App/Service 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 a NICE compliant App or Service. It describes how an App or Service can create and manage a connection with either a NICE compliant Device or NICE compliant Data Service. Once this connection has been established, the API that is described in the DataPipeline document is used to configure Nodes and the data flow between Nodes.

2. Overview

The NICE App is a software application that may execute as a web application, a cloud application or an application on a mobile, tv or pc platform. The App is selected by the User. The User associates the App with their account and may select which devices the App may access. Once the App has been granted permission to access the account, the App may request a Control Session with one or more NICE devices. These Control Sessions enable the App to set SceneModes and consume SceneData and SceneMarks. This API is described in the DataPipeline Specification. The App may have a direct interaction with an end user or may execute in the background. In this version of the spec a single App can manage a single Device at any time. There may be more than one App consuming SceneData and SceneMarks at any one time.

The App utilizes 4 major functions, Capabilities, SceneMode, SceneMark and SceneData. An App is capable of consuming NICE compliant data and also send Data to Devices.

An App shall **not implement** Source Node functionality and cannot receive control commands.

An App shall have:

- One or more Nodes implemented.
- A Single management Interface.
- At least one IP based connection to another Cloud Server or Device.
- Unique AppID, AppInstanceID, Private Signing Key and Private Encryption Key with an accompanying X.509 certificates available on the NICE Licensing Authority.

The Example below represent a basic App implementation with a Management Interface.

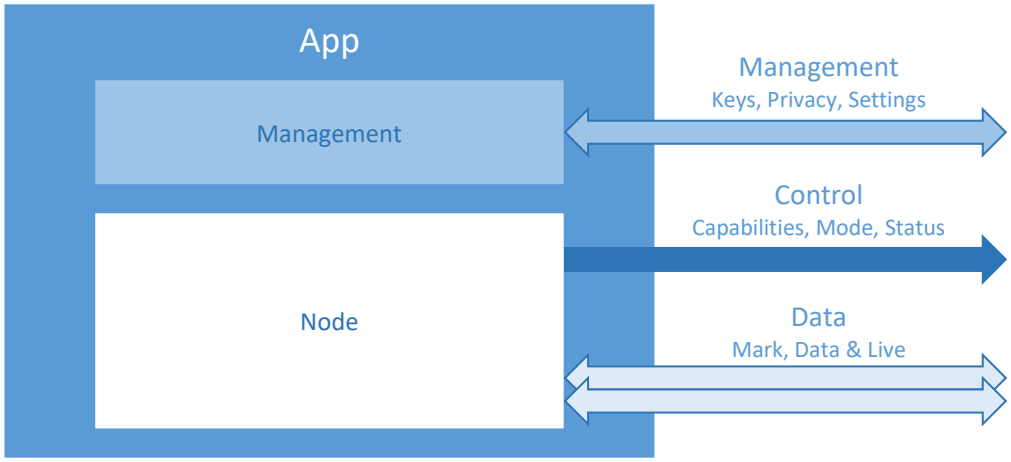

Figure 1. Basic App Implementation with a Single Node and a Management Interface

3. Interfaces

3.1. Management

A **Management Interface** of a App is responsible for configuring the interconnections between Apps to Devices and Cloud Service and setup the security and privacy objects.
The interface is mainly 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 and App.
- Getting the Device status.
- Getting the Device list linked to the App.

The Management session can use one of the following protocols:

1. MQTT
2. WebAPI

3.2. Control

The **Control Interface** is used by the App to configure Nodes within Devices and the Cloud to perform specific functions and to interact with other Nodes.

The Establishment of a Control session is made by the EstablishControlSession as part of the Management API.

The Control session can use one of the following protocols:

1. MQTT
2. WebRTC

3. WebAPI

The usage of these protocols is described by the Network Protocol specification. Once a Control connection has been established, the App or Service may utilize the API calls defined in DataPipeline specification. These APIs are used to query the capabilities of Nodes, to configure these Nodes and the flow of data between Nodes.

3.3. Data

The **Data Interface** of an App enables the App to exchange data with a Cloud Service or a Device. It could be either an event driven data(SceneData) or a live stream data(LiveData)

The establishment of the Data Session is set by the SceneMode as part of the Control API.

