Export Service to Osisoft PI

Description

Osisoft PI is the most used historian in the Oil&Gas industry. EdgeX needs to be capable to export data to such a platform.

The target is to deliver this feature as part of California Release.

Requirements

- Use PI Web API (restful API) for Data Ingestion
- Authenticate the EdgeX node through Basic Authentication
- Create new tags in PI Archive
- Export throughput up to 1,000 events/second from each EdgeX node
- Local buffering and retry mechanism with resilience to network downtime
- Local logging of error messages & events through EdgeX Log service

Design

This capability can be implemented by cloning the current EdgeX HTTPS REST export feature and modifying it accordingly to send messages in a format accepted by PI Web API. This is an approach that has been adopted by other IIoT platforms as well.

- **Assumptions**
  - There is a 1-to-1 relationship between EdgeX measurements and PI Tags. A table to map the datatypes shall be defined.
  - There is a 1-to-1 relationship between EdgeX events and PI Tag. A table to map the datatypes shall be defined.
  - The corresponding PI Tag will have a name that is obtained by concatenating the three following pieces (with "." in between)
    - Tag Prefix as defined in the next section "Step 1 - Configuration"
    - Device Name (as what is specified in EdgeX Metadata - this must be unique)
    - Measurement/Event Name (as per what is specified in EdgeX Metadata - this must be unique within the same device)
  - An example of PI Tag Name will be "deployment001.deviceABC.measurement005"

- **Step 1 - Configuration**
  - Parameters that define the configuration of the export service and should be set through Export Client API calls:
    - PI Web API Endpoint (e.g. https://mypiwebapiendpoint/piwebapi)
    - PI Archive Name (e.g. mypiarchive)
    - Export interval in milliseconds (e.g. 1000) - as an alternative this could be batch size
    - Username (e.g. piwebapiuser)
    - Password (e.g. piwebapipassword)
    - Tag Prefix (e.g. deployment0001)
    - Datapoint(s) that will be exported to corresponding PI tag(s)
  - As part of the registration to export client, it would be appropriate to verify connectivity/authentication.
    - Make a GET request to https://mypiwebapiendpoint/piwebapi/dataservers with the basic authentication header as per piwebapiuser/piwebapipassword
    - The result will look like the below. If there is an entry with "Name":"mypiarchive" then the registration can be considered successful and the WebId of the archive should be stored in EdgeX for usage in following steps
      -

```
{
  "Links": {},
  "Items": [
    {
      "WebId": "s0LOyweq8d50GzEPfROu0oqQMTAuMTI3LjkyLjEzNA",
      "Id": "7ab0ec2c-1daf-41e7-b310-f7d13aed28a9",
      "Name": "10.125.23.143",
      "Path": "\\\\PIServers[10.125.23.143]",
      "IsConnected": false,
      "ServerVersion": "",
      "Links": {
        "Self": "https://mypiwebapiendpoint/piwebapi/dataservers
/s0LOyweq8d50GzEPfROu0oqQMTAuMTI3LjkyLjEzNA",
        "Points": "https://mypiwebapiendpoint/piwebapi/dataservers
/s0LOyweq8d50GzEPfROu0oqQMTAuMTI3LjkyLjEzNA/points",
        "EnumerationSets": "mypiwebapiendpoint/piwebapi/dataservers
/s0LOyweq8d50GzEPfROu0oqQMTAuMTI3LjkyLjEzNA/enumerationsets"
      }
    },
    {
      "WebId": "s0AAAAAAAAAAD_____vIRf4ATkRQ",
      "Id": "00000000-0000-0000-ffff-ffffbc845fe0",
      "Name": "mypiarchive",
      "Path": "\\\\PIServers[mypiarchive]",
      "IsConnected": false,
      "ServerVersion": "",
      "Links": {
        "Self": "https://mypiwebapiendpoint/piwebapi/dataservers
/s0AAAAAAAAAAD_____vIRf4ATkRQ",
        "Points": "https://mypiwebapiendpoint/piwebapi/dataservers
/s0AAAAAAAAAAD_____vIRf4ATkRQ/points",
        "EnumerationSets": "https://mypiwebapiendpoint/piwebapi/dataservers
/s0AAAAAAAAAAD_____vIRf4ATkRQ/enumerationsets"
      }
    },
    {
      "WebId": "s0YLwbWI-V8kmIc_EPhJ9d-AVVNBLUgwTDY1UzE",
      "Id": "581bbc60-958f-49f2-8873-f10f849f5df8",
      "Name": "pisrv002",
      "Path": "\\\\PIServers[pisrv002]",
      "IsConnected": false,
      "ServerVersion": "",
      "Links": {
        "Self": "https://mypiwebapiendpoint/piwebapi/dataservers/s0YLwbWI-
V8kmIc_EPhJ9d-AVVNBLUgwTDY1UzE",
        "Points": "https://mypiwebapiendpoint/piwebapi/dataservers/s0YLwbWI-
V8kmIc_EPhJ9d-AVVNBLUgwTDY1UzE/points",
        "EnumerationSets": "https://mypiwebapiendpoint/piwebapi/dataservers
/s0YLwbWI-V8kmIc_EPhJ9d-AVVNBLUgwTDY1UzE/enumerationsets"
      }
    }
  ]
}
```


