource and the attachment.

Read:

Read by ID. Behavior like other FHIR resources.

Update:

- We decide to support updates so the content can be changed (you don’t need to change the reference in the medical resource if you upload a new version of a certain document).
- We decide not to support updates which enforces the clients to create a new Binary and update the reference to the Binary.

Delete:

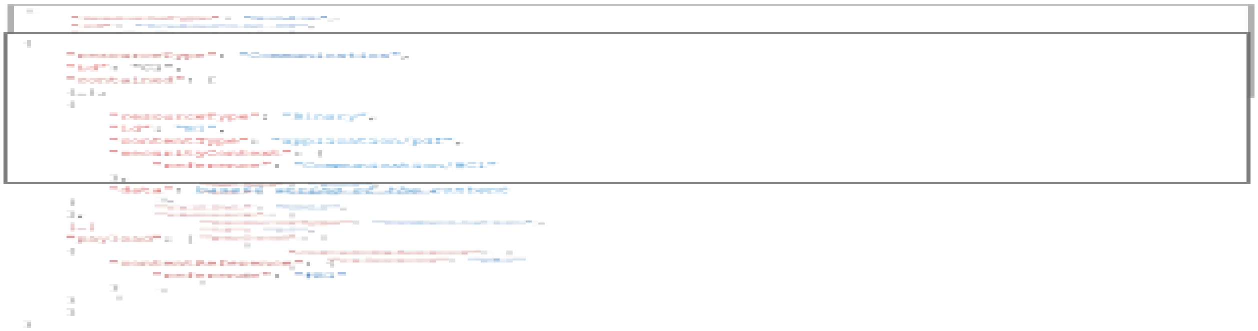
If we delete the Binary, the medical resource it was linked to will reference a deleted resource. The referential integrity will be broken.

Search:

No Search supported on Binary in the FHIR specifications.

5. Examples

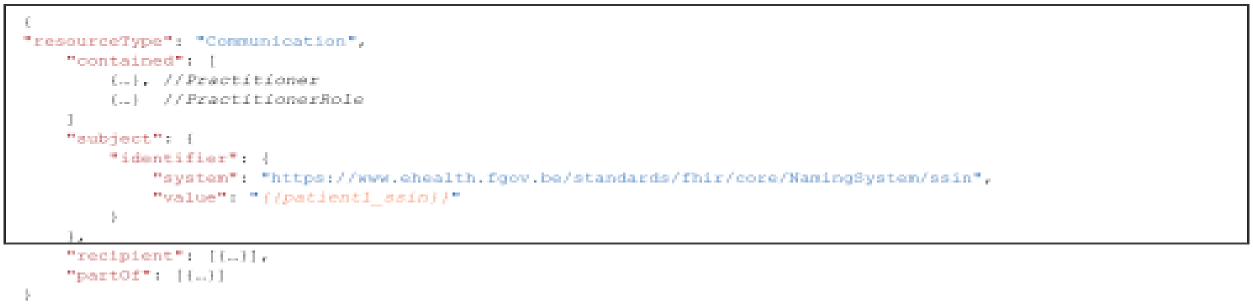
5.1 Creation of a Binary in a Transaction



```
{
    "resourceType": "Communication",
    "id": "C1",
    "contained": [
    {…},
    {
        "resourceType": "Binary",
        "id": "B1",
        "contentType": "application/pdf",
        "securityContext": {
            "reference": "Communication/#C1"
        },
        "data": Base64 String_of_the_content
    }
    ],
    [...]
    "payload": [
    {
        "contentReference": {
            "reference": "#B1"
        }
    }
    ]
}
```

5.2 Creation of a Binary in a contained resource

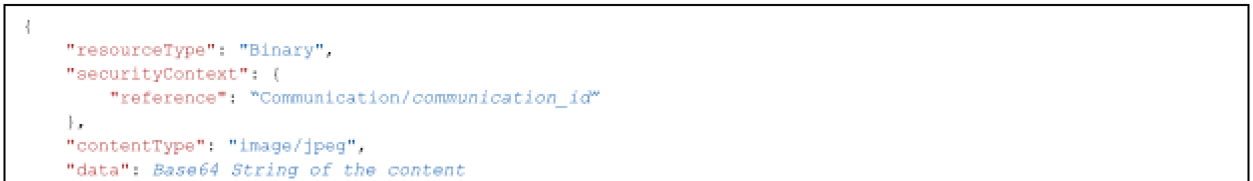
5.3 Creation of a Binary as a 'Standalone' resource



```
{
"resourceType": "Communication",
    "contained": [
        {…}, //Practitioner
        {…}  //PractitionerRole
    ]
    "subject": {
        "identifier": {
            "system": "https://www.ehealth.fgov.be/standards/fhir/core/NamingSystem/ssin",
            "value": "{{patient1_ssin}}"
        }
    },
    "recipient": [{…}],
    "partOf": [{…}]
}
```

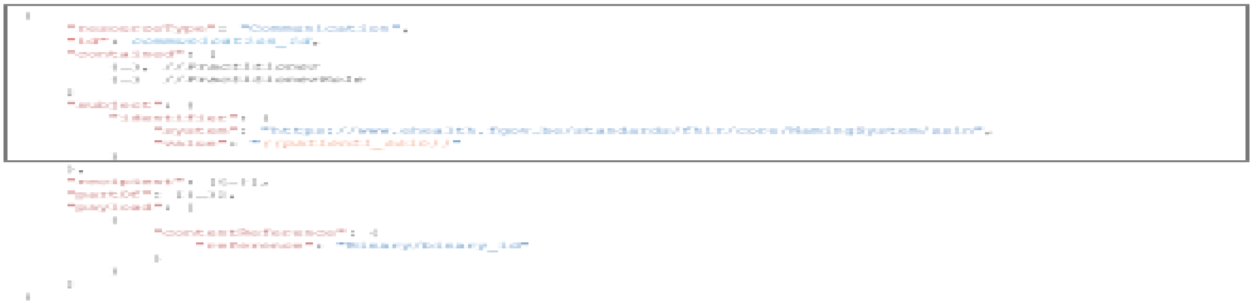
First POST request – Communication:

Second POST request - Binary:



```
{
    "resourceType": "Binary",
    "securityContext": {
        "reference": "Communication/communication_id"
    },
    "contentType": "image/jpeg",
    "data": Base64 String of the content
```

PUT request – Communication:



```
{
    "resourceType": "Communication",
    "id": communication_id,
    "contained": [
        {…}, //Practitioner
        {…}  //PractitionerRole
    ]
    "subject": {
        "identifier": {
            "system": "https://www.ehealth.fgov.be/standards/fhir/core/NamingSystem/ssin",
            "value": "{{patient1_ssin}}"
        }
    },
    "recipient": [{…}],
    "partOf": [{…}],
    "payload": [
        {
            "contentReference": {
                "reference": "Binary/binary_id"
            }
        }
    ]
}
```