NICE Data Pipeline Specification

Version 1.1.1

Scenera, Inc., Nikon Corporation, Sony Semiconductor Solutions Corporation, Wistron Corporation and Hon Hai Precision Industry Co., Ltd.(NICE Alliance Promoters) contributed to this document.

Edit History

Version	Date	Comments
1.1	26 Jan 2022	Initial Version 1.1 release
1.1.1	3 Mar 2022	Revision for publishing

Table of Contents

1. Scope

This specification covers the control and operation of a Data Pipeline. The APIs defined in this document are used once a Control Session has been established by an App or Data Service. This document also defines the format of the SceneMarks and SceneData that are generated by the configuration of the Data Pipeline.

2. Data Pipeline Overview

The purpose of the Data Pipeline is configuring a network of Nodes on different Devices to create a workflow that includes the capture of sensor data and its processing by distributed Artificial Intelligence and Computer Vision algorithms. The Data Pipeline is constructed using Nodes that may be distributed between different Devices and Cloud Services.

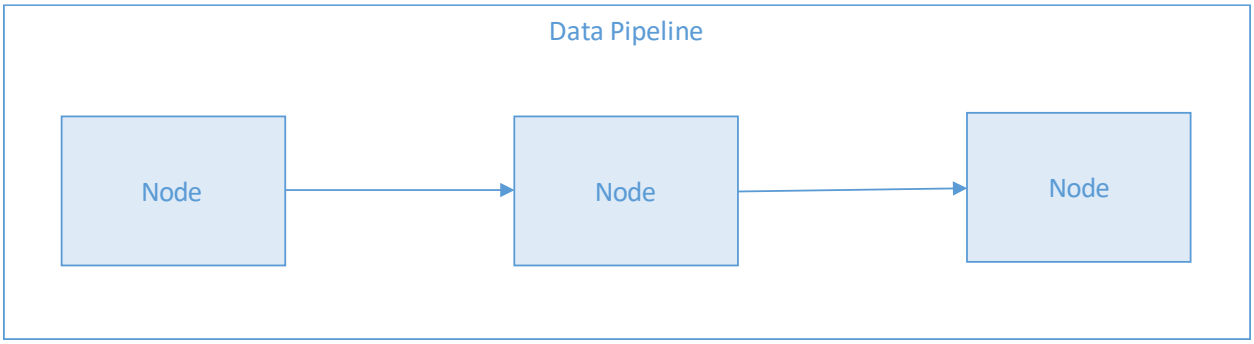

Figure 1: Data Pipeline

3. Device Node Overview

A Node is the basic component of the NICE Data Pipeline. A Node has Inputs, Outputs, Transducers (sensors or actuators) and a Process. A Node may capture an image, process its contents and generate a SceneMark and SceneData, it may input audio data and play it back through its speaker (Transducer) or simply take SceneMarks and SceneData as inputs and further process them.

The DataPipelineController sets the SceneMode for each Node. The SceneMode defines the type and amount of SceneData to be generated. This specification defines several SceneModes that simplify the configuration of AI and Computer Vision for each Node. The SceneMode also allows the DataPipelineController to define the kinds of analysis to be performed on the captured data. This results in a generation of a sequence of SceneMarks (which contains the metadata relevant to the capture) and SceneData (which are actual video, images or other data generated by the Scene). Examples of SceneModes are Face, Vehicle or Objects. In the case of Face SceneMode the AI and Computer Vision algorithms are set to look for faces within the images captured by the Node.

The App may receive this sequence of data in real time or retrieve the data at a later time. An App may utilize this data to provide users with summaries of events or notifications of specific events occurring.

The DataPipelineController configures a Data Pipeline by linking outputs of one Node to an input of another Node and setting the analysis function on each Node. In doing so the App defines a processing path for capturing data and processing them.

A single Device may have one or more Nodes. In this way the Data Pipeline may have multiple branches from a single sensor.

A Node may be connected to other Nodes on other entities or in the cloud using an IP based connectivity protocol. Nodes can also be connected to other Nodes within the device using interfaces which have been implemented by the Device maker.

4. Data Pipeline Interfaces

The **Control Interface** is used to configure a Data Pipeline. It enables the DataPipelineController to configure the Node's SceneMode and Data Interfaces. The Device Specification and the App/Service Specification describe how a Control Interface between a Device or Data Service's Controller and an App or Data Service is set up.

The **Data Interface** of the Node enables the Nodes to exchange data with other Nodes, Apps and Data Services. It could be an event driven data (SceneData).

A Node exposes a Control Interface for the Node to Apps or Data Services.

5. Device Node Properties

A Node may be implemented on any platform, for example embedded within a camera, executed on the cloud or as a mobile application. A NICE Device shall have at least one Node but may also have multiple Nodes. A Device may be a physical Device or may be virtualized on a server or in the cloud. The controlling App or Service interacts with the Device to determine which Nodes are present and interfaces to the Device's Controller to configure the Nodes internal to the Device.

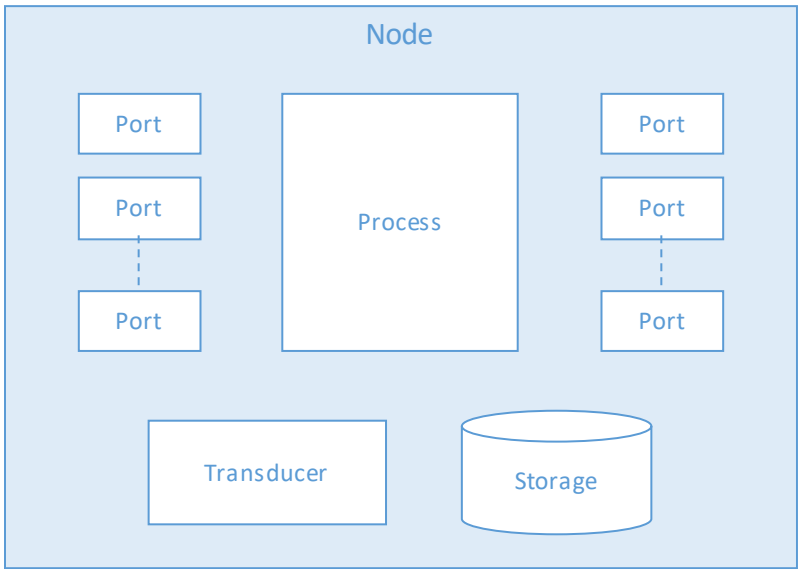

Figure 2. Components which make up a Node

A Node may have the following:

1. Transducer: A Transducer is either a sensor or an actuator which can convert data into a physical disturbance (for example a speaker). The following are examples of Transducers:
    1. Image sensor (image, depth, temperature camera) typically outputs a 2 dimensional array that represents a frame.
    2. Data sensor (humidity sensor, temperature sensor, etc.) typically outputs a text or JSON data structure.
    3. Audio microphone - produces a continuous sequence of audio samples.
    4. Speaker - takes as an input a sequence of audio samples and outputs them. This is the output for a stream going in the opposite direction to most streams in the system. Typically, from the cloud or App to the device rather than from the device to the cloud or application.
2. A Process is defined as a Computer Vision or Machine Learning algorithm that processes the sensor data. The following are examples of Processes:
    1. NICE SceneModes - these are NICE defined modes for processing images. NICE SceneModes are commonly used analysis types that are used in many applications.
    2. CustomAnalysis - is a specialized analysis algorithm that does not correspond to the existing defined SceneModes. This is an algorithm that can process an audio, image, video or data input.
3. Input - may be SceneData or SceneMarks and may be in a processed or unprocessed form. A source for the process may be one of the following:
    1. Output of a sensor internal or external to the device.
    2. Output located in a node on a different device.
    3. Output of a Node located within the device.
4. Output - an output may be SceneData or SceneMarks and may also be in a processed or unprocessed form. An output becomes a source for Node upstream from the current Node.

