

## UPLOADING A FILE FROM PO TO AWS s3 WITHOUT SIZE RESTRICTIONS

One of the many challenges that we face in On-Premise to Cloud migration is the difference in approach to integration. While solutions on the cloud have a great flexibility in making messages as granular as possible, on-premise solutions are bound to restricting factors like bandwidth, memory and message sizes. As a basic premise to SAP PO development, our aim as developers is to optimize the many factors of an integration requirement, message size being very important.

This document is aimed at providing one of the many solutions that we can adapt to transfer a file from PO to AWS s3 bucket without us having to worry about the size of the file. AWS has a list of APIs available for us to use to upload files into the s3 bucket. As mentioned above, AWS doesn't have a restriction on the size of the file received, but this issue is prominent in PO. You can find details of AWS API details here -

<https://docs.aws.amazon.com/AmazonS3/latest/API>Welcome.html>

Amazon gives us 3 options to upload files into s3 via APIs

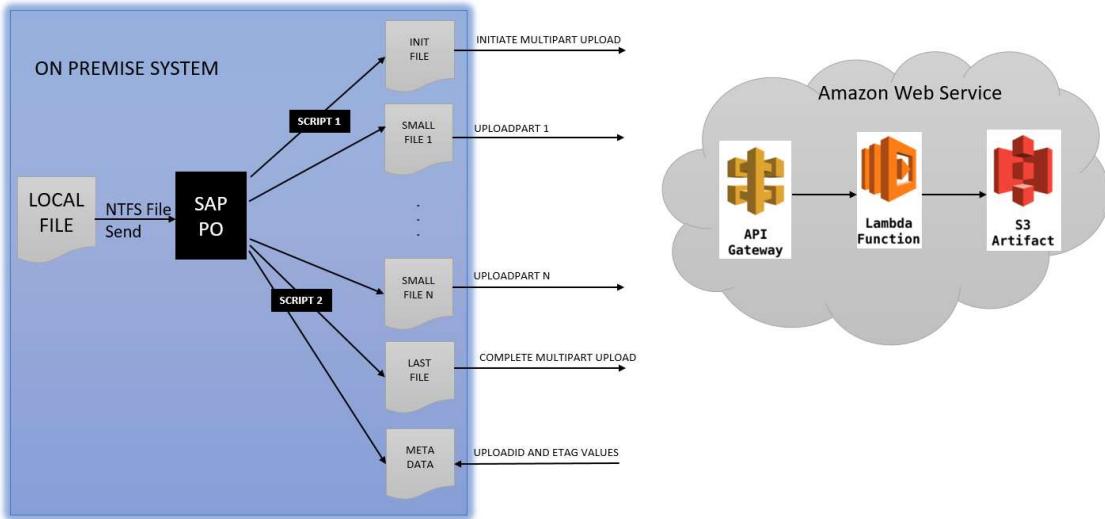
Single Chunk Upload - <https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-header-based-auth.html>

Multi Chunk Upload - <https://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-streaming.html>

Multi Part Upload - <https://docs.aws.amazon.com/AmazonS3/latest/dev/mpuoverview.html>

We had a file size restriction of ~900KB which wasn't very promising. Amazon's Single Chunk Upload was the best option. This is covered extensively in Rajesh PS's blog - <https://blogs.sap.com/2019/05/31/integrating-amazon-simple-storage-service-amazon-s3-and-sap-ecc-v6.0-via-sap-pi-v7.5-using-aws-signature-v5-and-signing-algorithm-hmac-sha256/>

Multi-chunk upload was not feasible for PO since it requires an HTTP connection to be open until all the chunks have been transferred. That leaves us with the Multipart Upload option. I will attempt to cover how I transferred a huge file using multipart upload. The basic design is as follows:

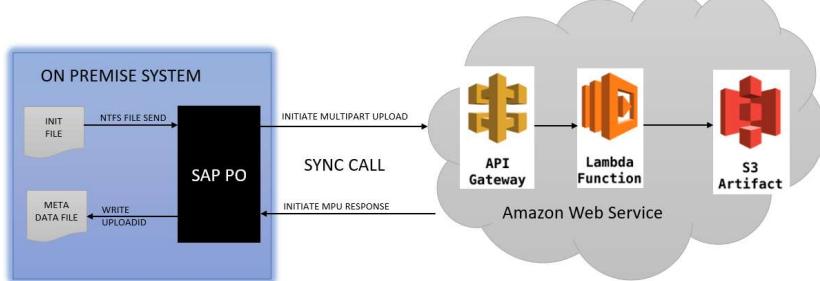


## AWS MULTIPART UPLOAD APIs

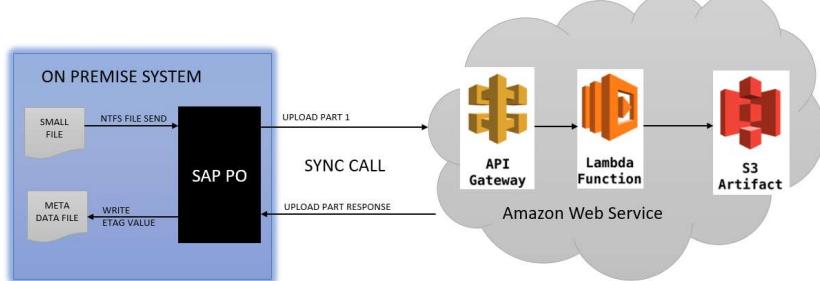
1. Multipart Upload works by initiating an API. AWS returns us an UploadId. The smaller files are then sent one by one with the UploadId and part number. AWS allows for parallel sending of the multiple parts, but we will need to know the sequence number beforehand, which is not possible in this design. AWS returns an ETag for each file uploaded. After all the parts have been uploaded, you will then need to call the Complete Multipart Upload API with all the ETags and the Part Number. This enables AWS to ‘stitch’ back the multiple small files into one large file based on the details sent. I’ve had to develop the AWS signature from scratch: <https://github.com/pnathan01/AWSSAPMultipartUploadLibrary>. I also took inspiration from Rajesh’s blog for single chunk upload(link in the last section)

The following is a diagrammatical representation of each of the 3 types of calls made to s3 API.

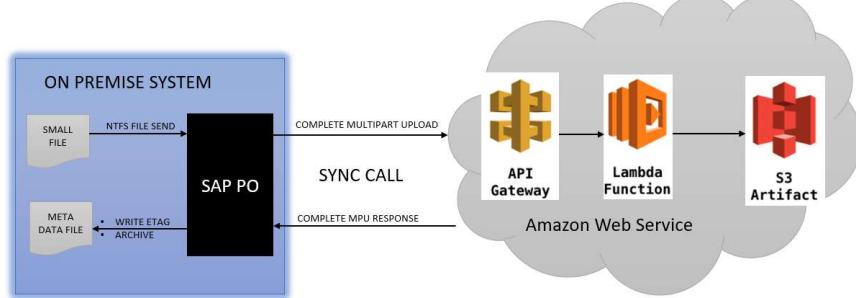
### INITIATE MULTIPART UPLOAD



### UPLOADPART API



### COMPLETE MULTIPART UPLOAD



## IMPLEMENTATION IN PO

As you can see all the above communication are Async-Sync bridges. I have created 4 ICOs – 1 for Request with routing to different APIs and 3 different ICOs for the response part. All 3 of the response ICOs will write into the metadata file.

Please note that I have not accounted for any data conversion, format checks or mapping of the data in the file. Technically they are pass through scenarios, except Complete Multipart Upload, but we have to use mappings to calculate AWS API signatures.

The following configuration will need to be done beforehand:

1. Implement certificates, if any
2. Enable large message handling. This is because AWS requires all smaller files of a multipart upload, except the last one, to be 5 MB or larger.

### Create ESR Objects

We have 2 different software components for source and target systems. So this scenario is from Software Component 1 to Software Component 2. Also, we will be using a common structure to write the metadata content like UploadIDs and ETags into the metadata file.

1. Data Types
  - a. Request Data Types in both software components.

**Display Data Type**

Name	ManagementFee
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	[REDACTED]

Classification: Free-Style Data Type | Qualify Schema: None

Type Definition | XSD

Name	Category	Type	Occurrence	Default
ManagementFee	Complex Type			
Records	Element		0..unbounded	
text	Element	xsd:string	0..1	

- b. Response Data Type in Software Component 1

**Display Data Type**

Name	MultipartUploadResponse
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	

Classification: Free-Style Data Type | Qualify Schema: None

Type Definition | XSD

Name	Category	Type	Occurrence
MultipartUploadResponse	Complex Type		
Record	Element		0..1
Bucket	Element	xsd:string	0..1
Key	Element	xsd:string	0..1
UploadId	Element	xsd:string	0..1
ETag	Element	xsd:string	0..1

- c. Request Data Type in Software Component 2. CompleteMultipartUpload Data Type will not be a pass through mapping because there is actual data that is required to be sent.

**Display Data Type**

Name	ManagementFee
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	[REDACTED]

Classification: Free-Style Data Type Qualify Schema: None

Type Definition XSD

Name	Category	Type	Occurrence	Default
ManagementFee	Complex Type			
Records	Element		0..unbounded	
text	Element	xsd:string	0..1	

**Display Data Type**

Name	CompleteMultipartUpload
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	AWS Complete Multipart Upload Structure

Classification: Free-Style Data Type Qualify Schema: None

Type Definition XSD

Name	Category	Type	Occurrence
CompleteMultipartUpload	Complex Type		
Part	Element		0..unbounded
ETag	Element	xsd:string	0..1
PartNumber	Element	xsd:integer	0..1

- d. Response Data Type in Software Component 2. The InitiateMultipartUploadResult data type is used to capture the response of both Initiate MPU and Upload Part because the UploadID and ETags are part of the header.

