App Development Guidelines [Protocol Aspect]

OpenSDK

2023-06-27
Version: 1.0

Contact Information
Hanwha Vision Co., Ltd.
Hanwha Vision 6, Pangyo-ro 319beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, 13488, KOREA
www.hanwhavision.com

Hanwha Vision America
500 Frank W. Burr Blvd. Suite 43 Teaneck, NJ 07666
hanwhavisionamerica.com

Hanwha Vision Europe
Heriot House, Heriot Road, Chertsey, Surrey, KT16 9DT, United Kingdom
hanwhavision.eu

Hanwha Vision Middle East FZE
Jafza View 18, Office 2001-2003, Po Box 263572, Jebel Ali Free Zone, Dubai, United Arab Emirates
www.hanwhavision.com/ar

Table of Contents

1. Scope

This document covers what app developers need to consider for handling events and metadata delivery from protocol point of view. Also how app configuration api can be exposed.

2. Introduction

When developing new application for the camera, in addition the functionality, app developers need to consider following topics for smooth integration with VMS/Client applications.

1. Event Delivery
2. Metadata Delivery
3. Configuration API

3. Event delivery

Camera supports following event delivery mechanism to deliver camera and app events to the connected VMS / other clients.

- SUNAPI
  - Eventstatus txt format (Old format)
  - Eventstatus json format (Old format)
  - Eventstatus Schemabased Dynamic Event txt format
  - Eventstatus Schemabased Dynamic Event json format
- ONVIF
  - ONVIF Event service PULL
  - ONVIF Event service Base Notification
  - RTP Metadata
- MQTT

In order to deliver app genereated events to the client connected to the camera using SUNAPI or ONVIF, app needs to notify the event based on the steps below.

1. Registering the dynamic event schema
   This is required to notify client, what events can be expected from the app and what parameters and values are supported in the event.
   Can refer to OpenSDK API Documentation **Add dynamic Event schema** section.
2. Sending dynamic event
   When the event is detected by the app, app can send that event through the opensdk to the camera and this will be delivered in camera supported prototocols (SUNAPI and ONVIF).
   Can refer to OpenSDK API Documentation **Send dynamic event** section.

3.1. Dynamic Event Schema

When registering event schema using the opensdk, following fields needs to be considered

3.1.1. Type

When **Type** is set as **META**, event can follow any xml format to send the notification and the notification is delivered only in Sunapi Dynamic Event Json Format as it is[camera does not parse the content of the xml]

EventNotification	Support
SUNAPI-Eventstatus TXT	No
SUNAPI-Eventstatus Json	No
SUNAPI-Eventstatus Schemabased TXT	No
SUNAPI-Eventstatus Schemabased Json	Yes
ONVIF-PULL Notification	No
ONVIF-BASE Notification	No

Example meta type schema for licenseplate reading app,

```
<tt:MessageDescription>
    <tt:Source>
        <tt:SimpleItemDescription Name="VideoSourceToken" Type="tt:ReferenceToken"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItemDescription Name="LicensePlateNumber" Type="xsd:string"/>
    </tt:Data>
</tt:MessageDescription>
```

NOTE | Minify(remove spaces) above xml before passing to opensdk.

This is notified in evenstatus dynamic event like below,

```
{
    "EventName": "OpenSDK.CarRec.LicesnePlateNumber",
    "Time": "2021-12-20T08:57:36.883+00:00",
    "Source": {
        "Channel": 0,
        "AppName": "CarRec",
        "AppID": "CarRec",
        "AppEvent": "LicensePlateNumber",
        "Type": "Meta"
    },
    "Data": {
        "Info": "<tt:Message><tt:Source><tt:SimpleItem Name=\"VideoSourceToken\" Value=\"
0\"/></tt:Source><tt:Data><tt:SimpleItem Name=\"LicensePlateNumber\" Value=\"ABC-1234
\"/></tt:Data></tt:Message>"
    }
}
```


When **Type** is set as **EVENT**, event follows onvif event xml template [Onvif Core Specification Message Description Language section 9.4.3] as below, when event is marked as **EVENT** type the source and data sections are parsed and provided both in, Sunapi Dynamic Event and ONVIF Event service.

