**Group Hierarchy**

# **GCS group**

Main process group that contains 5 groups. Each of the process groups is responsible for something. For example, “**Data\_Processing**” group is responsible for all data consuming from **Kafka**, data transformations and storing the data to storage.

# **Data\_Processing**

## Get\_Data\_And\_Set\_Dest

Group that contains flow which consume data from **Kafka** and create “flow file” to work with, updates type of the “flow file” and get information like **topic description**, “**\_timeTag**” and “**source**” from “flow file’s” content.

**Processors:**

* Consume\_Data: Consumes data from Kafka and creates flow file based in consumed data.
* Update\_Type\_And\_Get\_Topic: Updates “**mime.type**” attribute to **json** format and gets topic description from topic.
* RouteOnAttribute: Routes flow files based on their topic.
* Get\_Time\_Tag/Get\_Time\_Tag\_and\_Source: Gets “**\_timeTag**”, “**\_source**” data from flow file’s content and routes to next group.
* Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

## Process\_Data\_Group

Group that contains flow which transforms flow file to csv, enriches data and store flow file as a temp file up to needed size(default=100MB).

### Enrich\_Data\_Group

Group that contains “Lookup\_Group” that enriches data by looking up to “**full\_config.csv**” (Created by “**Full\_Confg\_Setup**” group), filters data, updates name, inserts datetime to “flow file” content.

#### Lookup\_Group

Group that contains flow which uses lookup processors for enriching “flow file” with parameters that are taken from “**full\_config.csv**”

**Processors:**

* + - Domain\_Type\_Lookup: Based on “**kafka.key**” gets value from “**full\_config.csv”** “**KEY**” / ”**Data Group**” (Key / Value) column and sets on “flow file” attribute that names “**domain\_type**”.
    - If\_Exists\_ = continue: Check if “**domain\_type**” is null, if False, then route to next processor.
    - Get\_Topic-Decription\_and\_Topic-Key: Adds “**topic.description**” and “**kafka.key**” to content, this data needed for creating full keys, they are used to get data from “**full\_config.csv**”.
    - Set\_Full\_Key: Transforms content to a table, to perform lookups and get needed data from “**full\_config.csv**”. Full key’s structure is: “{topic.description}.{kafka.key}.{parameter}”.
    - Convert\_Json\_To\_Csv: Converts flow file structure from json to csv file.
    - Interface\_No\_Lookup: Based on “{**topic.description**}.{**kafka.key}**” gets value from “**full\_config.csv”** “**topicANDkey**”/ ”**Interface-ID**” (Key / Value) column and sets on “flow file” attribute that names “**interface\_no**”.
    - Platform\_Type\_Lookup: Based on “**kafka.key**” gets value from “**full\_config.csv”** “**KEY**”/ ”**LRU.RFL**” (Key / Value) column and sets on “flow file” attribute that names “**interface\_no**”.
    - Filter\_Enabled-C\_Value: Based on values in column “**enables-c**” removes values from parameters value in flow file content.
    - Get\_Name-C: Gets “**NAME-C**” values from “**full\_config.csv**” based on "**full\_key**" value.
    - Get\_Enabled-C: Gets “**Enabled-C**” values from “**full\_config.csv**” based on "**full\_key**" value.
    - Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

**Processors:**

* + Filter\_notFounded\_Parameters: Filter content from flow file, that had been not found by “**Lookup\_Group**”
  + Filter\_Columns:Removes all columns beside “**Parameters**” and “**Values**”
  + Transform\_To\_Convertable\_Json:Transforms content to convertible csv structure.
  + Convert\_Json\_To\_Csv: Converts to CSV from JSON
  + Update\_Filename: Updates Filename and formats “**\_timeTag**” data from UNIX format to INT
  + Insert\_Datetime: Inserts “**\_timeTag**” as “**Datetime”** to “flow file” content.
  + Merge\_Records: Merges “flow files” with same names.

**Processors:**

* Store\_Data: processor that uses groovy script to append data (Default Nifi processors can’t append)
* Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

## Compress\_Hash\_Store

Flow which has groups that compress the data file, creates hash file for data file, and creates control files.

### Compress\_File

Group that gets files from temp storage (NFS), gets needed attributes from temporary filename, updates filename and destination folder, sets file number and compresses the file.

