Device Specification

Version 1.0.1

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

Revision History

Version	Date	Comments
0.9rc1	13 Nov 2018	First draft
0.9rc2	25 Feb 2019	Second draft
0.9	25 Mar 2019	Final draft
1.0	22 May 2019	Final release
1.0.1	20 Dec 2019	No change from version 1.0

Contributors

Name	**Company**
Andrew Wajs	Scenera
Aviram Cohen	Scenera
Munehiro Shimomura	Sony
Hironori Miyoshi	Sony
Wendy Tin	Wistron

Table of Contents

1. Scope

This document describes the implementation of a NICE compliant Device. This includes how the device is managed and configured.

2. Device Overview

A NICE Device shall contain at least one or more Nodes. A Node is a basic unit of the Data Pipeline which contains at least inputs or outputs, a sensor or actuator and a computer vision process. The Device shall manage the following functions on behalf the Nodes that it contains:

1. Network Connectivity
2. Security of access to the Device, the protection of connections to the Device.
3. Privacy of data that is generated by Nodes within the Device.

Each Device shall have a DeviceID which is a unique identifier for the Device. The Device shall validate Entities that are communicating with the Device and provide access to resources on the Device based on the Permission that has been provided by the Entity in communication with the Device. The Access Management for Devices is defined in the Authentication Specification.

A Device shall be managed by one NICE Account Service at a time. A Device may be transferred from one NICE Account Service to another. The NICE License Authority shall be responsible for ensuring that the Device is allocated to a specific NICE Account Service.

The NICE Account Service enables Apps and Data Services to interact with the Device and to configure the Nodes that are housed in the Device. The Management interface shall be used to configure the connections to the device and set up a connection fabric between devices. Once these connections are in place, Nodes on different Devices can interact with each other as if they are on the same Device.

A Device shall have:

- One or more Nodes implemented.
- A Single management Interface.
- At least one IP based connection for Management, Control and Data.
- A Unique DeviceID, Secret Key with an accompanying X.509 certificate available on the NICE Licensing Authority.

The Example below represents a Device implementation that implements a single Source Node and has a Management Interface.

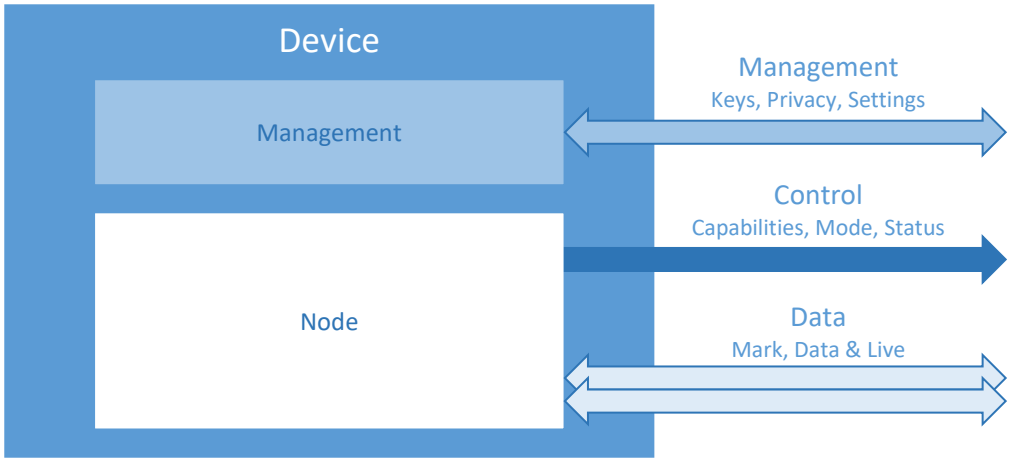

Figure 1: Device Implementation with a Single Node and a Management Interface

2.1. Interfaces

2.1.1. Management Interface

A **Management Interface** of a Device shall be used to configure the interconnections between Devices and Cloud Services and setup the security and privacy objects.
The interface shall be used for:

- Setting the Control and Data Protocols to be used for each connection.
- Setting the security credentials to enable secure communication between Devices, Cloud Services and App.
- Configuration of the Device parameters.
- Getting the Device status.

The Management session shall use one of the following protocols:

1. MQTT
2. WebAPI

2.1.2. Control Interface

The **Control Interface** of a Node shall enable an App or Cloud to manage the Node within a Device. Commands sent to the Node are addressed to the Node by referencing the NodeID for that Node.