The Data session can use one of the following protocols:

1. MQTT
2. WebRTC
3. WebAPI

The usage of these protocols is described by the Network Protocol specification. The transfer of data over the Data Interface is described in the Data Pipeline specification.

4. Management of NICE Apps

The App life cycle has the following steps:

1. The App Developer registers as an App Developer with NICE. This process is performed through the NICE App Developer portal.
2. The App Developer develops the App and makes the App available to Users. This process is performed through the NICE App Developer portal. The App may be implemented as a Web App, an Android App, an iOS App or as a combination of these or other technologies. The App makes use of the NICE APIs to access a User's Devices and Data. The further processing of this data and the presentation to the User is beyond the scope of the NICE specifications.
3. The User selects the App and provides it with access to their Devices and their data.
4. The App establishes a Control Session or Sessions with either Devices or Data Services. These Control Sessions are used to define a DataPipeline which comprises multiple Nodes distributed across Devices and Cloud Services.
5. The App consumes the SceneMarks and SceneData that are generated by the DataPipeline that it has created.
6. The App stops the DataPipeline and becomes inactive.
7. Steps 4 through 6 are repeated.

4.1. NICE App Developer Access to User Account Data

The architecture for a NICE App or Service may have a cloud application which interacts with the User's Devices and Data, a mobile/PC application that interacts with the user or a combination of cloud, mobile and PC applications. In the case that the App Developer has a cloud application that interacts with the User's Devices and Data, the App Developer shall be provided with credentials to enable the interaction with the NICE system.

The NICE App Developer is issued a Private Encryption Key, Private Signing key and corresponding X.509 certificates by the NICE Account Service. The Privacy Management Service utilizes this public key to encrypt Privacy Objects to enable the NICE App to access SceneData.

4.2. A NICE App Instance

In the case that the App Developer has a mobile or PC based App which interacts directly with the User's Devices and Data, each instance of the App shall be provided with credentials to enable the interaction with the User's Devices and Data.

An instance of a NICE App may be issued a Private Encryption Key, Private Signing key and corresponding X.509 certificates by the NICE Account Service. The Privacy Management Service utilizes this public key to encrypt Rights Objects to enable the NICE Service Provider to access SceneData. Each instance of application may have its own public private key pairs.

5. API

NICE Application APIs are for third-party apps and app server. Service developers will have access to the NICE Account Services , Data Services, Media Services and Direct access to NICE Devices.

5.1. Management Interface

5.1.1. SetAppSecurityObject

Function

The App shall be capable of receiving the AppSecurity Object from the NICE Account Service.

The "SetAppSecurityObject" sets the AppSecurity Object to the App Developer, including the keys, URI and credentials for the NICE Account Service Connection.

This API is called from NICE Account Service. The App establishes an MQTT session to the NICE Account Service server described in the AppSecurity Object.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	NICEAS
Callee	APP

Request Parameters

AppSecurity Object

Acknowledgement Parameters

Empty

5.1.2. GetAppSecurityObject

Function

The App Developer must have the ability to request the AppSecurityObject configuration form the NICE LA or NICE AS server.

The "GetAppSecurityObject" requests the AppSecurityObject from either the NICE LA or NICE AS.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	APP
Callee	NICELA

Request

Empty

Acknowledgement Parameters

AppSecurity Object

5.1.3. WebRTCSignaling

Function

This command is used for exchanging Session Description Protocol (SDP) between WebRTC peers. It shall be triggered when establishing WebRTC connection or starting Live Streaming. When setting up a WebRTC session the peers exchange SDP Offers and Answers to negotiate the configuration of the WebRTC session.

The App calls this command to send the Offer SDP to the peer and receive the Answer SDP.