EventNotification	Support
SUNAPI-Eventstatus TXT	No
SUNAPI-Eventstatus Json	No
SUNAPI-Eventstatus Schemabased TXT	No
SUNAPI-Eventstatus Schemabased Json	Yes
ONVIF-PULL Notification	Yes
ONVIF-BASE Notification	Yes

Example event type schema for objectdetection,

```
<tt:MessageDescription IsProperty="true">
    <tt:Source>
        <tt:SimpleItemDescription Name="VideoSourceToken" Type="tt:ReferenceToken"/>
        <tt:SimpleItemDescription Name="RuleName" Type="xsd:string"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItemDescription Name="State" Type="xsd:boolean"/>
        <tt:SimpleItemDescription Name="ClassTypes" Type="tt:StringList"/>
    </tt:Data>
</tt:MessageDescription>
```

NOTE Minify(remove spaces) above xml before passing to opensdk.

This event is notified in SUNAPI schema based eventstatus like below

```
{
    "EventName": "OpenSDK.WiseAI.ObjectDetection",
    "Time": "2022-07-27T07:50:02.179+00:00",
    "Source": {
        "Channel": 1,
        "AppName": "WiseAI",
        "AppID": "WiseAI",
        "AppEvent": "ObjectDetection",
        "Type": "Event",
        "VideoSourceToken": "VideoSourceToken-1",
        "RuleName": "ObjectDetectionRule-1"
    },
    "Data": {
        "ClassTypes": "",
        "State": false
    }
}
```

3.1.2. IsOnOffEvent

This parameter used to define events which has **State** parameter in Data section of the XML. If a event is marked as **isOnOffEvent**, that event will be additionally delivered in SUNAPI Eventstaus txt notification (old format) Example event type schema for objectdetection,

```
<tt:MessageDescription IsProperty="true">
    <tt:Source>
        <tt:SimpleItemDescription Name="VideoSourceToken" Type="tt:ReferenceToken"/>
        <tt:SimpleItemDescription Name="RuleName" Type="xsd:string"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItemDescription Name="State" Type="xsd:boolean"/>
        <tt:SimpleItemDescription Name="ClassTypes" Type="tt:StringList"/>
    </tt:Data>
</tt:MessageDescription>
```

NOTE Minify(remove spaces) above xml before passing to opensdk.

EventNotification	Support
SUNAPI-Eventstatus TXT	Yes
SUNAPI-Eventstatus Json	Yes
SUNAPI-Eventstatus Schemabased TXT	Yes
SUNAPI-Eventstatus Schemabased Json	Yes
ONVIF-PULL Notification	Yes
ONVIF-BASE Notification	Yes

3.1.3. isPropertyEvent

When registering the schema **isPropertyEvent** field is is can be optionally enabled, when the event needs to be notified as **Property** event in ONVIF **GetEventProperties** api response. You can refer to Onvif Core Specification section 9.4.2 to understand property events.

3.2. Sending Dynamic Event

Event XML should follow the XML Schema registered initially, example xml notification message for above objectdetection event schema, would look like,

```
<tt:Message UtcTime="2022-07-20T12:24:57.628">
    <tt:Source>
        <tt:SimpleItem Name="VideoSourceToken" Value="VideoSourceToken-0"/>
        <tt:SimpleItem Name="RuleName" Value="ObjectDetectionRule"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItem Name="State" Value="true"/>
        <tt:SimpleItem Name="ClassTypes" Value="Person Vehicle"/>
    </tt:Data>
</tt:Message>
```

Following fields need to be updated when notifying the event using opensdk **send_dynamicEventSchema** API.

3.2.1. type

This parameter needs to be set to inform this notification is of type **META: 0** or **EVENT: 1**

3.2.2. State

If the event is a **true** / **false** type event, it can be set to 0 for **false** and 1 for **true**. If the event is not a true false event it can be set to 0.

3.2.3. isOnOffEvent

This parameter needs to be enabled when the xml has **State** parameter in it.

3.2.4. channel

This parameter is used to notify, the event is applicable for which channel(Video Source) of the camera, index starts with **0**.

3.2.5. isPropertyEvent

Optionally app can send this feild as **1** if state changes needs to be maintained, else **0** can be sent.

3.2.6. metaEventSourceToDataDeleteorMaintain

When this parameter **metaEventSourceToDataDeleteorMaintain** is set to **0** and **state** parameter in event notifiation structure is **0** the previous xml content stored in the camera backend will be cleared. (ie., when state is false, the xml content is cleared)

When **metaEventSourceToDataDeleteorMaintain** is set to **1**, B.E will maintain the old xml value, irrespective of **state** parameter value in event notification structure. (In this case xml content will not be cleared)

3.3. Predefined Field names in event

For client intergration support, its recommended to follow some of the **predefined** fields in **Source** and **Data** section when applicable for smooth integration with existing clients.

