

Memoria ingesta de datos de OpenSky con NIFI

1) Obtención de datos desde la API de OpenSky Network

En primer lugar, necesitaba obtener los datos desde la API de OpenSky. Al analizar el enlace proporcionado, noté que los valores de las fechas estaban en un formato numérico poco común. Tras investigar, descubrí que la API utiliza timestamps UNIX, que representan el número de segundos transcurridos desde el 1 de enero de 1970 (UTC).

Para establecer el período de consulta adecuado, convertí las fechas al formato correspondiente. En este caso, configuré el rango desde el 24 de diciembre de 2020 a las 12:00:00 PM UTC hasta el mismo día a las 01:00:00 PM UTC, obteniendo los siguientes valores de timestamp:

- 24 de diciembre de 2020, 12:00:00 PM UTC = 1608804000
- 24 de diciembre de 2020, 01:00:00 PM UTC = 1608807600

<https://opensky-network.org/api/flights/all?begin=1608804000&end=1608807600>

Para realizar la ingesta de datos, utilicé el procesador InvokeHTTP en NiFi, el cual se encarga de realizar solicitudes HTTP a un servidor web. Este procesador permite enviar peticiones de diversos tipos como GET, POST, PUT y DELETE a una URL específica. Es particularmente útil para interactuar con APIs, obtener datos de servicios web y enviar información a servidores remotos, facilitando la integración de datos desde fuentes externas.

Settings

Scheduling

Properties

Relationships

Comments

Required field

+ Verification ✓

Property	Value
HTTP Method	GET
HTTP URL	https://opensky-network.org/api/fligh...
HTTP/2 Disabled	False
SSL Context Service	No value set
Connection Timeout	30 secs
Socket Read Timeout	60 secs
Socket Write Timeout	15 secs
Socket Idle Timeout	5 mins

Click the button above to verify this component.

Stopped

Cancel Apply

2) Conversión los datos del formato JSON a AVRO

El siguiente paso consistió en convertir los datos que se recibían en formato JSON a formato AVRO. Para ello, utilicé el procesador ConvertRecord en NiFi, configurando un JsonTreeReader como lector y un AvroRecordSetWriter como escritor. Este proceso asegura que los datos se estructuren de manera eficiente para su almacenamiento y posterior procesamiento, ya que el formato AVRO ofrece varias ventajas. Permite compresión, integración de esquemas y es ideal para sistemas que procesan grandes volúmenes de datos, optimizando tanto el almacenamiento como la transferencia entre sistemas distribuidos.

Edit Processor | ConvertRecord 2.2.0

Settings

Scheduling

Properties

Relationships

Comments

Required field

+ Verification ✓

Property	Value
Record Reader	JsonTreeReader
Record Writer	AvroRecordSetWriter
Include Zero Record FlowFiles	true

Click the button above to verify this component.

Stopped

Cancel Apply

3) Limpieza de los nulos

A continuación, realizamos la limpieza de los datos eliminando aquellas filas que contenían valores nulos. Para lograr esto, utilizamos el procesador QueryRecord de NiFi, el cual permite ejecutar consultas SQL sobre los datos de flujo. Esta herramienta facilita el filtrado y la eliminación de registros nulos de manera eficiente. Además, QueryRecord ofrece gran flexibilidad para aplicar transformaciones y otras operaciones sobre los datos, lo que optimiza el proceso de limpieza dentro del flujo de trabajo en NiFi.

Edit Processor | QueryRecord 2.2.0

Settings

Scheduling

Properties

Relationships

Comments

Required field

+ Verification

✓

Property	Value
Record Reader	<div><div></div>AvroReader</div>
Record Writer	<div><div></div>AvroRecordSetWriter</div>
Include Zero Record FlowFiles	<div><div></div>true</div>
Default Decimal Precision	<div><div></div>10</div>
Default Decimal Scale	<div><div></div>0</div>
Filter	<div><div></div>SELECT * FROM FLOWFILE WHERE ic...</div>

Click the button above to verify this component.

Stopped

Cancel

Apply

La consulta SQL utilizada fue la siguiente:

```
SELECT * FROM FLOWFILE
WHERE icao24 IS NOT NULL
      AND firstSeen IS NOT NULL
      AND estDepartureAirport IS NOT NULL
      AND lastSeen IS NOT NULL
      AND estArrivalAirport IS NOT NULL
      AND callsign IS NOT NULL
      AND estDepartureAirportHorizDistance IS NOT NULL
      AND estDepartureAirportVertDistance IS NOT NULL
      AND estArrivalAirportHorizDistance IS NOT NULL
      AND estArrivalAirportVertDistance IS NOT NULL
      AND departureAirportCandidatesCount IS NOT NULL
      AND arrivalAirportCandidatesCount IS NOT NULL
```

4) Conversión fecha de formato UNIX a “yyyy/MM/dd HH:mm:ss”

Para la conversión de la fecha, utilizamos el procesador UpdateRecord, el cual permite modificar o actualizar los valores de los registros dentro de un flujo de datos.

