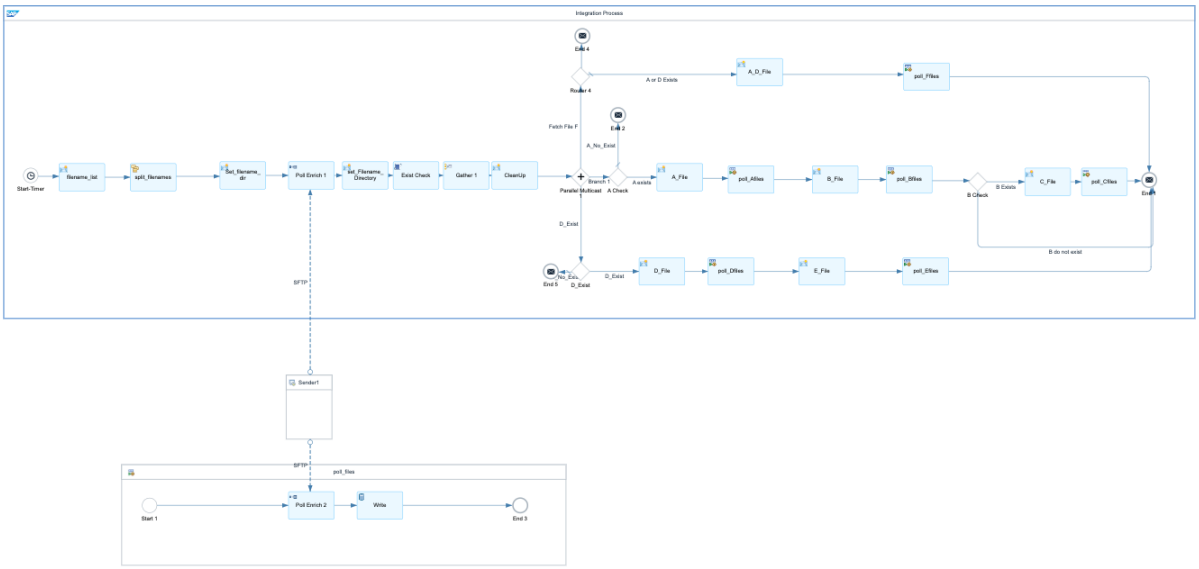
Use Case



Overview:

This Iflow is specifically designed to handle a use case involving nested files with dependencies and specific conditions for their retrieval and processing. The workflow follows these detailed steps:

1. **Retrieve File A**: The process starts by retrieving File A.

2. **Retrieve File B**: File B is retrieved only if File A was successfully retrieved and exists.

3. **Retrieve File C**: File C is retrieved only if File B was successfully retrieved and exists.

4. **Retrieve File D**: File D is retrieved independently of the existence of Files A, B, or C.

5. **Retrieve File E**: File E is retrieved only if File D was successfully retrieved and exists.

6. **Retrieve File F**: File F is retrieved if either File A or File D was successfully retrieved and exists.

This Iflow ensures that the sequence and dependencies of file retrieval are strictly followed to meet specific business requirements. The design effectively uses parallel multicast and routers to manage the conditions and dependencies for processing different files, ensuring a robust and logical workflow.

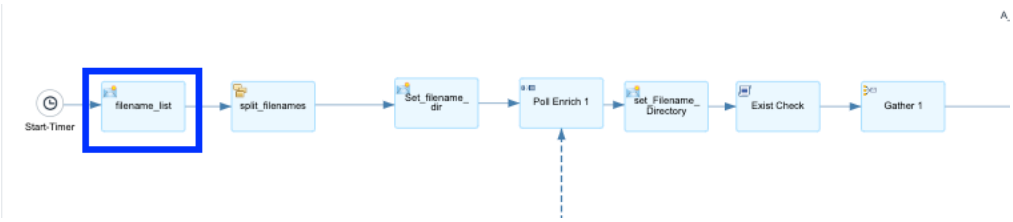
This image illustrates a series of steps in an Iflow. Here is a detailed description:

