ARGO Generic AI Integration Sample Code Guide Version 1.2

1. System Architecture

System Architecture for sample code

1.1 Spark_Test_Prog

1.1.1 What is Spark_Test_Prog

Spark_Test_Prog is a program designed to simulate Argo behavior. It encompasses the following key features:

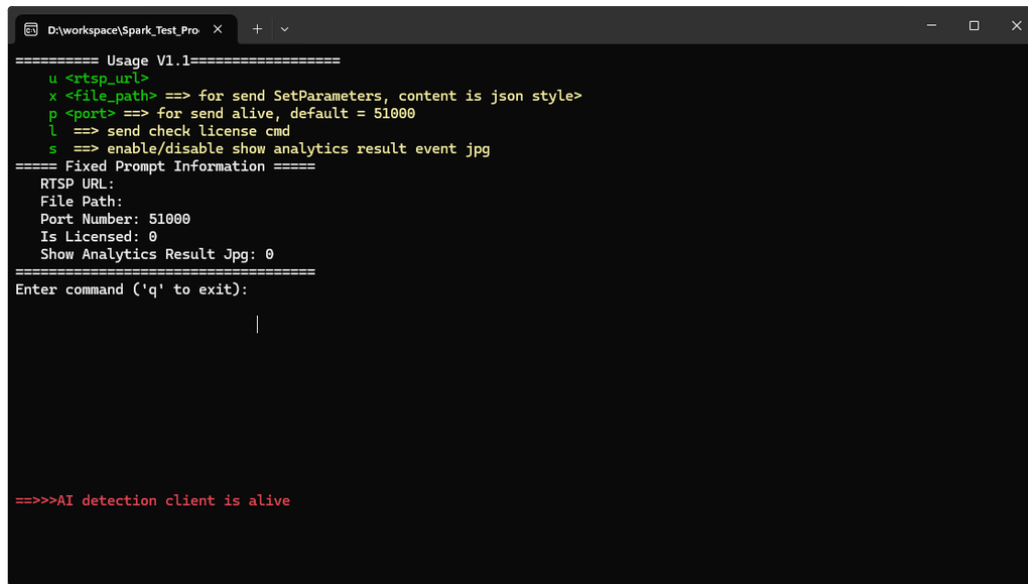1. Capability to receive RTSP (Real-Time Streaming Protocol) streams

2. Functionality to display video frames

3. All corresponding functions necessary for integrating Generic AI

This program serves as a valuable tool for developers looking to create their own Generic AI Wrapper. It facilitates AI integration with Argo by providing a testing environment that mimics Argo's behavior and interactions.

1.1.2 Primary Use Case

Developers can utilize Spark_Test_Prog to assist in the development and testing of Generic AI Wrappers intended for integration with Argo. This allows for a more streamlined development process and helps ensure compatibility before final implementation.

1.1.3 Interface

1.1.4 Spark_Test_Prog Command Instructions

RTSP URL Setting

Command: `u <rtsp_url>` Example: `u rtsp://admin:admin@192.168.1.219/stream1`

Sets the RTSP URL. After setting, it will automatically connect, decode, and send to shared memory. **Currently supports decoding up to 1920x1080.**

SetParameters JSON File

Command: `x <file_path>` Example: `x D:\param.json`

Sets the JSON content for SetParameters, used to configure the recognition algorithm.

Port Number Setting

Command: `p <port_num>` Example: `p 51000`

Sets the port for the corresponding **SampleWrapper** to activate. In Argo, each channel corresponds to a **SampleWrapper** for executing recognition. Current setting: Port number suffix is the channel number.

Example: To activate recognition for the 7th channel, set to 51007. The **SampleWrapper** will also be launched using the corresponding port.

License Check (optional function)

Command: `l`

Sends a command to **SampleWrapper** to check if the license exists.

Toggle Analytics Result Event Image Display

Command: `s`

Toggles the display of JPG image files in the analytics result event on or off.

Note: In Argo, each channel will open a corresponding **SampleWrapper** to perform recognition.

1.1.5 Spark_Test_Prog Operation Guide

Setup Procedure

1. Set the port number

2. Configure the RTSP URL

3. Set the JSON file for SetParameter

Important Notes

- It's crucial to set the channel number and port number before configuring the RTSP URL.

