XXII

Data warehouse connector

CORE V.4

Table of contents

This document aims to introduce the data warehouse connector integrated into Core 4.

CORE Engine

Configuration

The **Site Name** and **Site ID**, essential for various settings communications, have been relocated to the **Settings / Application** tab.

The **Data Warehouse** tab allows adding the configuration for communication with a data warehouse.

The **Data Warehouse** tab is accessible to users with the following profiles: **XXII Admin**, **Super Admin**, and **Integrator**.

The configuration determining which database to use and how to connect to them is **given by the front settings page**. We upload a JSON file and then the settings backend will publish a message containing the info of the json file under the key config on one of theses 2 routes: `/register/webhook/connector_name` or `/webhook/connector_name/update`.

The generic message sent on this route is the following:

```
Unset
{
  "vms": "<connector-name>",
  "isActive": "bool",
  "url": "str",
  "config": {
    "key1": "value1",
    "key2": "value2"
  }
}
```

- `vms` represents the vms name, **it's not used by the database connector**.
- `isActive` represents whether the connector is active or not. If set the false all events sent through MQTT are ignored by the connector.
- `url` represents an url to connect to, **it's not used by the database connector**.
- `config` a dictionary created from the uploaded JSON file containing the connection information.

In this version, BigQuery is the only data warehouse supported by the connector. Listed below is the information to be provided under the key `config`, which will be uploaded to the front settings page:

BigQuery

When connecting to BigQuery you should upload a json document following this format:

The generic message sent on this route is the following:

```
Unset
{
  "database_name": "bigquery",
  "metadata": {
    "Driver": "Simba Google BigQuery ODBC Connector",
    "OAuthMechanism": 0,
    "Email": "<sa-email>",
    "KeyFileContent": {
      "type": "service_account",
      "project_id": "<project-id>",
      "private_key_id": "<private-key-id>",
      "private_key": "<private-key>",
      "client_email": "<sa-email>",
      "client_id": "<client-id>",
      "auth_uri": "https://accounts.google.com/o/oauth2/auth",
      "token_uri": "https://oauth2.googleapis.com/token",
      "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
      "client_x509_cert_url": "<client-x509-cert-url>",
      "universe_domain": "googleapis.com"
    },
    "Catalog": "<project-id>",
    "DefaultDataset": "<dataset-id>"
  }
}
```

- `database_name` represents the name of the database to use. Use `bigquery`
- The dictionary under `KeyFileContent` is the service account given by the infra team. It's usually available on Bitwarden. The service account gives specific rights on specific project.
- `Email` represents the email of the service account copy the one under the `client_email` key in the service account
- `Catalog` represents the project_id you will find it in the service account just copy it.
- `DefaultDataset` represents the dataset_id the infra team is responsible for creating the dataset either ask them or go on the Gcloud console and in your project choose the right dataset to put data in

CORE Insights

Database schemas

This pages presents the two target SQL tables and each of their key. It also explains how each event type in the platform ends up in one of the two SQL tables.

CountEvent Table

Count Event		
STRING	id	PK
STRING	skill_id	
STRING	skill_name	
STRING	site_id	
STRING	site_name	
STRING	camera_id	
STRING	camera_name	
TIMESTAMP	creation_date	
STRING	class_label	
INTEGER	class_count	

The `CountEvent` table stores information related to counted events, such as skills, sites, cameras, and classification details. In this table each row represents a recurrent event sent each 5 secs whatever happens representing the count by class. The table has the following columns:

- `id` (PK): A unique identifier for each class of each event.
- `skill_id`: The identifier of the skill associated with the event.
- `skill_name`: The name of the skill associated with the event.
- `site_id`: The identifier of the site where the event occurred.
- `site_name`: The name of the site where the event occurred.
- `camera_id`: The identifier of the camera that captured the event.
- `camera_name`: The name of the camera that captured the event.
- `creation_date`: The timestamp when the event was created in the platform (different from the insertion date).
- `class_label`: The label associated to the count.
- `class_count`: The count associated to the class label.

