HOW TO.

Generic Data Push Integration Guide

May 27, 2021
Rev. 1.0

Contents

Introduction 3
System Architecture 4
Sample Data Output 5
Configuration 8
Example Implementation 9
Additional Resources 14

Introduction

People-counting technologies are widely used in many types of businesses and public locations. By automatically counting the number of persons that enter and exit, people counters provide visitor statistics that enable businesses to analyze trends and optimize their operations.

Axis offers the analytics AXIS People Counter, which can be uploaded to any compatible Axis camera, and AXIS P8815-2 3D People Counter, which consists of a dedicated camera with embedded analytics. Both solutions provide reliable results when properly installed in suitable locations.

Figure 1. A 3D solution uses information from its two sensors to evaluate depth in the image.

AXIS P8815-2 3D People Counter calculates the depth within the maximum counting area in order to measure the height and the size of the object. It is suitable for crowded scenarios and scenes with challenging light conditions like strong sunlight, glares and shadows.

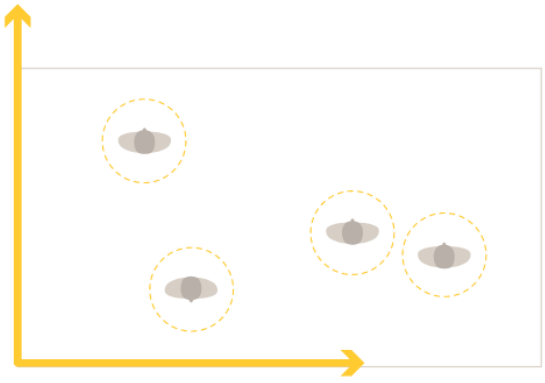


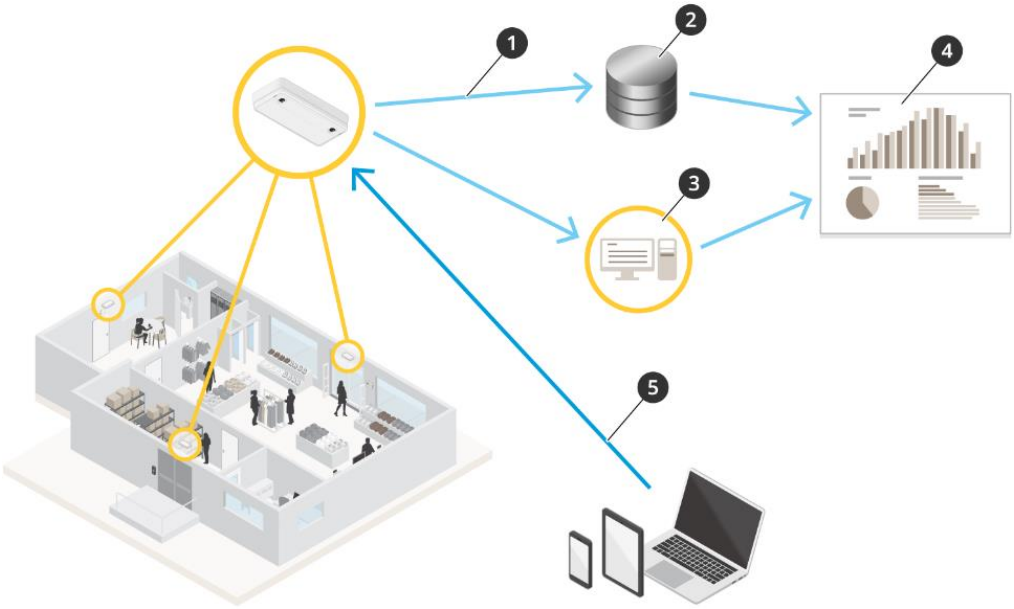
Figure 2. A 2D solution registers moving objects on one plane, seen from above.

AXIS People Counter is a 2D solution that can be used on a wide range of Axis cameras. This makes it an easy addition to a surveillance site standardized on a specific Axis camera model or when there are special requirements on camera capabilities and features.

System Architecture

Axis people-counting solutions offer a wide array of reporting capabilities to enable the integration of people counting data collected by the sensor into third-party services and applications.

The following diagram depicts a traditional deployment of Axis people-counting solutions in which the sensor data is integrated into third-party applications and services. It focuses on the essential components of the system and disregards other aspects such as network security components which would typically also be present in the overall architecture of a typical system.