The Entities managing the Data Pipeline shall be provided credentials by the NICE Account Service that enable the Entity to communicate with the Device. An Entity managing the Device shall be granted an Access Token to access the Device. This Access Token may have a finite duration and the Entity managing the Device may be required to renew the Access Token if the token expires. The Access Token may also be revoked by the NICE Account Service. The processing of the Access Token is described in

the NICE Authentication Specification. The Entity shall use the API defined in the Data Pipeline Specification to manage the configuration of the Nodes on the Device.

The Control session shall use one of the following protocols:

1. MQTT
2. WebRTC
3. WebAPI

For More information please refer to the NICE Network Protocol Specification.
For More information about the Control interface please refer to the Data Pipeline specification.

2.1.3. Data

The **Data Interface** of the Device enables the Device to exchange data with a Cloud Service or an Application.
It could be either an event driven data (SceneData) or a Live Stream using WebRTC as a delivery mechanism.

The establishment of the Data Session shall be set by the SceneMode as defined in the Data Pipeline Specification.

The Data session shall use one of the following protocols:

1. MQTT
2. WebRTC
3. WebAPI

For More information please refer to the protocol section in this specification.
For More information about the Data interface please refer to the Data Pipeline specification.

3. Management Interface

3.1. GetManagementEndPoint

Function

The "GetManagementEndPoint" request the ManagementEndPoint Object to the NICE LA.
If the NICE LA receives this request, the NICE LA returns the ManagementEndPoint Object. The Object includes the information to connect Management Interface by MQTT.

Protocol(s) Used to Make Calls

WebAPI

Direction

Caller	DEVICE

Callee	NICELA

Request Parameters

DeviceID

Acknowledgement Parameters

ManagementEndPoint Object

3.2. SetManagementObject

Function

The Device shall have the API to accept the Management Object configuration.

The "SetManagementObject" sets the Management Object to the Device. This Object shall include the keys, URI and credentials for the NICE Account Service Connection.

This API is called from NICE LA. The Device establishes an MQTT session to the NICE AS server described in the Management Object.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	NICELA
Callee	DEVICE

Request Parameters

Management Object

Acknowledgement Parameters

Empty

3.3. GetNodeList

Function

Retrieve the list of Nodes on a specific device.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	NICELA
Callee	DEVICE

Request Parameters

Empty

Acknowledgement Parameters

NodeList Object

3.4. GetManagementObject

Function

The Device must have the API to request the Management Object configuration form the NICE LA server.

The "GetManagementObject" request the Management Object to the NICE LA.
If the NICE LA receives this request, the NICE LA returns the Management Object.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	DEVICE
Callee	NICELA

Request Parameters

Empty

Acknowledgement Parameters

Management Object

3.5. SetControlObject

Function

The Device shall have the API to accept the AS credential object configuration.

The "SetControlObject" shall set the Control Object in the Device.
This call shall be made by the NICE AS. The Device shall establish a connection based on the requested protocol in the object. This API overwrites the settings previously configured with this API.

Protocol(s) Used to Make Calls

MQTT Management

WebAPI

Direction

Caller	NICEAS
Callee	DEVICE

Request Parameters

ControlObject

Acknowledgement Parameters

Empty

3.6. GetControlObject

Function

The Device shall support the API to request the Control Object from the NICE AS server.

The "GetControlObject" requests the Control Object from the NICE AS.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	DEVICE
Callee	NICEAS

Request Parameters

Empty

Acknowledgement Parameters

ControlObject

3.7. GetDeviceStatus

Function

The Device shall return the DeviceStatus Object in response to this request.

Protocol(s) Used to Make Calls

MQTT Management

Web API

Direction

Caller	NICEAS, NICELA
Callee	DEVICE

Request Parameters

Empty

Acknowledgement Parameters

DeviceStatus Object

3.8. GetLog

Function

The response shall contain the Log Object shall include a list of messages that have not been correctly responded to. The LogDuration Object that is passed in the request specifies the period of time for which log information is required.

Protocol(s) Used to Make Calls

MQTT Management

Web API

Direction

Caller	NICEAS, NICELA
Callee	DEVICE

Request Parameters

LogDuration Object

Acknowledgement Parameters

Log Object(s)

3.9. DeleteLog

Function

The Device shall delete log file entries as defined in the LogDuration Object.