TriggerObjectEvent Table

TriggerObjectEvent			
STRING	id	PK	
STRING	skill_id		
STRING	skill_name		
STRING	site_id		
STRING	site_name		
STRING	camera_id		
STRING	camera_name		
TIMESTAMP	creation_date		
STRING	object_label		
FLOAT	object_validity_period		NULLABLE
JSON	metadata		NULLABLE

The TriggerObjectEvent table stores information related to triggered object events, such as skills, sites, cameras, object labels, and metadata. In this table each row represents an insertion triggered by a particular event. The table has the following columns:

`id` (PK): A unique identifier for each object of each event.
`skill_id`: The identifier of the skill associated with the event.
`skill_name`: The name of the skill associated with the event.
`site_id`: The identifier of the site where the event occurred.
`site_name`: The name of the site where the event occurred.
`camera_id`: The identifier of the camera that captured the event.
`camera_name`: The name of the camera that captured the event.
`creation_date`: The timestamp when the event was created in the platform (different from the insertion date)
`object_label`: The label assigned to the object.
`object_validity_period`: The period for which the object stayed in the zone. This field is only use for the `Average time per zone` skill type. This field can be `null`.
`metadata`: Additional metadata associated with the object. It's currently used for the JSON result of the VLM statistics skill. This field can be `null`.

Relation between Skill type and the SQL tables

The database connector receives 5 different types of messages on MQTT each representing one kind of event, each of these events need to be transformed and stored in one of the two SQL tables (`CountEvent` or `TriggerObjectEvent`).

Detection alerts

Alerts are stored in the `TriggerObjectEvent` table since we can decompose each detection alerts into several objects. It also represents an event that is triggered by something in that case when an object enters a zone for example.

Below is the MQTT message we receive when dealing with a detection alert.

Unset
```
{
  "_id": "5763c538-8553-11ef-b3a8-9adaac3f08a3",
  "zoneId": "66fa60eae507e5540ef2ee57",
  "zoneName": "Custom Detection",
  "jobId": "66fa60e2a82b19d64648dac5",
  "jobName": "Camera 22",
  "creationDate": "2024-10-08T08:57:26.608Z",
  "metadata": {
    "objects": {
      "1279963": {
        "label": "car",
        "boundingBox": [
          0.24433593451976776,
          0.4784722725550334,
          0.05937499925494194,
          0.07152777910232544
        ]
      },
      "1279967": {
        "label": "person",
        "boundingBox": [
          0.246,
          0.887,
          0.123,
          0.002
        ]
      }
    }
  }
}
```

This message would give the following insertion in the `TriggerObjectEvent` table:

id	site_id	site_name	camera_id	camera_name	skill_id
bcff6b72-9d18-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	66fa60e2a82b19d64648dac5	Camera 22	66fa60eae507e5540ef2ee57
c04987a4-9d18-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	66fa60e2a82b19d64648dac5	Camera 22	66fa60eae507e5540ef2ee57

skill_name	creation_date	object_label	object_validity_period	metadata
Custom Detection	2024-10-08T08:57:26.608000 UTC	car	NULL	NULL
Custom Detection	2024-10-08T08:57:26.608000 UTC	person	NULL	NULL

Crowd detection alerts

This kind of event is only used for the `Grouping of people (Détection de foule avec seuil)` skill when we want to raise an alert whenever the count of people in the zone reaches a defined threshold

This event is stored in the `TriggerObjectEvent` **table** since we can decompose each alert into several "pseudo-object" each containing the alert for each class. It also represents an event that is triggered by something in that case when the crowd in a zone reaches a defined threshold.

Below is the MQTT message we receive when dealing with a crowd detection alert.

Unset
```
{
  "_id": "5763c538-8553-11ef-b3a8-9adaac3f08a3",
  "zoneId": "66fa60eae507e5540ef2ee57",
  "zoneName": "Grouping of people",
  "jobId": "66fa60e2a82b19d64648dac5",
  "jobName": "Camera 22",
  "creationDate": "2024-10-08T08:57:26.608Z",
  "metadata": {
    "count": 13,
    "classes": {
      "car": 3,
      "person": 10
    }
  }
}
```

This message would give the following insertion in the `TriggerObjectEvent` table:

id	site_id	site_name	camera_id	camera_name	skill_id
1dfb9284-9d19-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	66fa60e2a82b19d64648dac5	Camera 22	66fa60eae507e5540ef2ee57
25790474-9d19-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	66fa60e2a82b19d64648dac5	Camera 22	66fa60eae507e5540ef2ee57

skill_name	creation_date	object_label	object_validity_period	metadata
Grouping of people	2024-10-08T08:57:26.608000 UTC	car	NULL	NULL
Grouping of people	2024-10-08T08:57:26.608000 UTC	person	NULL	NULL

Line based countings

Line countings are stored in the `TriggerObjectEvent` table since we can decompose each counting into several objects. It also represents an event that is triggered by something in that case when an object crosses a line.

