Authentication Specification

Version 0.9

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

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 how access to Devices and Apps are managed, including the life cycle for the Device from its manufacture to its deployment and decommissioning and similarly how Apps are deployed. It describes how these processes are managed by the NICE License Authority and NICE Account Service.

2. Overview

The NICE System is made up of Devices, Data Services and Apps which interact with each other. The Authentication Specification describes how these interactions are managed by the NICE License Authority and the NICE Account Service. The Device has the following stages in its life cycle: Manufacture, Installation, Usage and Decommissioning. Through each stage of the Device's life cycle the Device is provisioned with credentials that enable it validate Apps and Data Services that interact with the Device. The operation of the Device through each stage of its life cycle is described in the Device Life Cycle Section.

Similarly for Apps and Data Services there are requirements for credentials to be distributed to Apps and Data Services to enable them to interact with Devices or other Data Services. This is described in the App/Service Specification.

The NICE License Authority provides keys, credentials and certificates for Devices that are manufactured. The Device manufacturer shall insert these into the Device in accordance to the security requirements for each item. The NICE LA shall manage the transition of the Device from manufacture to first installation. During first installation the Device shall be linked to a User's Account on a NICE Account Service. This process enables the NICE Account Service to manage the Device on behalf of the User who has the account with the NICE Account Service (NICE AS).

A Device can be decoupled from the User's Account and return to the same state that it was when first manufactured where the NICE License Authority is responsible for managing the access to the device. The device may be installed and associated with a different User Account on either the same or different NICE Account Service.

Apps shall be managed by the NICE Account Service. The App developer develops Apps and shall register the App as being available to be linked to User Accounts. The User shall use their User Account on the NICE Account Service to link the App to the Devices and Data associated with the User's Account. When the User has provided permission to the NICE Account Service to link the App to their User Account, the NICE Account Service shall provide Access Tokens to the App Instance or the App Provider's server to enable interaction with one or more of the Devices and Data Services provided for the User.

The process of managing access to Devices or Data Services shall be performed in accordance to the OAuth 2.0 specification. An Access Token is provided to each Entity to enable it to interact with the Device or Data Service. The Access Token contains the parameters for access (time window, permissions etc) and is protected using the public key provided in the X.509 certificate for the Device or Data Service and authenticated by the NICE Account Service key. This enables a Device or Data Service to determine that the Entity that is requesting access has been granted permission by the NICE Account Service.

All communication between Devices, Data Services and Apps shall be protected using either Transport Layer Security or Datagram Transport Layer Security.

3. NICE Life Cycle

The NICE Device has stages in its life cycle where different authorities have control over aspects of the device. At manufacture of the device the NICE licensing authority provides credentials to the device that enable secure communication, secure access and data protection. The NICE Device Life Cycle contains the following steps:

1. Manufacture of Device
2. Installation Phase
3. Operational Phase
4. Uninstallation Phase

3.1. Manufacture of Device

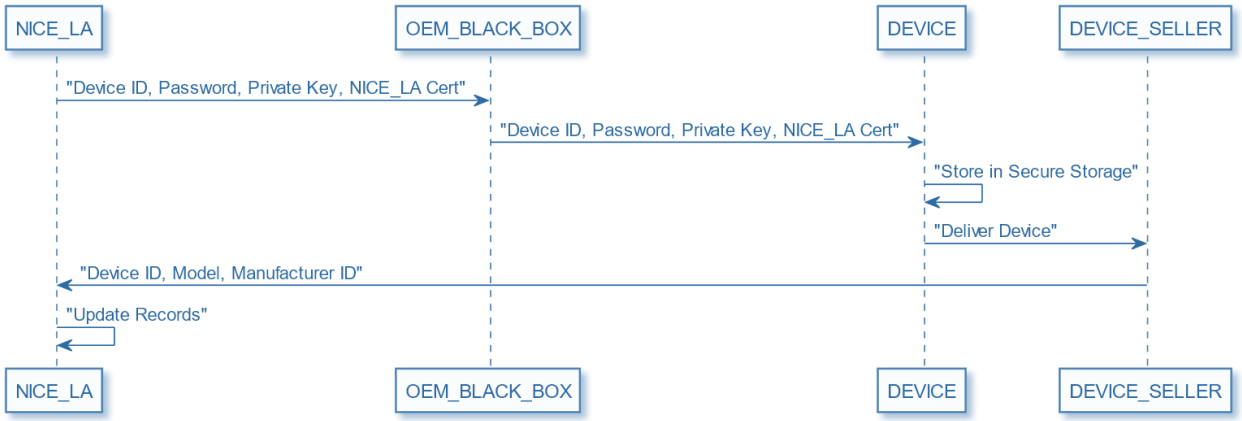

Figure 1. Manufacture of Device

The NICE License Authority securely transfers the Device credentials in the "Device Security Object" to a secure "Black Box" within the manufacturing environment. The Black Box is a secure execution environment which communicates with the device on the manufacturing line and transfers the credentials to the device.