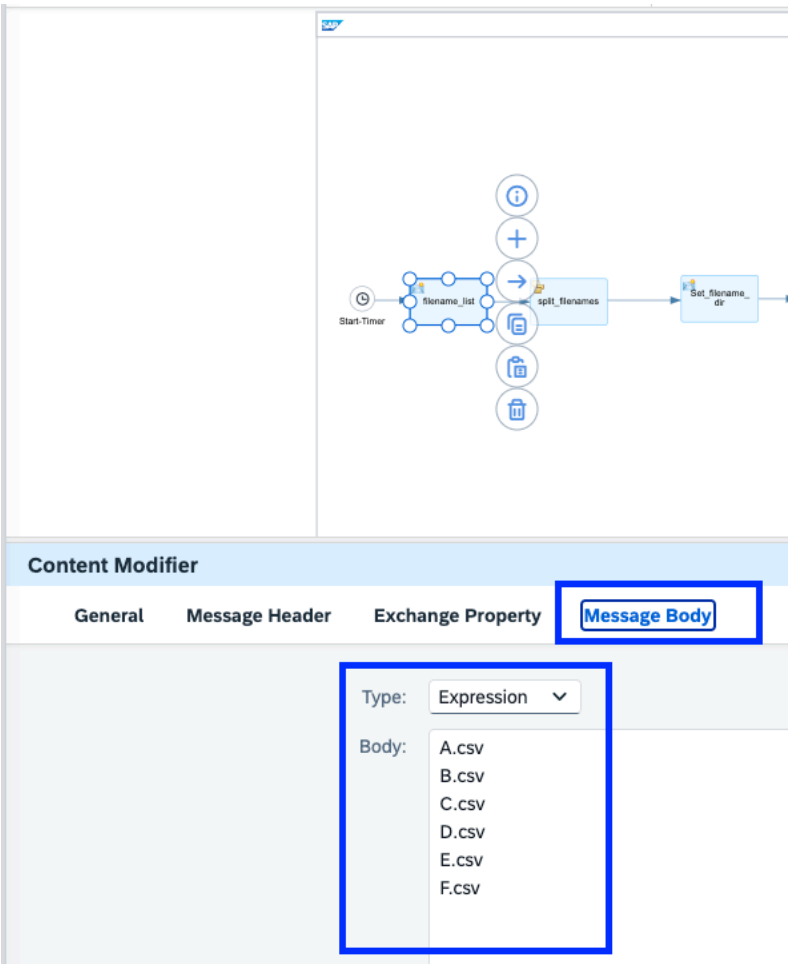
1. **filename_list**:

- **Type**: Content Modifier

- **Function**: Sets up a list of filenames to be processed.

- **Content**: Lists filenames such as A.csv, B.csv, C.csv, etc.





2. **split_filenames**:

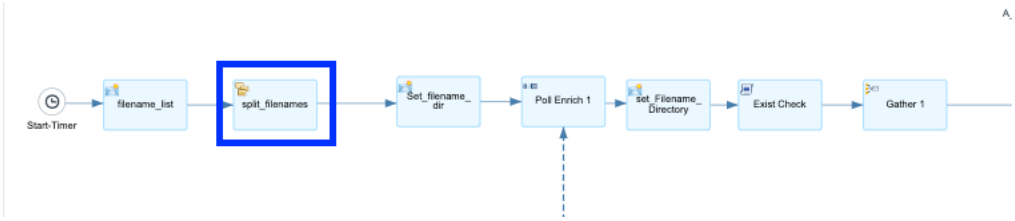
- **Function**: Splits the list of filenames into individual filenames for separate processing.

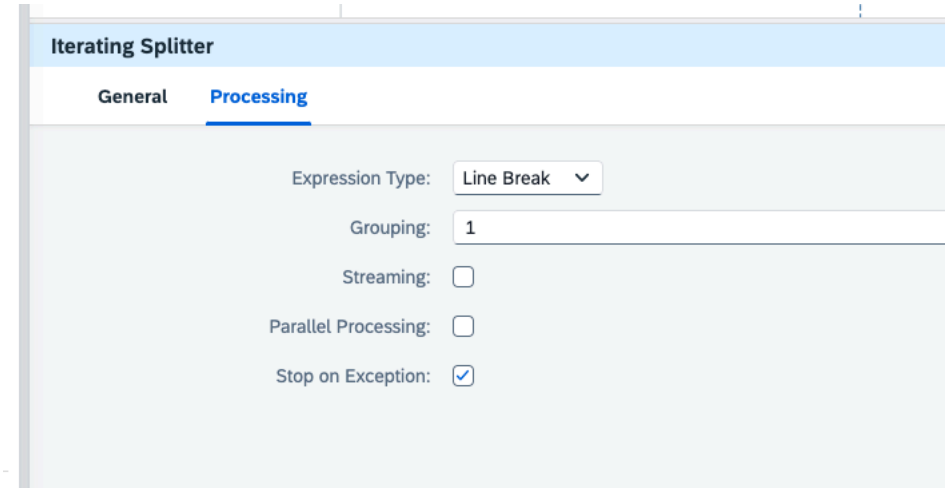
Settings in the Processing Tab:

- Expression Type: Set to "Line Break," meaning the filename list is split based on line breaks.