Protocol(s) Used to Make Calls

MQTT Management

Web API

Direction

Caller	NICEAS, NICELA
Callee	DEVICE

Request Parameters

LogDuration Object

Acknowledgement Parameters

Empty

3.10. SetFirmwareUpdateObject

Function

The Device shall support the API to accept the FirmwareUpdate object. The receipt and correct receipt of this object shall cause the the Device to fetch a file containing the Firmware from the server defined in the FirmwareUpdate Object if the Firmware File has not already been downloaded to the Device.

The "SetFirmwareUpdateObject" sets the FirmwareUpdate Object to the Device.

This API shall be called from NICE License Authority or from the NICE Account Service.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	NICELA
Callee	DEVICE

Request Parameters

FirmwareUpdate Object

Acknowledgement Parameters

Empty

3.11. GetFirmwareUpdateObject

Function

The Device shall support the API to request the FirmwareUpdate Object configuration form the NICE LA or NICE AS server.

The "GetFirmwareUpdateObject" request the FirmwareUpdate Object from the NICE LA or from the NICE AS.
If the NICE LA or NICE AS receives this request, it shall return the FirmwareUpdate Object.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	DEVICE
Callee	NICELA, NICEAS

Request Parameters

Empty

Acknowledgement Parameters

FirmwareUpdate Object

4. Data Objects

4.1. Management Object

The Management Object is provided to a Device when it is linked to a User's Account. The Object contains information that allows the Device to verify messages sent to the Device by the NICE AS or Entities that the NICE AS authorizes to communicate with the Device.

The Object shall always be encrypted under the Device Public Key and shall be Signed with a Private Key that is certified by the NICE LA for the purpose of being used to sign these Objects. In case the object is not signed or encrypted it shall be rejected by the Device. If the signature validation fails, the Object shall be rejected by the Device. If the object is incorrectly formed it shall be rejected by the Device.

The revoked jti list is the complete list of revoked Access Token IDs for the device. This list shall overwrite any previous list stored in the Device.

The "AllowedTLSRootCertificates" provides a list of allowed root certificates for TLS communication. Each entry in the array contains a Root Certificate that is valid for TLS communication. Certificates in this list are appended to any existing listed TLS Root Certificates that are stored in the device. The list is explicitly provided here so as not required to have the actual certificate signed by the NICE LA. There is also an option to delete Certificates that have been provided previously to the Device.