In **Source** section of the xml following fields are defined,

Field Name	Type	Description
VideoSourceToken or VideoSource	tt:ReferenceToken	Used to identify the video source for which the event is applicable, for new events **VideoSource** is more preferred.
RuleName or Rule	xsd:string	Used to identify the detection rule uniquely, example if there are multiple regions configured for detection, each region can have its own rule name to uniquely identify that region. For new events **Rule** is more preferred.
AnalyticsConfiguration	tt:ReferenceToken	Used to notify the analytics configuration token.(This is optional field and not commonly used)

In **Data** section of the xml following fields are defined,

Field Name	Type	Description
State	xs:boolean	Used to notify the event state.
ObjectId	tt:StringList	This field can be used to notify object ids which triggered this event. Its space separated values example "233 234 241".
ClassTypes	tt:StringList	This field can be used to notify which class types triggered this event, example "Person Vehicle Face"
Count	xs:integer	This field can be used to notify any count information in the event.
ObjectType	xs:string	To notify which object type triggered this event.
Direction	xs:string	This field can be used to notify direction string. Example RightToLeftIn or LeftToRightIn in case of object counting.
Action	tt:StringList	To notify which actions triggered the event, eg incase of Area based rule, it can be Enter, Exit, Loitering etc.,
Path	xs:string	Used to provide REST API path in configuration change event, to notify client which api triggered this.

Field Name	Type	Description
ImageUri	xs:anyURI	Used to pass object image url in the event
SceneImageUri	xs:anyURI	Used to pass scene image url in the event
BoundingBox	tt:Rectangle	Used to pass the bounding box information in the event[Normalized values -1 to 1] as ElementItem Example element item `<tt:ElementItem Name="BoundingBox"> <tt:Rectangle top="0.5" left="-0.5" bottom="-0.5" right="0.5" /> </tt:ElementItem>`
LicensePlateNumber	xs:string	Used to pass the licenseplate number
LicensePlateInfo	tt:LicensePlateInfo	Used to pass the licenseplate number as ElementItem (Complex type) along with other information regarding the license plate. Example element item `<tt:ElementItem Name="LicensePlateInfo"> <tt:LicensePlateInfo> <tt:PlateNumber Likelihood="0.8">33T2332</tt:PlateNumber> <tt:PlateType Likelihood="0.75608">Normal</tt:PlateType> <tt:CountryCode Likelihood="0.75608">US</tt:CountryCode> </tt:LicensePlateInfo> </tt:ElementItem>`
VehicleInfo	tt:VehicleInfo	Used to pass the Vehicle information as ElementItem Example element item `<tt:ElementItem Name="VehicleInfo"> <tt:VehicleInfo> <tt:Type Likelihood="0.8">Car</tt:Type> </tt:VehicleInfo> </tt:ElementItem>`

3.4. Event XML to JSON Conversion

Since app will be notifying the event in XML format, camera needs to convert this XML into JSON when delivering event over SUNAPI **Eventstatus Schemabased JSON**, during conversion basic data types are preserved as per below table.

XML Type	JSON Type
xs:float	number
xs:integer	number
xs:boolean	boolean
all other types	string

For example following objectdetection event is converted to JSON Like below, when converting to json under source section, additional information like channel, appid, app event and type is added by default.

XML EVENT

```
<tt:Message UtcTime="2022-07-20T12:24:57.628">
    <tt:Source>
        <tt:SimpleItem Name="VideoSourceToken" Value="VideoSourceToken-0"/>
        <tt:SimpleItem Name="RuleName" Value="ObjectDetectionRule"/>
    </tt:Source>
    <tt:Data>
        <tt:SimpleItem Name="State" Value="true"/>
        <tt:SimpleItem Name="ClassTypes" Value="Person Vehicle"/>
    </tt:Data>
</tt:Message>
```

JSON EVENT (Afer conversion)

```
{
    "EventName": "OpenSDK.WiseAI.ObjectDetection",
    "Time": "2022-07-20T12:24:57.628",
    "Source": {
        "Channel": 1,
        "AppName": "WiseAI",
        "AppID": "WiseAI",
        "AppEvent": "ObjectDetection",
        "Type": "Event",
        "VideoSourceToken": "VideoSourceToken-0",
        "RuleName": "ObjectDetectionRule"
    },
    "Data": {
        "ClassTypes": "Person Vehicle",
        "State": false
    }
}
```

4. Metadata Delivery

For metadata, ONVIF Metadata schema [Onvif Metadata Schema] format is used, to understand the onvif metadata format Onvif analytics specification Scene Description section.