- Grouping: Set to 1, meaning each split will contain one filename.
- Streaming: Unchecked, indicating that streaming is not enabled.
- Parallel Processing: Unchecked, indicating that parallel processing is not enabled.
- Stop on Exception: Checked, meaning the process will stop if an exception occurs during processing.

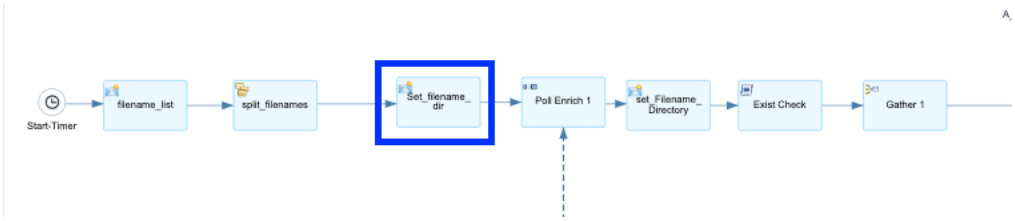
These settings ensure that the filename list is split into individual filenames based on line breaks, with each filename processed at a time. If an exception occurs during processing, the flow will stop to prevent further errors.

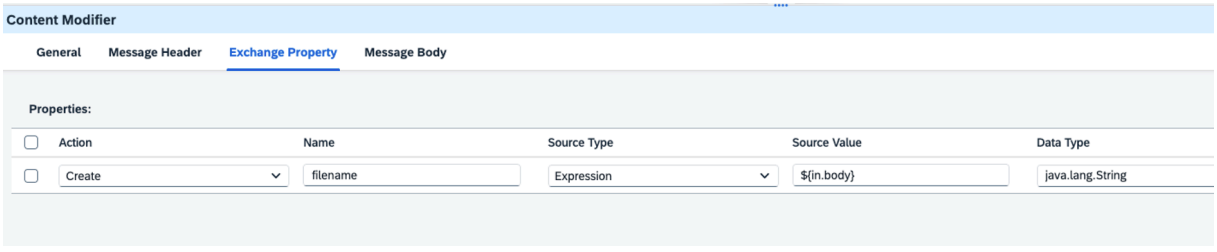




3. **Set_filename_dir**:

- **Function**: This step comes after the `split_filenames` step and is used to set an exchange property for the filename.





Settings in the Exchange Property Tab:

- Action: Selected as "Create," indicating the creation of a new exchange property.
- Name: Set to "filename," indicating that the name of the exchange property will be "filename."
- Source Type: Selected as "Expression," indicating that the source value is an expression.
- Source Value: Set to `${in.body}`, indicating that the source value is the content of the message body, i.e., the current filename being processed.
- Data Type: Set to `java.lang.String`, indicating that the data type is a string.

These settings are used to extract the current filename from the message body and create an exchange property named "filename." This allows for easier reference and processing of the filename in subsequent steps.

4. **Poll Enrich 1**:

- **Function**: Polls to retrieve detailed information about the files, ensuring their existence and gathering necessary metadata.