- After startup, the program will send an "Alive" message to **SampleWrapper** every 30 seconds to confirm its existence.

- All setup messages and results will be displayed in the lower half of the window.
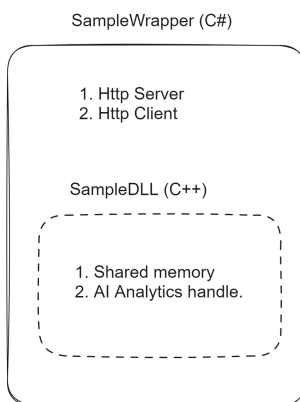
Operation Flow

1. Start Spark_Test_Prog

2. Use the `p` command to set the port number

3. Use the `u` command to set the RTSP URL

4. Use the `x` command to set the SetParameter JSON file

5. Monitor the lower half of the window for setup messages and results

Continuous Operation

- The program will automatically send "Alive" messages every 30 seconds

- Use other commands as needed (e.g., `l` for license check, `s` to toggle analytics result display)

- Continue monitoring the lower half of the window for ongoing messages and results

Remember to refer to the command instructions for specific syntax of each command.

1.2 SampleWrapper

SampleWrapper (C#)

1. Http Server
2. Http Client

SampleDLL (C++)

1. Shared memory
2. AI Analytics handle.

1.2.1 Overview

SampleWrapper is a program implemented in C# that serves as an interface between ARGO and AI recognition functionalities.

1.2.2 Architecture

1. C# Layer:
   - Implements HTTP client and HTTP server functionalities

- Handles communication with ARGO
2. C++ DLL Layer:
    - Manages Shared Memory
    - Contains AI recognition-related functionalities

1.2.3 Key Features

1. HTTP Client/Server: Implemented in the C# layer for communication.
2. Shared Memory: Implemented in the C++ DLL layer for efficient data sharing.
3. AI Recognition: Should be implemented in the C++ DLL layer to utilize image data from Shared Memory.

1.2.4 Data Flow

1. Image data is received through Shared Memory.
2. AI recognition is performed on this data in the C++ DLL layer.
3. Recognition results are returned to the C# layer.
4. Results are sent to ARGO via the HTTP server.

1.2.5 Usage by ARGO

- ARGO launches a separate instance of SampleWrapper for each channel.
- Each instance is started with different parameters.

Example: To start AI recognition for Channel 7, ARGO would execute:

```
1  SampleWrapper.exe port=51007
```

1.2.6 Development Guidelines

When developing AI recognition features:

1. Implement the core functionality in the C++ DLL layer.
2. Use Shared Memory to access image data for recognition.
3. Ensure results are properly communicated back to the C# layer for transmission to ARGO.

2. ARGO AI Integration Specifications

2.1. Communication APIs

2.1.1 Set Algorithm Parameters (ARGO → SampleWrapper.exe)

- Method: POST
- URL: http://127.0.0.1:{port_num}/SetParameters
- MIME: application/json
- HTTP Code: Only 200
- Content: See detailed explanation

> Note: The last digit of the port number represents the channel number. For example, 51007 for channel 7.

2.1.2. Program Channel Alive (ARGO → SampleWrapper.exe)

- Method: GET

- URL: http://127.0.0.1:{port_num}/Alive

- MIME: plain/text

- HTTP Code: Only 200

- Content: Empty string

> ▤ Note: The last digit of the port number represents the channel number. For example, 51007 for channel 7.

2.1.3. Program License Exist (ARGO → SampleWrapper.exe)

- Method: GET

- URL: http://127.0.0.1:{port_num}/GetLicense

- MIME: plain/text

- HTTP Code: Only 200

- Content: Empty string

> ▤ Note: The last digit of the port number represents the channel number. For example, 51007 for channel 7.

2.1.4. Event Parameters (SampleWrapper.exe → ARGO)

- Method: POST

- URL: http://127.0.0.1:{portnum on SetParameters json}/PostAnalyticsResult

- MIME: application/json

- HTTP Code: Only 200

- Content: See detailed explanation

> ▤ Note: The last digit of the port number represents the channel number. For example, 51007 for channel 7.