During manufacture the device shall have the following credentials inserted into the device:

1. Device ID: Permanent ID of the device. Readable by User.
2. Private Signing Key of the Device: Permanent Key of Device - Shall be securely handled and is only used for generating an Application level signature of Objects generated by the Device. It cannot be updated.
3. Private Encryption Key of the Device: Permanent Key of the Device - Shall be securely handled and is only used to decrypt encrypted objects that are sent to the Device. It cannot be updated.
4. X.509 Certificate and Private Key for TLS communication security. Device Certificate for Transport Level Security or Datagram Transport Level Security.
5. Device TLS Private Key: Private Key used for TLS security only.
6. NICELA Root Certificate: Root X.509 Certificate for the NICE Licensing Authority.
7. Master Issuer ID. This is the value of the "iss" field that must be present in a JSON Web Token.

8. Firmware Source ID: ID of the Entity that may sign FirmwareUpdate objects sent to the Device. The Firmware Source may be the device Manufacturer, the Device Seller, the NICE Account Service or any other entity that is authorized to manage Firmware updates.
9. Firmware Update Service Certificate: X.509 Certificate for the Firmware Update Service provided by the entity corresponding to the FirmwareSourceID.
10. NICELA URI: URI for the NICELA the device uses this URI to download settings for its operation.
11. Allowed TLS Root Certificates: which are the Root Certificates that the device is allowed to accept for securing TLS communication. This allows the device to interact with browsers and other devices using other Certificate Authorities than the NICE LA.

The above credentials, with the exception of the certificate for the Firmware Update Service, are permanent for the entire life cycle of the device.

The Device ID, Unique Password, Private Key and X.509 certificates are issued by the NICE License Authority (LA).

3.2. Installation Phase

The end user can link devices to their account.

1. The End User shall access the Device ID and Device Password that has been provided with the Device.
    - There may be several methods used to deliver the Device Password to the end user.
    - These may include having the Device ID and Device Password visible on the device.
2. The Device Password shall only be used to register the Device to a User Account. The Device ID and Device Password is entered on the NICE LA to enable the NICE LA to allocate the Device to a NICE Account Service.
3. The User shall log into their NICE Account Service account. The User shall be directed to the NICE Licensing Authority. The NICE Licensing Authority shall accept the Device credentials provided by the User and shall provide an AccessToken to the User's NICE Account Service. This AccessToken enables the NICE Account Service to interact with the Device.

The NICE LA will only allow the linkage of the device to a NICE Service Account and provide the device's X.509 certificate if the Device Seller has already registered the device with the NICE LA.

Each device shall have the NICE Licensing Authority Root Certificate stored in the device.

The Management Object is signed using the Private Signing Key of the NICE Licensing Authority (as defined in the X.509 certificate stored in the device) and shall be encrypted with the unique Public Encryption Key corresponding to the Private Encryption Key of the Device. The Device Private Encryption Key shall be stored in the device during manufacture.

When the User wishes to assign the device to the NICE Account Service, the User shall initiate the NICE Account Management Application to perform the OAuth2 session with the NICE License Authority. The User shall use the factory configured Device ID and Password to enable the NICE License Authority to generate an Access Token for the NICE Account Service to access the Device.

The NICE License Authority shall provide the Management Object to the Device.

This object shall contain the NICE Account Service ID that the device is being linked to and an expiry time for the message. If the message is received outside of the time defined by the start and expiry times it shall be ignored.

This object is delivered by the NICE LA to the Device.The Device Management Object shall be signed with the Private Signing Key that can be validated by the NICE License Authority and encrypted with the Device Encryption Public Key.