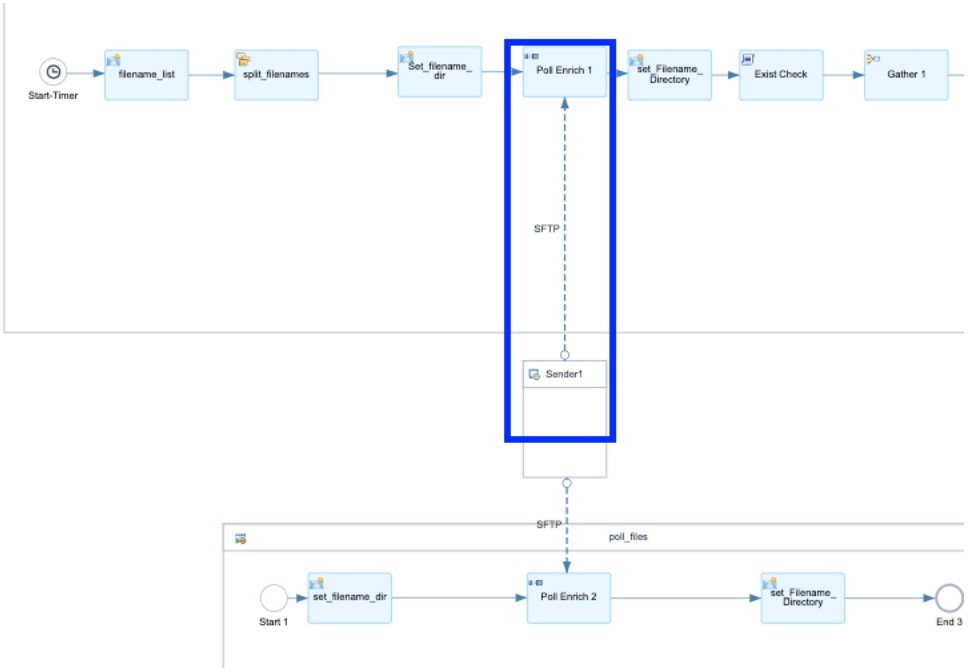
Settings in the Source Tab:

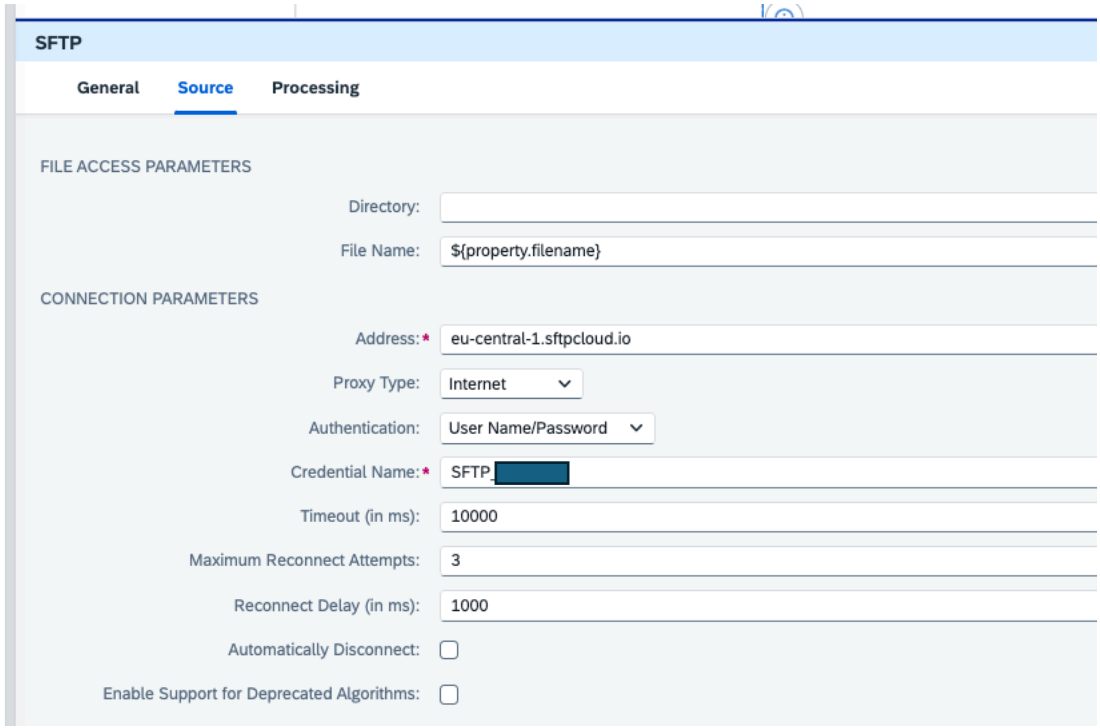
File Access Parameters:

- Directory: (left blank) - The directory on the SFTP server where the files are located.
- File Name: `${property.filename}` - The filename to be retrieved, dynamically set using the exchange property `filename` created earlier.

