NICE App/Service Specification

Version 1.1

Edit History

Version	Date	Comments
1.1	26 Jan 2022	Initial Version 1.1 release

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 operation of the Service or App is described in the DataPipeline Specification.

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 DataPipelineController to set SceneModes. The SceneMode triggers the generation of 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 DataPipelineController can manage a single Device at any time. The DataPipelineController may configure the Device to send SceneMarks and SceneData to multiple end points or the Service may process and distribute SceneMarks and SceneData to multiple Applications. 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.

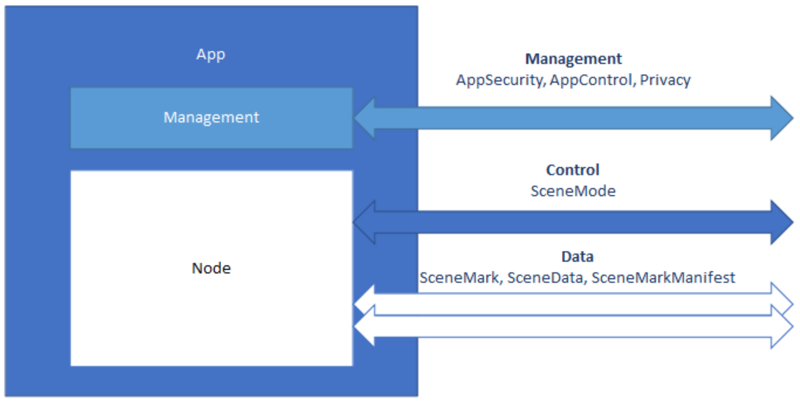

App
Management
Node
Management
AppSecurity, AppControl, Privacy
Control
SceneMode
Data
SceneMark, SceneData, SceneMarkManifest

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

3. Interfaces

3.1. Management

A **Management Interface** of an 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 list linked to the App.

The Management session uses the HTTPS protocol.

3.2. Control

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

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 a Service or App enables it to exchange data with a Cloud Service or a Device.

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

The Data session shall use the HTTPS protocol.

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. The App utilizes the Management interface to download the AppSecurity Object that enables it to interact with the user's account.
4. The App extracts the Control end points from the AppSecurity Object and is now able to request the AppControl Object which provides it with credentials to access DataServices.
5. 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 one or more Nodes distributed across Devices and Cloud Services. The Nodes in the cloud are configured by the Data Pipeline Controller.
6. The App consumes the SceneMarks and SceneData that are generated by the DataPipeline that it has created.
7. The App stops the DataPipeline and becomes inactive.
8. Steps 4 through 6 are repeated.

4.1. NICE App Developer Access to User’s Device and 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.

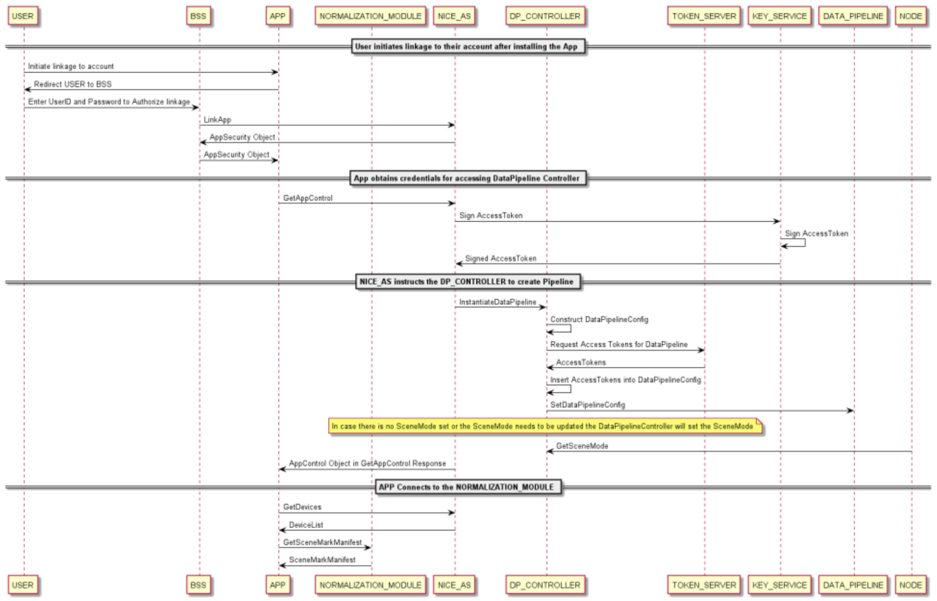
4.2. NICE App Instance

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 is issued an X.509 certificates corresponding with the App instance Public Key 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.