A Node shall have at least one Input or Output and at least a Transducer or a Process.

In case the of the Transducer being a sensor the Node shall have at least an Output.

5.1. Node/Port Identification

Each Node shall be uniquely identifiable in the NICE ecosystem. A Node shall have a NodeID which is unique within the Device hosting the Node. Each Device is given a DeviceID which is unique in NICE system. The combination of DeviceID and NodeID uniquely identifies a Node in NICE system.

Similarly each Port shall have a unique PortID within the the Node. The combination of the DeviceID, NodeID and PortID uniquely identifies a Port.

DestinationIDs referencing Ports shall adhere to this structure. This structure is defined in the NICE Network Protocol Specification.

5.2. Examples of Nodes

This section provides some examples of Nodes with different capabilities. These provide a reference for what a Node can do and how Nodes may be linked together to create a Data Pipeline.

5.2.1. Sensor Node with Motion Detection

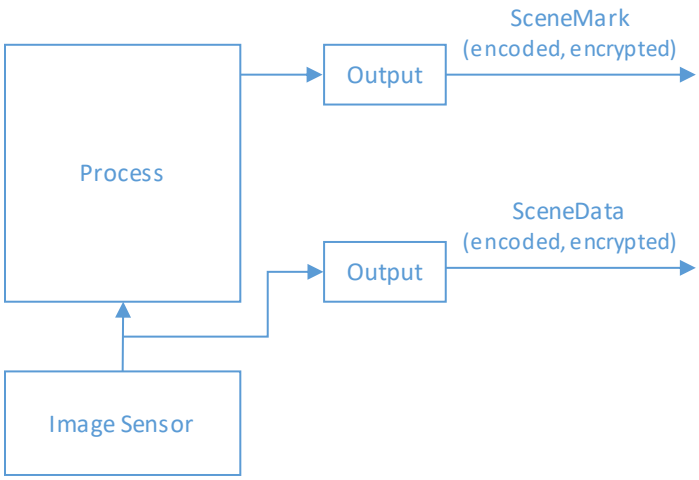

Figure 3. Image Sensor

In this Node there is an image sensor which is the Transducer and there is a Process which can detect motion. There are no Inputs and there are two outputs. One Output outputs SceneData which is a video stream that is encoded and encrypted. The second Output are SceneMarks which are JSON objects that are encoded and possibly encrypted. A port implements one of the protocols defined in the Network Protocols Section. The configuration of the Port, the Transducer and the Process is set using the SceneMode for the Node.

The Configuration of the Port includes the encoding of the data and the encryption of data. It also contains the destination for the port.

5.2.2. Node with Speaker

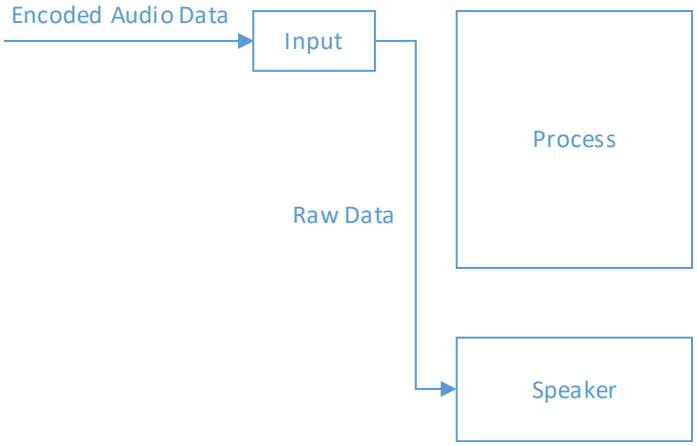

Figure 4. Node with Speaker

This is a simple Node that comprises only a Speaker which is the Transducer in the Node. It has a single Port that takes as an Input an Audio stream that is encoded and may be encrypted. Again the SceneMode configuration for the Node determines the encoding of the Audio, its source and the output level.

5.2.3. Node with Computer Vision Algorithm

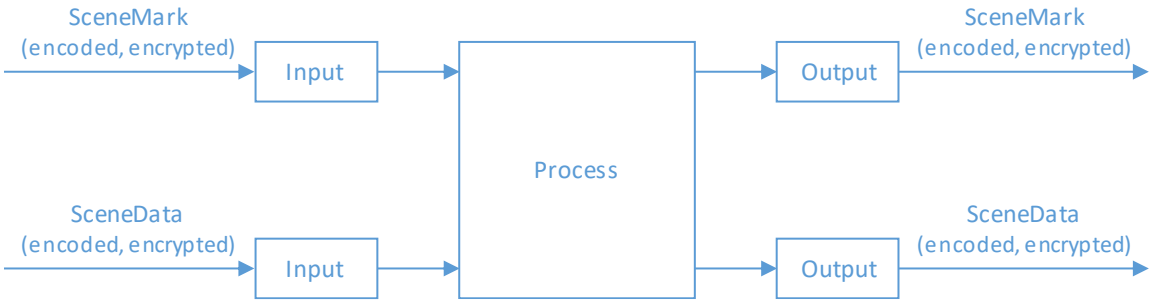

Figure 5. Node with Process

This node does not have a Transducer. It receives SceneData and SceneMarks from other Nodes and performs a process on the data. The Process, its Inputs and Outputs are defined by setting the SceneMode for the Node.

6. Setting Up a Data Pipeline

The Data Pipeline for Device Node shall be constructed by a DataPipelineController. The DataPipelineController shall request permission from the NICE Account Service to set up a Control Session. Once this permission has been provided the DataPipelineController shall interact with each Node to configure the SceneModes. The SceneMode Object defines the functionality of the Node and the interconnection with other Nodes.

The DataPipelineController may also configure a Data Pipeline for App. A DataPipelineController provides a Control API to NICE Apps to provide specific services to NICE Apps. The DataPipelineController takes the role of defining a Data Pipeline using Devices and Services. Data Service may further enhance the output of the Data Pipeline by analyzing data or performing cross analysis of data from multiple Devices for example. The Data Service may also aggregate requests for data from multiple Apps. A device can interact with a single NICE App or Data Service at a time. However multiple Apps can interact with a Data Service.