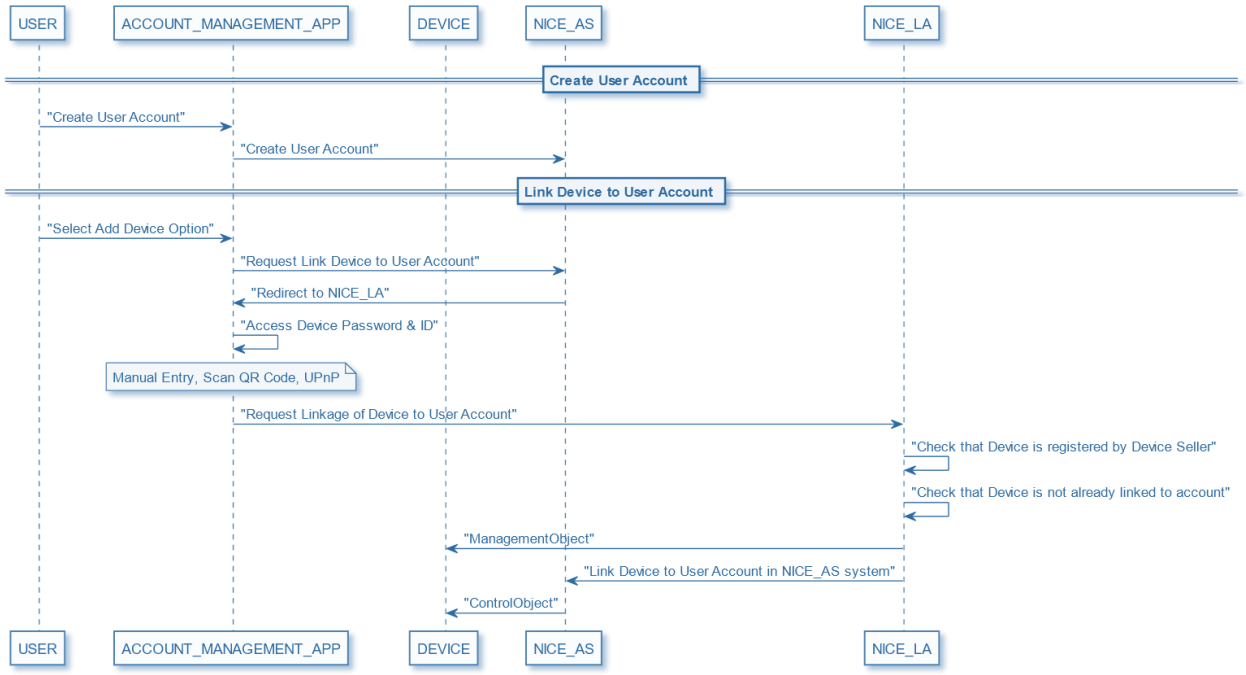

Figure 2. Linking a Device to a User Account

3.3. Operational Phase

When the device has been associated with a user's account, the NICE Account Service shall send the Control Object to the Device. The NICE Account Service uses the same Access Token to log into the device and to supply the device with the Control Object. The Control Object shall be signed with the NICE Account Service Private key and the encrypted using the Device Public Key.

The Control object shall be signed with the Private Signing Key of the NICE Account Service and encrypted with the Public Encryption Key of the Device.

The Device is now associated with the User's Account and may be utilized by Data Services or Apps.

3.4. Uninstallation Phase

The End User may decide to use a different NICE Account Service Provider or may give away or sell the Device to another User.

- The End User shall first delete the Device from their Account.
- The NICE Account Server shall communicate with the NICE LA to indicate that the Device is no longer associated with a User Account. The NICE LA shall register the Device as being in the state where the device is un-allocated to a User. The Device is ready for use by another User.

4. Device Network Security

Network Security covers how the communication links between Entities within the NICE system are secured. All communication shall be protected using either TLS or DTLS (depending on the network protocol). The authentication shall always be mutual.

The DeviceControl Object shall contain a list of end points with which the Device may communicate. The Device shall enforce the encryption settings defined for the end points in the Control Object.

5. Prevention of Replay Attacks

A replay attack is where a message is captured by the attacker and resubmitted at a later time to reset the Device into an earlier state. To prevent this type of attack, the Management and Control objects shall have a time window parameter within the object outside of which the object shall not be processed by the receiving device. The time window is enforced by each Entity having a Trusted Time Clock. The synchronization of the clock is described in the Device Trusted Time Section.

6. Device Trusted Time

The Device shall maintain a Trusted Time Clock when powered up. A Trusted Time Clock shall be initialized using the Trusted Time Protocol defined in this document and shall be resistant to attempts to change the value of the Trusted Time Clock or change the operation of the Trusted Time Clock. On power up the device shall request a time stamp from the NICE time server using the Trusted Time Protocol. Once the trusted time stamp has been received it is used to synchronize the trusted time clock within the Device.

7. Firmware Signing

For updates of firmware for the device the file containing the firmware update shall be signed with the Firmware Update Public Key. The Firmware Update Public Key for the validation of the signature shall be loaded into the device either at manufacture or subsequently in the Management Object.

The Device shall accept updates to its firmware at any stage of its life cycle irrespective of whether it has been registered by the Device Seller with the NICE LA or associated with a NICE Account Service.

The FirmwareUpdate Object that is generated for the firmware shall contain the following data fields:

1. **Period for which it is valid.**
    1. This defines a window of time during which the signature may be processed and the firmware is accepted by the device.
    2. The device shall use the time stamp that it has obtained and maintained through the secure time protocol to check whether the signature is within the required time window.

- The device shall store the value of the "Minimum Acceptable Version" and "Oldest Valid Date" fields in its secure storage and shall check the incoming FirmwareUpdate object against these parameters before processing the object.

- The FirmwareUpdate object shall have these fields present otherwise it is an invalid firmware signature object and shall be rejected.

The loader which updates the firmware on the device shall first generate the hash value for the file containing the new version of firmware, then validate the signature for the new version before enabling the firmware to execute on the device.

- The NICE License Authority shall provide device manufacturers with Private/Public key pairs to enable the manufacture to sign firmware.
- The NICE Licensing Authority may also generate new Private/Public key pairs for manufacturers in case of compromise of the manufacturing signing key.
- The public key of this pair may be delivered to the device in a Device Management Object.
- The Management Object containing the updated Public Key shall have the same fields as the code update JSON structure – valid time period and lowest version number.

This shall be processed in the same way as for accepting a new firmware image.