- One of the following 3 errors may occur during this verification:
  - Unable to reach the endpoint (may be due to different reasons such as connectivity, firewall blocking port 443, DNS resolution...). In this case a timeout is necessary to pop out an error after X amounts of seconds/retries
  - 401 Unauthorized (Wrong credentials, either username or password)
  - mypiarchive does not exist in the list of registered PI Archives (or it may be that the server is registered with the hostname while the specified endpoint is an IP address)

- **Step 2 - Initialization**
  - The following steps need to be executed after the initial configuration, any time the configuration changes or whenever the Export Distribution service restarts.
    - Verify if the tags are already existing in the PI Archive.
      - For each configured tag, the following query must be executed.
      - GET https://mypiwebapiendpoint/piwebapi/search/query?q=name:tagname&scope=pi:mypiarchive
      - If the tag already exists, the reply will look like:

```
```

```
{
  "TotalHits": 1,
  "Links": {
    "Next": "https://mypiwebapiendpoint/piwebapi/search/query?
q=name%3Atagname&scope=pi%3Amypiarchive&count=10&start=10",
    "First": "https://mypiwebapiendpoint/piwebapi/search/query?
q=name%3Atagname&scope=pi%3Amypiarchive&count=10",
    "Last": "https://mypiwebapiendpoint/piwebapi/search/query?
q=name%3Atagname&scope=pi%3Amypiarchive&count=10&start=0"
  },
  "Errors": [],
  "Items": [
    {
      "Name": "tagname",
      "Description": "Some Description",
      "MatchedFields": [
        {
          "Field": "name"
        }
      ],
      "ItemType": "pipoint",
      "AFCategories": [],
      "UniqueID": "\\\\{044de274-8d72-4cb6-839b-9e415e793dd8}\\?3",
      "WebId": "P0dOJNBHKNtkyDm55BXnk92AAwAAAAU1JWR0RZUExNT1NJRDAzX
ENEVDE1OA",
      "UoM": "deg. c",
      "DataType": "float32",
      "Links": {
        "Self": "https://mypiwebapiendpoint:443/piwebapi/points/P0d
OJNBHKNtkyDm55BXnk92AAwAAAAU1JWR0RZUExNT1NJRDAzXENEVDE1OA"

      },
      "Score": 13.62607
    }
  ]
}
```


In this case the WebId of the tag must be read and stored in EdgeX in order to be used in further iterations. If the tag does not exist, the reply will look something like:


```
{
  "Links": {
    "Next": "https://mypiwebapiendpoint/piwebapi/search/query?q=name%
3ACDT15&scope=pi%3Amypiarchive&count=10&start=10",
    "First": "https://mypiwebapiendpoint/piwebapi/search/query?q=name%
3ACDT15&scope=pi%3Amypiarchive&count=10",
    "Last": "https://mypiwebapiendpoint/piwebapi/search/query?q=name%
3ACDT15&scope=pi%3Amypiarchive&count=10&start=0"
  },
  "Errors": [],
  "Items": []
}
```


- In this case the tag will have to be created with the following command
- POST https://mypiwebapiendpoint/piwebapi/dataservers/{webId}/points where WebId is riferred to the target dataserver. The payload of the POST request should look like the following:


```
{
  "Name": "deployment001.deviceABC.measurement005",
  "Descriptor": "Some description as per EdgeX Metadata",
  "PointClass": "classic",
  "PointType": "Float32",
  "EngineeringUnits": "",
}
```


- **Step 3 - Data Export**
  - At each Export Interval as per defined in the configuration, a new POST call should be fired as below:

- POST https://mypiwebapiendpoint/piwebapi/streamsets/recorded with the following JSON payload containing a section for each WebId corresponding to the tags


```
[
        {"WebId":"P0dOJNBHKNtkyDm55BXnk92AAwAAAAU1JWR0RZUExNT1NJRDAzXENEVDE1OA",
        "Items":[
                {"Timestamp":"2018-01-16T20:31:50.267Z","Value":15308.23},
                {"Timestamp":"2018-01-16T20:31:51.234Z","Value":15308.88},
                {"Timestamp":"2018-01-16T20:31:51.357Z","Value":15308.04},
                {"Timestamp":"2018-01-16T20:31:51.678Z","Value":15309.15},
                {"Timestamp":"2018-01-16T20:31:51.907Z","Value":15309.65}
                ]
        },
        {"WebId":"P0dOJNBHKNtkyDm55BXnk92A3OM1AAU1JWR0RZUExNT1NJRDAzXEZPR0hPUk4uQ1JJTw",
        "Items":[
                {"Timestamp":"2018-01-16T20:31:50.653Z","Value":15308.67},
                {"Timestamp":"2018-01-16T20:31:51.124Z","Value":15308.86},
                ]
        },
]
```


  - A successful request will provide a "202 Accepted" response
  - The activities (request/reply) should be logged through EdgeX log service

Possible improvements for the next version (Delhi Release):

- Possibility to perform data backfill (From StartTimestamp to EndTimestamp) - this may require an architectural change in Export Services
- Manage conflicts with overlapping tag names between different EdgeX deployments
- Support Kerberos authentication (Will need to specify in the configuration Keytab, Kdc, Principal and Realm)

Possible improvements for the long term:

- High throughput up to 100,000 events/second from each EdgeX node by using an ad-hoc OMF connector to send messages to PI Connector Relay with OMF

Code

To be updated.

Current status

Waiting for Go Export Client & Distribution to support HTTPS REST which is mandatory for this export services to work