The Data Service shall have the same Control API as the Device Node. It may also extend the API to provide enhanced services. The Data Service shall abstract the Nodes from Devices. The NICE App does not need to set up a Control Session with each Device Node. The NICE App can create a Control Session with the Data Service Controller to request data and query which nodes are available. The Data Service takes care of creating the Control Session with each Node.

7. Overview of Objects in the Data Pipeline

There are 3 major Objects that exist within the Data Pipeline:

1. SceneMode
2. SceneMark
3. SceneData

4. The SceneMode Object is used to define the configuration of each Node and the interconnection between Nodes. The SceneMark and SceneData Objects are representations of the Data that is processed by the Data Pipeline.
5. The Pipeline produces a sequence of SceneMark and SceneData objects that are consumed by either the Data Service or the NICE App. SceneMark Objects may be manipulated by different Nodes in the Data Pipeline. This may entail adding additional fields to the SceneMark object that are the result of the Nodes processing either the SceneMark or SceneData. Nodes may also further generate SceneData that is the result of processing previous SceneMarks and SceneData. For example a Node that is capable of detecting faces may process a video frame from SceneData generated by a previous Node and extract the rectangles from the frame which correspond to the faces that have been detected. These may be extracted and encoded separately as new SceneData.
6. Once each Node has been configured bya SceneMode Object, the Data Pipeline is constructed and produces a sequence of SceneModes and SceneData in accordance to the SceneModes provided to each Node.

SceneMode	SceneMark & SceneData

Configuration of the Network
 	
• Configuration of Sensor(s)
• Setting of Processing
  ○ SceneModes
  ○ Analysis Level
  ○ Audio Processing
  ○ Triggers for SceneMark Generation
• Input and Output Stream Mappings & Configurations
	SceneMark
	
	• Formatted Data Describing Scene Captured by Network of Cameras
	• Time Stamps
	• Objects Detected, Labelled and Characterized
	
	SceneData
	
	• Video, Audio and Sensor Data
	  ○ Reduced to Relevant Clips Associated with SceneMarks Describing the Scene

7.1. SceneMode

The SceneMode Object is used to define the SceneMode including the inputs and outputs to and from the Node that implements the SceneMode.

A SceneMode defines which type of data is to be prioritized by the capture of frames and the processing of the captured frames.

The SceneMode also provides the configuration of a Node. The SceneMode defines the inputs and outputs, the configuration of the sensor and the processing of data within the Node. A SceneMode defines a workflow which specifies the settings for one or more sensor devices, as well as other necessary sensor behaviors. It also defines the Computer Vision (CV) and/or AI algorithms are to be engaged in the Node for processing the captured data. It also determines the requisite SceneData and possibly also SceneMarks and their content.

For example Face SceneMode will prioritize the capture of faces within a sequence of frames. When a face is detected, the camera system will capture frames with the faces present where the face is correctly focused, illuminated, and where necessary, sufficiently zoom to enable facial recognition to be executed with the highest possible chance of success.

When more than one face is detected, the camera shall capture as many faces as possible correctly. The camera may use multiple frames with different settings optimized for the faces in view. For example, for faces close to the camera the camera is focused close, and then for faces further away, digital zoom and longer focus should be used.

7.1.1. SceneMode Configuration

The SceneMode determines the data that is expected to be generated for each Device.

The SceneMode defines the data that the Device should prioritize. The SceneMode also incorporates the trigger conditions which shall be used to generate a SceneMark.

The configuration is made up of the following components:

1. SceneMode
2. Analysis Stage
3. SceneData Generation Parameters
4. Analysis Region
5. Scheduling

7.1.2. Types of SceneMode

The following SceneModes are defined:

1. Motion
2. Face
3. Human
4. Vehicle
5. Label
6. Animal
7. TextLogoQRCode
8. Custom
9. Loitering
10. Intrusion
11. Falldown
12. Violence
13. Fire
14. Abandonment
15. SpeedGate
16. Xray
17. Facility

Label SceneMode is a generalized labeling of images captured by the camera. This is typically the output of a general purpose labeling CNN trained on something like the ImageNet data set. This is in contrast with Face or Vehicle SceneMode where the CNN is trained on a specialized data set of vehicles or faces.

The SceneMode may generate data fields in the SceneMark associated with other SceneModes. The purpose of the SceneMode is guide the capture of images to suit the mode and define a minimum workflow for generating the data is defined by the SceneMode.

At the App level the App does not have insight into the specific configuration of the Device and how the Device is capturing images. The SceneMode configures the Device to produce the data required by the Data Pipeline.

7.1.3. Analysis Stage

Each SceneMode shall have an Analysis Stage. The Analysis Stage defines a condition upon which a SceneMark shall be generated and the SceneData defined for the SceneMode shall be captured and processed.

The AnalysisStage is one of the following:

1. Motion - the Process is capable of detecting motion within the field of view.
2. Detect - the Process is capable of detecting the item associated with the Scene. For example in the case of SceneMode being set to Face, Item Detected means that a Face has been detected. Detected means that a region has been identified where there is a high probability that a face is present within the region.
3. Recognize - the Process is capable of identifying the detected item. For example in the case of the Label SceneMode, where an item has been detected "Recognized" means the item can be labelled. In the case of Face SceneMode, "Recognized" means that the ID of the face has been determined. The SceneMode configuration enables reference images for categories that are to be identified.
4. Characterize - means that a characteristic for the Item can be determined. For example in Face SceneMode "Characterized" means that some feature of the detected face has had an attribute associated with it. For example a mood has been attributed to the detected face.

A Process in a Node is capable of performing at least one of these Analysis Levels.

7.1.4. SceneMark Generation and Appending to SceneMarks

In case of a Node that is generating a SceneMark the AnalysisStage indicates the condition for the generation of the SceneMark:

- "Motion": the SceneMark shall be generated when motion is detected within the field of view.
- "Detect": the SceneMark shall be generated when the item defined by the SceneMode is detected.
- "Recognize": the SceneMark is generated when a specific instance of an item has been identified or recognized.
- "Characterize": the SceneMark is generated when a characteristic of an item is detected. For example in case of a face when a particular expression is detected.

In case of a Node that is receiving a SceneMark as an input, the Node may append further information to the SceneMark:

- "Motion": is not relevant as it is only used as a trigger to generate a SceneMark.
- "Detect": information regarding the detected item is appended to the SceneMark.
- "Recognize": information regarding the identified instance is appended to the SceneMark.

"Characterize": the characteristic of an item is appended to the SceneMark.

7.1.5. AnalysisThreshold

The SceneMode shall define a threshold for which the SceneMark shall be generated. For example in the case of Human SceneMode a threshold 0.7 means that a SceneMark shall be generated if the Human detection algorithm has a greater than 70% certainty that a human has been detected.

7.1.6. LabelRefDataList

The App may provide reference images to enable the Recognition step. For example in Face SceneMode, the recognition step will require reference face to match the detected face against. The SceneMode has a field for reference images which can be used to enable this match when required. For example one or more images of a person may be supplied. These images are processed in accordance with the trained algorithm and programmed into the AI model for the SceneMode to enable the matching to occur. In other instances the AI model may be already capable of recognition. For example the model has specific objects, animals or vehicles already programmed into it.