1. AXIS P8815-2 3D People Counter or AXIS People Counter sends data automatically to a remote HTTPS destination through the data push functionality.
2. A third-party database or service is used to store and process the data received from the AXIS P8815-2 3D People Counter or AXIS People Counter.
3. Alternatively, the AXIS P8815-2 3D People Counter or AXIS People Counter sends data to a local instance of AXIS Store Data Manager where the data is stored.
4. A third-party reporting platform is used to visualize the collected data from the third-party database or AXIS Store Data Manager.
5. Alternatively, a built-in REST API allows third-party applications to request data directly from the AXIS P8815-2 3D People Counter or AXIS People Counter.

This document focuses specifically on the “Generic Data Push” feature (1) used to send raw data from the sensor to a specified HTTPS endpoint in JSON format (2) and provides a comprehensive step-by-step integration guide including a sample implementation to help get you started.

For additional information about the other integration capabilities depicted in the above illustration and complete API specifications of the AXIS P8815-2 3D People Counter or AXIS People Counter, please refer to the application manual.

Sample Data Output

The JSON data generated by the Genetic Data Push feature has the following structure:


```
{
  "apiName":"Axis Retail Data",
  "apiVersion":"0.4",
  "utcSent":"2021-04-13T09:22:24Z",
  "localSent":"2021-04-13T11:22:24",
  "data":{
    "utcFrom":"2021-04-13T09:19:00Z",
    "utcTo":"2021-04-13T09:22:00Z",
    "localFrom":"2021-04-13T11:19:00",
    "localTo":"2021-04-13T11:22:00",
    "measurements":[
      {
        "kind":"people-counts",
        "utcFrom":"2021-04-13T09:19:00Z",
        "utcTo":"2021-04-13T09:20:00Z",
        "localFrom":"2021-04-13T11:19:00",
        "localTo":"2021-04-13T11:20:00",
        "items":[
          {
            "direction":"in",
            "count":0,
            "adults":0
          },
          {
            "direction":"out",
            "count":0,
            "adults":0
          }
        ]
      }
    ]
  },
  "sensor":{
```

```
    "application":"AXIS 3D People Counter",
    "applicationVersion":"10.5",
    "timeZone":"Europe/Stockholm",
    "name":"axis-accc8ef3d92e",
    "serial":"accc8ef3d92e",
    "ipAddress":"192.168.1.106"
  }
}
```


Field Description

Field	Description
apiName	Name of the API being utilized
apiVersion	Version number of the API being utilized
utcSent	UTC date and time when the data was delivered
localSent	Local date and time when the data was delivered
data	Array containing counting data information
data[ ].utcFrom	UTC start date and time of the counting data
data[ ].utcTo	UTC end date and time of the counting data
data[ ].localFrom	Local start date and time of the counting data
data[ ].localTo	Local end date and time of the counting data
data[ ].measurements	Array containing counting data measurements
data[ ].measurements[ ].kind	Type of data provided by the Axis network camera
data[ ].measurements[ ].utcFrom	UTC start date and time of the counting data for specific time interval
data[ ].measurements[ ].utcTo	UTC end date and time of the counting data for specific time interval
data[ ].measurements[ ].localFrom	Local start date and time of the counting data for specific time interval
data[ ].measurements[ ].localTo	Local end date and time of the counting data for specific time interval
data[ ].measurements[ ].items	Array containing the counting data for specific time interval
data[ ].measurements[ ].items[ ].direction	Direction of the data recorded during specific time interval – in or out
data[ ].measurements[ ].items[ ].count	Number of counts recorded during specific time interval
data[ ].measurements[ ].items[ ].adults	Number of adult counts recorded during specific time interval **(only applicable to AXIS P8815-2 3D People Counter)**
sensor	Array containing sensor specific information

sensor[ ].application	Type of people counter application
sensor[ ].timeZone	Selected time zone for the Axis network camera
sensor[ ].name	Device name provided from the application
sensor[ ].serial	Serial number of the Axis network camera
sensor[ ].ipAddress	IP address of the Axis network camera

NOTE

A response of 200 OK from the recipient is necessary for the Axis device to send counting data. Should there be an interruption in the communication between the camera and the recipient, the camera will automatically re-send all counting data to the recipient since the last 200 OK response was received once the communication resumes.

NOTE

At the time of writing, the latest protocol version is 0.4. If the data being sent does not correspond to the description stated in this document, it is necessary to modify the version of the protocol.