Connection Parameters:

- Address: `eu-central-1.sftpcloud.io` - The address of the SFTP server.
- Proxy Type: `Internet` - The type of proxy used to connect to the SFTP server.
- Authentication: `User Name/Password` - The method of authentication for accessing the SFTP server.
- Credential Name: `SFTP_IXXXXX` - The name of the credential used for authentication.
- Timeout (in ms): `10000` - The connection timeout set to 10,000 milliseconds.
- Maximum Reconnect Attempts: `3` - The maximum number of attempts to reconnect if the initial connection fails.



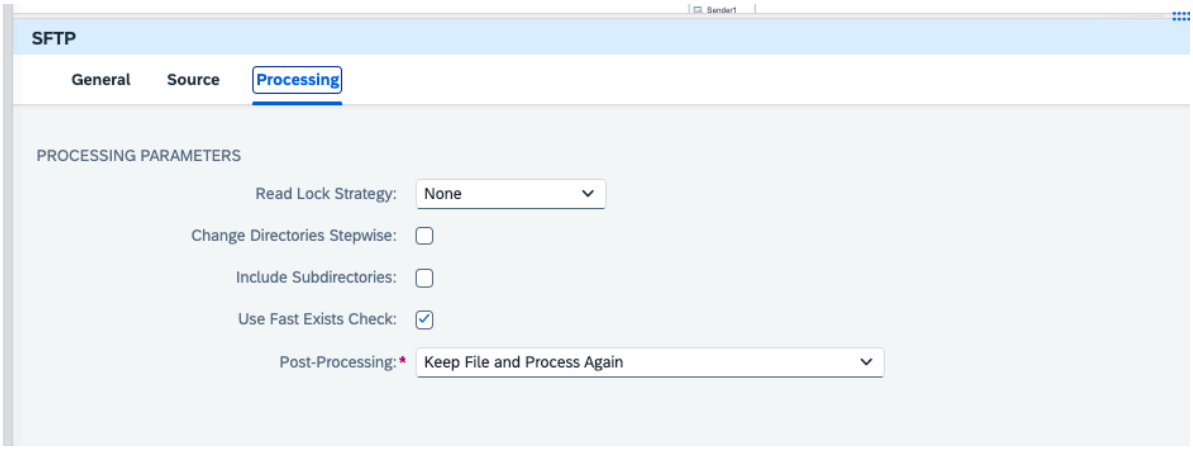


SFTP Step - Processing Tab:

Processing Parameters:

- Read Lock Strategy: Set to "None," indicating that no read lock strategy is applied during file processing.
- Change Directories Stepwise: Unchecked, meaning the process will not change directories step by step.
- Include Subdirectories: Unchecked, indicating that subdirectories are not included in the file processing.

- Use Fast Exists Check: Checked, enabling the fast check to determine if a file exists.
- Post-Processing: Set to "Keep File and Process Again," indicating that after processing, the file will be kept and can be processed again.

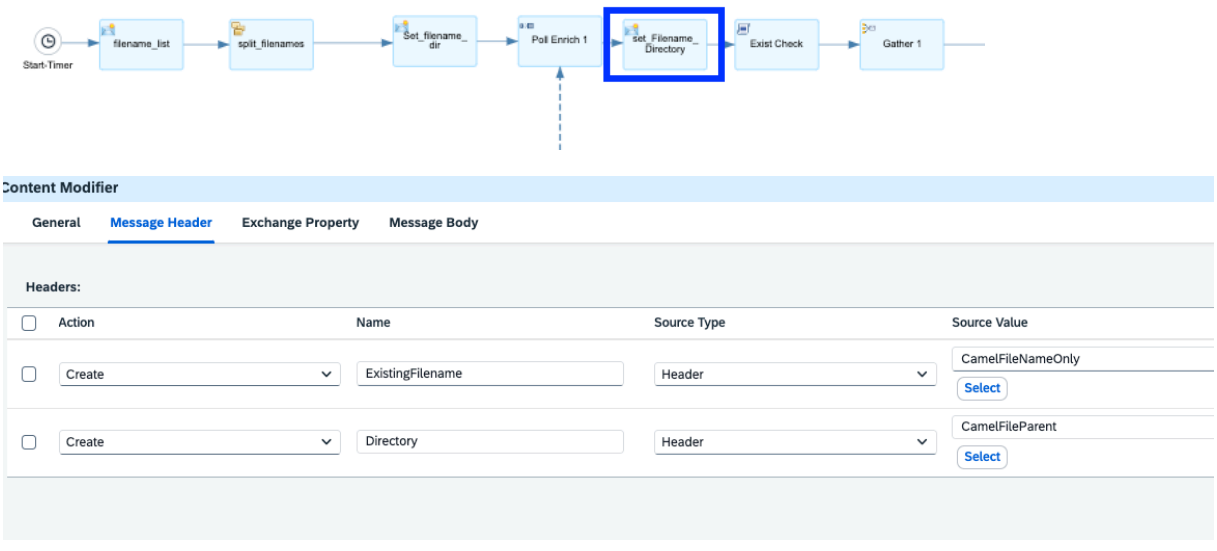


5. **Set_Filename_Directory**:

- **Function**: This is used to modify message headers for subsequent processing.