4.1. General Information on Metadata

Basic metadata with frame information would look like below, a frame can have mutiple objects and each objects should have its own bounding box information and a unique object id. If the app supports class type idenfication, then it can optionally include the **Class** related information and other optional fields like vehicleinfo, HumanFace etc., . In addition to the class type information, sample object captured can also be notified in **ImageRef** with relative url.

```
<?xml version="1.0" encoding="UTF-8"?>
<tt:MetadataStream
    xmlns:tt="http://www.onvif.org/ver10/schema"
    xmlns:ttr="https://www.onvif.org/ver20/analytics/radiometry"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
    xmlns:tns1="http://www.onvif.org/ver10/topics"
    xmlns:tnssamsung="http://www.samsungcctv.com/2011/event/topics"
    xmlns:fc="http://www.onvif.org/ver20/analytics/humanface"
    xmlns:bd="http://www.onvif.org/ver20/analytics/humanbody">
    <tt:VideoAnalytics>
        <tt:Frame UtcTime="2022-08-16T10:16:09.024Z">
            <tt:Transformation>
                <tt:Translate x="-1.0" y="1.0"/>
                <tt:Scale x="0.001042" y="-0.001852"/>
            </tt:Transformation>
            <tt:Object ObjectId="1416469">
                <tt:Appearance>
                    <tt:Shape>
                        <tt:BoundingBox left="1435.0" top="186.0" right="1481.0"
bottom="223.0"/>
                        <tt:CenterOfGravity x="1458.0" y="204.5"/>
                    </tt:Shape>
                    <tt:Class>
                        <tt:Type Likelihood="0.9">Vehicle</tt:Type>
                    </tt:Class>
                    <tt:ImageRef>
/download/objectid_1416469_1660612569024.jpg</tt:ImageRef>
                </tt:Appearance>
            </tt:Object>
        </tt:Frame>
    </tt:VideoAnalytics>
</tt:MetadataStream>
```

NOTE	The image naming follows below naming rule, objectid_<object id>_<timestamp in Millisecs>.jpg

To understand Transformation and Bounding box information, 5.2.2 Spatial Relation section of Onvif analytics specification can be referred.

Objects within a frame can also have **parent child** relation, one example is Vehicle object and Licenceplate object, Example of Parent child metadata, check the highlighted line in xml below,

```
<?xml version="1.0" encoding="UTF-8"?>
<tt:MetadataStream xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:ttr="https://www.onvif.org/ver20/analytics/radiometry" xmlns:wsnt="http://docs.oasis-
open.org/wsn/b-2" xmlns:tns1="http://www.onvif.org/ver10/topics"
xmlns:tnssamsung="http://www.samsungcctv.com/2011/event/topics"
xmlns:fc="http://www.onvif.org/ver20/analytics/humanface"
xmlns:bd="http://www.onvif.org/ver20/analytics/humanbody">
    <tt:VideoAnalytics>
        <tt:Frame UtcTime="2019-06-10T12:24:57.321">
            <tt:Transformation>
                <tt:Translate x="-1.0" y="-1.0"/>
                <tt:Scale x="0.003125" y="0.00416667"/>
            </tt:Transformation>
            <tt:Object ObjectId="12">
                <tt:Appearance>
                    <tt:Shape>
                        <tt:BoundingBox left="20.0" top="180.0" right="100.0"
bottom="30.0"/>
                        <tt:CenterOfGravity x="60.0" y="80.0"/>
                    </tt:Shape>
                    <tt:Class>
                        <tt:Type Likelihood="0.9">Vehicle</tt:Type>
                    </tt:Class>
                </tt:Appearance>
            </tt:Object>
        </tt:Frame>
        <tt:Frame UtcTime="2019-06-10T12:24:57.721">
            <tt:Transformation>
                <tt:Translate x="-1.0" y="-1.0"/>
                <tt:Scale x="0.003125" y="0.00416667"/>
            </tt:Transformation>
            <tt:Object ObjectId="14" Parent="12">
                <tt:Appearance>
                    <tt:Shape>
```

```
                    <tt:BoundingBox left="40.0" top="150.0" right="70.0"
bottom="100.0"/>
                    <tt:CenterOfGravity x="57.0" y="130.0"/>
                </tt:Shape>
                <tt:Class>
                    <tt:Type Likelihood="0.6">LicensePlate</tt:Type>
                </tt:Class>
            </tt:Appearance>
        </tt:Object>
    </tt:Frame>
</tt:VideoAnalytics>
</tt:MetadataStream>
```

Regarding sending metadata from APP, refer OpenSDK API document **Send Metadata to RTP Client** section.