**Processors:**

* + Get\_Files: Gets files which size is equal to 100MB.
  + End\_Day\_Get\_Files: Gets files at 23:30 at any size.
  + Get\_Source: Gets “**Source**”, “**topic**”and “**topic.key**”attributes from temporary filename.
  + Update\_Filename\_and\_Destination: Updates to needed structure filename and sets destination. Also routes to “**Create\_Control\_Group**”.
  + Set\_File\_Number: With script adds number for file enumeration.
  + Gzip\_File: Compresses the file with GZIP compression format and routes to “**Hash\_Data\_File**”.
  + Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

### Create\_Control\_File

Group that creates control file for specific data file.

**Processors:**

* + Fetch\_Full\_Config\_File: Gets “**full\_config.csv**”file.
  + Get\_Name\_Attributes: Sets attributes from file name to create with them control file in future.
  + Get\_Config\_Version: Gets config version.
  + Get\_Relevant\_Data\_From\_Config: With all attributes we get all needed data from “flow file” content.
  + Set\_Filename: Gets name from control file’s content.
  + Remove\_Filename\_Column\_From\_Content: Removing “Filename” column from “flow file” content.
  + Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

### Hash\_Data\_File

Group that creates hash file to specific data file.

**Processors:**

* + Hash\_Zip\_File: Uses “SHA3-256” algorithm to hash content of a file and creates attribute in flow file with it.
  + Set\_Hash\_String: Replaces all content with hash string.
  + Set\_Name\_And\_File\_Type: Sets name for hash file and changes type to “**text/plain**”.
  + Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

## Transfer\_To\_Storage

Group which has a flow that sets needed attributes and stores all files to **MinIO**.

**Processors:**

* + Set\_Desttination\_For\_Files: updates “**destination**” attribute with needed path for minio bucket and creates “**filepath**” attribute that will be used as a “s3.tag” – will help in CCN.
  + Store\_Files: Stores file in **MinIO** at “**destination**”
  + Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

## Full\_Config\_Setup

Group which has a flow that creates “**full\_config.csv**” based on **“configB.csv”** and “**configC.csv**”.

### Set\_Full\_Config

Group that creates file with versions of config files, sets a name, and wires data between “**configB.csv**” and “**configC.csv**”.

### Setup\_File

Group that creates file with versions of config files and sets a name for “**configB.csv**”. or “**configC.csv**”

**Processors:**

* + - Set\_Name: Updates filename based on which config file is it.
    - Route\_On\_Filename: Routes to flow based on which config file is it and sends a “flow file” further to create filename with version of a configuration file.
    - Attributes\_To\_Csv: Creates 2 columns csv with “**filename**” and “**config.version**” for creating control file names based on configuration file version.
    - Set\_Version\_Control\_Filename: Sets a name for a version control file name.
    - Put\_Version\_Control\_File: Stores version control file name to where all config files are stored.
    - Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

**Processors:**

* + LookupRecord\_For\_ConfigB: Uses “**configB.csv**” to lookup data from “**configC.csv**” based on “**ID-B**” column.
  + LookupRecord\_For\_ConfigC: Uses “**configC.csv**” to lookup data from “**configB.csv**” based on “**ID-B**” column.

### Create\_Full\_Config

Transforms rows to needed structure for “**full\_config.csv**”, sets name and stores it to filesystem to work with it.

**Processors:**

* + Route\_On\_Config\_Filename: Based on config filename routes to row transformation process.
  + Based\_On\_ConfigB/ConfigC\_Row\_transformation: Transforms rows to needed structure and adds needed keys to work with.
  + Merge\_Rows: Merges all rows to single csv file.
  + Update\_Filename: Sets name to “**full\_config.csv**”.
  + Save\_Full\_Config: Stores “**full\_config.csv**” file to NFC file system to work with.
  + Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

**Processors:**

* Gets\_Config\_File\_Names: Gets filenames based on their timestamp from **MinIO** (Bucket = “**config-files**”)
* Fetch\_Config\_Files: Gets files from **Minio** (Bucket = “**config-files**”) based onprevious filenames.
* Split\_Json: Splits csv files that are routed from “**Set\_Full\_Config**” to rows and routes them to **“Create\_Full\_Config”**.
* Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

# Kafka\_Integration

Has a group in it which has a flow that check if there is an update in topics by hashing data and evaluating the hash strings. If there was an update, new list of topics will be saved and used flow file will be routed to other groups that will restart “**Kafka\_Consumer**”.