The FirmwareUpdate object shall be signed by the Firmware Update Source **Private Key**. The certificate containing the key to be used to validate the FirmwareUpdate object is either inserted during the manufacturing process or over ridden using the Management Object.

- The device shall verify the object before processing it.
- The FirmwareSourceID field much match that of the device otherwise the signature shall be rejected.
- The Device shall have the X.509 certificate and ID for the Firmware Update Source.
- The FirmwareSourceID may be changed via the Device Management Object.

The validation of the firmware image follows these steps:

1. Check the version and date fields in the FirmwareUpdate Object to ensure that they are within the current date and that the version is allowable under the previously downloaded version control settings.
2. Check the Subject Field in the Firmware Source's X.509 certificate against the Seller ID in the device.
3. Validate the Sellers X.509 certificate using the NICE LA X.509 certificate permanently loaded into the device at manufacture.
4. Create a Hash of the Code image using SHA 256. Compare this value to that carried in the hash field in the FirmwareUpdate object.
5. Validate the FirmwareUpdate Object using the Public Key in the X.509 certificate of the Device Seller.

If all of these checks pass, the firmware can be uploaded and the version and date fields can be updated.

8. OAuth 2.0 Usage in the NICE System

The NICE System shall adhere to the OAuth 2.0 standard for managing Access Tokens used by Entities to establish connections to other Entities.

The **OAuth2** specification defines the process for a server to grant an access token to a client to enable the client to access a resource.

- The client presents the access token to the resource server and if the resource server determines that the token is valid the it provides access.
- OAuth2 does not specify the structure of the token or how the token is distributed to the resource server.

8.1. Linking an AppInstance to an Account ID Using NICE AccountID and Password

The linkage of the App to an AccountID is handled through a single OAuth session. The App instantiates a browser session with a URI made up of the following:

https://AuthenticationServerAddress/auth?response_type=code&client_id=AppID&redirect_uri=REDIRECT_URI&state=1234zyx

The "https://AuthenticationServerAddress/auth?response_type=code&client_id=AppID&redirect_uri=REDIRECT_URI" string is provided in the AppSecurity Object under the "AuthenticationServerURI" field. The App shall generate a RandomNumber as a state value and append to the above as "&state=RandomNumber".

The RFC8252 OAuth 2.0 for Native Apps describes methods for how a native App can initiate a browser session for the User to authorize the native apps access to the protected resource. The User interacts with the instantiated browser session, enters their user name and password which is validated by the authorization server. The authorization server returns the Access Token to the browser together with a REDIRECT_URI value which enables the browser to link back to the Apps backend server or to the App itself (depending on the value of REDIRECT_URI).

9. JSON Web Token Usage in the NICE System

RFC 7519 defines JSON Web Tokens.

- These are tokens containing required and optional fields defined in a JSON format.
- The encryption and authentication of these tokens are defined in the JOSE specifications also developed by the IETF.

The NICE Account Service shall issue Access Tokens that conform to the JSON Web Token definition. The client performs the following steps to use the Token to access a resource:

1. The client makes a request to the NICE AS OAuth2 server.
2. The NICE AS OAuth2 server shall issue a token in the format defined by the JSON Web Token format. This token validated using the NICE AS Private Signing Key.
3. The client presents the token to the Device, requesting access to a resource on the Device.
4. The Device checks the X.509 certificate that is referenced in the Access Token and validates the contents of the Access Token using the Public Key referenced in the NICE AS OAuth2 server's X.509 certificate.
5. The Device shall also check the fields to check the whether the Access Token should provide access to the resource. For example are the "valid to" and "not before" fields within the current date.

9.1. App Access to Devices and Services

The NICE Account Service controls which SceneData and Devices may be accessed by an App. The App shall do the following steps to access User's Account and Devices:

1. An App requests access to a User's Account using the OAuth 2.0 protocol.
2. The NICE Account Service acts as the OAuth defined authorization server.
3. If the User provides permission for the App to access its Devices, the NICE Account Service will provide the App an Access token which will allow the App to connect to a device to set up a control session. This transfer of the Access Token to a Native App shall be protected by the Proof Key for Code Exchange (PKCE [RFC 7636]) extension to OAuth.
4. The App does not process the contents of the token. These are processed by the device or the server holding the Scene Data.
5. If the Device or the Server verifies the Access Token the App can access data or settings in the Device.

In the case of SceneData, there may be a second layer of protection provided by the Privacy Management System. When interacting with Data that is protected by the Privacy Management System, App shall be provided with a Privacy Object which provides SceneEncryptionKeys to decrypt the Data. The Device shall adhere to the processing rules that are defined by the Privacy Object.

9.2. NICE Application Access Restriction Policies

Figure 3. App Roles and Permissions

The App is provided an Access Token to enable Access to Data or Devices. The Access Token shall contain a "Permissions" field which defines the APIs that are accessible to the bearer of the Access Token.