**Display Data Type**

Name	InitiateMultipartUploadResult
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	

Classification: Free-Style Data Type Qualify Schema: None

Type Definition XSD

Name	Category	Type	Occurrence
InitiateMultipartUploadResult	Complex Type		
Bucket	Element	xsd:string	0..1
Key	Element	xsd:string	0..1
UploadId	Element	xsd:string	0..1

**Display Data Type**

Name	CompleteMultipartUploadResult
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	

Classification: Free-Style Data Type   Qualify Schema: None

Type Definition: XSD

Name	Category	Type	Occurrence
CompleteMultipartUploadResult	Complex Type		
Location	Element	xsd:string	0..1
Bucket	Element	xsd:string	0..1
Key	Element	xsd:string	0..1
ETag	Element	xsd:string	0..1

2. Message Types

a. Request Message Type in Software Component 1

**Display Message Type**

Name	ManagementFeeMsg
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	

Name \*

Data Type Used: ManagementFee

b. Response Message Type in Software Component 1

**Display Message Type**

Name	MultipartUploadResponseMsg
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	

Name \*

Data Type Used: MultipartUploadResponse

c. Request Message Type in Software Component 2

**Display Message Type**

Name	ManagementFeeMsg
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	

Name \*

Data Type Used: ManagementFee

**Display Message Type**

Name	CompleteMultipartUpload
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	[REDACTED]
Name *	
Data Type Used	CompleteMultipartUpload

- d. Response Message Type in Software Component 2 - I've changed the namespace for these structures because I didn't want to develop code/adapter module to handle this separately.

Just like the data type, this message type is common for both Initiate MPU and UploadPart API Response structures.

**Display Message Type**

Name	InitiateMultipartUploadResult
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	[REDACTED]
Name *	
Data Type Used	InitiateMultipartUploadResult
XML Namespace	<a href="http://s3.amazonaws.com/doc/2006-03-01/">http://s3.amazonaws.com/doc/2006-03-01/</a>

**Display Message Type**

Name	CompleteMultipartUploadResult
Namespace	[REDACTED]
Software Component Version	[REDACTED]
Description	[REDACTED] AWS - s3 - Complete Multipart Upload Result Structure
Name *	
Data Type Used	CompleteMultipartUploadResult
XML Namespace	<a href="http://s3.amazonaws.com/doc/2006-03-01/">http://s3.amazonaws.com/doc/2006-03-01/</a>

### 3. Service Interfaces

- a. One outbound service for all requests from Software Component 1

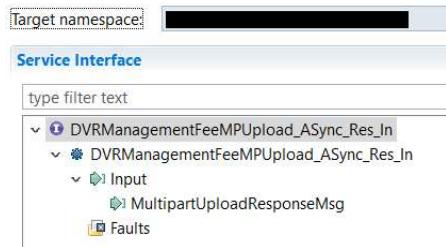
Target namespace: [REDACTED]

**Service Interface**

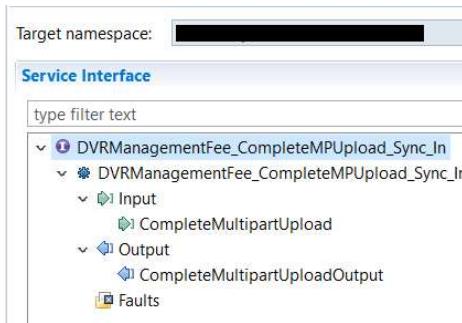
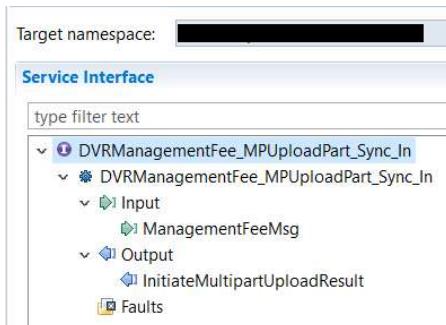
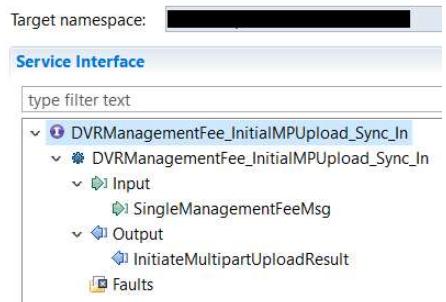
type filter text

- ▼ **DVRManagementFeeMPUpload\_ASync\_Req\_Out**
  - ▼ **DVRManagementFeeMPUpload\_ASync\_Req\_Out**
    - ▼ **Input**
      - ManagementFeeMsg**

- b. One inbound service for all responses from Software Component 2



- c. 3 Synchronous inbound services to send the request messages



- d. 3 outbound asynchronous services to receive the response – You will notice that we have separate services for InitialMPU and UploadPart even though the structure is the same. This is because the mappings for them are different.

Target namespace: [REDACTED]

**Service Interface**

type filter text

- ▼ ⓘ DVRManagementFee\_InitialMPUpload\_Res\_ASync\_Out
  - ❖ DVRManagementFee\_InitialMPUpload\_Res\_ASync\_Out
  - ❖ Input
    - ▷ InitiateMultipartUploadResult

Target namespace: [REDACTED]

**Service Interface**

type filter text

- ▼ ⓘ DVRManagementFee\_MPUploadPart\_Res\_ASync\_Out
  - ❖ DVRManagementFee\_MPUploadPart\_Res\_ASync\_Out
  - ❖ Input
    - ▷ InitiateMultipartUploadResult

Target namespace: [REDACTED]

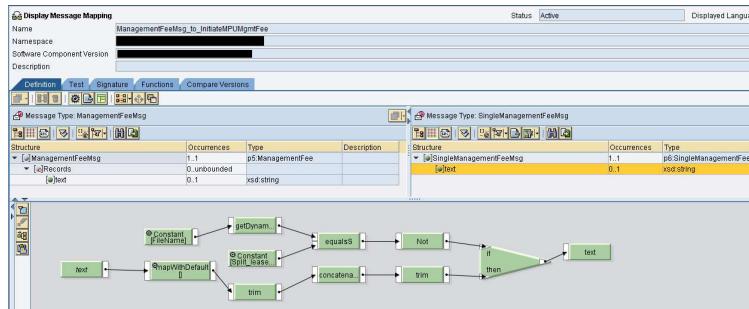
**Service Interface**

type filter text

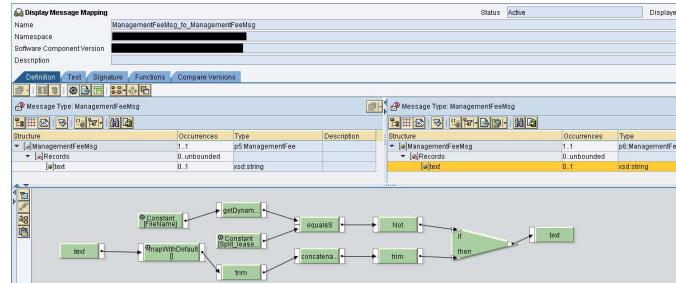
- ❖ DVRManagementFee\_CompleteMPUploadPart\_Res\_ASync\_Out
  - ❖ DVRManagementFee\_CompleteMPUploadPart\_Res\_ASync\_C
    - ❖ Input
      - ▷ CompleteMultipartUploadResult

2. Mapping Programs – I've added a parameter called action because I reuse the same library code for both single chunk and multipart upload. The difference is required to calculate the AWS signature for the payload. The mappings themselves are pretty straightforward except for the signature calculation, which is handled entirely in the library. Code for AWS Signature Calculation:  
<https://github.com/pnathan01/AWSSAPMultipartUploadLibrary>

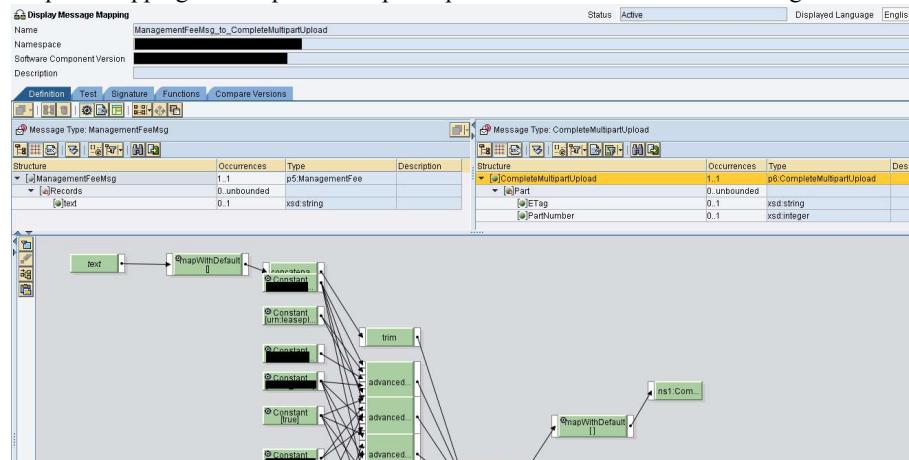
e. Request Mapping – Initiate Multipart Upload.