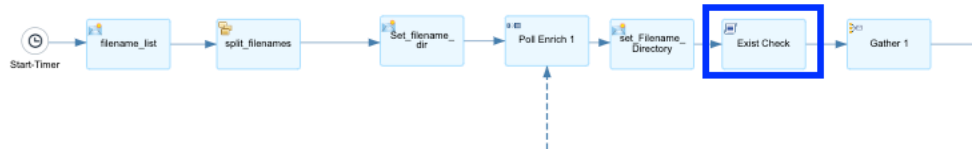
Message Header Tab Settings:

- Headers: Set to "Create," indicating that new headers are being created.
- Name: The first header is named "ExistingFilename." The second header is named "Directory."
- Source Type: Both headers have the source type set to "Header," meaning the values are taken from existing message headers.
- Source Value: For "ExistingFilename," the source value is set to `CamelFileNameOnly`, which extracts only the filename from the existing headers. For "Directory," the source value is set to `CamelFileParent`, which extracts the parent directory from the existing headers.



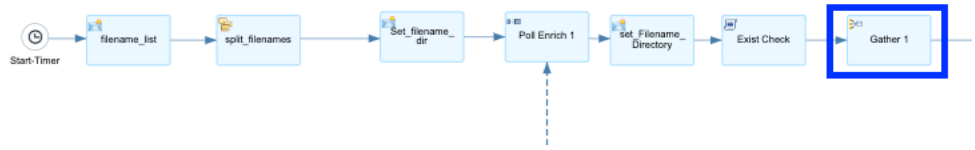
6. **Exist Check**:

- **Function**: The "Groovy Script" step named "Exist Check" is used to execute custom Groovy code within the Iflow. This script will typically be used to check the existence of files or directories as part of the integration process, ensuring that the necessary files are available before proceeding to subsequent steps.



7. **Gather 1**:

- **Function**: Aggregates the processing results, preparing to enter the subsequent multicast and routing stages.



This image shows a detailed part of the Iflow focusing on the parallel multicast and routing logic. Here is the description:

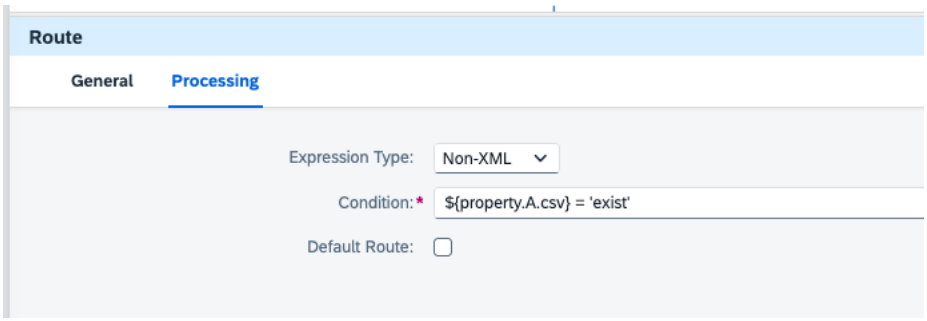
1**Parallel Multicast**:

- **Function**: Splits the flow into multiple branches for parallel processing.

2. **Branch 1**:

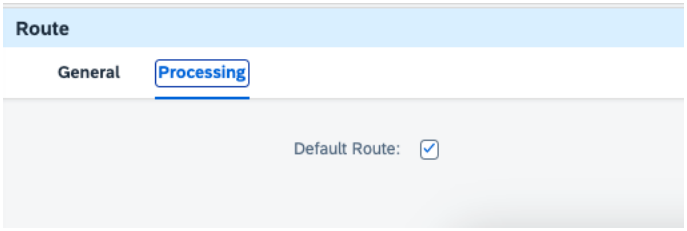
- **Router 1**:

- **Condition**: Checks the existence of File A (`A_Exist`).