This is possible by issuing the following request:

AXIS P8815-2 3D People Counter

Format: JSON

Method: PATCH

http://<servername>/a3dpc/api/settings

JSON input parameters
{"push_protocol_version":"V0_4"}

AXIS People Counter

http://CAMERAIP/axis-cgi/param.cgi?action=update&root.Tvpc.GenericHTTPPost0ProtocolVersion=V0_4

Configuration

To enable the “Generic Data Push” feature within the AXIS P8815-2 3D People Counter or AXIS People Counter application interface:

To make the device push data regularly to a remote destination:

1. Go to Setup > Counter and check that the device has a name.
2. On the AXIS P8815-2 3D People Counter, go to Setup > Reporting and turn on Report to server. On the AXIS People Counter, go to Settings > Reporting > Push Reporting.
3. In the URL field, enter the address of the server, for example https://example.com/path.
4. In the Send interval field, set how often you want the device to send data to the server. This setting is only available on the AXIS P8815-2 3D People Counter.
5. Optionally, enter a token to let the device authenticate itself to the server in the API token field.
6. Click Test connection.

   If the connection is successful, it’s indicated by a white check mark.
7. Click Save.

NOTE

The remote destination must be configured with HTTPS using a certificate validated by a public or custom root CA, which must also be installed on the Axis network camera in order to ensure proper handshake and encryption of the data. The certificate can be installed on the Axis device by accessing Device settings > TCP/IP > Security tab. Proceed to click on the + sign in CA certificates and upload the certificate.

This step can be ignored if using a root CA issued by a trusted certificate authority.

NOTE Once the SSL certificate has been installed and the data push feature enabled, the AXIS P8815-2 3D People Counter or AXIS People Counter will immediately begin sending stored counting data in the desired time interval to the recipient specified in the Reporting page.

Example Implementation

Axis has provided a sample HTTPS server written in Python that receives and parses the data sent from the AXIS P8815-2 3D People Counter or AXIS People Counter.

NOTE The following sample implementation requires protocol version is 0.4 of the Generic Data Push feature.

To make use of this example, it is necessary to install Python version 3 on the host from which the script will be run. Python is a standard component in Linux operating systems but needs to be manually installed on machines running Windows operating systems. The steps to install Python are outside the scope of this document, but the following page is a good resource to get Python installed.

https://wiki.python.org/moin/BeginnersGuide/Download

To use the sample HTTPS server:

1. Copy and paste the following code snippet in a notepad document and save it as push_server.py. Please ensure that the file is saved with .py as the file extension, otherwise it might not be possible to run the script.

```
#!/usr/bin/env python3

from http.server import BaseHTTPRequestHandler, HTTPServer
import logging
import json
import ssl
import socket
```

```
class Handler(BaseHTTPRequestHandler):
    def _set_response(self):
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()

    def do_GET(self):
        raise NotImplementedError("GET request not supported")

    def do_POST(self):
        # Gets the size of data
        content_length = int(self.headers["Content-Length"])
        # Gets the data itself
        post_data = self.rfile.read(content_length)
        logging.debug(
            "POST request,\nPath: %s\nHeaders:\n%s\n\nBody:\n%s\n",
            str(self.path),
            str(self.headers),
            post_data.decode("utf-8"),
        )
        self.parse_and_print(post_data)
        self._set_response()
        self.wfile.write("POST request for {}".format(
            self.path).encode("utf-8"))

    def parse_and_print(self, post_data):
        sensor_data = json.loads(post_data)
        sensor_name = sensor_data["sensor"]["name"]
        sensor_mac = sensor_data["sensor"]["serial"]
        sensor_timezone = sensor_data["sensor"]["timeZone"]
        sensor_IP = sensor_data["sensor"]["ipAddress"]
        timestamp = sensor_data["localSent"]
        if "data" not in sensor_data:
            # Test connection message, no data appended
            print("Connection from %s established successfully..." % sensor_IP)
        else:
            measurements = sensor_data["data"]["measurements"]
            for measurement in measurements:
                if measurement["items"] is None:
                    time_from = measurement["localFrom"]
                    time_to = measurement["localTo"]
                    print("Name: %s" % sensor_name)
                    print("Serial Number: %s" % sensor_mac)
                    print("Timezone: %s" % sensor_timezone)
                    print("Timestamp: %s" % timestamp)
                    print("From: %s" % time_from)
                    print("To: %s" % time_to)
                    print("In: 0")
                    print("Out: 0")
```

```
            print()
        else:
            counts = measurement["items"]
            in_counts = counts[0]["count"]
            out_counts = counts[1]["count"]
            time_from = measurement["localFrom"]
            time_to = measurement["localTo"]
            print("Name: %s" % sensor_name)
            print("Serial Number: %s" % sensor_mac)
            print("Timezone: %s" % sensor_timezone)
            print("Timestamp: %s" % timestamp)
            print("From: %s" % time_from)
            print("To: %s" % time_to)
            print("In: %s" % in_counts)
            print("Out: %s" % out_counts)
            print()


def run(server_class=HTTPServer, handler_class=Handler, port=4443):
    logging.basicConfig(level=logging.INFO)
    hostname = socket.gethostname()
    ip_address = socket.gethostbyname(hostname)
    server_address = ("", port)
    httpd = server_class(server_address, handler_class)
    httpd.socket = ssl.wrap_socket(
        httpd.socket,
        certfile="server-cert.pem",
        keyfile="server-key.pem",
        server_side=True,
    )

    print("Starting sample push server on", ip_address, "on port 4443...")
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()
    print("Stopping sample push server...")


if __name__ == "__main__":
    from sys import argv

    if len(argv) == 2:
        run(port=int(argv[1]))
    else:
        run()
```