f. Request Mapping – Upload Part



g. Request Mappings – Complete Multipart Upload – Conversion and Formatting



```

public void transform(TransformationInput transformationInput, TransformationOutput transformationOutput) throws StreamTransformationException {
    try {
        InputStream inputstream = transformationInput.getInputPayload().getInputStream();
        OutputStream outputstream = transformationOutput.getOutputPayload().getOutputStream();
        byte[] b = new byte[inputstream.available()];
        inputstream.read(b);
        String input = new String(b);
        input = input.replaceAll(""","");
        input = input.replaceAll("ns1","");
        input = input.replaceAll(" xmlns:ns1='urn:leaseplan.com:au:xi:unifi:dvr'");
        String splitStr [] = input.split("<CompleteMultipartUpload");
        input = "<CompleteMultipartUpload"+splitStr[1];
        TransformerFactory tFactory = TransformerFactory.newInstance();
        Transformer transformer = tFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,"yes.");
        outputstream.write(input.getBytes());
    } catch (Exception exception) {
        getTrace().addDebugMessage(exception.getMessage());
        throw new StreamTransformationException(exception.toString());
    }
}

```

h. Response Mapping - Initiate Multipart Upload – Java code to handle empty message

The screenshot shows two message mapping configurations in Oracle SOA Studio:

- InitiateMultipartUploadResult\_Namespace\_Mapping:**

```
/*
 * You can create source text for the class body in area 'Attributes and Methods'.
 * Exceptions are import statements; you must enter these in the import table. You
 * can use the inner classes, methods, and attributes defined here in the user-defined
 * functions and in the methods "init, cleanUp"
 * You can also use classes from imported archives that are in the same software
 * component version as the function library, or in a sub software component version.
 * You must specify the imported archives in table 'Archives Used'
 */
public void transform(TransformationInput transformationInput, TransformationOutput transformationOutput) throws StreamTransformationException {
    try {
        InputStream inputStream = transformationInput.getInputPayload().getInputStream();
        OutputStream outputStream = transformationOutput.getOutputPayload().getOutputStream();

        byte[] b = new byte[inputStream.available()];
        inputStream.read(b);
        String encoding = "UTF-8";
        String inputXML= "";
        String input = new String(b);
        getTrace().addInfo("input"+input);
        input = input.replaceAll("<InitiateMultipartUploadResult", "<ns0:InitiateMultipartUploadResult");
        input = input.replaceAll("xmlns=", "xmlns:ns0=");
        input = input.replaceAll("</InitiateMultipartUploadResult", "</ns0:InitiateMultipartUploadResult");
        outputStream.write(input.getBytes());
    } catch (Exception exception) {
        getTrace().addDebugMessage(exception.getMessage());
        throw new StreamTransformationException(exception.toString());
    }
}
```
- InitiateMultipartUploadResult\_to\_MultipartUploadResponseMsg:**

Structure	Occurrences	Type	Description
InitiateMultipartUploadResult	1..1	pb:InitiateMultipartUploadResult	
Bucket	0..1	xsd:string	
Key	0..1	xsd:string	
UploadId	0..1	xsd:string	

Message Type: MultipartUploadResponseMsg

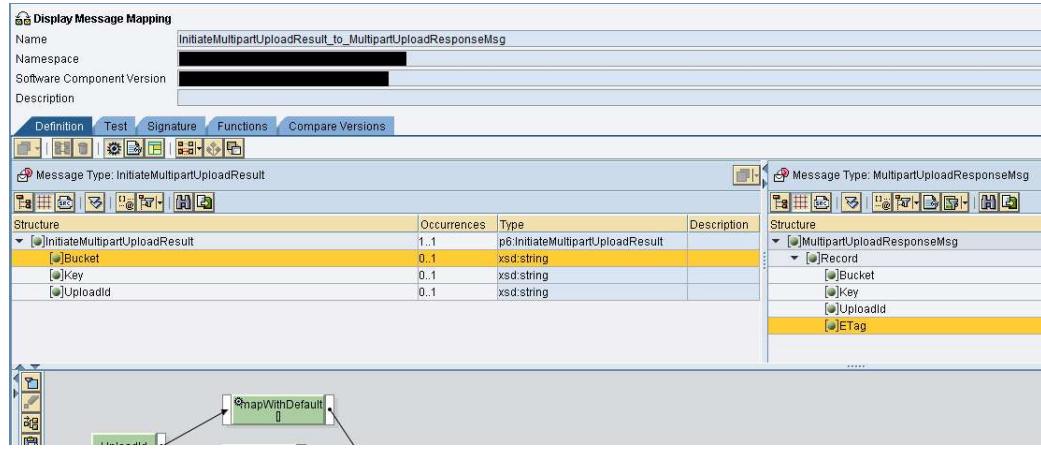
Structure:

  - MultipartUploadResponseMsg
    - Record
      - Bucket
      - Key
      - UploadId
      - ETag

i. Response Mapping – Upload Part – Java code to handle empty message received.

The screenshot shows the Java code for handling empty multipart upload part messages:

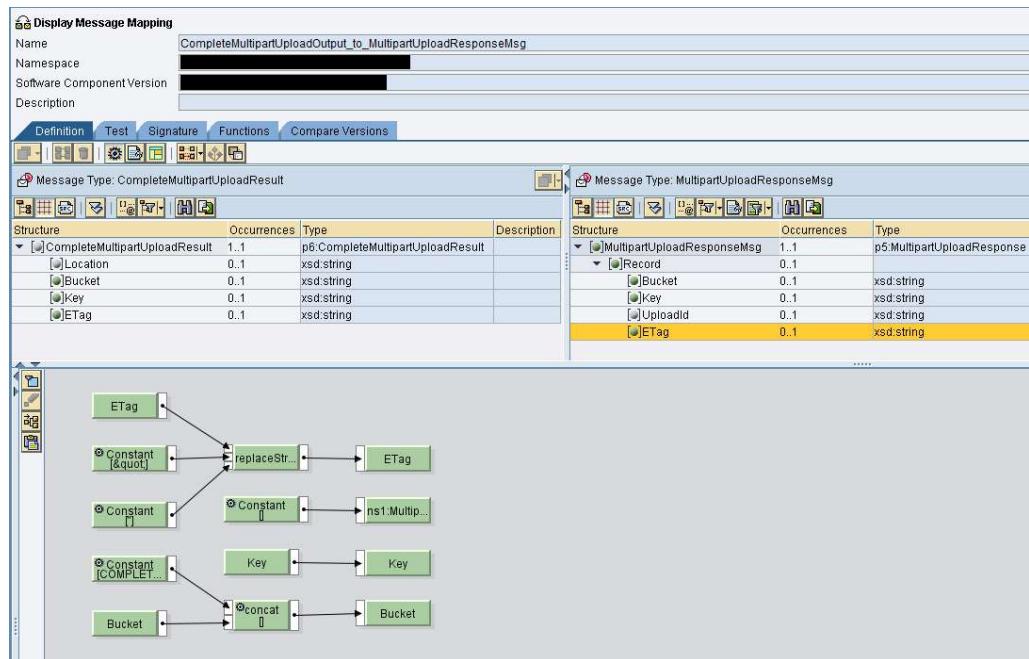
```
public void transform(TransformationInput transformationInput, TransformationOutput transformationOutput) throws StreamTransformationException {
    try {
        InputStream inputStream = transformationInput.getInputPayload().getInputStream();
        OutputStream outputStream = transformationOutput.getOutputPayload().getOutputStream();
        String input = "<?xml version='1.0' encoding='UTF-8'?>"+'\n' +
                     "<ns0:InitiateMultipartUploadResult xmlns:ns0='http://s3.amazonaws.com/doc/2006-03-01'>"+'\n' +
                     "<Bucket>"+'\n' +
                     "<Key>"+'\n' +
                     "<UploadId>"+'\n' +
                     "</ns0:InitiateMultipartUploadResult>";
        outputStream.write(input.getBytes());
    } catch (Exception exception) {
        getTrace().addDebugMessage(exception.getMessage());
        throw new StreamTransformationException(exception.toString());
    }
}
```



- j. Response Mapping – Complete Multipart Upload. The response is namespace-agnostic, causing issues in PI.

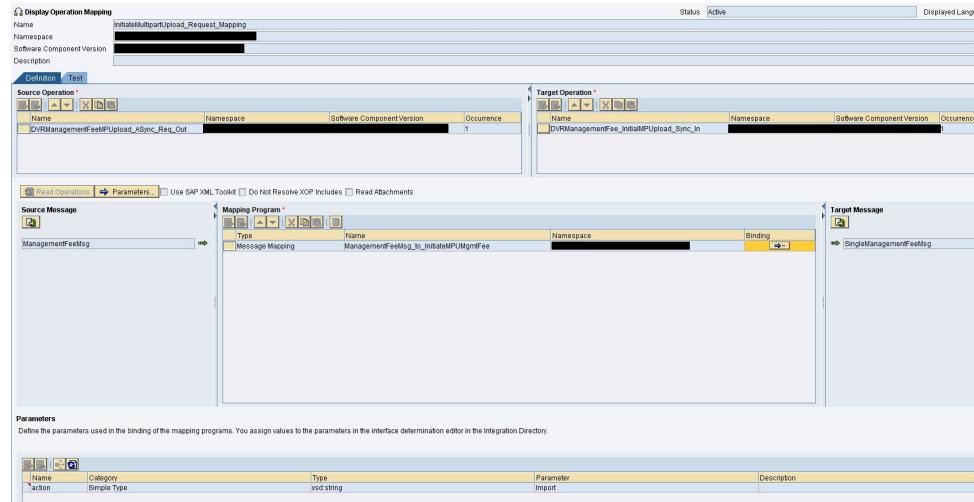
```
public void transform(TransformationInput transformationInput, TransformationOutput transformationOutput) throws StreamTransformationException {
    try {
        InputStream inputStream = transformationInput.getInputPayload().getInputStream();
        OutputStream outputStream = transformationOutput.getOutputPayload().getOutputStream();

        byte[] b = new byte[inputStream.available()];
        inputStream.read(b);
        String encoding = "UTF-8";
        String inputXML = "";
        String input = new String(b);
        getTrace().addInfo("input=" + input);
        input = input.replaceAll("xmlns:", "xmlns:ns0=");
        input = input.replaceAll("CompleteMultipartUploadResult", "ns0:CompleteMultipartUploadResult");
        input = input.replaceAll(""", "'");
        // input = input.replaceAll("CompleteMultipartUploadOutput", "<ns0:CompleteMultipartUploadOutput>");
        outputStream.write(input.getBytes());
    } catch (Exception exception) {
        getTrace().addDebugMessage(exception.getMessage());
        throw new StreamTransformationException(exception.toString());
    }
}
```