Five permissions are currently defined:

- Management - the bearer of the AccessToken may make calls to the Management API for the Device.
- Control - the bearer of the AccessToken may make calls to the Control API for Nodes in the Device.
- Data - the bearer of the AccessToken may request SceneMarks, SceneData or LiveStreams from the Device.
- Status - the bearer of the AccessToken may request the Status of the Device.
- Test - there are no specific Test APIs or modes defined in this version of Specification. However should the Device Maker provide such APIs or later versions of the NICE specifications of the Test APIs, the bearer of the AccessToken may access these APIs.

Role-based access control includes 3 actors:

- Access principal: an object that represents an App, group of Apps requesting access to NICE services
- Resource principal: an object that represents the NICE server services (API, service…)

- Role principal: an node that represents the Access restriction

The following functionality is used to perform the access-control:

- Role assignment: link a role to a permission
- Role scope: limit what service the role is allowed to access
- Role management : add and remove permission of APP_ID

The following diagram illustrates the steps in the process of access control, in which PA (Policy Administration) creates and manages policies and PE (Policy Enforcement) is responsible for authorization decisions based on policies.

NICE Role-based access control has the following components:

- Policy Administration (PA) : PA creates and manages policies.
- Policy Decision (PD) : PD is responsible for storing and analyzing policy information from APP request.
- Policy Enforcement (PE) : PE is responsible for authorization decisions based on policies.
- Policy Content (PC): PC provides importation attribute values such as APP_ID, Access role, Access level…

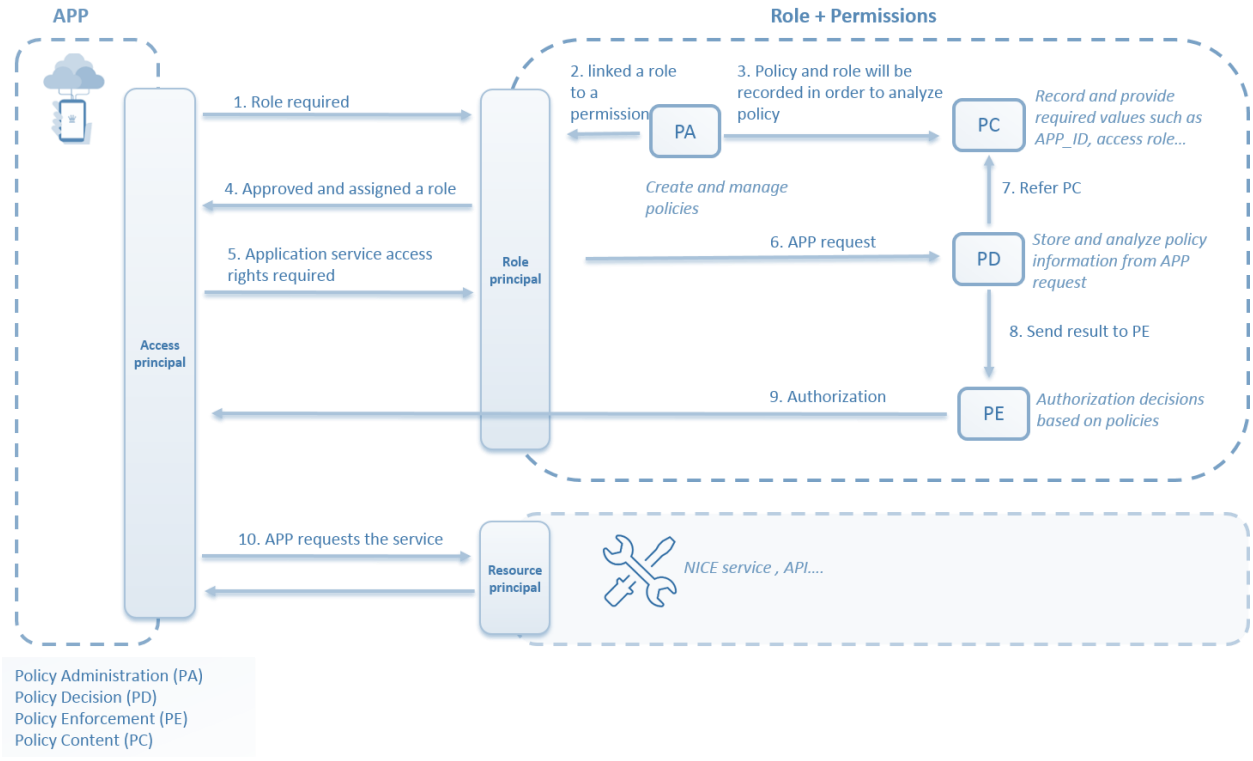

Figure 4. Role and Permissions Architecture

Rule is unit of the policy and PA combines rules to form Policy or set of policies. Furthermore, PA assigns certain roles to the App (based on APP_ID) for developers applying for a new App when service/API is requested. The role is checked against assigned role from the repository by PD (PC must be recorded in order to analyze policy).

9.3. JWT (JSON Web Tokens) Format

RFC 7519 defines the format for JSON Web Tokens. This specification defines the specific usage of fields defined in this RFC.

9.3.1. Authentication

The token shall be authenticated using the ES256 algorithm (ECDSA using P-256 and SHA-256).

