
Mobile SDK Guide

Advanced Authentication

Version 6.2

Legal Notice

For information about legal notices, trademarks, disclaimers, warranties, export and other use restrictions, U.S. Government rights, patent policy, and FIPS compliance, see <https://www.netiq.com/company/legal/>.

Copyright © 2018 NetIQ Corporation, a Micro Focus company. All Rights Reserved.

Contents

About NetIQ Corporation	5
About this Book	7
1 Prerequisites	9
2 Use Case	11
3 Example	13
4 Creating the Mobile SDKs	15
Representing the Geo Location	15
Static Public Member Functions.	15
Managing the Authentication Handler Interface	15
Public Member Functions.	15
Creating a Data Representation and Converting Bytes and Hex Strings	17
Public Member Functions.	18
Static Public Member Functions.	19
Handling Enrollment Result	20
Public Member Functions.	20
Example	21
5 Managing Endpoints	23

About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity and risk—and how we can help you control them.

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

Enabling critical business services, better and faster

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

Our Philosophy

Selling intelligent solutions, not just software

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate—day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

Driving your success is our passion

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with—for a change. Ultimately, when you succeed, we all succeed.

Our Solutions

- ♦ Identity & Access Governance
- ♦ Access Management
- ♦ Security Management
- ♦ Systems & Application Management
- ♦ Workload Management
- ♦ Service Management

Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

Worldwide:	www.netiq.com/about_netiq/officelocations.asp
United States and Canada:	1-888-323-6768
Email:	info@netiq.com
Web Site:	www.netiq.com

Contacting Technical Support

For specific product issues, contact our Technical Support team.

Worldwide:	www.netiq.com/support/contactinfo.asp
North and South America:	1-713-418-5555
Europe, Middle East, and Africa:	+353 (0) 91-782 677
Email:	support@netiq.com
Web Site:	www.netiq.com/support

Contacting Documentation Support

Our goal is to provide documentation that meets your needs. The documentation for this product is available on the NetIQ Web site in HTML and PDF formats on a page that does not require you to log in. If you have suggestions for documentation improvements, click **Add Comment** at the bottom of any page in the HTML version of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

NetIQ Communities, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, NetIQ Communities helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit community.netiq.com.

About this Book

This guide is designed for the developers and describes how to create and edit the SDKs for mobile apps. It outlines the prerequisites and guides you through the different functions in the SDKs.

Intended Audience

This book provides information for individuals responsible for understanding administration concepts and implementing a secure, distributed administration model.

1 Prerequisites

You must have the following prerequisites:

You can download the sdk at

2 Use Case

Bob is currently using the Advanced Authentication mobile application. However, he wants options that allow him to create the mobile apps according to his requirements. These requirements can be to change the look and feel of the app,

With the new mobile SDKs, Bob can edit the SDKs and bundle them with the project to create his own app for iOS and Android.

3 Example

You can allow users to authenticate to another user's account by using their own authenticators. For example, if the share authenticator option is enabled, the secretary's account can be shared with the account of boss and the secretary will be able to authenticate to the account of boss by using her own authenticators.

4 Creating the Mobile SDKs

You can create the Mobile SDKs.

Representing the Geo Location

The `AuCoreMobileSDK.Geocoordinate` Class Reference is used to represent the geo location in the mobile application.

Static Public Member Functions

`static Geocoordinate FromJson (string json)`

The following table describes the properties of this member function.

Property	Functions	Description
double Accuracy [get, set]	double <code>AuCoreMobileSDK.Geocoordinate</code> <code>.Accuracy[get], [set]</code>	Accuracy of the location in meters
double Latitude [get, set]	double <code>AuCoreMobileSDK.Geocoordinate</code> <code>.Latitude[get], [set]</code>	Latitude in degrees
double Longitude [get, set]	double <code>AuCoreMobileSDK.Geocoordinate</code> <code>.Longitude[get], [set]</code>	Longitude in degrees
DateTime Timestamp [get, set]	DateTime <code>AuCoreMobileSDK.Geocoordinate</code> <code>.Timestamp[get], [set]</code>	Time at which the location was determined

Managing the Authentication Handler Interface

The `AuCoreMobileSDK.IAuthenticationHandler` interface reference must be implemented by the class that contains a code for the authentication events. This interface is called from **`AuCoreMobileSDK.Sdk`**.