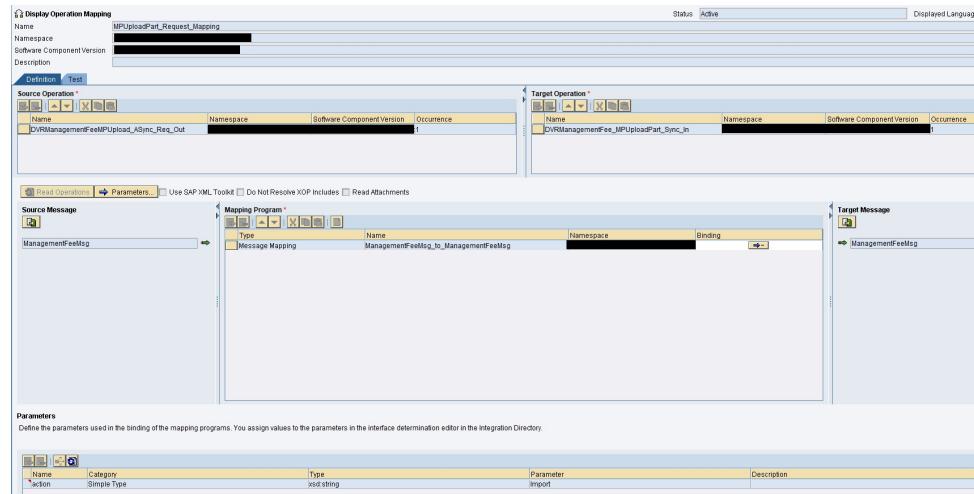


## 4. Operation Mappings

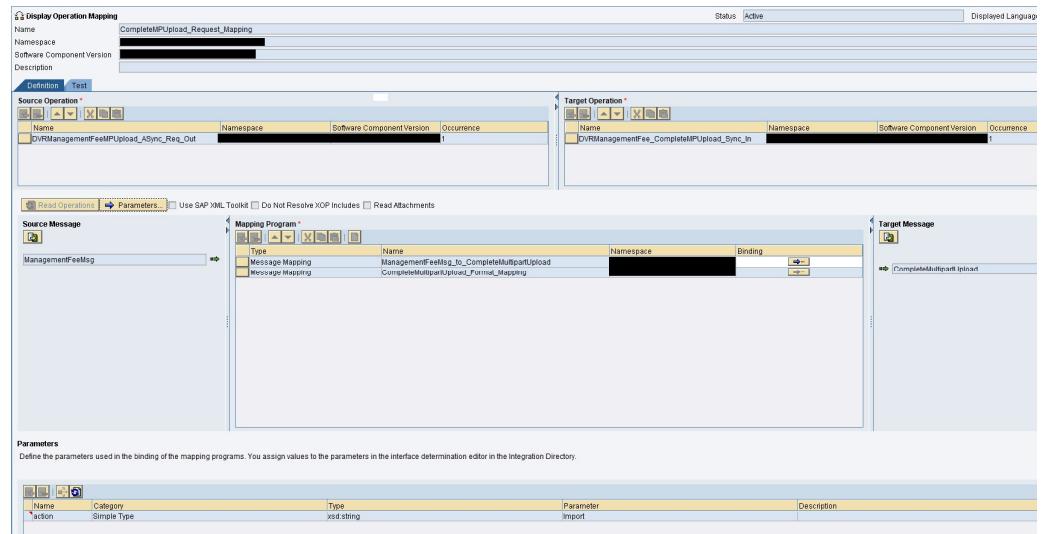
### a. Request OM – Initiate Multipart Upload



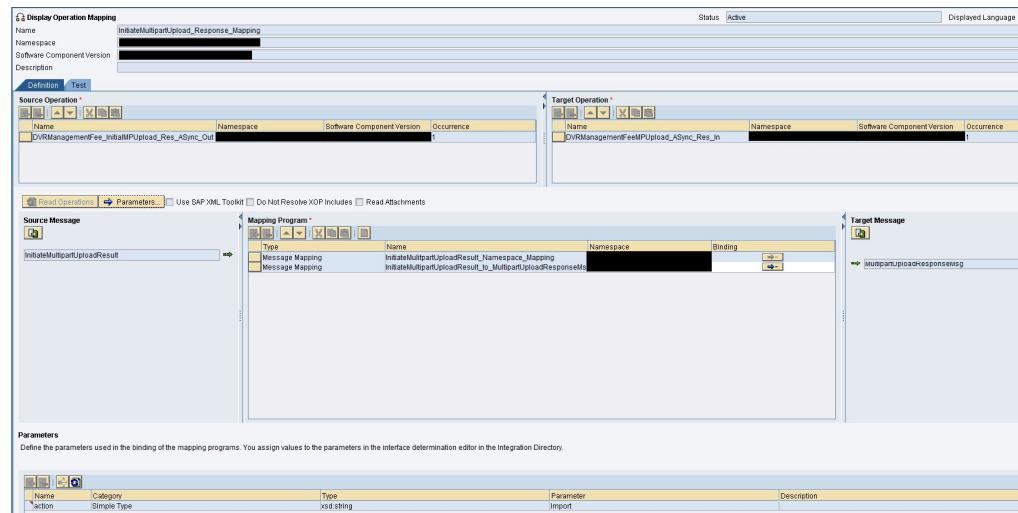
### b. Request OM – Upload Part



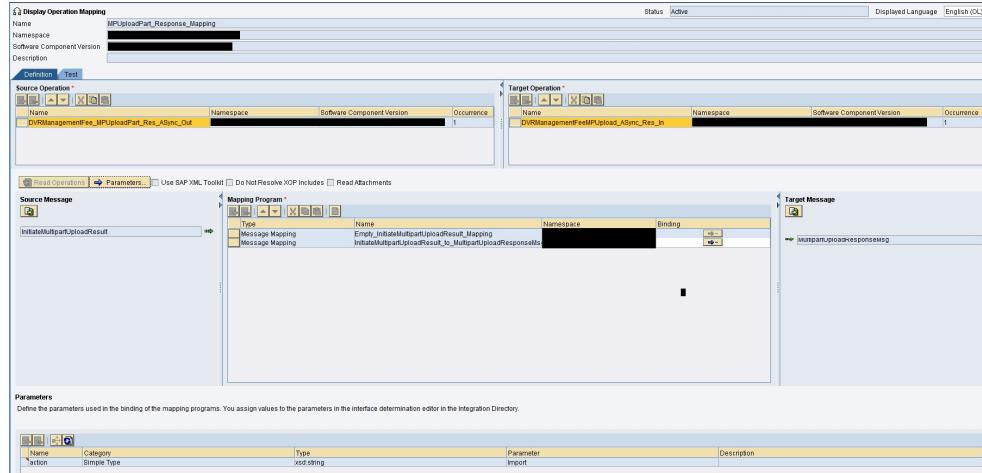
c. Request OM – Complete Multipart Upload



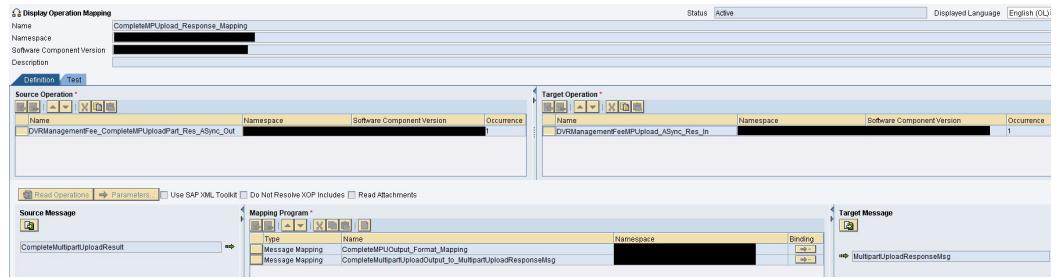
d. Response OM - Initiate Multipart Upload



e. Response OM – Upload Part



f. Response OM – Complete Multipart Upload



### Create ID Objects

We will create the following ID objects

- 1 ICO for splitting the files
- 1 ICO for sending out the request part of the ASync-Sync bridge
- 3 ICOs for receiving the response

## 1. File Split Scenario

- a. Sender Channel – The file is split based on number of records because AWS MPU has a condition that all multipart files except the last one be larger than 5 MB.