2.2. Suggested Detailed Settings (JSON)

2.2.1. Set Algorithm Parameters

```
 1  {
 2      "version": "1.2",
 3      "analytics_event_api_url":"http://127.0.0.1:9901/PostAnalyticsResult",
 4      "image_width": 1280,
 5      "image_height":720,
 6      "jpg_compress":20,
 7      "rois":[
 8          {"sensitivity": 50,
 9          "threshold":   50,
10          "rects":
11          [
12              {"x": 100 , "y":100},
13              {"x": 200 , "y":200},
14              {"x": 300 , "y":300},
15              {"x": 400 , "y":400}
16          ]},
17          {"sensitivity": 50,
18          "threshold":   50,
```

```
19          "rects":
20          [
21              {"x": 500 , "y":500},
22              {"x": 600 , "y":600},
23              {"x": 700 , "y":700},
24              {"x": 800 , "y":800}
25          ]}
26      ]
27  }
```

★The supported settings allow up to 10 ROIs, with each ROI containing a maximum of 10 points.

2.2.2 Recognition Event Results

```
1  {
2    "version":"1.2",
3    "port_num": 1,
4    "keyframe": "/9j/4AAQSkZJR...",
5    "timestamp": 15003215760000,
6    "rois_rects": [
7       [10, 20, 30, 40],
8       [440, 104, 608, 214],
9       [8, 8, 300, 300]
10    ]
11  }
```

> ⚠ • For deep integration, please define your own algorithm parameter settings and provide   documentation.
>
>   • "keyframe" is in JPG format and then encoded into a Base64 JPG string.
>
>   • "version" is used to define the version number of the transmitted JSON.

2.3. Shared Memory Settings

Using Windows shared memory, currently supporting a maximum resolution of 1920x1080.

```
1  const int MMF_DATA_HEADER = 0x1234;
2  const int MMF_DATA_FOOTER = 0x4321;
3  struct MMF_Data_Generic
4  {
5      __int64 header = MMF_DATA_HEADER;
6      //video status : 0=no use , 1=new frame, 2=detection got frame
7      int image_status = 0;
8      //resolution
9      int image_width = 0;
10     int image_height = 0;
11     //video size
12     int image_size = 0;
13     //timestamp in ARGO FileTime style
14     uint64_t  timestamp = 0;
15     //video data
16     unsigned char* image_data[1920 * 1080 * 3];
17     __int64 footer = MMF_DATA_FOOTER;
18  };
```

* `image_status` : 0 = no use, 1 = new frame, 2 = detection got frame
* `timestamp` : in ARGO FileTime style

  ARGO FileTime Style is base on Windows File time.

```
1  inline uint64_t GetFileTimeNow()
2  {
3      FILETIME ft;
4      GetSystemTimeAsFileTime(&ft);
5      ULARGE_INTEGER ularge;
6      ularge.LowPart = ft.dwLowDateTime;
7      ularge.HighPart = ft.dwHighDateTime;
8      return ularge.QuadPart;
9  }
```

- Shared mem Name Rule : ChannelFrame_%d
  %d is PortNum

2.4. Execution and Naming Conventions

Executable Naming Flexibility

SampleWrapper.exe can be renamed to any desired name. ARGO supports different program names for integration. However, the parameter format remains fixed and must be adhered to.