Public Member Functions

- ♦ `void OnCheckPendingAuthenticationFinished (bool thereIsPendingAuthentication, IConfirmation confirmation)`

This method is called when the server responds for checking a pending authentication request.

- ♦ `void AuCoreMobileSDK.IAuthenticationHandler.OnCheckPendingAuthenticationFinished(bool thereIsPendingAuthentication, IConfirmation confirmation)`

Parameter	Description
thereIsPendingAuthentication	true if there is a pending authentication on the server.
confirmation	Interface to communicate with Confirmation object. You must call <code>AuCoreMobileSDK.IConfirmation.SendConfirmation</code> method if <code>thereIsPendingAuthentication</code> <code>thereIsPendingAuthentication == true</code> .

- ♦ `void OnAuthenticatorDeleted (string authenticatorId)`

This method is called when an authenticator is deleted from the server and this event is detected by the Confirmation object. This method is called after the local authenticator has been deleted.

- ♦ `void AuCoreMobileSDK.IAuthenticationHandler.OnAuthenticatorDeleted (string authenticatorId)`

Parameters:

Parameter	Description
authenticatorId	The ID of deleted authenticator.

- ♦ `void OnSendConfirmationFinished (ConfirmResult result, IConfirmation confirmation)`

This method is called after the server returns the result of confirmation which had been performed by calling `AuCoreMobileSDK.IConfirmation.SendConfirmation`.

- ♦ `void AuCoreMobileSDK.IAuthenticationHandler.OnSendConfirmationFinished (ConfirmResult result, IConfirmation confirmation)`

Parameters:

Parameter	Description
result	<code>AuCoreMobileSDK.JsonObjects.ConfirmResult</code>
confirmation	<code>AuCoreMobileSDK.IConfirmation</code> . You don't need to call anything from <code>confirmation</code> <code>confirmation</code>

- ♦ `void OnSendConfirmationError (string description, IConfirmation confirmation)`

This method is called when an error occurs during the processing of `AuCoreMobileSDK.IConfirmation.SendConfirmation`.

- ♦ `void AuCoreMobileSDK.IAuthenticationHandler.OnSendConfirmationError (string description, IConfirmation confirmation)`

Parameters:

Parameter	Description
description	Error description
confirmation	AuCoreMobileSDK.IConfirmation You don't need to call anything from confirmation confirmation

Example

```
using System;
using AuCoreMobileSDK;
using AuCoreMobileSDK.JsonObjects;

namespace TestConsoleApp
{
    class AuthenticationHandler : IAuthenticationHandler
    {
        public void OnCheckPendingAuthenticationFinished(bool
thereIsPendingAuthentication, IConfirmation confirmation)
        {
            if (thereIsPendingAuthentication)
            {
                confirmation.SendConfirmation(true, null);
            }
        }

        public void OnAuthenticatorDeleted(string authenticatorId)
        {
            Console.WriteLine(@"Authenticator ID deleted: {0}", authenticatorId);
        }

        public void OnSendConfirmationFinished(ConfirmResult result, IConfirmation
confirmation)
        {
            Console.WriteLine(@"Result: {0}", Enum.GetName(typeof(ConfirmResult),
result));
        }

        public void OnSendConfirmationError(string description, IConfirmation
confirmation)
        {
            Console.WriteLine(@"Error: {0}", description);
        }
    }
}
```

Creating a Data Representation and Converting Bytes and Hex Strings

The AuCoreMobileSDK.HexData Class Reference is used for the universal data representation and for conversion between bytes and hex.string. This class inherits IEquatable<HexData> and ICloneable.

Public Member Functions

- ◆ HexData (string hex)

AuCoreMobileSDK.HexData.HexData (string hex)

Creates HexData object from hex.string. It internally uses Convert.ToByte and can throw exceptions.