The screenshot shows the SAP Fiori interface for configuring a communication channel. The top section displays basic channel information: Communication Channel, Party, Communication Component, and Description. Below this, the 'Content Conversion' tab is selected, showing settings for Adapter Type (File), Transport Protocol (File System (NFS)), Message Protocol (File Content Conversion), and Adapter Engine (Central Adapter Engine). Under 'Content Conversion Parameters', fields include Document Name (ManagementFeeMsg), Document Namespace, Document Offset, Recordset Name (Records), Recordset Namespace, Recordset Structure (Records,1), Recordset Sequence (Ascending), Records per Message (30000), Key Field Name, and Key Field Type (String (Case-Sensitive)). A table below lists adapter-specific message attributes with their values:

Name	Value
ignoreRecordsetName	true
Records.endSeparator	'\n'
Records.fieldSeparator	'\n'
Records.fieldNames	text

**Adapter-Specific Message Attributes**

- Set Adapter-Specific Message Attributes
- File Name
- Directory
- File Type
- Source File Size
- Source File Time Stamp

Here I call 2 scripts, before and after the message processing. The first one creates a file of the format <FILE\_NAME>\_00000000-000000-000.txt. This will be the first file called which will trigger the Initiate Multipart Upload.

The second script creates the last file of the format <FILE\_NAME>\_99991231-235959-000.txt and Temp\_Metadata.txt. This file is the one to initiate the Complete Multipart Upload. The metadata file is the one that will hold all the UploadIDs and ETags for logging, audit and monitoring purposes.

I haven't provided the script code because it is simple, and it is dependent on your OS.

**Run Operating System Command Before Message Processing**

Command Line: sh [REDACTED] Create\_Initial\_File.sh  
 Timeout (secs): 90  
 Terminate Program After Timeout

**Run Operating System Command After Message Processing**

Command Line: sh [REDACTED] Create\_Complete\_File\_1.sh  
 Timeout (secs): 90  
 Terminate Program After Timeout

b. Receiver Channel

**Display Communication Channel**

Communication Channel: [REDACTED]  
 Party: [REDACTED]  
 Communication Component: [REDACTED]  
 Description: [REDACTED]

**Parameters** **Identifiers** **Module**

Adapter Type \*: File http:  
 Sender  Receiver  
 Transport Protocol \*: File System (NFS)  
 Message Protocol \*: File Content Conversion  
 Adapter Engine \*: Central Adapter Engine

**Target** **Processing** **Content Conversion** **Advanced**

**File Access Parameters**

Target Directory \*: [REDACTED]  
 Create Target Directory  
 File Name Scheme \*: Split [REDACTED]

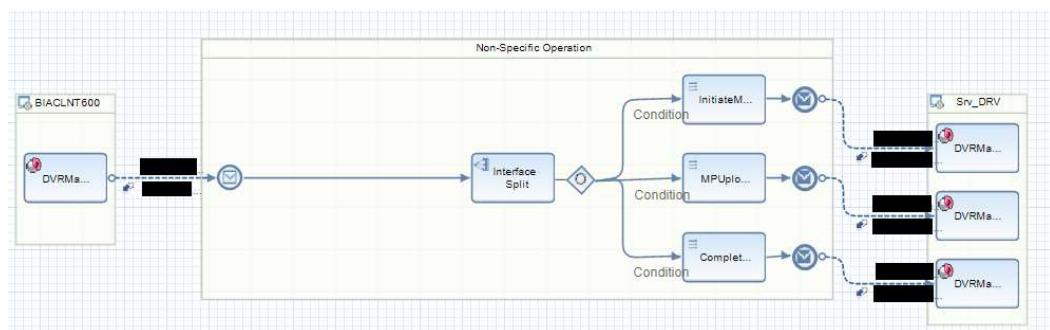
**Content Conversion Parameters**

Recordset Structure \*: Records

Name	Value
Records.fieldSeparator	'\n'
Records.endSeparator	'\n'
Records.fieldNames	text
Records.addHeaderLine	0

2. Request Send Scenario

a. ICO



Receiver Interfaces *		Maintain Order at Runtime	Operation Mapping	Name
Condition	(FileName = Split)	.00000000-000000-000.csv)	InitiateMultipartUpload_Request_Mapping	D
	(FileName ≠ Split)	.00000000-000000-000.csv AND FileName ≠ Split	MPUploadPart_Request_Mapping	D
	(FileName = Split)	.99999999-235959-000.csv)	CompleteMPUpload_Request_Mapping	D

b. Sender Channel

**Display Communication Channel**

Communication Channel	LPAU_BI_AU_DVR_ManagementFee_File_S		
Party	[REDACTED]		
Communication Component	[REDACTED]		
Description	[REDACTED]		
<b>Parameters</b>	<b>Identifiers</b>	<b>Module</b>	
Adapter Type *	File	http://sa...	
<input checked="" type="radio"/> Sender <input type="radio"/> Receiver			
Transport Protocol *	File System (NFS)		
Message Protocol *	File Content Conversion		
Adapter Engine *	Central Adapter Engine		
<b>Source</b>	<b>Processing</b>	<b>Content Conversion</b>	<b>Advanced</b>
<b>File Access Parameters</b>			
Source Directory *	[REDACTED]		
File Name *	[REDACTED]		
<input type="checkbox"/> Advanced Selection for Source File			
<input type="checkbox"/> Additional File(s)			
<b>Run Operating System Command Before Message Processing</b>			
Command Line	sleep 600		
Timeout (secs)	600		
<b>Content Conversion Parameters</b>			
Document Name	ManagementFeeMsg		
Document Namespace	[REDACTED]		
Document Offset			
Recordset Name	Records		
Recordset Namespace			
Recordset Structure *	Records,*		
Recordset Sequence *	Ascending		
Recordsets per Message			
Key Field Name			
Key Field Type	String (Case-Sensitive)		
<b>Adapter-Specific Message Attributes</b>			
<input checked="" type="checkbox"/> Set Adapter-Specific Message Attributes			
<input checked="" type="checkbox"/> File Name			
<input checked="" type="checkbox"/> Directory			
<input checked="" type="checkbox"/> File Type			
<input checked="" type="checkbox"/> Source File Size			
<input checked="" type="checkbox"/> Source File Time Stamp			