Below is the MQTT message we receive when dealing with a line based counting.

Unset

```
{
  "_id": "6ba7b810-9dad-11d1-80b4-00c04fd430c8",
  "jobId": "6152e414a05a3408cbdc8d41",
  "jobName": "Camera 22",
  "countingId": "6152e414a05a3408cbdc8d42",
  "zoneId": "6152e3134b1215a02bcaa9c3",
  "zoneName": "Custom Counting line",
  "creationDate": "2021-09-28T21:12:35.777",
  "receivedDate": "2021-09-28T21:12:36.777",
  "count": 5,
  "delta": 3,
  "objects": [
    {
      "label": "person",
      "count": 5,
      "delta": 3,
      "changes": [
        {
          "id": 1,
          "delta": 1,
          "validityPeriod": null
        },
        {
          "id": 2,
          "delta": 1,
          "validityPeriod": null
        },
        {
          "id": 3,
          "delta": 1,
          "validityPeriod": null
        }
      ]
    }
  ]
}
```

This message would give the following insertion in the `TriggerObjectEvent` table:

id	site_id	site_name	camera_id	camera_name	skill_id
49c62654-9d19-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	6152e414a05a3408cbdc8d41	Camera 22	6152e3134b1215a02bcaa9c3
5021b4dc-9d19-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	6152e414a05a3408cbdc8d41	Camera 22	6152e3134b1215a02bcaa9c3
577e1c98-9d19-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	6152e414a05a3408cbdc8d41	Camera 22	6152e3134b1215a02bcaa9c3

skill_name	creation_date	object_label	object_validity_period	metadata
Custom Counting line	2021-09-28T21:12:35.777000 UTC	person	NULL	NULL
Custom Counting line	2021-09-28T21:12:35.777000 UTC	person	NULL	NULL
Custom Counting line	2021-09-28T21:12:35.777000 UTC	person	NULL	NULL

Zone based countings

Zone countings are stored in the `CountEvent` table since we can decompose each counting into several classes. It also represents an event that is recurrent since a zone counting skill sends the count by class in the zone each 5 secs. We send it every 5 secs even the count for the class did not change

Below is the MQTT message we receive when dealing with a zone based counting.

Unset
```
{
  "_id": "6ba7b810-9dad-11d1-80b4-00c04fd430c8",
  "jobId": "6152e414a05a3408cbdc8d41",
  "jobName": "Camera 22",
  "countingId": "6152e414a05a3408cbdc8d42",
  "zoneId": "6152e3134b1215a02bcaa9c3",
  "zoneName": "Custom counting zone",
  "creationDate": "2021-09-28T21:12:35.777",
  "receivedDate": "2021-09-28T21:12:36.777",
  "count": 5,
  "delta": 3,
  "objects": [
    {
      "label": "person",
      "count": 5,
      "delta": 3
    }
  ]
}
```

This message would give the following insertion in the `CountEvent` table:

id	site_id	site_name	camera_id	camera_name	skill_id
ed146eec-9d19-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	6152e414a05a3408cbdc8d41	Camera 22	6152e3134b1215a02bcaa9c3

skill_name	creation_date	class_label	class_count
Custom Counting zone	2021-09-28T21:12:35.777000 UTC	person	5

Average time per zone

Average time per zone is a special kind of zone based counting.

Average time per zone events are stored in the `TriggerObjectEvent` table. It represents an event that is triggered every time an object enters or exits a zone. If an object exits the zone the time it spent in the zone is recorded in the `validityPeriod` field and then an insertion is made into BigQuery only for this object.

Below is the MQTT message we receive when dealing with a zone based counting.