4.1.1. Data Structure

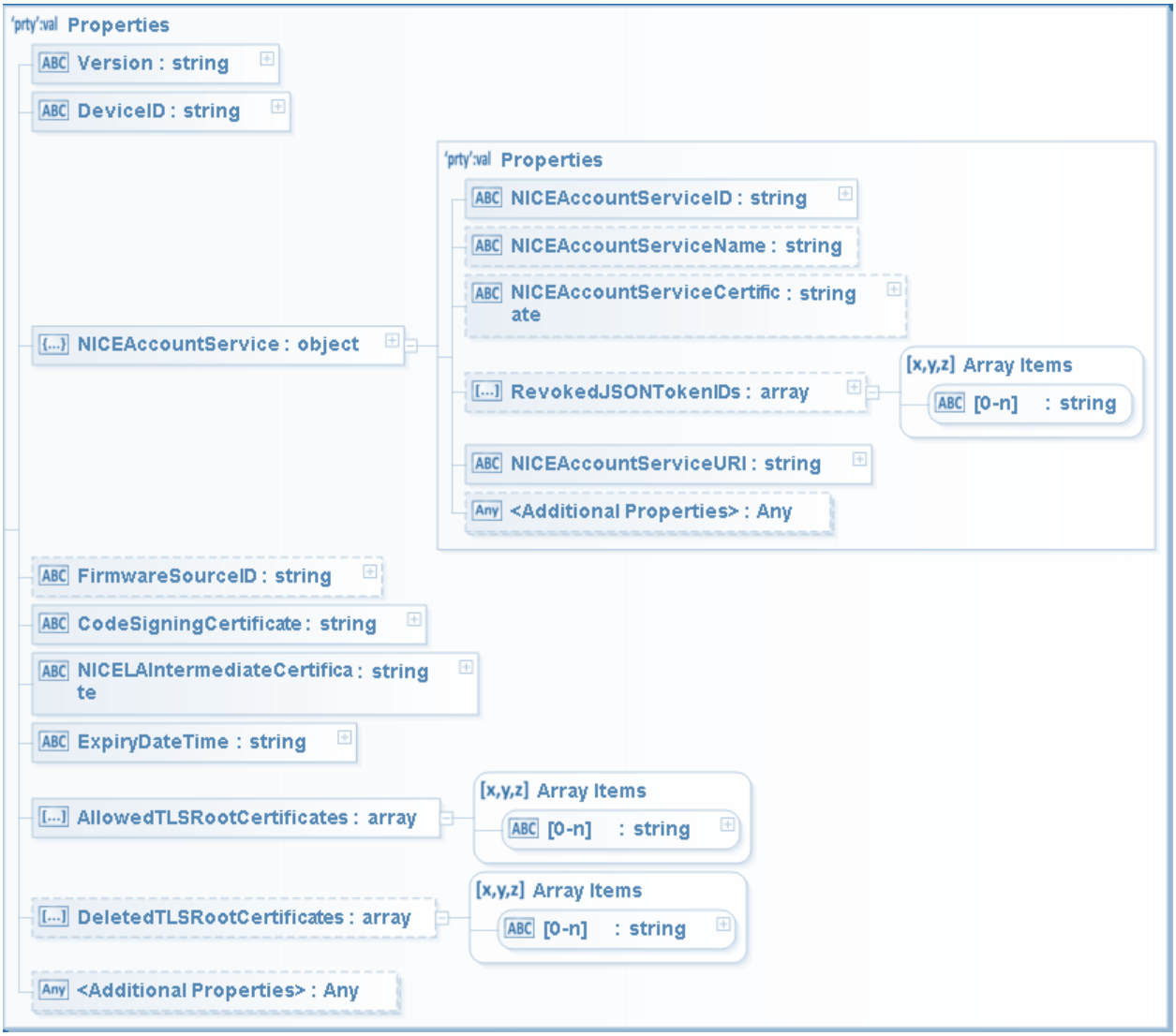

Figure 2. Management Object Data Structure

4.1.2. JSON Object

Management Object

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "Management",
    "description": "The device shall first decrypt this object using its private key.
It then extracts and validates the NICELA working X.509 certificate and validates this
certificate. It then uses the public key from this certificate and validates the
message. The validate includes the entire decrypted document structure. ",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "DeviceID": {
            "type": "string",
            "description": "Permanent ID of the device. Readable by end user."
        },
        "NICEAccountService": {
            "type": "object",
            "description": "Management layer messages can only be sent and received
using these credentials to encrypt at an application level.",
            "properties": {
                "NICEAccountServiceID": {
                    "type": "string",
                    "description": "Unique ID for the NICE Account Service Provider"
                },
                "NICEAccountServiceName": {
                    "type": "string"
                },
                "NICEAccountServiceCertificate": {
                    "type": "string",
                    "description": "Base64 encoded X.509 certificate for the NICE
Account Service "
                },
                "RevokedJSONTokenIDs": {
                    "type": "array",
                    "description": "JSON Token IDs (jti) that are revoked ",
                    "uniqueItems": true,
                    "items": {
                        "type": "string"
                    }
                },
                "NICEAccountServiceURI": {
                    "type": "string",
                    "description": "URI for the MQTT broker for the NICE Account
Service"
                }
            },
            "required": [
                "NICEAccountServiceID",
                "NICEAccountServiceURI"
            ]
        },
        "FirmwareSourceID": {
            "type": "string",
```

