xml-reader-producer

Release 1.0.6

Luan Moreno

Mar 17, 2021

Table of Contents

Chapter 1

User's Guide

With this documentation in hand it will be possible to better understand the functioning of our producer. He is responsible for converting CF-E XMLs to events and sends them to Kafka.

1.1 Objects

In this module is the main object of the application. With the event object, it will be possible to start fragmentation of CF-E.

1.1.1 The main event

class objects.imais_cfe_events.**Events** (*xml*)

Class for dividing the event into different streams to facilitate data processing.
When sending an XML to your builder it is possible to split CF-e by issuer, recipient, items etc.
…

Parameters **xml** (*str*) – XML containing the CF-e.

get_dest ( )
Returns identification group of the recipient of the CF-e.

get_det ( )
Returns Products and Services detailing group of CF-e.

get_emit ( )
Returns CF-e issuer identification group.

get_ide ( )
Returns CF-E identification information group.

get_inf_adic ( )
Returns Additional Information Group.

get_key ( )
Returns the CF-E key.

get_pgto ( )
Returns CFe Payment Information Group.

get_total ( )
Returns CF-e Total Values Group.

xml2json ( )
Converts the XML received on the builder in a JSON.

1.2 Metadata

With this module, you can perform the database settings. We are using Postgres.

1.2.1 Communication class with the database

class metadata.postgres.**Postgres**
Class that allows you to connect to the Postgres database.

With this class you can:

- Create the database in the instance of Postgres.
- Create the table where processing logs will be saved (processed_files).
- Inserts the metadata records from the files in the database

static create_database ( )
Create the database in the instance of Postgres.
Using the information contained in the environment variables, the create or recreates the database method.

static create_tables ( )
Create the table where processing logs will be saved (processed_files).
Using the information of the environment variables to connect to the database, create or re-create the "processed_files" table

static insert_rows_metadata_processed_files (*rows*)
Inserts the metadata records from the files in the database
Using the information of the environment variables to connect to the database and the data sent by parameter, the method inserts the file metadata in the database
…

Parameters **rows** (*dict*) – Records to be entered in the database.

1.2.2 Configuring the database

metadata.create_env.**create_database** ( )
This method will create the database and the table where the logs will be registered

1.3 Datastores

Nesse módulo temos os módulos que farão a comunicação com o Kafka e o Azure BlobStorage

1.3.1 Connection with Azure Blob Storage

This is the module responsible for communicating with Azure Blobstorage You can read the files in the containers and make the separation of valid files. Valids will be suitable for processing and invalids will quarantine.

Reading the files

class datastores.blob_storage.read_blob_files.**BlobStorage** (*blob_storage_conn_str*, *container_base*, *container_processed*, *container_quarantined*)

A class that allows you to connect to Azure Blob Storage
With this class it will be possible to identify the metadata of the files, copy the files between the containers, delete the files and process the files present in the container
…

Parameters
- **blob_storage_conn_str** (*str*) – Connection string with Blob Storage.
- **container_base** (*str*) – Container where files will be read.
- **container_processed** (*str*) – Container where files will be sent after being processed.
- **container_quarantined** (*str*) – Container that stores quarantine files.

copy_blob_files (*file_name*)

Copies the file from one container to the other.
…

Parameters **file_name** (*str*) – File to be copied.

delete_blob_files (*file_name*)

Deletes the file from a container. Used to clean the source location and reduce space usage
…

Parameters **file_name** (*str*) – File to be deleted.

get_file_metadata_info ( )

Returns the metadata of files in a container.
…

Returns **dict_metadata** – A list that contains the metadata for all files in the container.

Return type list

process_blob_files ( )

Process files within a container.
Processing takes place in 3 steps:
- Treatment and data ingestion in Kafka topics (CFE data and your items).
- Moves processed files to the successfully processed container.
- Retrieve information to share with the metadata repository (Postgres).

static utc_to_local (*utc_dt*)

Converts the utc time to the location
…

Parameters **utc_dt** (*datetime*) – A datetime containing the value to be converted.

Returns A datetime containing the converted value.

Return type datetime

Validating the files

class datastores.blob_storage.validate_file_type.**StorageFileTypeValidator**

Class that validates the files present in the read container.
Validation takes place in 3 steps:
- The contents of the files present in the container are read.

- Identified the file type by the root node of the XML file.
- If the file is not of the CFE type, it will be sent to the "quarantine" container, where it will be analyzed later.

run ( )
Performs the file validation process.

1.3.2 Connection with Kafka Broker

This is the module responsible for communicating with kafka broker Here are producer settings, the methods used for Send CF-e data and its items to their respective topics and Because Callback will be registered.

Creating Communication with Kafka

Kafka Producer Settings

datastores.kafka.producer_settings.**producer_settings_json** (*app_name*, *broker*)
Returns the default settings for a Kafka Producer.
…

Parameters
- **app_name** (*str*) – Name of the application that is sending the information to Kafka. This name will be used internally for logs and monitoring.
- **broker** (*str*) – Name of the Broker to be ingested by the producer.

Returns **settings** – A dictionary with the settings to be used by the producer.

Return type dict

Registering Callbacks

datastores.kafka.delivery_reports.**on_delivery_json** (*err*, *msg*)
Method used to log ingestion callbacks in Kafka
…

Parameters
- **err** (*str*) – String with error information, if any.
- **msg** (*obj*) – Object with the information from where the message was saved.

Feeding Topics

CF-E Topics

datastores.kafka.json.cfe_json_producer.**cfe_json_producer** (*xml*)
Performs xml/event decomposition in Kafka topics.
The following topics are fed by this routine:
- Ide_Json: CF-e identification information group
- Emit_Json: CF-e issuer identification group
- Dest_Json: Identification group of the recipient of the CF-e
- Det_Json: Products and Services detailing group of CF-e
- Total_Json: CF-e Total Values Group
- Pgto_Json: CFe Payment Information Group
- Inf_Adic_Json: Additional Information Group

The routine happens in 3 steps:
- Creation of KafkaProducer
- XML decomposition
- Sending events to Kafka

…

Parameters **xml** (*str*) – XML string to be decomposed

Items topic

datastores.kafka.json.cfe_items_json_producer.**cfe_items_json_producer**(*xml*)

Performs the sending of items (Tag Det) from xml to Kafka.
The following topics are fed by this routine:
- Items_Json: Product and Service Detailing Group of CF-e
The routine happens in 3 steps:
- Creation of KafkaProducer
- XML decomposition, searching for items in CF-e
- Sending events to Kafka
…

Parameters **xml** (*str*) – XML string to be decomposed

d

m

o

B

C

D

E

G

I

M

O

P

R

S

U

X