Protocol(s) Used to Make Calls

MQTT

Direction

In case of sending offer SDP:

Caller	DEVICE
Callee	NICEAS, NICEMS

In case of sending answer SDP:

Caller	APP, NICEMS
Callee	DEVICE

Request Parameters

Signaling

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "Signaling",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "WebRTCversion": {
            "type": "string",
            "description": "WebRTC version."
        },
        "sessionDescription": {
            "type": "string",
            "description": "WebRTC 1.0 compliant SDP string."
        }
    },
    "required": [
        "Version",
        "WebRTCversion",
        "sessionDescription"
    ]
}
```

Response Parameters

Empty.

5.1.4. GetDevices

Function

This command is used to get a list of all devices registered under the relevant user account.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	APP
Callee	NICEAS

Request Parameters

Empty

Response

DeviceList Object

5.1.5. SetDevices

Function

This command is used to update the list of devices associated with a user's account. This may be sent in case the User acquires a new Device or delinks an existing Device from their Account.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	NICEAS
Callee	APP

Request Parameters

DeviceList Object

Response Parameters

Empty

6. Data Objects

6.1. DeviceList Object

This object contains the list of DeviceIDs associated with a User's account.

DeviceList

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "DeviceList",
    "description": "List of Devices Associated with an Account",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "AccountID": {
            "type": "string"
        },
        "DeviceList": {
            "type": "array",
            "uniqueItems": true,
            "items": {
                "type": "object",
                "properties": {
                    "DeviceID": {
                        "type": "string"
                    },
                    "Status": {
                        "type": "string",
                        "enum": [
                            "Connected",
                            "Unconnected"
                        ]
                    },
                    "Description": {
                        "type": "string"
                    },
                    "BrokerURI": {
                        "type": "string",
                        "description": "URI for the Broker which will maintain the
queues to communicate with the Device."
                    },
                    "MQTT QOS Level": {
                        "type": "integer",
                        "description": "MQTT QOS level.",
                        "enum": [
```

```
                            0,
                            1,
                            2
                        ]
                    }
                },
                "required": [
                    "Status",
                    "DeviceID",
                    "BrokerURI",
                    "MQTT QOS Level"
                ]
            }
        }
    },
    "required": [
        "DeviceList",
        "AccountID",
        "Version"
    ]
}
```

6.2. AppSecurity Object

The AppSecurity Object provides key and credential material that enables the NICE App Developer to process an Privacy Object sent to the App Developer to enable the developer the access specific SceneData or SceneMarks.

Where the App Developer wishes for a downloaded App to access either the Users Account, devices, or SceneData, the AppSecurity Object shall be provided to the App developer per instance of App. The App developer shall ensure that the security assets are protected when downloaded into a consumer device.

Where it is necessary to manage to the security of data to a specific instance of App. The key material should be embedded in the App in a secure manner to enable the App to process the Privacy Objects sent to the App instance.

AppSecurity

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "AppSecurity",
    "description": "Application ID for the App developer",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "AppDeveloperID": {
            "type": "string",
            "description": "UserID for the Developer who develops the App."
        },
        "AppID": {
            "type": "string",
            "description": "Global Identifier for an App"
        },
```

```
        "AppInstanceID": {
            "type": "string",
            "description": "Identifier for the specific instance of the App"
        },
        "AppInstancePrivateKey": {
            "type": "object",
            "description": "This private key is used by the App to decrypt SceneData
and SceneMarks protected using the Privacy Management System.",
            "properties": {
                "EncryptionKeyID": {
                    "type": "string",
                    "description": "Key ID of the Public Key that has been used to
encrypt the AppInstancePrivateKey"
                },
                "EncryptedAppInstancedKey": {
                    "type": "string"
                }
            },
            "required": [
                "EncryptionKeyID",
                "EncryptedAppInstancedKey"
            ]
        },
        "AppInstanceCertificate": {
            "type": "string",
            "description": "Certificate for this App. "
        },
        "NICELARootCertificate": {
            "type": "string"
        },
        "SecurityLevel": {
            "type": "string",
            "description": "If ServerOnly the key shall alway held securely in a cloud
application. If EdgeApplication the Keys may be inserted into mobile or web
application.",
            "enum": [
                "ServerOnly",
                "EdgeApplication"
            ]
        }
    },
    "required": [
        "Version",
        "NICELARootCertificate",
        "AppID",
        "AppInstanceCertificate",
        "AppDeveloperID",
        "AppInstanceID",
        "SecurityLevel",
        "AppInstancePrivateKey"
    ]
}
```