```
            "description": "ID of the party that is responsible for signing the
firmware update. When checking the X.509 certificate for the code image signature this
ID must match that of the X.509 certificate."
        },
        "CodeSigningCertificate": {
            "type": "string",
            "description": "X.509 Certificate containing verification key for firmware
upgrades. The NICE LA will use this field to update the certificate from the device
manufacturer to the device seller."
        },
        "NICELAIntermediateCertificate": {
            "type": "string",
            "description": "Base64 encoded X.509 certificate for the NICELA working
Certificate. This certificate is signed using the NICE LA root key."
        },
        "ExpiryDateTime": {
            "type": "string",
            "description": "Message is no longer valid after this date in ISO 8601
format."
        },
        "AllowedTLSRootCertificates": {
            "type": "array",
            "uniqueItems": true,
            "items": {
                "type": "string",
                "description": "Allowed X.509 root certificates that may be used for
Management Level TLS communication"
            }
        },
        "DeletedTLSRootCertificates": {
            "type": "array",
            "uniqueItems": true,
            "items": {
                "type": "string",
                "description": "TLS certificates to be deleted from the Device for TLS
communication."
            }
        }
    },
    "required": [
        "CodeSigningCertificate",
        "NICEAccountService",
        "AllowedTLSRootCertificates",
        "NICELAIntermediateCertificate",
        "DeviceID",
        "ExpiryDateTime",
        "Version"
    ]
}
```

4.2. DeviceControl Object

This Object is provided by the NICE AS to the Device to manage the processing of Access Tokens that the NICE AS provides to Entities that are enabled to interact with the Device.

The Object shall always be encrypted under the Device Public Key and shall be Signed with a Private Key that is certified by the NICE AS for the purpose of being used to sign these Objects. In case the object is not signed or encrypted it shall be rejected by the Device. If the signature validation fails, the Object shall be rejected by the Device. If the object is incorrectly formed it shall be rejected by the Device.

The revoked jti list is the complete list of revoked Access Token IDs for the device. This list shall overwrite any previous list stored in the Device.

4.2.1. Data Structure

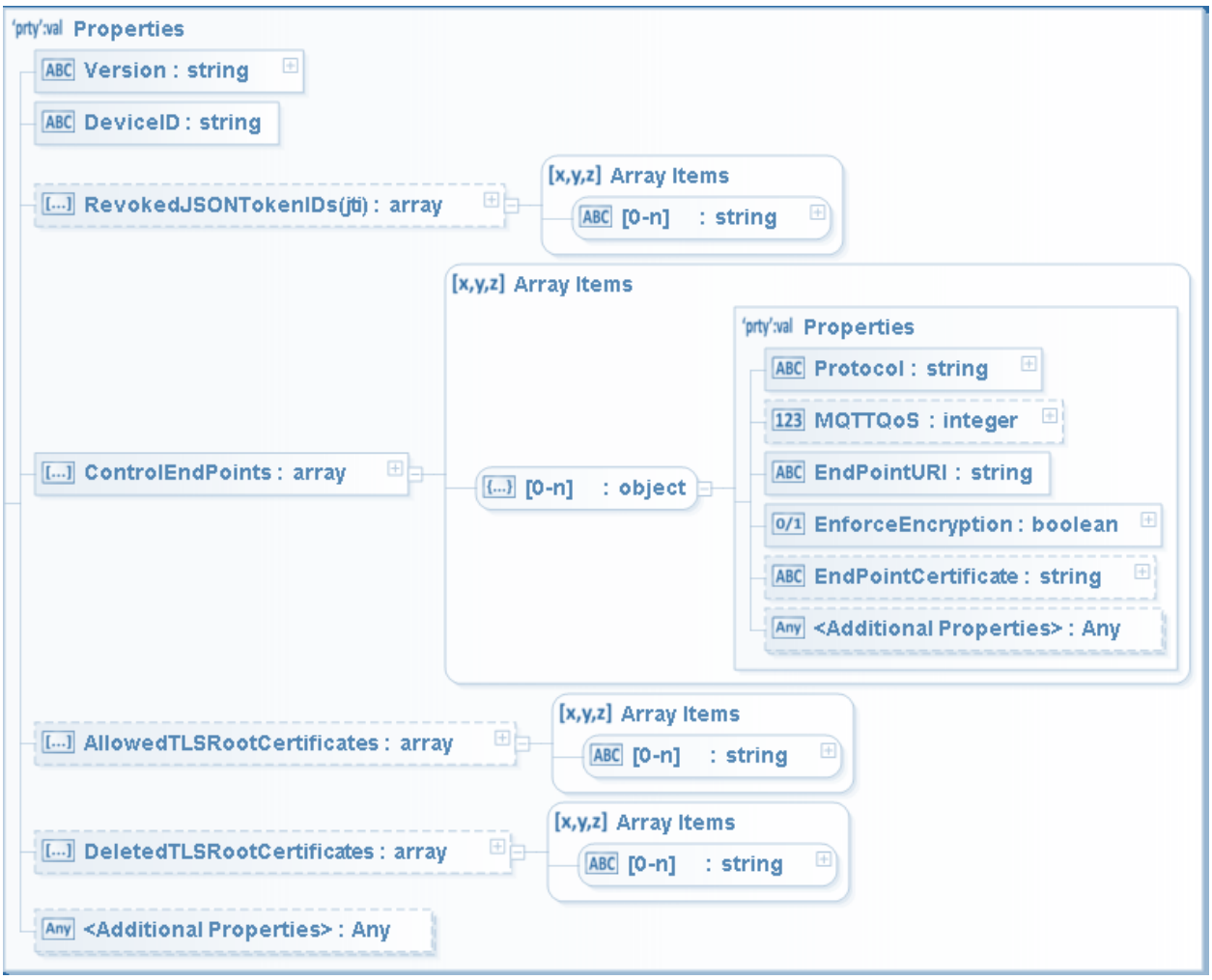

Figure 3. DeviceControl Object Data Structure

4.2.2. JSON Object

DeviceControl

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "DeviceControl",
    "description": "This object sets up the permissions and end points for the Control