## Check\_Topics

**Processors:**

* + Ping\_Kafka: Generates “flow file” with kafka’s bridge url.
  + Get\_Kafka\_Topics: Sends HTTP request to Kafka Bridge to get a list of topics.
  + Update\_Name\_And\_Type: Updates “flow file” name and “**mime.type**”.
  + Clean\_Topics: Cleans topic list with script based on regex.
  + Hash\_Topic\_List: Creates attribute “**hash.values**” based on hash value of a “flow\_file” content.
  + Get\_Saved\_Topics: Gets a file with list of saved topics.
  + isUpdated: Evaluates hash values of both lists and check if there is a difference between lists, routes to “**Route\_If\_Updated**” if there is a difference.
  + Route\_If\_Updated: Routes to “**Set\_Name**” and to “**Set\_Consume\_Data\_Processors\_Att**”

if filename equals to “new\_topics.txt”.

* + Set\_Name: updates name from “new\_topics.txt” to "topics.txt”
  + Save\_Updated\_Topics: Store topic list to NFC storage.
  + Set\_Consume\_Data\_Processors\_Att: Creates needed attributes that will help to find needed processor and to work with it through NIFI’s API, routes to “**Find\_Processor\_Service**”.
  + Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

# Find\_Processor\_Service

Group that finds needed processor based on attributes that passed by previous groups and the routes it to “**Stop\_And\_Run\_Processor\_Service**”.

**Processors:**

* \* **Recursive Flow**
* Check\_If\_Group\_Found: Check if targeted Group had ben found, if **FOUND** – “Flow file” will be routed to “**Get\_Processors\_Data\_From\_Group**”.
* Get\_Group\_Data: Gets group data that is provided by Nifi’s api.
* Split\_by\_Groups: Splits provided json file by groups.
* Get\_Group\_ID\_and\_Name: For each splited group, will creates attributes based on “**ID**” and “**Group Name**” that in content.
* \* **End Of Recursive Flow**
* Get\_Prossesors\_Data\_From\_Group: Gets all processors data from specific group.
* Split\_By\_Processors: Splits “flow file” by “processors” objects in json.
* Get\_Prossesors\_ID\_and\_Name: For each splited group, will creates attributes based on “**ID**” and “**State**” and “**Name**”, that in content.
* Check\_if\_Processors: Check if needed processor found, if **FOUND** – “Flow file” will be routed to “**Stop\_And\_Run\_Processor\_Service**”.
* Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

# Stop\_And\_Run\_Processor\_Service

Group that communicates with Nifi’s api to start and stop processors.

## Run\_Processor

Group that gets “flow files” from “**Processor\_isRunning**” when attributes “**state**” = **STOPPED**, “**actions.performed** = **Less than 2**

**Processors:**

* + Get\_Processor\_Data: Gets information about processor. Uses “**processorsID**” “flow file” attribute.
  + Convert\_To\_Needed\_Structure: Converts json to needed structure to work with.
  + Change\_Stop\_To\_Run: Changes state in json’s content from “**STOP**” to “**RUN**”
  + Run\_Processor: Send request to Nifi’s api with updates json to run processor.
  + Update\_State\_And\_Actions: Updates “**state**”, “**actions.performed**” attributes and routes back to previous group.
  + Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

## Stop\_Processor

Group that gets “flow files” from “**Processor\_isRunning**” when attributes “**state**” = **RUNNING**, “**actions.performed** = **Less than 2**

**Processors:**

* + Get\_Processor\_Data: Gets information about processor. Uses **processorsID**}“flow file” attribute.
  + Convert\_To\_Needed\_Structure: Converts json to needed structure to work with.
  + Change\_Run\_To\_Stop: Changes state in json’s content from “**RUN**” to “**STOP**”
  + Run\_Processor: Send request to Nifi’s api with updates json to run processor.
  + Update\_State\_And\_Actions: Updates “**state**”, “**actions.performed**” attributes and routes back to previous group.
  + Set\_Group\_Name: Creates attribute on flow file and sets a group name is error was occurred and routes flow file to “**Log\_Error\_Message**” to “**Error\_Message\_Group**”

**Processors:**

* Processor\_isRunnnig?: Processor that routes “flow files” to needed group based on conditions:
  + If **state == RUNNING AND action.performed < 2:** Route to Stop processor.
  + If **state == STOPPED AND action.performed < 2:** Route to Run processor.
  + If **action.performed >= 2:** Terminate “Flow file”.

# Error\_Message\_Group

Contains flow that writes logs about flow files that had been routed from “Error\_Mesage” flow.

**Processors:**

* Log\_Message: Logs error message
* Log\_Attribute: Logs “flow file” attributes and content