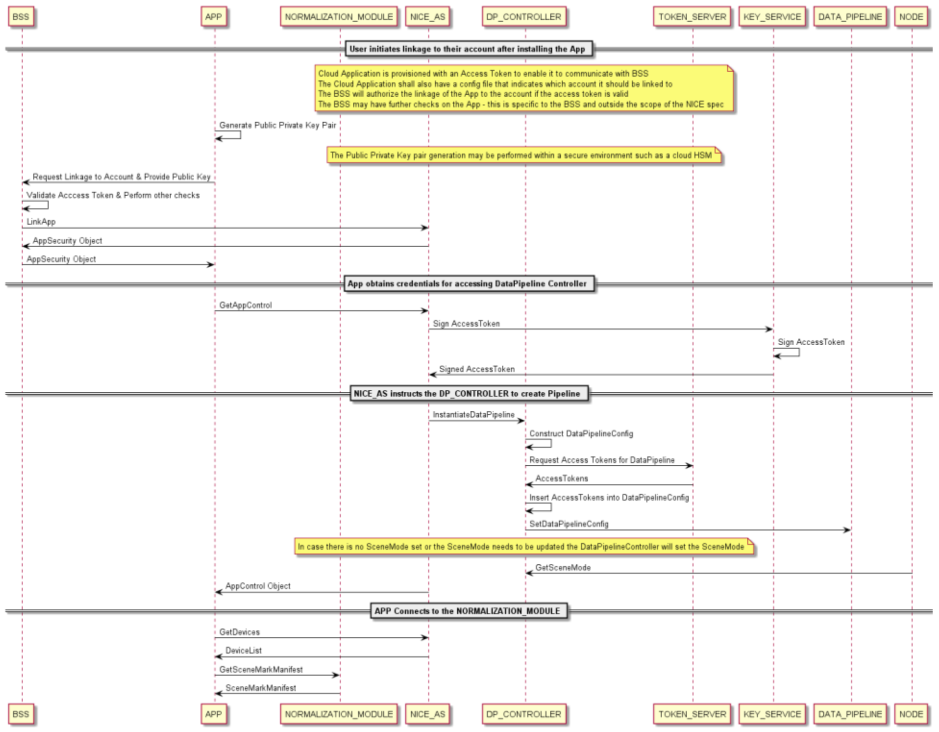
4.3. App Linkage to User's Account

Apps may be implemented as a mobile or as a cloud application. The following sequence diagram shows how an App is linked to an account.


USER
BSS
APP
NORMALIZATION_MODULE
NICE_AS
DP_CONTROLLER
TOKEN_SERVER
KEY_SERVICE
DATA_PIPELINE
NODE
User initiates linkage to their account after installing the App
Initiate linkage to account
Redirect USER to BSS
Enter UserID and Password to Authorize linkage
LinkApp
AppSecurity Object
AppSecurity Object
App obtains credentials for accessing DataPipeline Controller
GetAppControl
Sign AccessToken
Sign AccessToken
Signed AccessToken
NICE_AS instructs the DP_CONTROLLER to create Pipeline
InstantiateDataPipeline
Construct DataPipelineConfig
Request Access Tokens for DataPipeline
AccessTokens
Insert AccessTokens into DataPipelineConfig
SetDataPipelineConfig
In case there is no SceneMode set or the SceneMode needs to be updated the DataPipelineController will set the SceneMode
GetSceneMode
AppControl Object in GetAppControl Response
APP Connects to the NORMALIZATION_MODULE
GetDevices
DeviceList
GetSceneMarkManifest
SceneMarkManifest

When the App redirects the User to the BSS to authorize the linkage of the App to their account, the App passes a public key that corresponds to a private key generated in the App.

In case of a cloud application the interaction between the BSS and the cloud application may be simplified. The following sequence diagram shows how this interaction may occur. This interaction avoids the requirement for a user to enter their password and id to link the app to their account. The Cloud Application is provided with an Account ID and Access Token which it uses to link itself to an account.


BSS
APP
NORMALIZATION_MODULE
NICE_AS
DP_CONTROLLER
TOKEN_SERVER
KEY_SERVICE
DATA_PIPELINE
NODE
User initiates linkage to their account after installing the App
Cloud Application is provisioned with an Access Token to enable it to communicate with BSS
The Cloud Application shall also have a config file that indicates which account it should be linked to
The BSS will authorize the linkage of the App to the account if the access token is valid
The BSS may have further checks on the App - this is specific to the BSS and outside the scope of the NICE spec
Generate Public Private Key Pair
The Public Private Key pair generation may be performed within a secure environment such as a cloud HSM
Request Linkage to Account & Provide Public Key
Validate Acccess Token & Perform other checks
LinkApp
AppSecurity Object
AppSecurity Object
App obtains credentials for accessing DataPipeline Controller
GetAppControl
Sign AccessToken
Sign AccessToken
Signed AccessToken
NICE_AS instructs the DP_CONTROLLER to create Pipeline
InstantiateDataPipeline
Construct DataPipelineConfig
Request Access Tokens for DataPipeline
AccessTokens
Insert AccessTokens into DataPipelineConfig
SetDataPipelineConfig
In case there is no SceneMode set or the SceneMode needs to be updated the DataPipelineController will set the SceneMode
GetSceneMode
AppControl Object
APP Connects to the NORMALIZATION_MODULE
GetDevices
DeviceList
GetSceneMarkManifest
SceneMarkManifest

4.4. Implementation of Apps and Services

Apps may be implemented as a mobile App running on Android or IoS or as a Cloud Application.