Configuramos las propiedades correspondientes para reemplazar las fechas en las columnas deseadas. El siguiente código se utilizó para convertir las fechas en formato Unix a un formato legible:

```
${field.value:multiply(1000):formatInstant("yyyy/MM/dd HH:mm:ss", "GMT+3")}
```

Este código multiplica el valor de la fecha por 1000 (para convertirlo de segundos a milisegundos) y luego lo formatea al formato "yyyy/MM/dd HH:mm:ss", ajustado a la zona horaria GMT+3. Este proceso se aplicó tanto para /firstSeen como para /lastSeen.

Edit Processor | UpdateRecord 2.2.0

Settings

Scheduling

Properties

Relationships

Comments

Required field

+ Verification

✓

Property	Value
Record Reader	<div>AvroReader</div>
Record Writer	<div>AvroRecordSetWriter</div>
Replacement Value Strategy	<div>Literal Value</div>
/firstSeen	<div><div><div></div></div><div><code>\${field.value:multiply(1000):formatInst...</code></div></div>
/lastSeen	<div><div><div></div></div><div><code>\${field.value:multiply(1000):formatInst...</code></div></div>

Click the button above to verify this component.

Stopped

Cancel

Apply

5) Reemplazar los valores de los aeropuertos a un formato legible

Para reemplazar los valores de los aeropuertos por nombres legibles, creamos un servicio que compara los códigos de los aeropuertos con los nombres correspondientes, almacenados en un archivo CSV. Para ello, utilizamos el procesador LookupRecord junto con un servicio SimpleCsvLookupService. Configuramos este servicio para que realizara la comparación utilizando las columnas relevantes del CSV y especificamos la ruta del archivo. Este proceso permitió traducir los códigos de los aeropuertos a sus nombres completos y legibles para su análisis.

Controller Service Details


SimpleCsvFileLookupService 2.2.0







Settings

Properties

Comments

Required field

Verification 

Property	Value
CSV File	 /Users/pedrotoledano/Desktop/MIOTI/...
CSV Format	 Default Format
Character Set	 UTF-8
Lookup Key Column	 ICAO
Ignore Duplicates	 true
Lookup Value Column	 Airport name

Close

En el procesador LookupRecord, tuvimos que añadir una propiedad denominada "key", donde indicamos la columna que contiene los códigos de los aeropuertos. Esta columna es la que se utilizaría para realizar la búsqueda en el archivo CSV y asociar los códigos con los nombres correspondientes.

Required field

+ Verification



Property	Value	
Lookup Service	Replace_estDepartureAirport	
Root Record Path	/	
Routing Strategy	Route to 'success'	
Record Result Contents	Insert Entire Record	
Record Update Strategy	Use "Result RecordPath" Property	
Result RecordPath	/estDepartureAirport	
Cache Size	0	
key	/estDepartureAirport	

Click the button above to verify this component.

Stopped ▾

Cancel

Apply

Este proceso tuvimos que repetirlo dos veces para realizar la conversión en las columnas estDepartureAirport y estArrivalAirport. Para cada una, configuramos un servicio diferente, añadiendo un SimpleCsvLookupService independiente.

6) Cambio de formato de AVRO a CSV

Una vez completadas todas las transformaciones necesarias, el siguiente paso fue cambiar el formato de los datos, pasando de AVRO a CSV. Para ello, utilizamos el procesador ConvertRecord, configurando un AvroReader como lector y un CSVRecordSetWriter como escritor.

Required field

+ Verification



Property	Value	
Record Reader	AvroReader	
Record Writer	CSVRecordSetWriter	
Include Zero Record FlowFiles	true	

Click the button above to verify this component.

7) Actualizar atributos del CSV

Como el archivo obtenido no se guardaba correctamente al exportarlo como .csv, decidimos añadir un atributo adicional. En la propiedad filename, configuramos el nombre del archivo para asegurarnos de que se guardara correctamente con el formato deseado.

Edit Processor | UpdateAttribute 2.2.0

SettingsSchedulingPropertiesRelationshipsComments

Required field+ Verification✓

Property	Value
Delete Attributes Expression	<i>No value set</i>
Store State	Do not store state
Stateful Variables Initial Value	<i>No value set</i>
Cache Value Lookup Cache Size	100
filename	<code>\${filename}.csv</code>

Click the button above to verify this component.

StoppedCancelApply

8) Exportar el CSV

Por último, añadimos el procesador PutFile para exportar los archivos CSV generados a la ruta específica.

Edit Processor | PutFile 2.2.0

SettingsSchedulingPropertiesRelationshipsComments

Required field+ Verification✓

Property	Value
Directory	<code>/Users/pedrotoledano/Desktop/MIOTI...</code>
Conflict Resolution Strategy	fail
Create Missing Directories	true
Maximum File Count	<i>No value set</i>
Last Modified Time	<i>No value set</i>

Click the button above to verify this component.