communication.",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
```

```
                "1.0"
            ]
        },
        "DeviceID": {
            "type": "string"
        },
        "RevokedJSONTokenIDs(jti)": {
            "type": "array",
            "description": "List of revoked jti values",
            "uniqueItems": true,
            "items": {
                "type": "string",
                "description": "jti values that are revoked"
            }
        },
        "ControlEndPoints": {
            "type": "array",
            "uniqueItems": true,
            "items": {
                "type": "object",
                "properties": {
                    "Protocol": {
                        "type": "string",
                        "enum": [
                            "WebRTC",
                            "MQTT"
                        ]
                    },
                    "MQTTQoS": {
                        "type": "integer",
                        "enum": [
                            0,
                            1,
                            2
                        ]
                    },
                    "EndPointURI": {
                        "type": "string"
                    },
                    "EnforceEncryption": {
                        "type": "boolean",
                        "description": "If true data objects are encrypted and
signed."
                    },
                    "EndPointCertificate": {
                        "type": "string",
                        "description": "If true the certificate for the end point
shall be present in this field."
                    }
                },
                "required": [
                    "Protocol",
                    "EndPointURI",
                    "EnforceEncryption"
                ]
            }
        },
        "AllowedTLSRootCertificates": {
            "type": "array",
            "description": "Each entry in the array contains a Root Certificate that
is valid for TLS communication. Only entries in this field shall be accepted by the
Device. ",
            "uniqueItems": true,
```

```
            "items": {
                "type": "string",
                "description": "Allowed X.509 root certificates that may be used for
Management Level TLS communication"
            }
        },
        "DeletedTLSRootCertificates": {
            "type": "array",
            "description": "Root Certificates in this list are to be deleted from the
Device. ",
            "uniqueItems": true,
            "items": {
                "type": "string",
                "description": "Allowed X.509 root certificates that may be used for
Management Level TLS communication.\n"
            }
        }
    },
    "required": [
        "ControlEndPoints",
        "DeviceID",
        "Version"
    ]
}
```