Unset

```
{
  "_id": "6ba7b810-9dad-11d1-80b4-00c04fd430c8",
  "jobId": "6152e414a05a3408cbdc8d41",
  "jobName": "Camera 22",
  "countingId": "6152e414a05a3408cbdc8d42",
  "zoneId": "6152e3134b1215a02b2a49c3",
  "zoneName": "Average time per zone",
  "creationDate": "2021-09-28T21:12:35.777",
  "receivedDate": "2021-09-28T21:12:36.777",
  "count": 5,
  "delta": 3,
  "objects": [
    {
      "label": "person",
      "count": 5,
      "delta": 1,
      "changes": [
        {
          "id": 1,
          "delta": -1,
          "validityPeriod": 10.5
        },
        {
          "id": 2,
          "delta": 1,
          "validityPeriod": null
        },
        {
          "id": 3,
          "delta": 1,
          "validityPeriod": null
        }
      ]
    }
  ]
}
```

This message would give the following insertion in the `TriggerObjectEvent` table:

id	site_id	site_name	camera_id	camera_name	skill_id
a47027b2-9d19-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	6152e414a05a3408cbdc8d41	Camera 22	6152e3134b1215a02bcaa9c3

skill_name	creation_date	object_label	object_validity_period	metadata
Average time per zone	2021-09-28T21:12:35.777000 UTC	person	10.5	NULL

VLM Statistics

Statistics events are stored in the `TriggerObjectEvent` table since each event represents an object when it triggers something. Info on the object is then added by a VLM under the field `objectMetadata`.

Below is the MQTT message we receive when dealing with a statistic.

Unset
```
{
  "_id": "6152e414a05a3408cbdc8d40",
  "zoneId": "6152e3134b1215a02bcaa9c3",
  "zoneName": "VLM statistics",
  "jobId": "6152e414a05a3408cbdc8d41",
  "jobName": "Camera 22",
  "creationDate": "2021-09-28T21:12:35.777",
  "label": "car",
  "objectId": 1337,
  "objectMetadata": {
    "color": "red",
    "brand": "Renault"
  }
}
```

This message would give the following insertion in the `TriggerObjectEvent` table:

id	site_id	site_name	camera_id	camera_name	skill_id
bfc0fb4e-9d1a-11ef-8be1-db4ea7cac089	03e00cfe-98b5-43dd-b3bb-4ae4c496f3e7	XXII DEV	6152e414a05a3408cbdc8d41	Camera 22	6152e3134b1215a02bcaa9c3

skill_name	creation_date	object_label	object_validity_period	metadata
VLM statistics	2021-09-28T21:12:35.777000 UTC	car	NULL	{"color": "red", "brand": "Renault"}

How to query the database

You will find below explanation on the recommended way to query the database.

TriggerObjectEvent table

Get the number of object for each label and for each skill

Unset

```
SELECT
  skill_id,
  skill_name,
  object_label,
  COUNT(object_label) AS count_by_label
FROM
  `project_id.dataset_id.TriggerObjectEvent`
GROUP BY
  object_label,
  skill_id,
  skill_name
```

Get the average time per zone for each label for each skill

Unset

```
SELECT
  skill_id,
  object_label,
  AVG(object_validity_period) AS avg_validity_period
FROM
  `project_id.dataset_id.TriggerObjectEvent`
WHERE
  object_validity_period IS NOT NULL
GROUP BY
  skill_id,
  object_label
```

Get the number of corresponding persons to each profile combination

Unset
```
WITH ExtractedMetadata AS (
  SELECT
    skill_id,
    camera_id,
    EXTRACT(DATE FROM creation_date) AS day_date,
    EXTRACT(HOUR FROM creation_date) AS hour,
    JSON_VALUE(METADATA, '$.age') AS age,
    JSON_VALUE(METADATA, '$.gender') AS gender,
    JSON_VALUE(METADATA, '$.style') AS style
  FROM
    `project_id.dataset_id.TriggerObjectEvent`
)

SELECT
  skill_id,
  camera_id,
  day_date,
  hour,
  age,
  CASE
    WHEN age != 'underage' THEN gender
    ELSE NULL
  END AS gender,
  CASE
    WHEN age != 'underage' THEN style
    ELSE NULL
  END AS style,
  COUNT(*) AS count_by_type
FROM
  ExtractedMetadata
WHERE
  age IN ('underage', '18-30', '31-44', '45-54', '55-64', '65+')
  OR gender IN ('female', 'male')
  OR style IN ('casual', 'relaxed', 'athleisure', 'business casual',
'professional', 'chic', 'undetermined')
GROUP BY
  skill_id,
  camera_id,
  day_date,
  hour,
  age,
  gender,
  style
```

CountEvent table

Get the average count in the zone for each skill by label

Unset

```
SELECT
  skill_id,
  class_label,
  AVG(class_count) AS avg_count_by_class
FROM
  `project_id.dataset_id.CountEvent`
GROUP BY
  skill_id,
  class_label
```