- The identification of the public key to be used to authenticate the message shall be carried using an X.509 certificate.
- This certificate shall be validated against the NICE LA root certificate.

9.3.2. Fields

The following example shows the JWT structure that the NICE LA would generate to enable the NICE Account Service to access a specific NICE device.

- Similar tokens would be generated for Apps to access a user's account or devices.
- The fields described below shall always be present.
- The values in the fields shall depend on:
    - Who is issuing the token.
    - Who is expected to receive the token.
    - Who is going to use the token.

The following is the payload of the token.

- All the fields defined below are **required** when used in the NICE Ecosystem.
- The token shall be encrypted using the public key of the device and signed with the private key of the issuing authority (NICE LA or NICE Account Service).

9.3.3. Access Token

- The "**iss**" field refers to the party generating the JWT.
- The "**sub**" field refers to the party that is going to use the token to access a service or a device.
- The "**aud**" field refers to the party that will receive and validate the token.
    - In case that the token is validated the party shall grant access to the holder of the token.
- The "**exp**" field refers to the expiry date for the token.
    - After this time the token shall not be accepted.
- The "**nbf**" field refers to the date before which the token is not valid.
- The "**iat**" field refers to the date of issue of the token.
- The "**jti**" field is a unique ID for the token.
    - A specific token can be revoked using the Management Object for the device issued by the NICE LA or NICE Account Service.
    - The device must check this field against the revocation list delivered in the Management Object.
- The **"Permissions"** field describes which APIs are allowed to be called by the bearer of the Access Token.
- The **"EncryptionEnforced"** field if true requires all Objects passed over the interface to be encrypted and authenticated. The Object shall be encrypted using the Device public key and the

Authentication shall be signed using the public key that is associated with the EndPointID that is indicated in the "sub" field. The certificate for the EndPointID shall be validated against the NICE LA or NICE AS certificates.

9.3.4. Types of Tokens

The following types of tokens are used in the NICE eco system:

- Token issued by the NICE LA to enable a NICE Account Service to access a NICE device.
- Token issued by a NICE Account Service to enable access to either a NICE device or a NICE account managed by the NICE Account Service.

	NICE LA to Enable Access to NICE Device	**NICE Account Service to Allow Access to Account**
"iss"	NICELicenseAuthority	NICEAccountServiceID
"sub"	NICEAccountServiceID	AppInstanceID
"aud"	DeviceID	AccountID DeviceID

The device receiving the token shall ensure that the following matches:

1. "iss" shall be the same the NICE License Authority provided during manufacture or the NICE Account Service ID provided when the device is associated with an end user account. This is carried as a data field in the Management Object.
2. "aud" is the EndPointID.
3. The entity identified in the "iss" field should match the X.509 certificate for the entity and the public key carried in that X.509 certificate shall be used to verify the token.

10. Control Interface

The Control Interface is used by the App to create a Data Pipeline to generate SceneMarks and SceneData.

10.1. EstablishControlSession

Function

The App uses this API Call to enable it communicate with Devices. Once the App has established a session with a Device it can use the APIs defined in the Data Pipeline Section to build a Data Pipeline using Nodes within the Device and connecting these Nodes to Nodes located in other Devices with which it has established Control Sessions.

The "EstablishControlSession" API fetches the 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. The Control Interface makes use of either MQTT or WebAPI. The Data Interface may use MQTT, WebRTC or WebAPI.

Protocol(s) Used to Make Calls

MQTT Management

WebAPI

Direction

Caller	NICE_AS
Callee	APP

Request Parameters

Empty

Acknowledgement Parameters

AppControl Object

10.2. CloseControlSession

Function

This API call is used to close an existing Control Session between an App and a Device. When this call is made an existing session with the Device is closed and the Device becomes free to be used by other Apps.

Protocol(s) Used to Make Calls

MQTT

WebAPI

Direction

Caller	APP
Callee	NICE_AS

Request Parameters

Empty

Acknowledgement Parameters

Empty

11. Trusted Time Interface

11.1. GetDateTime

Function

The APP must have the API to set the current time.

The RequestTime Object initiates a request for an authenticated time and date stamp from the NICE Account Service. The response from this request is used to synchronize the secure time clock within the trusted execution environment of the device.

Protocol(s) Used to Make Calls

MQTT Management

WebAPI

Direction

Caller	APP
Callee	NICE_LA, NICE_AS

Request Parameters

TrustedTimeRequest Object

Acknowledgement Parameters

TrustedTimeResponse Object

12. Data Objects

12.1. DeviceSecurity

The following JSON structure is provided by the NICE LA to the device manufacturer using a secured channel to the Manufacturer's server that is responsible for inserting these fields into the Device.