Parameter Format

The parameter format is standardized and must be maintained regardless of the executable name:

```
1  [executable_name].exe port=[port_number]
```
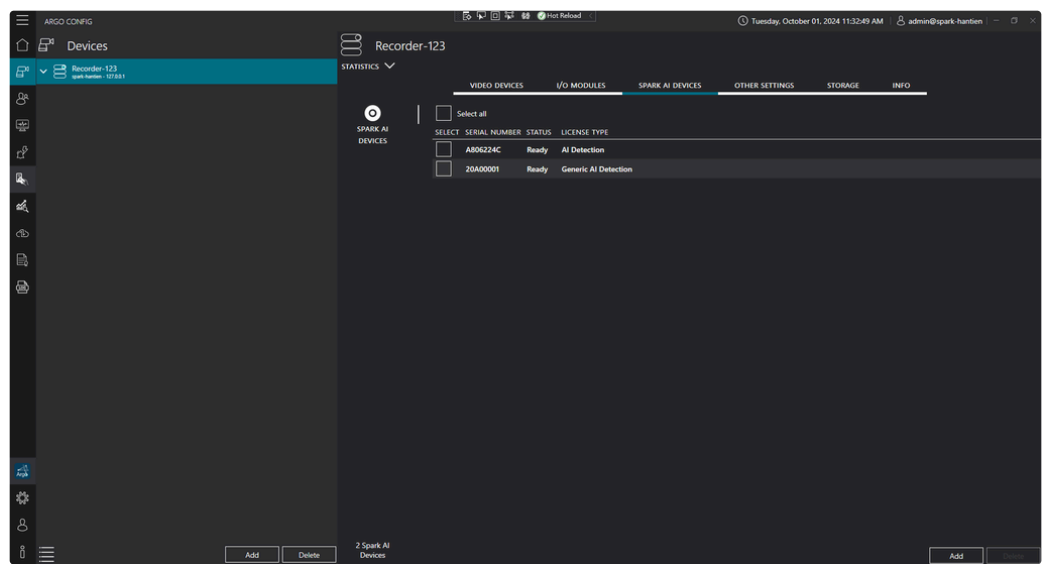
Example

```
1  yourprogramname.exe port=51007
```

Important Notes

1. While the executable name is flexible, ensure that ARGO is configured to call the correct filename.
2. The port number format (51000 + channel_number) must be strictly followed.
3. Always use the "port=" parameter as shown in the example.
4. Ensure that your program correctly interprets and uses the provided port number for communication with ARGO.
5. The executable must be a **signed application** to ensure it runs correctly and is not flagged as malicious by Windows or antivirus software.

3. ARGO AI Integration UI on Client and Config Client

3.1. Add Generic AI Detection on Spark AI Devices

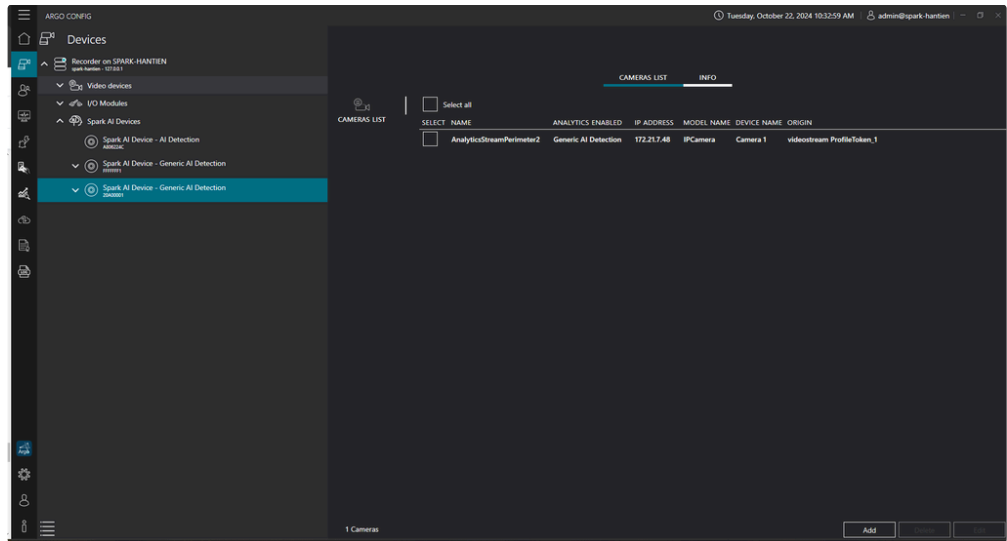Users can integrate AI into your Generic AI Device on the Spark AI Devices Page by clicking the Add button