4.3. FirmwareUpdate Object

This Object is provided when a new firmware image is to be loaded into the Device. The firmware image is checked against the signature provided in the object. The object also has fields that enable rollback and version control. The Object shall be encrypted with the Device Public key and Signed with a Public Key that is set by the NICE LA for the purposes of signing Firmware. This is defined by the FirmwareSourceID field in the Management Object provided to the Device.

The HashValue in the FirmwareUpdate Object shall be the hash of the entire code image using the SHA 256 algorithm. The firmware update shall be delivered in the Executable and Linkable Format. The Hash function shall cover the entire file that conforms to this format. This object shall be delivered as an encrypted and signed JSON object as defined in the Security and Privacy section. The signature that is provided by this format is used to ensure that the hash of the firmware image is cryptographically protected and any tampering with the firmware image can be detected. If the hash result carried within this object does not match the firmware image or the signature of the object fails the firmware update shall not proceed.

Firmware Update Object

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "FirmwareUpdate",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "ManufacturerID": {
            "type": "string"
        },
```

```
        "ModelType": {
            "type": "string"
        },
        "FirmwareSourceID": {
            "type": "string",
            "description": "The "
        },
        "SoftwareVersion": {
            "type": "number",
            "description": "Actual Version of Software this Object Refers To."
        },
        "IssueDate": {
            "type": "string",
            "description": "Date that the Signature was Generated On"
        },
        "OldestValidDate": {
            "type": "string",
            "description": "This field determines the oldest date that a signature may
be accepted."
        },
        "ValidDate": {
            "type": "string",
            "pattern": "This signature is valid until this date. "
        },
        "HashValue": {
            "type": "string",
            "description": "Hash of the file containing the code image. Hash is
generated using SHA 256."
        },
        "UpdateMethod": {
            "type": "string",
            "description": "Mechanism used to update code.",
            "enum": [
                "Immediate",
                "NextReset"
            ]
        },
        "FirmwareURI": {
            "type": "string",
            "description": "Location of the Firmware image. "
        }
    },
    "required": [
        "ManufacturerID",
        "ModelType",
        "SoftwareVersion",
        "IssueDate",
        "OldestValidDate",
        "ValidDate",
        "HashValue",
        "UpdateMethod",
        "FirmwareURI",
        "FirmwareSourceID",
        "Version"
    ]
}
```

4.4. DeviceStatus Object

The DeviceStatus provides the list of connections that have been configured for the Device. A Device Maker may provide StatusCodes in the StatusCodes data structure. The meaning of these codes shall be defined by the Device Maker.

4.4.1. Data Structure

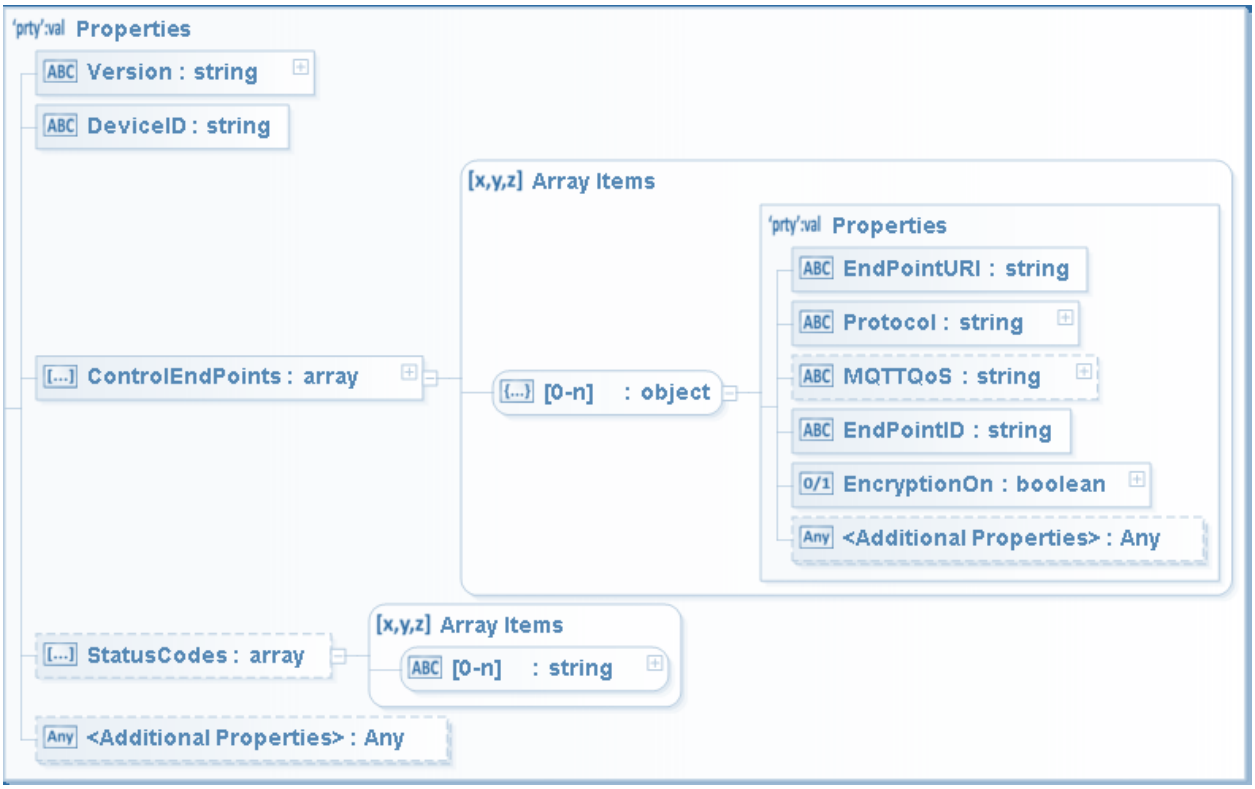

Figure 4. DeviceStatus Object Data Structure

4.4.2. JSON Object

DeviceStatus

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "DeviceStatus",
    "description": "This is the status of the Device.",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "DeviceID": {
            "type": "string"
```

```
        },
        "ControlEndPoints": {
            "type": "array",
            "description": "Control End Points are where the App can send control
commands to Devices. Each Control End Point Object contains the parameters for the end
points and an indication of which Device IDs are behind the End Point. ",
            "uniqueItems": true,
            "items": {
                "type": "object",
                "properties": {
                    "EndPointURI": {
                        "type": "string"
                    },
                    "Protocol": {
                        "type": "string",
                        "enum": [
                            "WebAPI",
                            "MQTT",
                            "WebRTC"
                        ]
                    },
                    "MQTTQoS": {
                        "type": "integer",
                        "enum": [
                            0,
                            1,
                            2
                        ]
                    },
                    "EndPointID": {
                        "type": "string"
                    },
                    "EncryptionOn": {
                        "type": "boolean",
                        "description": "If true then the Data Objects transfered over
this channel shall be encrypted and authenticated. The App requires an App Instance
Security Object with the appropriate key and certificate."
                    }
                },
                "required": [
                    "EndPointURI",
                    "Protocol",
                    "EndPointID",
                    "EncryptionOn"
                ]
            }
        },
        "StatusCodes": {
            "type": "array",
            "uniqueItems": true,
            "items": {
                "type": "string",
                "title": "StatusCode",
                "description": "Vendor defined status codes."
            }
        }
    },
    "required": [
        "ControlEndPoints",
        "DeviceID",
        "Version"
    ]
}
```