DeviceSecurity

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "DeviceSecurity",
    "description": "Provided to the manufacturer for provisioning to the device.",
    "additionalProperties": false,
    "properties": {
        "DeviceID": {
            "type": "string",
            "description": "Permanent ID of the device. Readable by end user."
        },
        "DevicePassword": {
            "type": "string",
            "description": "Password used to log into NICE LA device account and
associate device with NICE Account Service Provider. It shall not be accessible to the
Device software."
        },
        "DevicePrivateKey": {
            "type": "object",
            "description": "Permanent Private Key of Device - Shall be securely
handled. Used for Application level Decryption only.",
            "properties": {
                "EncryptionKeyID": {
                    "type": "string",
                    "description": "Key ID of the Public Key that has been used to
encrypt the AppInstancePrivateKey"
                },
                "EncryptedDeviceKey": {
                    "type": "string"
                }
            },
            "required": [
                "EncryptionKeyID",
                "EncryptedDeviceKey"
            ]
        },
        "DeviceTLSCertificate": {
            "type": "string",
            "description": "Device Certificate for Transport Level Security or
Datagram Transport Level Security"
        },
        "DeviceTLSPrivateKey": {
            "type": "string",
            "description": "Private Key used for TLS security only."
        },
        "NICELARootSigningCertificate": {
            "type": "string",
            "description": "X.509 Certificate for the Root Key for Keys that are used
to sign objects issued by the NICE LA"
        },
        "NICELARootEncryptionCertificate": {
            "type": "string",
            "description": "X.509 Certificate for the Root Key for Keys that are used
to encrypt objects that are sent to the NICE LA"
        },
        "MasterIssuerID": {
            "type": "string",
            "description": "\"iss\" field in a JWT issued by the NICE LA to allow
another entity to access the device"
        },
        "FirmwarePublisherID": {
```

```
            "type": "string",
            "description": "ID for the entity that shall sign any FirmwareUpdate
object that is sent to the Device."
        },
        "FirmwarePublisherCertificate": {
            "type": "string",
            "description": "X.509 Certificate containing verification key for firmware
upgrades"
        },
        "NICELAURI": {
            "type": "string",
            "description": "URI for the NICELA the device uses this URI to download
settings for its operation."
        },
        "AllowedTLSRootCertificates": {
            "type": "array",
            "description": "Each entry in the arracy contains a Root Certificate that
is valid for TLS communication. Only entries in this field shall be accepted by the
Device. ",
            "items": {
                "type": "string",
                "description": "Allowed X.509 root certificates that may be used for
Management Level TLS communication"
            }
        }
    },
    "required": [
        "MasterIssuerID",
        "NICELAURI",
        "DevicePassword",
        "DeviceTLSPrivateKey",
        "AllowedTLSRootCertificates",
        "DeviceID",
        "DeviceTLSCertificate",
        "NICELARootSigningCertificate",
        "NICELARootEncryptionCertificate",
        "DevicePrivateKey",
        "FirmwarePublisherID",
        "FirmwarePublisherCertificate"
    ]
}
```

12.2. AccessToken

The Access Token is used by an App to Access either a Device or a Data Service. The Token is encrypted using the Public Key for the Device or Data Service and is signed by the issuer of the token. The issuer may be either the NICE LA or the NICE AS. It shall be rejected if it is not signed and encrypted. It shall also be rejected if the signature fails. It shall be rejected if the object is misformed.

AccessToken

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "AccessToken",
    "additionalProperties": false,
    "properties": {
```

```
        "iss": {
            "type": "string",
            "description": "Issuer of Token"
        },
        "sub": {
            "type": "string",
            "description": "Subject of Token"
        },
        "aud": {
            "type": "string",
            "description": "Audience for Token"
        },
        "exp": {
            "type": "string",
            "description": "Expiry date of token"
        },
        "nbf": {
            "type": "string",
            "description": "Not valid before this date"
        },
        "iat": {
            "type": "string",
            "description": "Date of Issue"
        },
        "jti": {
            "type": "string",
            "description": "JSON Token ID unique identifier for this token. This
should be checked against revoked tokens for the device before accepting. "
        },
        "Permissions": {
            "type": "array",
            "items": [
                {
                    "description": "This defines which level of API is accessible to
the holder of this token. For example if management, then the bearer of the token may
make management API calls. ",
                    "enum": [
                        "Management",
                        "Control",
                        "Data",
                        "Status",
                        "Test"
                    ]
                }
            ]
        },
        "EnforceEncryption": {
            "type": "boolean",
            "description": "If TRUE then all Control Objects sent to via this
connection shall be encrypted and authenticated over and above the encryption provided
by TLS."
        }
    },
    "required": [
        "iss",
        "sub",
        "aud",
        "exp",
        "nbf",
        "iat",
        "jti",
        "Permissions",
        "EnforceEncryption"
```

```
    ]
}
```

12.3. TrustedTimeRequest

This Object is used by an Entity that is requesting a secured time stamp from either the NICE LA or the NICE AS. The request shall be encrypted using the Public Key of the NICE AS or NICE LA. The request shall be signed using the Private Key of the Entity making the request. The random number generated as part of the challenge protocol shall be generated by a random number generator that conforms to the Device or App guidelines specification.