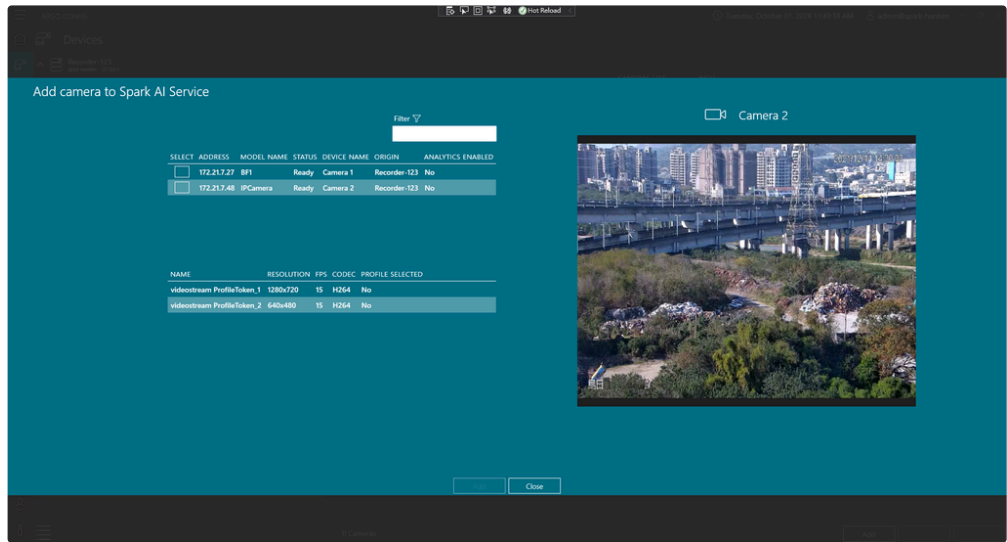**Steps to Manually Add a Generic AI Device**

1. Select **Detection Type** as **Generic AI Detection**.

2. Enter the Class name detected by your recognition program in the **Generic AI Detection Class Name** field.

3. Enter the name of your recognition program (e.g., *SampleWrapper.exe*) in the **Analytics Application Name** field.

4. Use the **Browse** button to select the recognition program's ZIP file for upload.

5. Choose the **License Serial Number** to use.

   ○ This License Serial Number corresponds to the **Generic AI Detection License** for Argo.

   ○ If needed, please contact customer support for assistance.

6. Enter the **port** used by the recognition program for Argo in the **Analytics Application Port** field.

7. Enter the **HTTP port** used by the recognition program to push data to Argo in the **HTTP Port** field.

8. Click Add Button to add Generic AI Device.

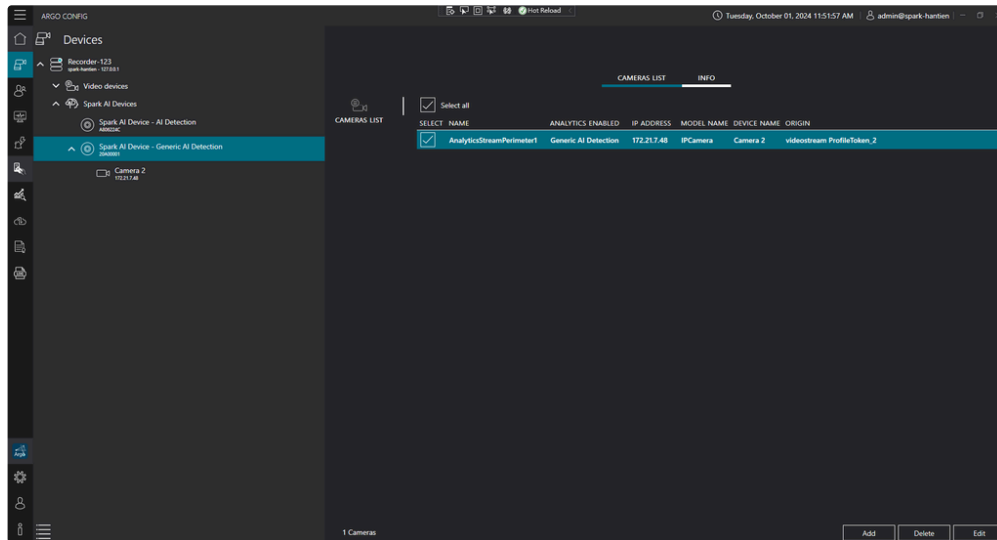## 3.2. Analyticsstreamperimeter on Spark AI Device - Generic AI Detection