### c. Receiver Channels

#### Channel 1 – Request to Initiate Multipart Upload

<p><b>Display Communication Channel</b></p> <p>Communication Channel [REDACTED]</p> <p>Party [REDACTED]</p> <p>Communication Component [REDACTED]</p> <p>Description [REDACTED]</p> <p><b>Parameters</b> <b>Identifiers</b> <b>Module</b></p> <p>Adapter Type * REST http://sap.com/xi  <input type="radio"/> Sender <input checked="" type="radio"/> Receiver</p> <p>Transport Protocol * HTTP</p> <p>Message Protocol * REST</p> <p>Adapter Engine * Central Adapter Engine</p> <p><b>General</b> <b>REST URL</b> <b>REST Operation</b> <b>Data Format</b> <b>Operation Rules</b> <b>Response</b></p> <p><b>Warning: Parameterizing HTTP Host or Port is a security risk (see SAP Note: 2174651)</b></p> <p>URL Pattern * https://(header_host)/(filename)?uploads</p> <p><input checked="" type="checkbox"/> Follow Server Redirects on HTTP GET calls</p> <p><b>Pattern Variable Replacement</b></p> <table border="1"> <tr> <td>Value Source</td> <td>Adapter-Specific Attribute</td> </tr> <tr> <td>Pattern Element Name *</td> <td>filename</td> </tr> <tr> <td>Adapter-Specific Attribute *</td> <td>Custom Attribute</td> </tr> <tr> <td>Attribute Name *</td> <td>1_attachmentName</td> </tr> </table> <p><b>Pattern Variable Replacement</b></p> <table border="1"> <tr> <td>Value Source</td> <td>Adapter-Specific Attribute</td> </tr> <tr> <td>Pattern Element Name *</td> <td>header_host</td> </tr> <tr> <td>Adapter-Specific Attribute *</td> <td>Custom Attribute</td> </tr> <tr> <td>Attribute Name *</td> <td>0_host</td> </tr> </table> <p><b>Pattern Variable Replacement</b></p> <table border="1"> <tr> <td>Value Source</td> <td>Adapter-Specific Attribute</td> </tr> <tr> <td>Pattern Element Name *</td> <td>header_date</td> </tr> <tr> <td>Adapter-Specific Attribute *</td> <td>Custom Attribute</td> </tr> <tr> <td>Attribute Name *</td> <td>0_x_amz_date</td> </tr> </table> <p><b>Pattern Variable Replacement</b></p> <table border="1"> <tr> <td>Value Source</td> <td>Adapter-Specific Attribute</td> </tr> <tr> <td>Pattern Element Name *</td> <td>header_authorization</td> </tr> <tr> <td>Adapter-Specific Attribute *</td> <td>Custom Attribute</td> </tr> <tr> <td>Attribute Name *</td> <td>1_Authorization</td> </tr> </table> <p><b>Pattern Variable Replacement</b></p> <table border="1"> <tr> <td>Value Source</td> <td>Adapter-Specific Attribute</td> </tr> <tr> <td>Pattern Element Name *</td> <td>header_contenthash</td> </tr> <tr> <td>Adapter-Specific Attribute *</td> <td>Custom Attribute</td> </tr> <tr> <td>Attribute Name *</td> <td>0_x_amz_content_sha256</td> </tr> </table> <p><b>Pattern Variable Replacement</b></p> <table border="1"> <tr> <td>Value Source</td> <td>Adapter-Specific Attribute</td> </tr> <tr> <td>Pattern Element Name *</td> <td>header_contentlength</td> </tr> <tr> <td>Adapter-Specific Attribute *</td> <td>Custom Attribute</td> </tr> <tr> <td>Attribute Name *</td> <td>0_content_length</td> </tr> </table> <p><b>Pattern Variable Replacement</b></p> <table border="1"> <tr> <td>Value Source</td> <td>Adapter-Specific Attribute</td> </tr> <tr> <td>Pattern Element Name *</td> <td>header_httpMethod</td> </tr> <tr> <td>Adapter-Specific Attribute *</td> <td>Custom Attribute</td> </tr> <tr> <td>Attribute Name *</td> <td>1_httpMethod</td> </tr> </table>	Value Source	Adapter-Specific Attribute	Pattern Element Name *	filename	Adapter-Specific Attribute *	Custom Attribute	Attribute Name *	1_attachmentName	Value Source	Adapter-Specific Attribute	Pattern Element Name *	header_host	Adapter-Specific Attribute *	Custom Attribute	Attribute Name *	0_host	Value Source	Adapter-Specific Attribute	Pattern Element Name *	header_date	Adapter-Specific Attribute *	Custom Attribute	Attribute Name *	0_x_amz_date	Value Source	Adapter-Specific Attribute	Pattern Element Name *	header_authorization	Adapter-Specific Attribute *	Custom Attribute	Attribute Name *	1_Authorization	Value Source	Adapter-Specific Attribute	Pattern Element Name *	header_contenthash	Adapter-Specific Attribute *	Custom Attribute	Attribute Name *	0_x_amz_content_sha256	Value Source	Adapter-Specific Attribute	Pattern Element Name *	header_contentlength	Adapter-Specific Attribute *	Custom Attribute	Attribute Name *	0_content_length	Value Source	Adapter-Specific Attribute	Pattern Element Name *	header_httpMethod	Adapter-Specific Attribute *	Custom Attribute	Attribute Name *	1_httpMethod	<p><b>HTTP Operation Source</b></p> <table border="1"> <tr> <td>Value Source</td> <td>Adapter-Specific Attribute</td> </tr> <tr> <td>Adapter-Specific Attribute *</td> <td>Custom Attribute</td> </tr> <tr> <td>Attribute Name *</td> <td>1_httpMethod</td> </tr> </table> <p><b>GET Operation</b></p> <p>Expression (Glob) GET</p> <p><b>POST Operation</b></p> <p>Expression (Glob) POST</p> <p><b>PUT Operation</b></p> <p>Expression (Glob) PUT</p> <p><b>DELETE Operation</b></p> <p>Expression (Glob) DELETE</p> <p><b>Request Format</b></p> <table border="1"> <tr> <td>Data Format</td> <td>XML</td> </tr> <tr> <td><input type="checkbox"/> Strip Operation from Message (Outer Element)</td> <td></td> </tr> <tr> <td>Character Set Name *</td> <td>UTF-8</td> </tr> </table> <p><b>Response Format (for Synchronous Messages)</b></p> <table border="1"> <tr> <td>Data Format</td> <td>XML</td> </tr> <tr> <td>Character Set</td> <td>Manual Value</td> </tr> <tr> <td>Character Set Name *</td> <td>UTF-8</td> </tr> </table> <p><b>XML/JSON Namespace Mapping</b></p> <p><input type="checkbox"/> Enable Namespace Mapping</p> <p><b>Additional HTTP Headers</b></p> <table border="1"> <thead> <tr> <th>Header Name</th> <th>Value Pattern</th> </tr> </thead> <tbody> <tr> <td>Host</td> <td>{header_host}</td> </tr> <tr> <td>x-amz-date</td> <td>{header_date}</td> </tr> <tr> <td>Authorization</td> <td>{header_authorization}</td> </tr> <tr> <td>x-amz-content-sha256</td> <td>{header_contenthash}</td> </tr> <tr> <td>Content-Length</td> <td>{header_contentlength}</td> </tr> <tr> <td>x-amz-storage-class</td> <td>REDUCED_REDUNDANCY</td> </tr> </tbody> </table>	Value Source	Adapter-Specific Attribute	Adapter-Specific Attribute *	Custom Attribute	Attribute Name *	1_httpMethod	Data Format	XML	<input type="checkbox"/> Strip Operation from Message (Outer Element)		Character Set Name *	UTF-8	Data Format	XML	Character Set	Manual Value	Character Set Name *	UTF-8	Header Name	Value Pattern	Host	{header_host}	x-amz-date	{header_date}	Authorization	{header_authorization}	x-amz-content-sha256	{header_contenthash}	Content-Length	{header_contentlength}	x-amz-storage-class	REDUCED_REDUNDANCY
Value Source	Adapter-Specific Attribute																																																																																								
Pattern Element Name *	filename																																																																																								
Adapter-Specific Attribute *	Custom Attribute																																																																																								
Attribute Name *	1_attachmentName																																																																																								
Value Source	Adapter-Specific Attribute																																																																																								
Pattern Element Name *	header_host																																																																																								
Adapter-Specific Attribute *	Custom Attribute																																																																																								
Attribute Name *	0_host																																																																																								
Value Source	Adapter-Specific Attribute																																																																																								
Pattern Element Name *	header_date																																																																																								
Adapter-Specific Attribute *	Custom Attribute																																																																																								
Attribute Name *	0_x_amz_date																																																																																								
Value Source	Adapter-Specific Attribute																																																																																								
Pattern Element Name *	header_authorization																																																																																								
Adapter-Specific Attribute *	Custom Attribute																																																																																								
Attribute Name *	1_Authorization																																																																																								
Value Source	Adapter-Specific Attribute																																																																																								
Pattern Element Name *	header_contenthash																																																																																								
Adapter-Specific Attribute *	Custom Attribute																																																																																								
Attribute Name *	0_x_amz_content_sha256																																																																																								
Value Source	Adapter-Specific Attribute																																																																																								
Pattern Element Name *	header_contentlength																																																																																								
Adapter-Specific Attribute *	Custom Attribute																																																																																								
Attribute Name *	0_content_length																																																																																								
Value Source	Adapter-Specific Attribute																																																																																								
Pattern Element Name *	header_httpMethod																																																																																								
Adapter-Specific Attribute *	Custom Attribute																																																																																								
Attribute Name *	1_httpMethod																																																																																								
Value Source	Adapter-Specific Attribute																																																																																								
Adapter-Specific Attribute *	Custom Attribute																																																																																								
Attribute Name *	1_httpMethod																																																																																								
Data Format	XML																																																																																								
<input type="checkbox"/> Strip Operation from Message (Outer Element)																																																																																									
Character Set Name *	UTF-8																																																																																								
Data Format	XML																																																																																								
Character Set	Manual Value																																																																																								
Character Set Name *	UTF-8																																																																																								
Header Name	Value Pattern																																																																																								
Host	{header_host}																																																																																								
x-amz-date	{header_date}																																																																																								
Authorization	{header_authorization}																																																																																								
x-amz-content-sha256	{header_contenthash}																																																																																								
Content-Length	{header_contentlength}																																																																																								
x-amz-storage-class	REDUCED_REDUNDANCY																																																																																								

### Processing Sequence

Number	Module Name	Type	Module Key
1	AF_Modules/StrictXml2PlainBean	Local Enterprise Bean	transform
2	AF_Modules/DynamicConfigurationBean	Local Enterprise Bean	sender
3	AF_Modules/RequestResponseBean	Local Enterprise Bean	req_resp
4	ReplaceString	Local Enterprise Bean	replace
5	sap.com/com.sap.aii.adapter.rest.app/RESTAdapterBean	Local Enterprise Bean	rest
6	AF_Modules/DynamicConfigurationBean	Local Enterprise Bean	receiver
7	AF_Modules/ResponseOnewayBean	Local Enterprise Bean	resp_oneway

### Module Configuration

Module Key	Parameter Name	Parameter Value
receiver	key.1	read http://sap.com/xi/XI/System/REST 1_fileName
receiver	value1	module.http://sap.com/xi/XI/System/REST
replace	param1	null;<?xml version="1.0" encoding="UTF-8"?> <ns0:InitiateMulti
replace	separator	;
req_resp	passThrough	true
resp_oneway	interface	DVRManagementFee_InitialMPUpload_Res_ASync_Out
resp_oneway	interfaceNamespace	[REDACTED]
resp_oneway	replaceInterface	true
rest	setIgnoreEmptyArrayValues	false
rest	setIgnoreElements	ManagementFeeMsg
sender	key.1	write http://sap.com/xi/XI/System/REST 1_fileName
sender	key.2	write http://sap.com/xi/XI/System/REST Content-Type
sender	value.1	module.http://sap.com/xi/XI/System/REST
sender	value.2	text/csv;charset=UTF-8
transform	Records.endSeparator	'\n'
transform	Records.fieldNames	text
transform	Records.fieldSeparator	'\n'
transform	Transform.ContentType	text/plain
transform	recordTypes	Records

## Channel 2 – Request to Upload Part

**Display Communication Channel**

Communication Channel [REDACTED]  
 Party [REDACTED]  
 Communication Component [REDACTED]  
 Description [REDACTED]

Parameters Identifiers Module

Adapter Type \* REST http://sap.com/

Sender  Receiver

Transport Protocol \* HTTP

Message Protocol \* REST

Adapter Engine \* Central Adapter Engine

General REST URL REST Operation Data Format Operation Rules Response

**Warning: Parameterizing HTTP Host or Port is a security risk (see SAP Note: 2174651)**

URL Pattern \* {URL}

Follow Server Redirects on HTTP GET calls

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	URL
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	1_URL