A public private key pair is generated for the application.

This may be generated automatically using a secure processing unit in the device in the case of an Android and IoS device, or it may be generated in a Hardware Security Module in the case of a cloud service or application.

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. GetAppControl

Function

The "GetAppControl” API fetches the AppControl Object from the NICE Account Service. This object includes the URI and credentials for the NICE Device that the App will interact with. There are two types of connection that the object provides to the App. The first is the Control Interface over which the App can configure the Nodes within the Device. The second is the Data interface over which the App can obtain data.

Protocol(s) Used to Make Calls

WebAPI

Direction

Caller	APP
Callee	NICEAS

Request Parameters

AppInstanceID Object

Response Parameters

AppControl Object

5.1.2. 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

WebAPI

Direction

Caller	APP
Callee	NICEAS

Request Parameters

AccountID Object

Response Parameters

DeviceList Object

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",
                        "description": "Indicates whether the device is active and
connected to the network. ",
                        "enum": [
                            "Connected",
                            "Unconnected"
                        ]
                    },
                    "Description": {
                        "type": "string",
                        "description": "Information provided by user during
configuration describing the device."
                    }
                },
                "required": [
                    "Status",
                    "DeviceID"
                ]
            }
        }
    },
    "required": [
        "DeviceList",
        "AccountID",
        "Version"
    ]
}
```

6.2. AppSecurity Object

The AppSecurity Object provides key and credential material that enables the NICE App Developer to process a 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

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "AppSecurity",
    "description": "Application ID for the App developer",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "AccountID": {
            "type": "string"
        },
        "AppDeveloperID": {
            "type": "string",
            "description": "UserID for the Developer who develops the App."
        },
        "AppID": {
            "type": "string",
            "description": "Global Identifier for an App"
        },
        "AppInstanceID": {
            "type": "string",
            "description": "Identifier for the specific instance of the App"
        },
        "AppInstanceCertificate": {
            "type": "string",
            "description": "Certificate for this App. "
        },
        "NICELARootCertificate": {
            "type": "string"
        },
        "NICEASEndPoint": {
            "$ref": "Definitions.json#/definitions/EndPoint"
        }
    },
    "required": [
        "Version",
        "NICELARootCertificate",
        "AppID",
        "AppInstanceCertificate",
        "AppInstanceID",
        "NICEASEndPoint",
        "AccountID"
    ]
}
```

6.3. AppControl Object

This Object is used to manage which Devices and Data Services an instance of an App has access to. The tokens contained within this object are already encrypted objects that conform to the Access Token format. The object may be encrypted under the Public for the App and signed using the Private Key for the NICE AS.

AppControl

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "AppControl",
    "description": "This object sets up the permissions and end points for the Control
and Data Layers.",
    "properties": {
        "Version": {
            "type": "string",
            "enum": [
                "1.0"
            ]
        },
        "AppID": {
            "type": "string"
        },
        "AppInstanceID": {
            "type": "string"
        },
        "ControlEndPoints": {
            "type": "array",
            "description": "Control End Points are where the App can send control
commands to Devices. Each Control End Point Object contains the parameters for the end
points and an indication of which Device IDs are behind the End Point. ",
            "uniqueItems": true,
            "items": {
                "type": "object",
                "properties": {
                    "EndPoint": {
                        "$ref": "Definitions.json#/definitions/EndPoint"
                    },
                    "EndPointType": {
                        "type": "string",
                        "enum": [
                            "SceneMode Source",
                            "Account Service"
                        ]
                    }
                }
            }
        },
        "DataEndPoints": {
            "type": "array",
            "uniqueItems": true,
            "items": {
                "$ref": "Definitions.json#/definitions/EndPoint"
            }
        }
    },
    "required": [
        "Version",
        "AppInstanceID",
        "AppID",
        "ControlEndPoints"
    ]
}
```

6.4. AppInstanceID Object

AppInstanceID

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "AppInstanceID",
    "description": "AppInstanceID - unique ID that is associated with a specific
App.",
    "properties": {
        "AppInstanceID": {
            "type": "string"
        }
    },
    "required": [
        "AppInstanceID"
    ]
}
```

6.5. AccountID Object

AccountID

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "type": "object",
    "title": "AccountID",
    "description": "Unique identifier for an Account.",
    "properties": {
        "AccountID": {
            "type": "string"
        }
    },
    "required": [
        "AccountID"
    ]
}
```