4.1.1. Extending class types or elements

Class/Type and sub types like VehicleInfo/Types can be extented with new values in addition the values defined in the metadata specfication Onvif Metadata Schema.

Other custom fields in the metadata can be included in the extension section where its applicable or at the end of the element or attribute section already defined.

4.2. Metadata Schema

Once finalizing what metadata to be sent, metadata scheme can be notified to camera using OpenSDK refer OpenSDK API document section **Set Metaframe Schema**

Since we follow onvif metadata format by default we could use below xsd schema or Onvif Metadata Schema file full file content.

```
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema">
<xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="https://www.onvif.org/ver10/schema/metadatastream.xsd"/>
            <xs:element name="MetadataStream" type="tt:MetadataStream"/>
</xs:schema>
```

If any new complexTypes are added it can be added to the schema.

4.3. Metadata Schema Capability

App needs to register the schema capabilty to inform the camera, what values are expected for each field

in metadata and also what dependency relationship one field has on another. Schema capability is expressed as simple xpath expressions and conditions.

OpenSDK API document **Set Metaframe Options** section can be referred for opensdk API to send the capability.

Example Capability json data with proper escape characters,

```
"{\"AppID\":\"ID\",\"Capabilities\":[{\"xpath\":\"//tt:VideoAnalytics/tt:Frame/tt:Object/tt
:Appearance/tt:LicensePlateInfo/tt:CountryCode\",\"type\":\"xs:string\",\"enum\":[\"KR\",\"
US\",\"CN\",\"FN\",\"IN\"]}]}"
```

To understand how client will use this capability with detailed example, Sunapi Opensdk Document **Metaframe Capability** section can be referred.

4.4. Namespaces Used in Metadata

Namespace needs to be added in the Metadata and Schema for clients to parse the xml properly. Commonly used namepace in metadata and schema are listed below. If any custom namespace is used in Framemetadata, that should also be added to the xml.

NameSpace	Description
xmlns:xs="http://www.w3.org/2001/XMLSchema"	General xml elements and datatypes refer this. **Note** > Used only when providing Schema and not used in FrameMetadata
xmlns:tt="http://www.onvif.org/ver10/schema"	Onvif defined datatypes and elements refer this. **Note** > Used both in Schema and FrameMetadata

NameSpace	Description
xmlns:fc="http://www.onvif.org/ver20/analytics/humanface"	Onvif defined humanface related schema. **Note** > Used both in Schema and FrameMetadata
xmlns:bd="http://www.onvif.org/ver20/analytics/humanbody"	Onvif defined humanbodya related schema. **Note** > Used both in Schema and FrameMetadata
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"	Eventstream in Metadata follows this namespace. **Note** > Not used in FrameMetadata Used only when EventMetadata is sent
xmlns:ttr="https://www.onvif.org/ver20/analytics/radiometry"	Eventstream in Metadata follows this namespace when Thermal events are notified. **Note** > Not used in FrameMetadata Used only when EventMetadata is sent

NameSpace	Description
xmlns:tns1="http://www.onvif.org/ver10/topics"	Eventstream in Metadata follows this namaspace for delivering event topic. **Note** > Not used in FrameMetadata Used only when EventMetadata is sent

5. APP Configuration API

App can have its own webpage and it can also optionally expose API to configure the APP. There are several options for exposing API to configure the APP, like grpc, basic cgi request etc., but we recommend **OpenAPI** based REST API support.

> **NOTE** Its not compulsory to support **OpenAPI** based API, its only a recommendation.

You can find more about specification and tools here [OpenAPI Specification , OpenAPI Generator]

References

- [1] Wisenet OpenPlatform SDK API Document
- [2] OpenAPI Tools
- [3] OpenAPI Specification
- [4] SUNAPI Event document
- [5] ONVIF Core Specification
- [6] ONVIF Analytics Specification
- [7] ONVIF Metadata Schema
- [8] Sunapi Opensdk document