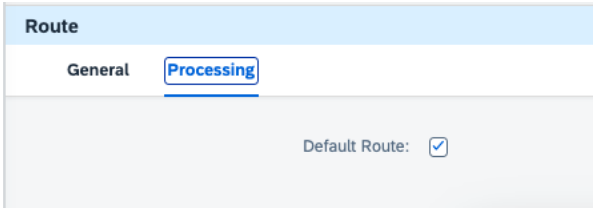
- **poll_AFiles**: If File A exists, this step polls for related files.

- **End 2**: If File A does not exist (`A_No_Exist`), the flow ends here.



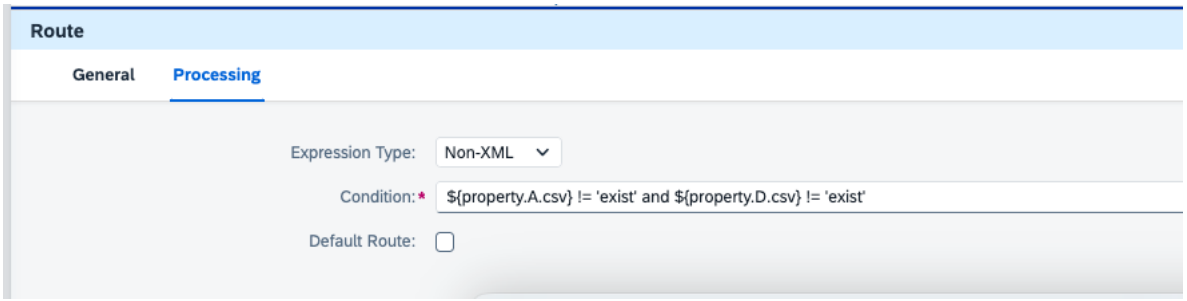
- **Router 4**:

- **Condition**: Checks the existence of either File A or File D (`A_D_Exist`).



- **poll_FFiles**: If either File A or File D exists, this step polls for related files.

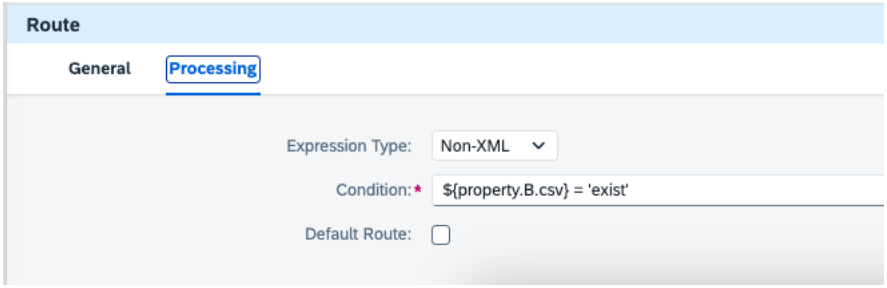
- **End 4**: If neither File A nor File D exists, the flow ends here.



3. **Branch 2**:

- **Router 2**:

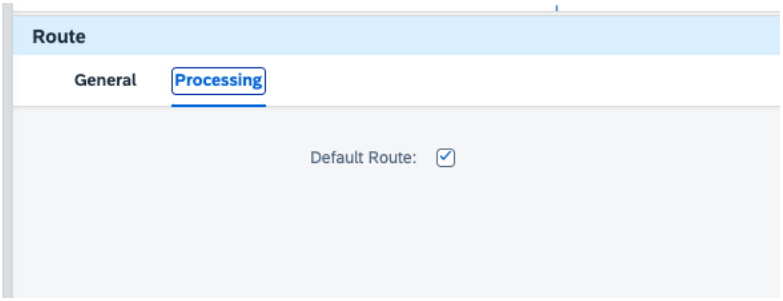
- **Condition**: Checks the existence of File B (`B_Exist`).



- B_Exist: If File B exists, the flow proceeds to poll_BFiles and poll_CFiles.
- B_No_Exist: If File B does not exist, the flow transport only AFile to the End1.

4. **Branch 3**:

- **Condition**: Checks the existence of File D (`D_Exist`).



- **poll_DFiles**: If File D exists, this step polls for related files.

- **poll_EFiles**: After polling for D files, this step checks for the existence of File E and polls accordingly.

- **End 5**: If File D does not exist (`D_No_Exist`), the flow ends here.