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_host
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	0_host

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_date
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	0_x_amz_date

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_authorization
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	1_Authorization

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_contenthash
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	0_X_amz_content_sha256

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_contentlength
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	0_content_length

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_httpMethod
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	1_httpMethod

**HTTP Operation Source**

Value Source	Adapter-Specific Attribute
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	1_httpMethod

**GET Operation**

Expression (Glob)	GET
-------------------	-----

**POST Operation**

Expression (Glob)	POST
-------------------	------

**PUT Operation**

Expression (Glob)	PUT
-------------------	-----

**DELETE Operation**

Expression (Glob)	DELETE
-------------------	--------

**Request Format**

Data Format	JSON
<input type="checkbox"/> Strip Operation from Message (Outer Element)	
Character Set Name *	UTF-8
<input type="checkbox"/> Convert XML Payload to JSON	

**Response Format (for Synchronous Messages)**

Data Format	KML
Character Set	HTTP Header

**XML/JSON Namespace Mapping**

<input type="checkbox"/> Enable Namespace Mapping	
---	--

**Additional HTTP Headers**

Header Name	Value Pattern
Host	{header_host}
x-amz-date	{header_date}
Authorization	{header_authorization}
x-amz-content-sha256	{header_contenthash}
Content-Length	{header_contentlength}
Content-Type	text/csv

**Processing Sequence**

Number	Module Name	Type	Module Key
1	AF_Modules/MessageTransformBean	Local Enterprise Bean	xmltocsv
2	AF_Modules/DynamicConfigurationBean	Local Enterprise Bean	sender
3	AF_Modules/RequestResponseBean	Local Enterprise Bean	req_resp
4	sap.com/com.sap.aii.adapter.restapp/RESTAdapterBean	Local Enterprise Bean	rest
5	AF_Modules/DynamicConfigurationBean	Local Enterprise Bean	receiver
6	AF_Modules/ResponseOnewayBean	Local Enterprise Bean	resp_oneway

Module Configuration		
Module Key	Parameter Name	Parameter Value
receiver	key.1	read http://sap.com/xi/XI/System/REST 1_fileName
receiver	value.1	module http://sap.com/xi/XI/System/REST
req_resp	passThrough	true
resp_oneway	interface	DVRManagementFee_MPUploadPart_Req_ASync_Out
resp_oneway	interfaceNamespace	[REDACTED]
resp_oneway	replaceInterface	true
rest	xml.processFieldNames	fromConfiguration
sender	key.1	write http://sap.com/xi/XI/System/REST 1_fileName
sender	value.1	module http://sap.com/xi/XI/System/REST
xmitocsv	Transform Class	com.sap.aii.messaging.adapter.Conversion
xmitocsv	Transform.ContentType	text/csv;charset=utf-8
xmitocsv	xml.addHeaderLine	0
xmitocsv	xml.conversionType	SimpleXML2Plain
xmitocsv	xml.documentName	ManagementFeeMsg
xmitocsv	xml.documentElementNamespace	urn:leaseplan.com:aux:unifil:dvr
xmitocsv	xml.fieldSeparator	'\n'
xmitocsv	xml.recordsetStructure	Records

### Channel 3 – Request to Complete Multipart Upload

<p><b>HTTP Operation Source</b></p> <p>Value Source <input checked="" type="checkbox"/> Adapter-Specific Attribute <input type="checkbox"/> Custom Attribute  <input type="checkbox"/> Attribute Name * <u>1_httpMethod</u></p> <p><b>GET Operation</b>  <input type="checkbox"/> Expression (Glob) <u>GET</u></p> <p><b>POST Operation</b>  <input type="checkbox"/> Expression (Glob) <u>POST</u></p> <p><b>PUT Operation</b>  <input type="checkbox"/> Expression (Glob) <u>PUT</u></p> <p><b>DELETE Operation</b>  <input type="checkbox"/> Expression (Glob) <u>DELETE</u></p>	<p><b>Request Format</b></p> <p>Data Format <input checked="" type="checkbox"/> JSON <input type="checkbox"/> Strip Operation from Message (Outer Element)  <input type="checkbox"/> Character Set Name * <u>UTF-8</u> <input type="checkbox"/> Convert XML Payload to JSON</p> <p><b>Response Format (for Synchronous Messages)</b></p> <p>Data Format <input type="checkbox"/> KML <input checked="" type="checkbox"/> HTTP Header  <input type="checkbox"/> Character Set <u>HTTP Header</u></p> <p><b>XML/JSON Namespace Mapping</b>  <input type="checkbox"/> Enable Namespace Mapping</p>	<p><b>Additional HTTP Headers</b></p> <table border="1"> <thead> <tr> <th>Header Name</th> <th>Value Pattern</th> </tr> </thead> <tbody> <tr> <td>x-amz-date</td> <td>{header_date}</td> </tr> <tr> <td>Authorization</td> <td>{header_authorization}</td> </tr> <tr> <td>x-amz-content-sha256</td> <td>{header_contenthash}</td> </tr> <tr> <td>Host</td> <td>{header_host}</td> </tr> <tr> <td>Content-Length</td> <td>{header_contentlength}</td> </tr> <tr> <td>Content-Type</td> <td>application/xml</td> </tr> </tbody> </table>	Header Name	Value Pattern	x-amz-date	{header_date}	Authorization	{header_authorization}	x-amz-content-sha256	{header_contenthash}	Host	{header_host}	Content-Length	{header_contentlength}	Content-Type	application/xml
Header Name	Value Pattern															
x-amz-date	{header_date}															
Authorization	{header_authorization}															
x-amz-content-sha256	{header_contenthash}															
Host	{header_host}															
Content-Length	{header_contentlength}															
Content-Type	application/xml															

**Display Communication Channel**

Communication Channel	[REDACTED]
Party	[REDACTED]
Communication Component	[REDACTED]
Description	[REDACTED]

**Parameters** **Identifiers** **Module**

Adapter Type \* REST http://sap.com/xi/XIS  
 Sender  Receiver

Transport Protocol \* HTTP  
Message Protocol \* REST  
Adapter Engine \* Central Adapter Engine

**General** **REST URL** **REST Operation** **Data Format** **Operation Rules** **Response Data**

**Warning:** Parameterizing HTTP Host or Port is a security risk (see SAP Note: 2174651)

URL Pattern \* [URL]  
 Follow Server Redirects on HTTP GET calls

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	URL
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	1_URL

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_host
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	0_host

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_date
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	0_x_amz_date

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_authorization
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	1_Authorization

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_contenthash
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	0_x_amz_content_sha256

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_contentlength
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	0_content_length

**Pattern Variable Replacement**

Value Source	Adapter-Specific Attribute
Pattern Element Name *	header_httpMethod
Adapter-Specific Attribute *	Custom Attribute
Attribute Name *	1_httpMethod

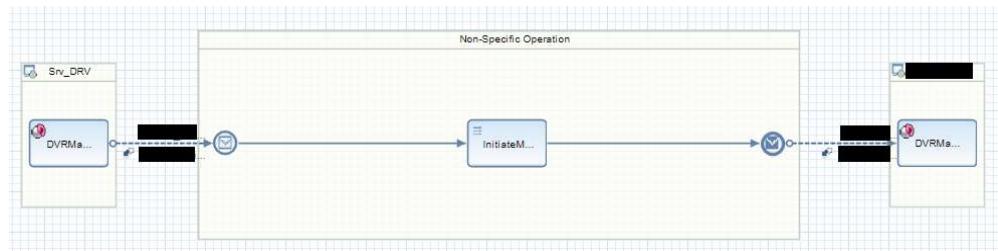
### Processing Sequence

Number	Module Name	Type	Module Key
1	localejbs/ReplaceString	Local Enterprise Bean	replace
2	AF_Modules/DynamicConfigurationBean	Local Enterprise Bean	sender
3	AF_Modules/RequestResponseBean	Local Enterprise Bean	req_resp
4	sap.com:com.sap.aii.adapter.rest.app/RESTAdapterBean	Local Enterprise Bean	rest
5	AF_Modules/DynamicConfigurationBean	Local Enterprise Bean	receiver
6	AF_Modules/ResponseOnewayBean	Local Enterprise Bean	resp_oneway

Module Configuration		
Module Key	Parameter Name	Parameter Value
receiver	key.1	read http://sap.com/xi/XI/System/REST 1_fileName
receiver	value.1	module.http://sap.com/xi/XI/System/REST
replace	param2	ns0:#emptyString
replace	separator	#
req_resp	passThrough	true
resp_oneway	interface	DVRManagementFee_CompleteMPUploadPart_Res_ASync_Out
resp_oneway	interfaceNamespace	[REDACTED]
resp_oneway	replaceInterface	true
sender	key.1	write http://sap.com/xi/XI/System/REST 1_fileName
sender	value.1	module.http://sap.com/xi/XI/System/REST

### 3. Response Receive Scenario – Initiate Multipart Upload

#### a. ICO



#### b. Sender Channel

Display Communication Channel

Communication Channel	[REDACTED]			
Party	[REDACTED]			
Communication Component	[REDACTED]			
Description				
Parameters	Identifiers	Module		
Adapter Type *	REST	http://sap.com/xi/XI/System		
<input checked="" type="radio"/> Sender	<input type="radio"/> Receiver			
Transport Protocol *	HTTP			
Message Protocol *	REST			
Adapter Engine *	Central Adapter Engine			
General	Channel Selection	REST Resources	REST Operation	Operation Determination
Input Message Format				
Data Format	XML			
Character Set	HTTP Content-Type Header			
Quality of Service				
Quality of Service	Exactly Once			