1. Select Spark AI Device- Generic AI Detection , you will see Add button on the right side.
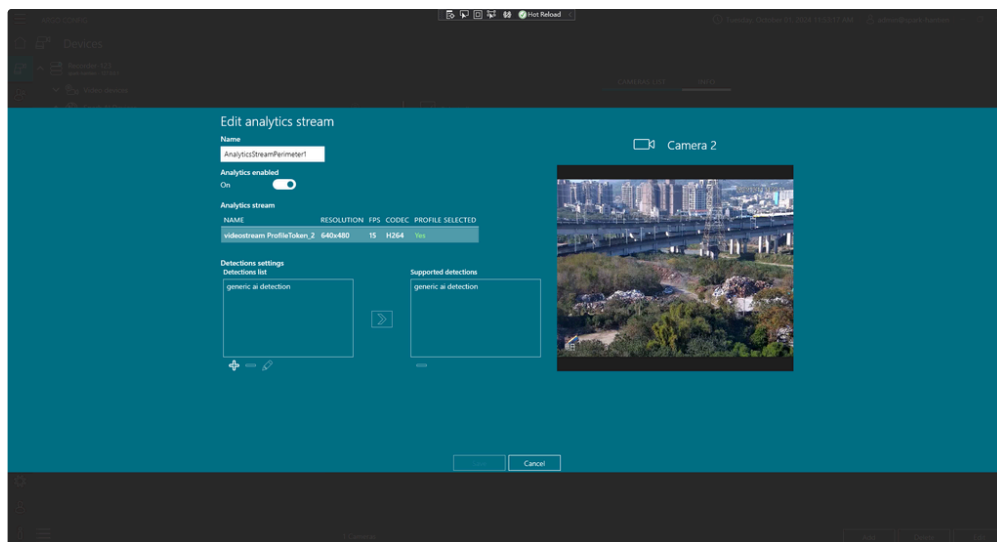

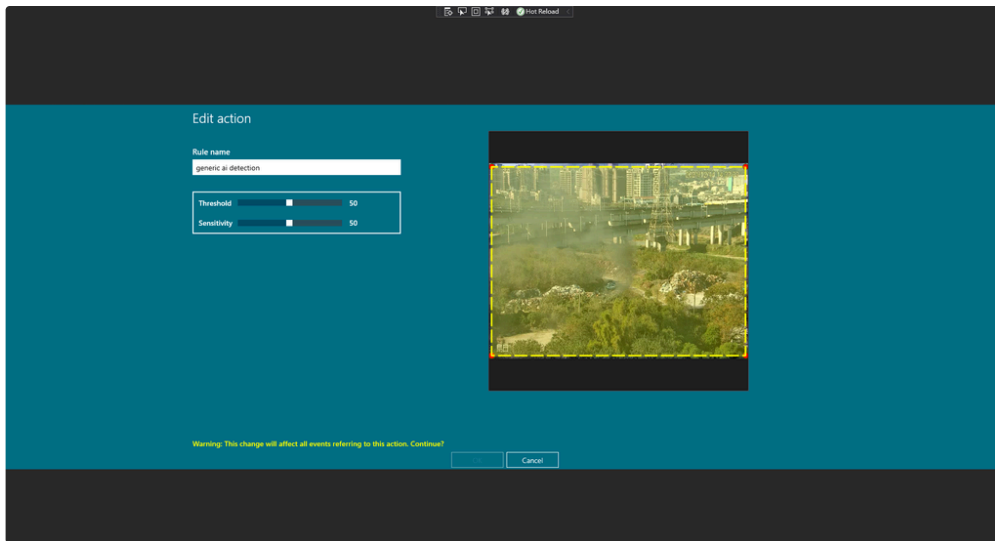2. Click Add button to Add Analytcisstreamperimeter