2. As HTTPS is required to ensure encryption of the communication between the device and receiving server, a root CA certificate and server certificate must be generated. The root CA certificate needs to be installed on the Axis device, while the server certificate is stored on the machine hosting the sample HTTPS server script.

   For the purposes of this exercise, the following steps provide guidance on how to generate a self-signed certificate using yourself as a root certificate authority.

> **NOTE** This configuration should only be used for testing and development purposes. It is not intended for use in a production environment.

   a. Install OpenSSL from http://openssl.org

   b. Add the path to the openssl.exe executable to your PATH variable. Refer to http://openssl.org for other configuration properties.
   c. Navigate to the same directory where you saved the push_server.py file.
   d. To generate a CA private key, issue the following command from a command prompt:

```
openssl genrsa -des3 -out CA-key.pem 2048
```

   e. To generate the root CA certificate, issue the following command:

```
openssl req -new -key CA-key.pem -x509 -days 1000 -out CA-cert.pem
```

> **NOTE** You will be prompted for information which will be incorporated into the certificate, such as Country, City, Company Name, etc. Remember what information you entered as you may get prompted for this information again at a later stage.

   You will need CA-key.pem and CA-cert.pem to create and sign the server certificate:

   f. Generate a new key by issuing the following command:

```
openssl genrsa -des3 -out server-key.pem 2048
```

   Generate a self-signing request:

   g. Locate the openssl.cnf file is in your OpenSSL installation directory.

h. Copy the openssl.cnf file to the directory where you saved the push_server.py file.

i. Enter the following command:

```
openssl req –new –config openssl.cnf –key server-key.pem –out signreq.csr
```

NOTE You will be prompted for information which will be incorporated into the certificate, such as Country, City, Company Name, etc. It is necessary to input the server's IP address or FQDN when generating the signing request.

j. Self-sign the certificate using your CA-cert.pem certificate. Enter the following command (all in one line):

```
openssl x509 -req -days 365 -in signreq.csr
-CA CA-cert.pem -CAkey CA-key.pem -CAcreateserial -out server-cert.pem
```

k. If necessary, modify the location and name of the server certificate on line 56 of the sample HTTPS server script.

3. Install the **CA-cert.pem** on the Axis device by accessing **System > Security > CA certificates** menu and clicking on the + sign.

To run the HTTPS server, open a terminal or command line on the host machine and navigate to the location where the push_server.py file is located. Execute the following command:

```
python push_server.py
```

The terminal should display the following information upon successfully running the script:

```
Starting sample push server on <host_ip> on port 4443.
```

Note the IP address and port that the server is running on.

At this point, the AXIS P8815-2 3D People Counter or AXIS People Counter can be configured as per the steps described in the "Configuration" chapter

NOTE If the connection is unsuccessful, please ensure that an exception for incoming traffic on port 4443 has been added to the firewall for the machine running the server.

using the server's IP address and port in the URL field, followed by the time interval in which the data should be sent. Click on "Test connection" button to ensure that the connection to the receiving server is successful and save the settings. The terminal on the host machine will display a "Connection from <camera_IP> established successfully..." upon successful communication with the AXIS P8815-2 3D People Counter or AXIS People Counter.

After communication between the device and the receiving server has been established successfully, the following output containing the counting data and relevant information should be received according to the time resolution selected in the "Send interval" field of the AXIS P8815-2 3D People Counter or AXIS People Counter reporting settings page.

```
Name: axis-accc8ef3d92e
Serial Number: accc8ef3d92e
Timezone: Europe/Stockholm
Timestamp: 2021-03-18T15:55:12
From: 2021-03-18T15:54:00
To: 2021-03-18T15:55:00
In: 3
Out: 4
```

NOTE When the communication between the device and the receiving server is successfully established, the device will automatically push any historical data stored within the device for up to 90 days.

Additional Resources

AXIS P8815-2 3D People Counter Manual

https://help.axis.com/axis-p8815-2

AXIS People Counter Manual

https://help.axis.com/axis-people-counter