#### Processing Sequence

Number	Module Name	Type	Module Key
1	AF_Modules/XMLAnonymizerBean	Local Enterprise Bean	0
2	CallSapAdapter	Local Enterprise Bean	rest

#### Module Configuration

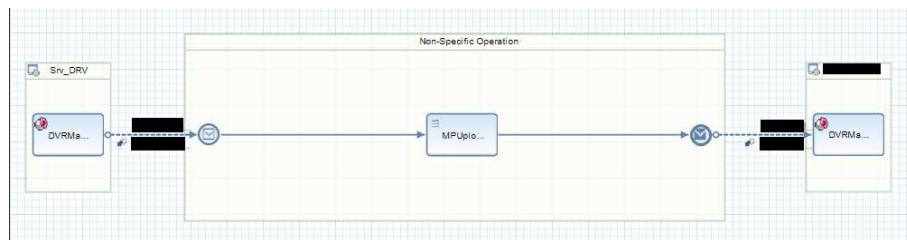
Module Key	Parameter Name	Parameter Value
0	anonymizer.acceptNamespaces	http://s3.amazonaws.com/doc/2006-03-01/ns0

c. Receiver Channel

Content Conversion Parameters	
Recordset Structure *	
Record	
Name	Value
Record.addHeaderLine	0
Record.fieldSeparator	
Record.endSeparator	'\n'

4. Response Receive Scenario – Upload Part

a. ICO



b. Sender Channel

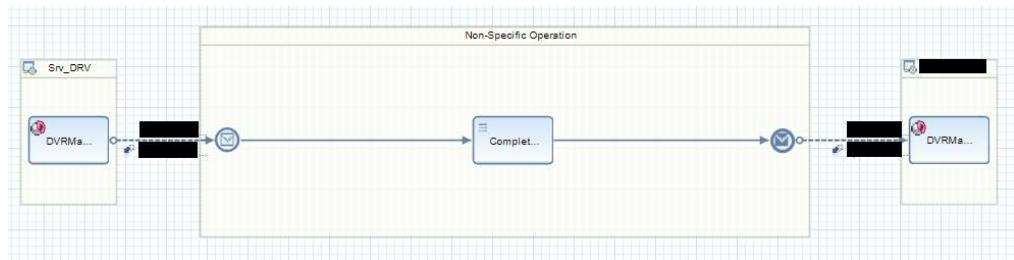
XI Dynamic Attribute	
Dynamic Attribute	Value Source
REST Operation (operation)	HTTP Operation

c. Receiver Channel

Name	Value
Record.addHeaderLine	0
Record.fieldSeparator	
Record.endSeparator	'\n'

5. Response Receive Scenario – Complete Multipart Upload

a. ICO



b. Sender Channel

Quality of Service	
Quality of Service	Exactly Once

c. Receiver Channel

Name	Value
Record.addHeaderLine	0
Record.fieldSeparator	
Record.endSeparator	'nl'

I have provided only the skeleton and the basic information to get this requirement. Because there is routing to multiple interfaces in the request part, I find it difficult to implement EOIO quality in this scenario. Hence I had to introduce a mandatory delay of 600 seconds. This was calculated based on the time it took to upload one part of the file, which was 6-7 minutes.

Also, after writing the Complete Multipart Upload, the metadata file is archived. This is a sample file to be uploaded.

File1.csv	8/7/2020 3:43 PM	Microsoft Excel Co...	83,086 KB
-----------	------------------	-----------------------	-----------

A file of this size is split into multiple files of size slightly larger than 5 MB.

20200807-183150_Temp_Metadata	8/7/2020 6:31 PM	Text Document	2 KB
20200807-182148-654_Split	8/7/2020 6:21 PM	Microsoft Excel Co...	715 KB
20200807-181147-788_Split	8/7/2020 6:11 PM	Microsoft Excel Co...	5,856 KB
20200807-180146-679_Split	8/7/2020 6:01 PM	Microsoft Excel Co...	5,854 KB
20200807-175144-806_Split	8/7/2020 5:51 PM	Microsoft Excel Co...	5,854 KB
20200807-174143-595_Split	8/7/2020 5:41 PM	Microsoft Excel Co...	5,857 KB
20200807-173142-292_Split	8/7/2020 5:31 PM	Microsoft Excel Co...	5,858 KB
20200807-172140-984_Split	8/7/2020 5:21 PM	Microsoft Excel Co...	5,856 KB
20200807-171139-878_Split	8/7/2020 5:11 PM	Microsoft Excel Co...	5,855 KB
20200807-170138-738_Split	8/7/2020 5:01 PM	Microsoft Excel Co...	5,854 KB
20200807-165137-529_Split	8/7/2020 4:51 PM	Microsoft Excel Co...	5,854 KB
20200807-164136-064_Split	8/7/2020 4:41 PM	Microsoft Excel Co...	5,852 KB
20200807-163134-891_Split	8/7/2020 4:31 PM	Microsoft Excel Co...	5,855 KB
20200807-162133-941_Split	8/7/2020 4:21 PM	Microsoft Excel Co...	5,854 KB
20200807-161132-967_Split	8/7/2020 4:11 PM	Microsoft Excel Co...	5,852 KB
20200807-160132-045_Split	8/7/2020 4:01 PM	Microsoft Excel Co...	5,853 KB
20200807-155131-185_Split	8/7/2020 3:51 PM	Microsoft Excel Co...	1 KB
20200807-154327-318_	8/7/2020 3:43 PM	Microsoft Excel Co...	83,086 KB

The metadata file contains the following data.

```
CREATEMULTIPARTUPLOAD|BUCKET: [REDACTED] |KEY: [REDACTED] |UPLOADID:PGZ0wZNmmPsRlyo7N1
UPLOADPART|ETAG:"e6b4f2e14139cc51f97aa71cc07cccc8"
UPLOADPART|ETAG:"230850302f0d58fc1f2bd98df3e92e71"
UPLOADPART|ETAG:"a5f3351f1d076136f2b7a80bb7573d"
UPLOADPART|ETAG:"1ff5aca1a0ea9a49cf29891d0b1eb34f"
UPLOADPART|ETAG:"bfacbbe1c53a6248eca4a16ef03f6829"
UPLOADPART|ETAG:"104dc1017e693b0d2b738063f3d341db"
UPLOADPART|ETAG:"a0901397e97a98e13848bb9da232c38"
UPLOADPART|ETAG:"84b7b275ec0fa8bae7002bbbd49363c2"
UPLOADPART|ETAG:"22d640fde392acf61045f7ae17827de7"
UPLOADPART|ETAG:"7d48879825fb8a9d6dafa3c3c5c457a"
UPLOADPART|ETAG:"b5c721468e0ce5eda93b1d11eaet783f5"
UPLOADPART|ETAG:"f8746f3abcf826dd270f11a5427b8f0"
UPLOADPART|ETAG:"1e2273cc79f5d0ea20d29039fa505d2"
UPLOADPART|ETAG:"143c3647bbd12370054f8a62e6b3b40a"
UPLOADPART|ETAG:"1731513ecbadee9ae65caab8e31483fb"
COMPLETEMULTIPARTOUTPUT|[REDACTED]"7d316b20ca2be814654fb462d494fb3b-15"
```

The above is written into this file by the 3 response mappings.

### **Conclusion:**

This way we don't have to worry about files of any size to be uploaded into AWS s3. But there are a few points to be considered here:

1. SAP PO has no provision for greater than or lesser than in the condition editor. This prevented us from routing the file to a single chunk or multipart upload based on file size.
2. AWS has no central repo for the structures they use. It would've been easier for us that way to just download and use them. There was quite some time spent on resolving namespace issues.
3. Java mapping could be used to eliminate the need for separate mappings to resolve namespace issues. Also, EOIO quality for file sender to multiple receivers is possible with java mapping.
4. The performance of this scenario seems exponential to the number of parts it divides into. This could be optimized if we process files of different sizes based on threshold values. For example: files of size 5 MB or lesser can be uploaded in a single chunk. Files larger than that could be uploaded using multipart.
5. The absence of EOIO necessitates that only one file be present in the folder at a time. In case of multiple files, there will be multiple metadata files created and the process will find writing back the correct ETag values to the correct metadata file confusing.

Feel free to suggest any improvements/suggestions/feedback because this was done due to a time crunch. There can be many more ways to optimize this.

Lastly, I've listed the links/people I referred to during the development of this requirement.

3. <https://blogs.sap.com/2019/07/18/idoc-rest-async-sync-scenario-approach-with-aleaud/>
4. <https://blogs.sap.com/2019/05/31/integrating-amazon-simple-storage-service-amazon-s3-and-sap-ecc-v6.0-via-sap-pi-v7.5-using-aws-signature-v5-and-signing-algorithm-hmac-sha256/>
5. <https://docs.aws.amazon.com/AmazonS3/latest/dev/mpuoverview.html>
6. <https://github.com/pnathan01/AWSSAPMultipartUploadLibrary>
7. <https://wiki.scn.sap.com/wiki/display/XI/Where+to+get+the+libraries+for+XI+development>
8. <https://launchpad.support.sap.com/#/notes/0001763011>
9. Robbie Cooray from AWS – He was my go-to person for valuable inputs and great ideas. Thanks Robbie! 😊