3. Users can select a camera device and stream for analysis . And Click add to create Analyticsstreamperimeter on Generic AI Detecion Device.
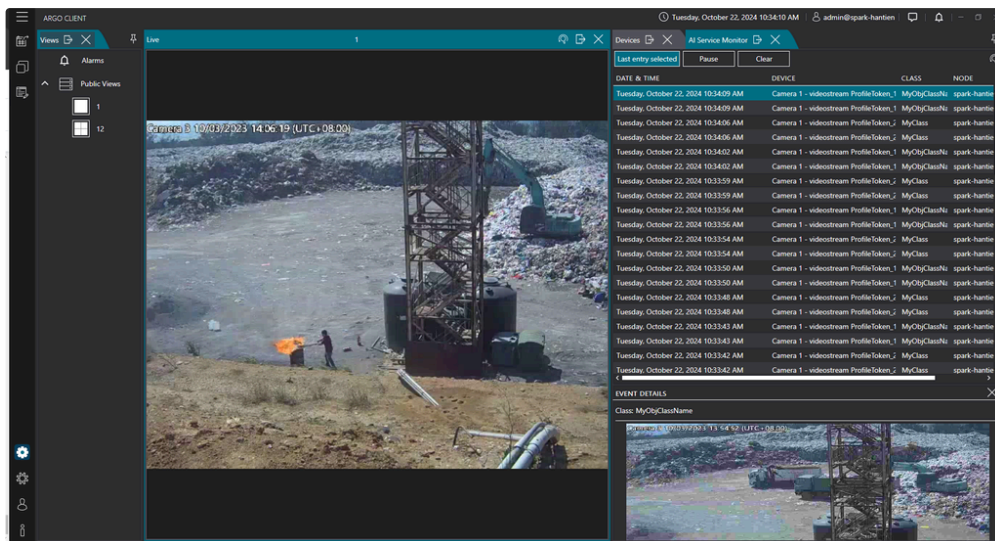

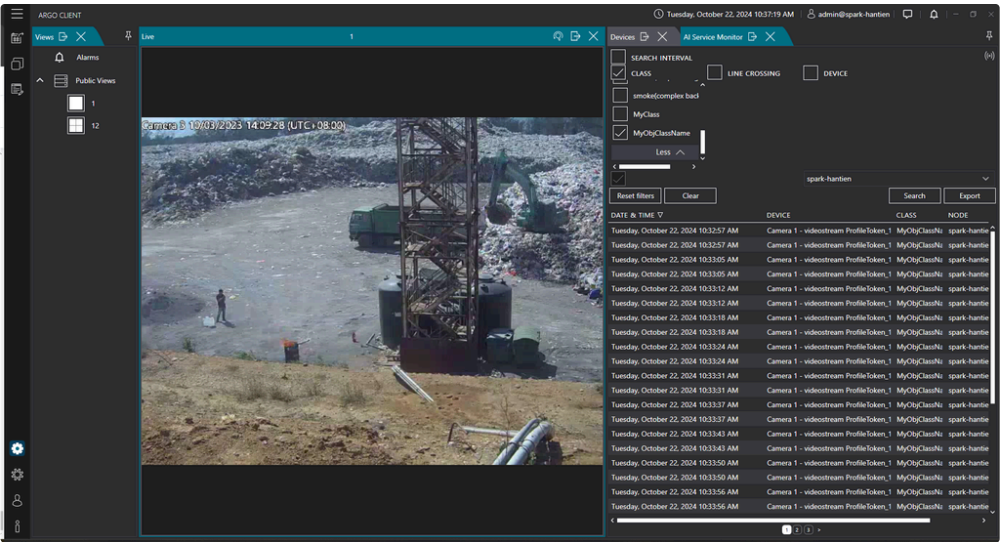4. User can edit Analyticsstreamperimeter by clicking Edit Button.

5. Users can click the **+** button to create a new detection, the **pen** button to edit detections, the **-** button to delete detections, and the **>** button to use detections.

6. Users can edit detection settings, such as threshold, sensitivity, and ROI region, on the Edit page.

7. After these settings are configured, the analytics will start to run.

8. Events pushed by the recognition program will be displayed in the **AI Service Monitor** on the client side

9. The user can search using the name previously entered in the "**Generic AI Detection Class Name**" field.

4. License

The Generic AI Detection License serial number and channel count are provided by Taiwan Spark Technology Co.,Ltd . If you wish to use the above features, please contact the appropriate representative.