Parameters

Parameter	Description
hex	hex.string

Exceptions

Exception	Description
System.FormatException	hex hex contains a character that is not a valid digit in the 16-base. The exception message indicates that there are no digits to convert if the first character in hex hex is invalid; otherwise, the message indicates that hex hex contains invalid trailing characters.

- ◆ HexData (byte[] data)

AuCoreMobileSDK.HexData.HexData (byte [] data)

Creates HexData object from a copy of source bytes.

Parameters

Parameter	Description
data	Source bytes

- ◆ override string ToString ()

override string AuCoreMobileSDK.HexData.ToString ()

Returns the HEX representation in the hex.string format.

- ◆ bool Equals (HexData other)

bool AuCoreMobileSDK.HexData.Equals (HexData other)

Determines whether the specified object is equal to the current object.

Parameters

Parameter	Description
other	The object to compare with the current object.

Returns true if the specified object is equal to the current object.

- ◆ override bool Equals (object obj)

override bool AuCoreMobileSDK.HexData.Equals (object obj)

Parameters

Parameter	Description
obj	The object to compare with the current object.

Returns true if the specified object is equal to the current object.

- ♦ `override int GetHashCode ()`
`override int AuCoreMobileSDK.HexData.GetHashCode ()`
Serves as the default hash function.
Returns a hash code for the current object.
- ♦ `HexData CloneData ()`
`HexData AuCoreMobileSDK.HexData.CloneData ()`
Returns a cloned copy of this object.
- ♦ `object Clone ()`
`object AuCoreMobileSDK.HexData.Clone ()`
Returns a cloned copy of this object.

Static Public Member Functions

- ♦ `static HexData operator+ (HexData a, HexData b)`
`static HexData AuCoreMobileSDK.HexData.operator+ (HexData a, HexData b)[static]`
Creates a new HexData object which is a result of concatenation two source objects.

Parameters

Parameter	Description
a	The right object
b	The left object

Returns a new object with a copy of the concatenated data

- ♦ `static bool operator!= (HexData obj1, HexData obj2)`
`static bool AuCoreMobileSDK.HexData.operator!= (HexData obj1, HexData obj2)[static]`

Parameters

Parameters	Description
obj1	First operand (left)
obj2	Second operand (right)

Returns true if `obj1 != obj2`

- ♦ `static bool operator== (HexData obj1, HexData obj2)`

```
static bool AuCoreMobileSDK.HexData.operator== (HexData obj1, HexData
obj2)[static]
```

Parameters

Parameters	Description
obj1	First operand (left)
obj2	Second operand (right)

Returns true if obj1 == obj2

Public Attributes

```
bool IsEmpty => _data == null || _data.Length == 0
```

```
bool AuCoreMobileSDK.HexData.IsEmpty => _data == null || _data.Length == 0
```

Checks if the object is empty.

Properties

```
byte [] Data [get]
```

```
byte [] AuCoreMobileSDK.HexData.Data[get]
```

Gets a copy of the stored bytes.

Example

```
using System;
using AuCoreMobileSDK;

namespace TestConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var arr = new byte[] {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15};
            var hexData = new HexData(arr);
            var s = hexData.ToString();
            Console.WriteLine(s);
        }
    }
}
```

Output:
000102030405060708090A0B0C0D0E0F

Handling Enrollment Result

The AuCoreMobileSDK.IEnrollResultHandler Interface Reference is used to handle Enrollment results.

Public Member Functions

- ♦ void OnEnrollError (string description, string detailDescription)

```
void AuCoreMobileSDK.IEnrollResultHandler.OnEnrollError (string description,
string detailDescription)
```

Error occurs during enrollment.

Parameters

Parameter	Description
description	Error text (brief)
detailDescription	Error text (details)

- ♦ void OnEnrollCompleted (string authenticatorId)
void AuCoreMobileSDK.IEnrollResultHandler.OnEnrollCompleted (string authenticatorId)
Enrollment has been completed. Authenticator is saved.