TrustedTimeRequest

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "TrustedTimeRequest",
    "additionalProperties": false,
    "properties": {
        "DeviceID": {
            "type": "string"
        },
        "RandomChallenge": {
            "type": "string",
            "description": "Random Number at least 64 bytes long, encrypted using the
public key of the NICE AS or NICE LA."
        },
        "EndPointCertificate": {
            "type": "string",
            "description": "X.509 Certificate for the EndPoint making the request"
        }
    },
    "required": [
        "DeviceID",
        "RandomChallenge",
        "EndPointCertificate"
    ]
}
```

12.4. TrustedTimeResponse

The server shall return the following structure with the Random Challenge correctly decrypted. The Object shall be encrypted using the Device Public Key and signed using a Private Key that is certified by either the NICE LA or NICE AS for this purpose.

TrustedTimeResponse

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
```

```
    "title": "TrustedTimeResponse",
    "description": "Response containing a time stamp for the requesitng device.",
    "additionalProperties": false,
    "properties": {
        "EndPointID": {
            "type": "string"
        },
        "ReturnedRandomChallenge": {
            "type": "string",
            "description": "Random challenge that was carried in the SecureTimeRequest
object to which this response is responding. "
        },
        "DateTimeStamp": {
            "type": "string",
            "description": "Time stamp in ISO 8601 format."
        }
    },
    "required": [
        "EndPointID",
        "ReturnedRandomChallenge",
        "DateTimeStamp"
    ]
}
```

12.5. AppControl

This Object is used to manage which Devices and Data Services an instance of an App has access to. The encryption and signing of this object is optional. The tokens contained within this object are already encrypted objects that conform to the Access Token format defined in this document. 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-04/schema#",
    "type": "object",
    "title": "AppControl",
    "description": "This object sets up the permissions and end points for the Control
and Data Layers.",
    "additionalProperties": false,
    "properties": {
        "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. ",
            "items": [
                {
                    "type": "object",
                    "properties": {
                        "EndPointURI": {
```

```
                            "type": "string"
                        },
                        "Protocol": {
                            "enum": [
                                "WebAPI",
                                "MQTT",
                                "WebRTC"
                            ]
                        },
                        "MQTTQoS": {
                            "enum": [
                                "0",
                                "1",
                                "2"
                            ]
                        },
                        "EndPointID": {
                            "type": "string"
                        },
                        "AccessToken": {
                            "type": "object",
                            "properties": {
                                "Token": {
                                    "type": "string",
                                    "description": "Actual Token to be used to access
End Point"
                                },
                                "StartDateTime": {
                                    "type": "string",
                                    "description": "Date before which the token is
invalid. Date in ISO8601 format."
                                },
                                "EndDateTime": {
                                    "type": "string",
                                    "description": "Date after which the Token is not
valid."
                                }
                            },
                            "required": [
                                "Token"
                            ]
                        },
                        "EncryptionOn": {
                            "type": "boolean",
                            "description": "If true then the Control Objects
transfered over this channel shall be encrypted and authenticated. The App requires an
App Instance Security Object with the appropriate key and certificate."
                        },
                        "EndPointCertificate": {
                            "type": "string",
                            "description": "If encryption is on then the public key in
this certificate shall be used to encrypt traffic and validate data objects. "
                        }
                    },
                    "required": [
                        "EndPointURI",
                        "Protocol",
                        "AccessToken",
                        "EndPointID",
                        "EncryptionOn"
                    ]
                }
            ]
```

```
        },
        "DataEndPoints": {
            "type": "array",
            "items": [
                {
                    "type": "object",
                    "properties": {
                        "EndPointURI": {
                            "type": "string"
                        },
                        "Protocol": {
                            "enum": [
                                "WebAPI",
                                "MQTT",
                                "WebRTC"
                            ]
                        },
                        "MQTTQoS": {
                            "enum": [
                                "0",
                                "1",
                                "2"
                            ]
                        },
                        "EndPointID": {
                            "type": "string"
                        },
                        "AccessToken": {
                            "type": "object",
                            "properties": {
                                "Token": {
                                    "type": "string",
                                    "description": "Actual Token to be used to access
End Point"
                                },
                                "StartDateTime": {
                                    "type": "string",
                                    "description": "Date before which the token is
invalid. Date in ISO8601 format."
                                },
                                "EndDateTime": {
                                    "type": "string",
                                    "description": "Date after which the Token is not
valid."
                                }
                            },
                            "required": [
                                "Token"
                            ]
                        },
                        "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."
                        },
                        "EndPointCertificate": {
                            "type": "string",
                            "description": "If encryption is on then the public key in
this certificate shall be used to encrypt traffic and validate data objects. "
                        },
                        "DataType": {
                            "enum": [
```

```
                                "SceneData",
                                "SceneMarks"
                            ]
                        }
                    },
                    "required": [
                        "EndPointURI",
                        "Protocol",
                        "AccessToken",
                        "DataType",
                        "EndPointID",
                        "EncryptionOn"
                    ]
                }
            ]
        }
    },
    "required": [
        "AppInstanceID",
        "ControlEndPoints",
        "DataEndPoints",
        "AppID"
    ]
}
```