4.5. LogDuration Object

This object defines the range of log information that is to be provided when used in a DeviceStatus request. If defines the range of log information to be deleted when used in a DeleteLog request.

LogDuration

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "LogDuration",
    "description": "Describes the period for which log information should be returned
in a DeviceObject",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "EndLogTime": {
            "type": "string",
            "description": "Latest time for which log information should be provided."
        },
        "StartLogTime": {
            "type": "string",
            "description": "Earliest time for which log information should be
provided."
        }
    },
    "required": [
        "EndLogTime",
        "StartLogTime",
        "Version"
    ]
}
```

4.6. Log Object

The following object is defined for the log entry for a message for which there has been failure to respond to the message correctly.

Log Object

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "Log",
    "description": "Structure for log entries ",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "FailedMessages": {
            "type": "array",
```

```
            "uniqueItems": true,
            "items": {
                "type": "object",
                "title": "Failed Message",
                "description": "Contains the Header for Messages that have not been
processed correctly or responded to.",
                "properties": {
                    "MessageHeader": {
                        "type": "string"
                    },
                    "Error": {
                        "type": "string",
                        "enum": [
                            "No Response",
                            "Authentication Failure of Response",
                            "Misformed Response"
                        ]
                    }
                },
                "required": [
                    "MessageHeader",
                    "Error"
                ]
            }
        }
    },
    "required": [
        "FailedMessages",
        "Version"
    ]
}
```

4.7. NodeList

NodeList

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "NodeList",
    "description": "List of Nodes within a Device",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "EndPointID": {
            "type": "string"
        },
        "Nodes": {
            "type": "array",
            "uniqueItems": true,
            "items": {
                "type": "string",
                "title": "NodeID",
                "description": "ID for Node "
            }
        }
    },
```

```
    "required": [
        "EndPointID",
        "Nodes",
        "Version"
    ]
}
```

4.8. ManagementEndPoint Object

ManagementEndPoint

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "ManagementEndPoint",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "EndPointURI": {
            "type": "string",
            "description": "End Point URI for management."
        },
        "Protocol": {
            "type": "string",
            "enum": [
                "MQTT"
            ]
        },
        "MQTTQoS": {
            "type": "integer",
            "enum": [
                0,
                1,
                2
            ]
        },
        "Topic": {
            "type": "object",
            "properties": {
                "Subscribe": {
                    "type": "object",
                    "properties": {
                        "Name": {
                            "type": "string",
                            "description": "Topic name"
                        },
                        "QoS": {
                            "type": "integer",
                            "description": "QoS level",
                            "enum": [
                                0,
                                1,
                                2
                            ]
                        }
                    },
                    "required": [
```

```
                        "Name"
                    ]
                },
                "Publish": {
                    "type": "object",
                    "properties": {
                        "Name": {
                            "type": "string",
                            "description": "Topic name"
                        },
                        "QoS": {
                            "type": "integer",
                            "description": "QoS level",
                            "enum": [
                                0,
                                1,
                                2
                            ]
                        }
                    },
                    "required": [
                        "Name"
                    ]
                }
            }
        }
    },
    "required": [
        "EndPointURI",
        "Protocol",
        "Version"
    ]
}
```