Parameters

Parameter	Description
authenticatorId	Authenticator ID

Example

```
using System;
using AuCoreMobileSDK;

/// <summary>
/// Demo code of enroll result handler interface implementation
/// </summary>
internal class EnrollResultHandler : IEnrollResultHandler
{
    public void OnEnrollError(string description, string detailDescription)
    {
        // something went wrong during enrollment
        // print message
        Console.WriteLine(@"Error: {0}", description);
        Console.WriteLine(@"Details: {0}", detailDescription);
    }

    public void OnEnrollCompleted(string templateId)
    {
        // a new authenticator has been enrolled
        // print message
        Console.WriteLine(@"Enrolled: {0}", templateId);
        // save the storage to file
        SdkWrapper.Instance.SaveStorageToFile();
    }
}
```


5 Managing Endpoints

Endpoints are devices where the Advanced Authentication server authenticates. An endpoint can be a Windows workstation for Windows Client endpoint, or Advanced Authentication Access Manager appliance for the NAM endpoint and so on.

The endpoints are automatically added when you install a plug-in such as NAM or install Windows Client. The RADIUS endpoint, an OSP endpoint that is used for WebAuth authentication, and Endpoint41 and Endpoint42 are the predefined endpoints.

NOTE: Endpoint41 and Endpoint42 are created for the integration with legacy NAM and NCA plug-ins, which are used in NAM 4.2 and earlier versions with Advanced Authentication 5.1.

The NAM and NCA plug-ins work with the hard coded endpoint ID and secret. In Advanced Authentication 5.2 and later, you must register the endpoints. This breaks the backward compatibility with old plug-ins. These two legacy endpoints allow to keep the old plug-ins working.

To configure an endpoint for Advanced Authentication, perform the following steps:

- 1 In the **Endpoints** section, click **Edit** against the endpoint you want to edit.
- 2 You can rename the endpoint, change its description or endpoint type.
- 3 Set **Is enabled** to **ON** to enable the endpoint.
- 4 Set **Is trusted** to **ON** if the endpoint is trusted. In some integrations such as Migration Tool, Password Filter, NAM, and NCA you must enable the **Is trusted** option for their endpoints.
- 5 Specify an **Endpoint Owner** if you have configured a specific chain to be used by the Endpoint owner only. This is a user account that must be able to use a different **chain** than the other users for authentication.

The Endpoint Owner feature is supported for Windows Client, Mac OS Client, and Linux PAM Client only.

NOTE: Additional information such as **Operating System**, **Software** version, **Last session** time and **Device** information are displayed. Also in **Advanced properties**, RAM information is displayed.

Advanced Authentication Windows Client 5.6 or newer, Advanced Authentication Linux PAM Client 6.0 or newer, Advanced Authentication Mac OS X Client 6.0 or newer must be installed on the endpoint.

- 6 Click **Save**.

You can create an endpoint manually. This endpoint can be used for the third-party applications that do not create endpoints.

To create an endpoint manually, perform the following steps:

- 1 In the **Endpoints** section, click **Add**.
- 2 On the **Add endpoint** page, specify a **Name** of the endpoint and its **Description**.
- 3 Set the **Type** to **Other**.
- 4 Set **Is enabled** to **ON**.

- 5 Set **Is trusted** to **ON** if the endpoint is trusted.
- 6 Leave **Endpoint Owner** blank.
- 7 Click **Save**. The **New Endpoint secret** window is displayed.
- 8 Take down the values specified in **Endpoint ID** and **Endpoint Secret** and place them in a secure place in your application.

NOTE: You will not be able to get the **Endpoint ID** and **Endpoint Secret** later on the appliance.

- 9 Click **OK**.

NOTE: **Tenancy settings** are not supported for Endpoints.

IMPORTANT: You must ensure not to remove an endpoint that has at least one component running on it such as Windows Client, Logon Filter, RD Gateway plug-in, or ADFS plug-in. Endpoint is removed automatically when you uninstall Windows Client. However you must remove the endpoint manually when you uninstall Logon Filter, RD Gateway plug-in or ADFS plug-in.

If you remove an endpoint accidentally, ensure to remove the records with prefix **endpoint*** from the %ProgramData%\NetIQ\Windows Client\config.properties file and re-start the machine. This recreates the endpoint.