Multiple Nodes may be configured to create a Pipeline where the first Node detects motion, the second Node detects the presence of a face and a third Node recognizes the Face. Alternatively a powerful Node may be capable of performing multiple of these operations at the same time for example a single Node may be capable of performing the steps of Face Detection and Face Recognition within the Node.

7.1.7. AnalysisRegion

If this is defined the analysis shall only be performed within the region defined by the array of points.

7.1.8. Scheduling

This defines the times when the analysis should be performed and SceneMarks generated.

7.1.9. CustomAnalysis

The Device may have one or more Modes that are proprietary to the device. The Device may have an Artificial Intelligence algorithm tailored for a specific purpose. For the purposes of this specification it is assumed that the model is trained a particular data set which has been labelled where each label is known to the developer of the algorithm. The algorithm is capable of locating items according to the networks training. The algorithm may offer further classification of the items. These two steps correspond to the Item Detected trigger and the Item Recognized trigger fields. If the implementation is capable of tracking recognized items, the trigger may also be generated if the item disappears.

The CustomAnalysisID is a unique ID for the analysis algorithm that is to be used by the processing stage.

7.2. SceneMark

A SceneMark is a compact representation of a recognized Scene of interest based on intelligent interpretation of the time- and/or location- correlated aggregated events. SceneMarks may be used to extract and present information pertinent to consumers of the sensor data. SceneMarks may also be used to facilitate the intelligent and efficient archival/retrieval of detailed information, including the raw sensor data. In this role, SceneMarks operate as an index into a much larger volume of sensor data.

7.2.1. SceneMark Identifier

The SceneMark Identifier is a unique ID for each SceneMark. This is used to track the SceneMark through the Data Pipeline and when it is stored. Refer to NICE Identifier Structure in detail.

7.2.2. SceneMark Generation

The Process shall trigger the generation of SceneData and a SceneMark when an event that conforms to the trigger condition defined in the SceneMode occurs. The SceneMark provides reference metadata for the SceneData that has been generated. The completeness of the SceneMark is determined by the analysis capabilities of the Node. If the device can only perform motion detection, a partial SceneMark shall be generated. Additional information may be added to the SceneMark by Nodes to which the SceneMark is an input.

The analysis shall also provide bounding boxes for each object detected with the probability of the label of the object. The labeling provided by the low level of analysis may be more refined than these categories. For example with vehicle it may include automobile and model.

The Custom analysis mode is a mode that is embedded in the device and is proprietary to the device.

7.2.3. SceneMark Caching

The allocation of the cache for SceneMarks and SceneData may be defined as part of the SceneMode configuration. In case that there is no definition of the SceneMode and SceneMark cache size, the cache shall be used as a single cache for both SceneData and SceneMarks. In case that the cache becomes full, the cache shall overwrite the oldest entries in the cache.

The source of SceneMarks that are cached may be other Nodes (as defined in the SceneMode) or the sensor within the Node.

In case SceneMarks are cached, the SceneMark manifest provides a means for navigating through SceneMarks in a logical manner based on time. The SceneMark manifest can be used to create playlists for video or other SceneData to create summaries of events.

The **SceneMarkManifest** is a list of SceneMark. The SceneMark Manifest is divided into time intervals, each interval period contains a list of SceneMarks that occurred within that interval. The Time Interval structure enables an App to navigate through the SceneMark manifest based on time.

The contents of the SceneMarkManifest may be based on the following criteria:

1. A list of all SceneMarks generated by a group of devices over a period of time.
2. A list of SceneMarks generated by a Control Session initiated by a NICE App.
3. A list of SceneMarks generated by a single Device.
4. Any combination of the above.

The Node that stores the SceneMarks shall maintain a SceneMark manifest and enable NICE Apps and Data Services to access this file so as to enable navigation through SceneMarks.

7.2.4. SceneMarkManifest Object

The SceneMarkManifest is a directory of SceneMarks with their storage locations. The SceneMarkManifest is organized in time periods to enable Apps to chronologically locate SceneMarks.

7.3. SceneData

SceneData is captured or provided by a group of one or more sensor devices and/or sensor modules, which includes different types of sensor data related to the Scene and also further processed or analyzed data. SceneData can be thought of as a sample or snapshot of a Scene. SceneData can also include different types of meta data from various sources. Examples include timestamps, geolocation data, ID for the sensor device, IDs and data from other sensor devices in the vicinity.

SceneData is generated by NICE devices in accordance with the SceneMode that has been configured by an App. SceneData is made up of still image, video, audio data.

A Node may support the following video codecs and container format:
- Codec format
  - H.264, ISO/IEC 14496-10 Advanced Video Coding
  - H.265, ISO/IEC 23008-2 HEVC
  - JPEG, ISO/IEC 10918-1:1994. Information technology -- Digital compression and coding of continuous-tone still images, lossless

- Container format for H.264, H.265 and JPEG
  - ISO/IEC, 14496-12:2004 (MPEG-4 Part 12: ISO base media file format) for audio and video.

- Container format for JPEG
  - JFIF, ISO/IEC 10918-5:2013. Information technology -- Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF)

Configuration of Devices may also be included as part of the SceneData. Where SceneData is cached or stored, SceneData can be navigated through by using SceneMarks with references to the SceneData file.

7.3.1. SceneData Identifier

The SceneData Identifier is a unique ID for each component of the SceneData. This is used to track the SceneData through the Data Pipeline and when it is stored. Refer to NICE Identifier Structure fore more detail.

7.3.2. SceneData Generation

SceneData shall be created by encoding the output of the sensor as either still images, video clips and/or audio. The duration of video generated in a response to a trigger is determined as one the parameters in the SceneMode. It is desirable to reduce the amount of video generated as SceneData to a minimum to minimize storage and bandwidth costs. For different applications there may be different requirements for different duration of video.

7.3.3. SceneData Configuration

The SceneMode determines which SceneData should be captured when a SceneMark is generated. For example if a SceneMark is generated then a period of video should be captured. The video may start before the time stamp of the SceneMark and end after the SceneMark's time stamp. Examples of SceneData that may be captured may be the following:

1. RGB, IR or RGB-IR including audio.
2. RGB or IR including audio.
3. Audio only.

The SceneMode defines the type and amount of SceneData that is generated when a Trigger occurs. For example, the SceneMode configuration may indicate that 10 seconds of video before the trigger and 30 seconds after the trigger shall be generated as SceneData. This is set in the SceneData configuration field of the SceneMode object. Multiple SceneMarks may reference a single video file of SceneData if triggers happen more rapidly than the period defined for SceneData. For example where multiple triggers occur within 30 seconds and the SceneData is defined for each trigger is 30 seconds. Where multiple triggers occur within those 30 seconds, the SceneMarks generated for each trigger shall reference the same video file that makes up the SceneData for the trigger.

8. Control Interface API

The following APIs may be called to configure the SceneMode.

8.1. GetSceneMode

Function

The Node uses this API to fetch the SceneMode from the DataPipeline Controller.

Protocol(s) Used to Make Calls

WebAPI

Direction

Caller	DEVICE
Callee	DataPipelineController

Request Parameters

SceneModeRequest Object

Acknowledgement Parameters

SceneMode Object

8.2. GetSceneMarkManifest

Function

Request a list of SceneMarks available on the Node.

Protocol(s) Used to Make Calls

WebAPI

Direction

Caller	APP, NICE AS
Callee	NICEDS

Request Parameters

SceneMarkManifestRequest

Acknowledgement Parameters

SceneMarkManifest Object

9. Data Interface

9.1. SetSceneData

Function

A Node, App or Data Service uses this call to upload new SceneData or overwrite SceneData to a location. The SceneData is uploaded as a number of sections to make a complete file. The receiving entity shall rebuild the original file when all of the sections have been received.

Protocol(s) Used to Make Calls

WebAPI

Direction

Caller	NICEDS, APP, DEVICE
Callee	NICEDS, APP

Request Parameters

DataSection Object

Acknowledgement Parameters

Empty.

9.2. SetSceneMark

Function

The Device must have the function of sending SceneMark to an App or a Cloud Service.

Protocol(s) Used to Make Calls

WebAPI

Direction

Caller	NICEDS, APP, DEVICE
Callee	NICEDS, APP

Request Parameters

SceneMark Object

Acknowledgement Parameters

Empty.

10. Data Objects

10.1. SceneMode Object

10.1.1. JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "SceneMode",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "SceneModeID": {
            "type": "string",
            "description": "SceneModeID unique within the scope of this node."
        },
        "NodeID": {
            "type": "string"
        },
        "Inputs": {
            "type": "array",
            "description": "Defines inputs for the Node from other Nodes either within
the device or external devices. String contains URI for the location of the input.\n",
            "uniqueItems": true,
            "items": {
                "anyOf": [
                    {
                        "type": "object",
```

```
                        "properties": {
                            "Type": {
                                "type": "string"
                            },
                            "VideoEndPoint": {
                                "type": "object",
                                "properties": {
                                    "CameraFPS": {
                                        "type": "integer"
                                    },
                                    "VideoURI": {
                                        "type": "string"
                                    },
                                    "Distance": {
                                        "type": "string"
                                    }
                                }
                            }
                        },
                        "required": []
                    },
                    {
                        "type": "object",
                        "properties": {
                            "PortID": {
                                "type": "string",
                                "description": "ID of the Port which the input is
made. This reference is used by functions in the Node to reference the Input."
                            },
                            "EndPoint": {
                                "$ref":
"Definitions.json#/definitions/NetworkEndPointSpecifier"
                            },
                            "Encryption": {
                                "$ref": "Definitions.json#/definitions/Encryption"
                            }
                        }
                    }
                ]
            }
        },
        "Outputs": {
            "type": "array",
            "description": "Defines outputs including data encoding, protocols and end
points.",
            "uniqueItems": true,
            "items": {
                "anyOf": [
                    {
                        "type": "object",
                        "title": "VideoConfiguration",
                        "description": "This configures the output of the video output
of the Node",
                        "properties": {
                            "Type": {
                                "type": "string",
                                "enum": [
                                    "Video"
                                ]
                            },
                            "PortID": {
                                "type": "string",
```

```
                                "description": "ID allocated within the Node for this
output. "
                            },
                            "FrameRate": {
                                "type": "number",
                                "description": "Frames per second"
                            },
                            "Resolution": {
                                "type": "object",
                                "description": "Resolution in pixels.",
                                "properties": {
                                    "Height": {
                                        "type": "integer"
                                    },
                                    "Width": {
                                        "type": "integer"
                                    }
                                },
                                "required": [
                                    "Height",
                                    "Width"
                                ]
                            },
                            "DestinationEndPointList": {
                                "type": "array",
                                "uniqueItems": true,
                                "items": {
                                    "$ref": "Definitions.json#/definitions/EndPoint"
                                }
                            },
                            "MediaFormat": {
                                "type": "string",
                                "enum": [
                                    "JPEG",
                                    "H.264",
                                    "H.265",
                                    "RAW"
                                ]
                            },
                            "StartTimeRelTrigger": {
                                "type": "number"
                            },
                            "EndTimeRelTrigger": {
                                "type": "number"
                            },
                            "MaxChunkSize": {
                                "type": "number"
                            },
                            "Encryption": {
                                "$ref": "Definitions.json#/definitions/Encryption"
                            }
                        },
                        "required": [
                            "PortID",
                            "Type"
                        ]
                    },
                    {
                        "type": "object",
                        "title": "ImageConfiguration",
                        "description": "Image output configuration",
                        "properties": {
                            "Type": {
```

```
                        "type": "string",
                        "enum": [
                            "Image"
                        ]
                    },
                    "PortID": {
                        "type": "string"
                    },
                    "ImageType": {
                        "type": "string",
                        "enum": [
                            "RGB",
                            "IR",
                            "RGBIR",
                            "Depth",
                            "Thermal",
                            "StereoRGB"
                        ]
                    },
                    "Resolution": {
                        "type": "object",
                        "description": "Resolution in pixels.",
                        "properties": {
                            "Height": {
                                "type": "integer"
                            },
                            "Width": {
                                "type": "integer"
                            }
                        },
                        "required": [
                            "Height",
                            "Width"
                        ]
                    },
                    "MediaFormat": {
                        "type": "string",
                        "enum": [
                            "JPEG",
                            "RAW"
                        ]
                    },
                    "DestinationEndPointList": {
                        "type": "array",
                        "uniqueItems": true,
                        "items": {
                            "$ref": "Definitions.json#/definitions/EndPoint"
                        }
                    },
                    "TimeRelTrigger": {
                        "type": "number"
                    },
                    "Encryption": {
                        "$ref": "Definitions.json#/definitions/Encryption"
                    }
                },
                "required": [
                    "PortID",
                    "Type"
                ]
            }
        ]
    }
```

```
        },
        "Mode": {
            "type": "object",
            "description": "Defines either NICE defined SceneMode or
DeviceDefinedAnalysis - which is a proprietary Computer Vision or AI in the device.",
            "properties": {
                "SceneMode": {
                    "type": "string"
                },
                "SceneModeConfig": {
                    "type": "array",
                    "description": "This defines the depth of analysis performed and
whether a result of an output can be used to drive a subsequent capture of frames.",
                    "uniqueItems": true,
                    "items": {
                        "type": "object",
                        "properties": {
                            "AnalysisStage": {
                                "type": "string",
                                "enum": [
                                    "Motion",
                                    "Detect",
                                    "Recognize",
                                    "Characterize"
                                ]
                            },
                            "CustomAnalysisID": {
                                "type": "string",
                                "description": "Each algorithm and set of weights has
a unique ID that is defined by NICE. This value shall be carried in this record."
                            },
                            "AnalysisDescription": {
                                "type": "string",
                                "description": "Description of algorithm."
                            },
                            "CustomAnalysisStage": {
                                "type": "string"
                            },
                            "LabelRefDataList": {
                                "type": "array",
                                "description": "For a specific label the following are
reference data such as images for the particular label. The Node shall process these
images to create the appropriate reference vector and store which RefDataIDs have been
used to create the vector. If new RefDataIDs are detected in the SceneMode object the
vector shall be regenerated with the listed RefData.",
                                "uniqueItems": true,
                                "items": {
                                    "type": "object",
                                    "properties": {
                                        "LabelName": {
                                            "type": "string",
                                            "description": "Label name for example for
facial recognition this would be the name or id of an individual."
                                        },
                                        "RefDataList": {
                                            "type": "array",
                                            "uniqueItems": true,
                                            "items": {
                                                "type": "object",
                                                "properties": {
                                                    "RefDataID": {
                                                        "type": "string"
                                                    },
```

```
                                                "RefDataEndPoint": {
                                                    "$ref":
"Definitions.json#/definitions/NetworkEndPointSpecifier"
                                                }
                                            },
                                            "required": [
                                                "RefDataID"
                                            ]
                                        }
                                    },
                                    "RefData": {
                                        "type": "array",
                                        "uniqueItems": true,
                                        "items": {
                                            "type": "object",
                                            "properties": {
                                                "RefDataID": {
                                                    "type": "string"
                                                },
                                                "RefData": {
                                                    "type": "string",
                                                    "description": "Reference data
encoded in Base64. For example an image of a persons face."
                                                },
                                                "Encryption": {
                                                    "$ref":
"Definitions.json#/definitions/Encryption"
                                                }
                                            },
                                            "required": [
                                                "RefDataID",
                                                "RefData",
                                                "Encryption"
                                            ]
                                        }
                                    },
                                    "ProcessingStage": {
                                        "type": "string",
                                        "description": "This indicates which
analysis stage should use the reference data.",
                                        "enum": [
                                            "CustomAnalysis",
                                            "Motion",
                                            "Detect",
                                            "Recognize",
                                            "Characterize"
                                        ]
                                    }
                                },
                                "required": [
                                    "ProcessingStage",
                                    "LabelName"
                                ]
                            }
                        },
                        "AnalysisThreshold": {
                            "type": "number",
                            "description": "The output of the analysis should be
greater than this value to trigger the Capture Sequence."
                        },
                        "AnalysisRegion": {
                            "type": "array",
                            "items": {
```

```
                                    "type": "object",
                                    "additionalProperties": {
                                        "not": {}
                                    },
                                    "properties": {
                                        "XCoord": {
                                            "type": "integer"
                                        },
                                        "YCoord": {
                                            "type": "integer"
                                        }
                                    },
                                    "required": [
                                        "XCoord",
                                        "YCoord"
                                    ]
                                }
                            },
                            "Scheduling": {
                                "type": "array",
                                "items": {
                                    "type": "object",
                                    "additionalProperties": {
                                        "not": {}
                                    },
                                    "properties": {
                                        "EndTime": {
                                            "type": "string"
                                        },
                                        "SchedulingType": {
                                            "type": "string"
                                        },
                                        "StartTime": {
                                            "type": "string"
                                        }
                                    },
                                    "required": [
                                        "EndTime",
                                        "SchedulingType",
                                        "StartTime"
                                    ]
                                }
                            },
                            "Encryption": {
                                "$ref": "Definitions.json#/definitions/Encryption"
                            }
                        },
                        "required": []
                    }
                },
                "SceneMarkInputList": {
                    "type": "array",
                    "description": "Locations from where incoming SceneMarks may be
received.",
                    "uniqueItems": true,
                    "items": {
                        "type": "object",
                        "properties": {
                            "SceneMarkInputEndPoint": {
                                "$ref":
"Definitions.json#/definitions/NetworkEndPointSpecifier"
                            },
                            "Encryption": {
```

```
                                "$ref": "Definitions.json#/definitions/Encryption"
                            }
                        },
                        "required": []
                    }
                },
                "SceneMarkOutputList": {
                    "type": "array",
                    "description": "Locations where SceneMarks should be sent to",
                    "uniqueItems": true,
                    "items": {
                        "type": "object",
                        "properties": {
                            "SceneMarkOutputEndPoint": {
                                "$ref":
"Definitions.json#/definitions/NetworkEndPointSpecifier"
                            },
                            "Encryption": {
                                "$ref": "Definitions.json#/definitions/Encryption"
                            }
                        },
                        "required": []
                    }
                }
            },
            "required": [
                "SceneMode",
                "SceneModeConfig"
            ]
        }
    },
    "required": [
        "Version",
        "SceneModeID",
        "NodeID"
    ]
}
```

10.2. SceneMarkManifest Object

10.2.1. JSON Schema

The following is the JSON schema for the SceneMark Manifest.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "SceneMarkManifest",
    "properties": {
        "Version": {
            "type": "string"
        },
        "StartDateTime": {
            "type": "string",
            "description": "Earliest SceneMark occurs after this time."
        },
        "EndDateTime": {
            "type": "string",
            "description": "Latest SceneMark occurs before this time."
```

```
        },
        "PageLength": {
            "type": "integer",
            "description": "Number of SeneMarks in the SceneMarkList."
        },
        "ListDates": {
            "type": "array",
            "description": "Dates of the SceneMarks in the list.",
            "items": {
                "type": "string"
            }
        },
        "SceneMarkList": {
            "type": "array",
            "description": "List of SceneMarks.",
            "items": {
                "type": "object",
                "properties": {
                    "SceneMarkID": {
                        "type": "string"
                    },
                    "NodeID": {
                        "type": "string"
                    },
                    "SceneMarkURI": {
                        "type": "string"
                    },
                    "TimeStamp": {
                        "type": "string"
                    },
                    "EventTypes": {
                        "type": "array",
                        "items": {
                            "type": "string"
                        }
                    },
                    "ItemIdentities": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "ItemType": {
                                    "type": "string"
                                },
                                "ItemIDs": {
                                    "type": "array",
                                    "items": {
                                        "type": "string"
                                    }
                                }
                            },
                            "required": [
                                "ItemType",
                                "ItemIDs"
                            ]
                        }
                    },
                    "SceneDataThumbnail": {
                        "type": "object",
                        "properties": {
                            "SceneDataThumbnailURI": {
                                "type": "string"
                            },
```

```
                            "EncryptionOn": {
                                "type": "boolean"
                            },
                            "SceneEncryptionKeyID": {
                                "type": [
                                    "string",
                                    "null"
                                ]
                            }
                        },
                        "required": [
                            "SceneDataThumbnailURI",
                            "EncryptionOn",
                            "SceneEncryptionKeyID"
                        ]
                    }
                },
                "required": [
                    "SceneMarkID",
                    "NodeID",
                    "SceneMarkURI",
                    "TimeStamp",
                    "EventTypes",
                    "ItemIdentities",
                    "SceneDataThumbnail"
                ]
            }
        },
        "ContinuationToken": {
            "type": [
                "string",
                "null"
            ],
            "description": "Token that should be used in the next call if there are
more SceneMarks from the initial query."
        }
    },
    "required": [
        "Version"
    ]
}
```

10.3. SceneMark Object

10.3.1. Data Structure

The SceneMark is a JSON document that contains metadata regarding events that have triggered the generation of the SceneMark.

The following are the types of data that are included in the SceneMark:

1. Thumbnail associated with the Scene.
2. Time Stamp for the generation of the SceneMark.
3. Attributes for the Scene.
4. List of devices that have generated SceneData associated with the Scene.
5. For each Node a list of streams that have been generated or processed by the device.

6. For each Node: settings, SceneMode, SceneData generated, Progress of Completion of SceneMode data generation, Metadata that has been generated regarding the Scene, thumbnails etc.

In addition to identifying specific objects, faces etc the SceneMark also has attribute fields which describe characteristics of the object identified. For example a recognized face may have the attribute "smiling" or a recognized object may have the attribute "yellow". These attributes are determined by the AI engines that are utilized to generate SceneMarks and are not fixed by the NICE standard.

There are also probability fields for the attributes or elements that are recognized in the SceneMark. These fields indicate the certainty of the result that has been supplied.

The algorithm ID that is used throughout the SceneMark is a unique identifier issued by the NICE alliance that indicates which algorithm has been used to generate the attribute. This unique id allows an App to verify the vendor of the algorithm, the type of algorithm and version of algorithm. On the basis of this ID the App can assess whether the attribute is useful and in the case of applications requiring an audit trail the algorithm ID can be used to track how attributes where generated.

SceneMarkID

The SceneMarkID value is generated if the SceneMark is generated for the first time. The SceneMarkID shall start with the value SM, followed by NodeID and an incrementing 8 byte value. Each SceneMarkID is unique.

TimeStamp

The TimeStamp for the trigger event that generated the SceneMark. The time format is according to the NICE Date Time Format specification.

Thumbnail

There are two options for carrying the thumbnail for the scene. The Thumbnail may be a JPEG file encoded as a base64 string. Alternatively the thumbnail will be a JPEG file referenced by a URI.

Encryption

The encryption of the SceneData is signaled through the "Encrypted" being set to TRUE. When encrypted is "TRUE" there is a link provided to the Rights Object Server where the Rights Object can be obtained for the SceneData field. There is also a Key ID that enables the server to reference the key to be provided in the Rights Object returned to the client.

Attributes

Attributes are characteristics that have been associated with the SceneMark.

For each attribute there is an algorithm ID which is used to track which algorithm was used to generate the attribute and the probability or score for the attribute that has been provided to the SceneMark.

Processing Status

This refers to the status of the processing of SceneMark for each camera. Each SceneMode defines a minimum set of data fields for a complete SceneMark. The data provided in the SceneMark may be added by different processes within a workflow defined for the SceneMode. The Processing Status field indicates which steps of the workflow have been completed.

Typical workflow entails the following steps:

1. Detect - identify regions where a thing such as a Face, Human, Vehicle, Object, Text/Logo/QR Code or Animal are located.
2. Recognize - specifically identify the thing that has been detected.
3. Categorize - additional categorization of the thing detected. For example is the face smiling, or the color or features of an object.

These fields are used by processes in the SceneMode workflow to determine what fields need to be added to the SceneMark.

SceneData

The object contains a list of SceneData that is associated with the SceneMark.

Format Type

The type of data file that is used to contain the SceneData.

Detected Items

Several types of Items may be detected and processed. The minimum set of Items to be detected are determined by the SceneMode. Additional categories of Items may be added to the SceneMark. For example the Face SceneMode requires facial information to be captured and processed. Other Items such as vehicles may optionally added to the SceneMark.

For each category there is an array of Items that have been identified under the category.

For each thing that has been captured there are the following:

1. Whether one of the categories has been detected.
2. If it has been detected has it been recognized as being a specific member of the category. For example with faces the identity of the face.
3. Any attribute for the thing listed - for example if a face what is the mood of the face.
4. Coordinates of a rectangle where the object is located in the image.
5. Thumbnail or URI referencing a thumbnail.

The SceneMark is a JSON document that contains metadata regarding events that have triggered the generation of the SceneMark.

10.3.2. JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "SceneMark",
```

```
    "description": "The SceneMark contains data describing what has been captured in
SceneData and either contains references to SceneData or contains the SceneData
itself.",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "TimeStamp": {
            "type": "string",
            "description": "Time stamp for when the SceneMark is first generated."
        },
        "SceneMarkID": {
            "type": "string",
            "description": "Unique ID for a SceneMark. This ID is unique across the
NICE ecosystem."
        },
        "DestinationID": {
            "type": "string",
            "description": "DataService or App ID initiated the request for this
SceneMark"
        },
        "SceneMarkStatus": {
            "type": "string",
            "enum": [
                "Removed",
                "Active",
                "Processed"
            ]
        },
        "NodeID": {
            "type": "string"
        },
        "PortID": {
            "type": "string"
        },
        "VersionControl": {
            "type": "object",
            "properties": {
                "DataPipelineInstanceID": {
                    "type": "string"
                },
                "VersionList": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "VersionNumber": {
                                "type": "number"
                            },
                            "DateTimeStamp": {
                                "type": "string"
                            },
                            "NodeID": {
                                "type": "string"
                            }
                        },
                        "required": [
                            "VersionNumber",
                            "NodeID",
                            "DateTimeStamp"
```

```
                        ]
                    }
                }
            }
        },
        "ThumbnailList": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "VersionNumber": {
                        "type": "number"
                    },
                    "SceneDataID": {
                        "type": "string",
                        "description": "SceneDataID should appear in the SceneDataList
that is included in the SceneMark."
                    }
                },
                "required": [
                    "VersionNumber".
                    "SceneDataID"
                ]
            }
        },
        "AnalysisList": {
            "type": "array",
            "uniqueItems": true,
            "items": {
                "type": "object",
                "properties": {
                    "VersionNumber": {
                        "type": "number"
                    },
                    "SceneMode": {
                        "type": "string"
                    },
                    "CustomAnalysisID": {
                        "type": "string",
                        "description": "Each algorithm and set of weights has a unique
ID that is defined by NICE. This value shall be carried in this record."
                    },
                    "AnalysisDescription": {
                        "type": "string"
                    },
                    "ProcessingStatus": {
                        "type": "string",
                        "enum": [
                            "Motion",
                            "Detect",
                            "Recognize",
                            "Characterize"
                        ]
                    },
                    "DetectedObjects": {
                        "type": "array",
                        "uniqueItems": true,
                        "items": {
                            "type": "object",
                            "properties": {
                                "VersionNumber": {
                                    "type": "number"
                                },
```

```
                                "NICEItemType": {
                                    "type": "string",
                                    "description": "NICE defines different DeviceModes
which target specific types of data associated with the DeviceMode."
                                },
                                "CustomItemType": {
                                    "type": "string",
                                    "description": "Devices may have proprietary AI
algorithms embedded in the device or processing node. If this algorithm was used, the
label generated by the algorithm shall be carried in this field."
                                },
                                "ItemID": {
                                    "type": "string",
                                    "description": "Unique ID that is associated with
this instance of object. This is an optional field that may be used to track objects
across different scenemarks."
                                },
                                "Probability": {
                                    "type": "number",
                                    "description": "Certainty of the Attribute
According to the Algorithm"
                                },
                                "Analysis": {
                                    "type": "object",
                                    "description": "Defines which types of analysis
have been used to identify or detect the object.\n",
                                    "properties": {
                                        "SceneMode": {
                                            "type": "string"
                                        },
                                        "CustomAnalysisID": {
                                            "type": "string"
                                        },
                                        "AnalysisDescription": {
                                            "type": "string"
                                        },
                                        "ProcessingStatus": {
                                            "type": "string",
                                            "enum": [
                                                "Motion",
                                                "Detect",
                                                "Recognize",
                                                "Characterize"
                                            ]
                                        }
                                    }
                                },
                                "Attributes": {
                                    "type": "array",
                                    "description": "Different AI algorithms are
capable of identifying different attributes of objects that have been identified. For
example if a face is detected the attribute may be \"smiling\". These attributes
depend on the AI algorithm used and are not specified.",
                                    "uniqueItems": true,
                                    "items": {
                                        "type": "object",
                                        "properties": {
                                            "Attribute": {
                                                "type": "string",
                                                "description": "Attribute of face
recognized - mood etc"
                                            },
                                            "ProbabilityofAttribute": {
```

```
                                                "type": "number",
                                                "description": "Degree of certainty of
the attribute"
                                            },
                                            "AlgorithmID": {
                                                "type": "string",
                                                "description": "Unique ID for the
algorithm."
                                            }
                                        }
                                    }
                                },
                                "BoundingBox": {
                                    "type": "object",
                                    "properties": {
                                        "XCoordinate": {
                                            "type": "integer"
                                        },
                                        "YCoordinate": {
                                            "type": "integer"
                                        },
                                        "Height": {
                                            "type": "integer"
                                        },
                                        "Width": {
                                            "type": "integer"
                                        }
                                    },
                                    "required": [
                                        "Height",
                                        "Width",
                                        "XCoordinate",
                                        "YCoordinate"
                                    ]
                                },
                                "ThumbnailSceneDataID": {
                                    "type": "string",
                                    "description": "SceneDataID should appear in the
SceneDataList that is included in the SceneMark."
                                },
                                "RelatedSceneData": {
                                    "type": "array",
                                    "items": {
                                        "type": "string",
                                        "description": "SceneDataIDs that are related
to the Detected Object. These should appear in the SceneDataList that follows."
                                    }
                                }
                            },
                            "required": []
                        }
                    }
                },
                "required": []
            }
        },
        "SceneDataList": {
            "type": "array",
            "items": {
                "$ref": "Definitions.json#/definitions/SceneData"
            }
        }
    },
```

```
    "required": [
        "SceneMarkID",
        "NodeID",
        "Version",
        "TimeStamp"
    ]
}
```

10.4. DataSectionID Object

This object carries the SectionID of data.

10.4.1. JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "DataSectionID",
    "description": "References a specific section in the SceneData file to be
returned.",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "EndPointID": {
            "type": "string"
        },
        "DataID": {
            "type": "string"
        },
        "DataSectionID": {
            "type": "string"
        }
    },
    "required": [
        "DataSectionID",
        "DataID",
        "Version"
    ]
}
```

10.5. DataSection Object

A DataSection Object may be used to transfer any large files between two locations. SceneData may be encapsulate using the DataSection Object structure.

10.5.1. JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
```

```
    "type": "object",
    "title": "DataSection",
    "description": "Section of Data that is transferred as part of a file transfer",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "DataID": {
            "type": "string",
            "description": "Unique ID associated with the Data. If the Data is
SceneData, the ID is made up of the prefix 'SD' followed by the unique Node ID for the
Node that generated the SceneData and an incrementing number starting at the value of
1000."
        },
        "FileType": {
            "type": "string",
            "description": "Type Of File being uploaded [ \"Video\", \"Image\",
\"Json\", ... ]"
        },
        "FileName": {
            "type": "string",
            "description": "Filename of the file being uploaded."
        },
        "PathURI": {
            "type": "string",
            "description": "The path to store the file on Server:\n Example:
\"\\Account0001\\files\\videos\\\""
        },
        "Section": {
            "type": "integer",
            "description": "This indicates the specific splice number of the file
being send."
        },
        "LastSection": {
            "type": "integer",
            "description": "This is the total number of splices that the file was
spliced into."
        },
        "HashMethod": {
            "type": "string",
            "description": "MD5 or SHA1 or SHA256 or CRC32 (Only required when the
last section is sent)",
            "enum": [
                "MD5",
                "SHA1",
                "SHA256"
            ]
        },
        "OriginalFileHash": {
            "type": "string",
            "description": "Hex String of Hash of the Original File. (Only required
when the last section is sent)"
        },
        "SectionBase64": {
            "type": "string",
            "description": "This is the File Section converted to a base64 string"
        }
    },
    "required": [
        "Section",
```

```
        "LastSection",
        "SectionBase64",
        "DataID",
        "Version"
    ]
}
```

10.6. SceneModeRequest Object

This object contains the NodeID and the specific SceneModeID on the Node. This Node shall return the SceneMode Object that is specified by the SceneMode ID in this object.

10.6.1. JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "SceneModeRequest",
    "description": "Contains NodeID and SceneModeID. Used to request a specific
SceneMode Object from a Node.",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "NodeID": {
            "type": "string"
        }
    },
    "required": [
        "Version",
        "NodeID"
    ]
}
```

10.7. SceneMarkManifestRequest Object

This structure is used to request the SceneMarkManifest for the SceneMarks. The contents of the Manifest are filtered using parameters in the Object. There is also a continuation token which allows the server to track which part of the list should be returned to the caller.

10.7.1. JSON Schema

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "SceneMarkManifestRequest",
    "description": "This data structure has a set of filters used to filter out
SceneMarks required.",
    "properties": {
        "NodeIDs": {
            "type": "array",
```

```
            "description": "List of Nodes that generated SceneMarks. SceneMarks from
these Nodes only are requested.",
            "items": {
                "type": "string"
            }
        },
        "StartTime": {
            "type": "string",
            "description": "Only SceneMarks after this time are requested."
        },
        "EndTime": {
            "type": "string",
            "description": "No SceneMarks after this time."
        },
        "PageLength": {
            "type": "integer",
            "description": "Number of SceneMark entries to be returned in the
SceneMarkManifest."
        },
        "ReturnPage": {
            "type": "boolean",
            "description": "If false then only the dates are returned - this speeds up
the response for queries where the date range is required only."
        },
        "ListEventTypes": {
            "type": "array",
            "description": "Types of events to filter upon.",
            "items": {
                "type": "string"
            }
        },
        "ListNICEItemTypes": {
            "type": "array",
            "description": "List of detected items to filter upon.",
            "items": {
                "type": "string"
            }
        },
        "ContinuationToken": {
            "type": "string",
            "description": "Used to manage sequential requests for pages of
SceneMarks. Acts as a book mark to indicate to the server the current location in the
list."
        }
    },
    "required": [
        "NodeIDs",
        "StartTime",
        "EndTime",
        "PageLength",
        "ReturnPage",
        "ListEventTypes",
        "ListNICEItemTypes",
        "ContinuationToken"
    ]
}
```