



NetIQ Advanced Authentication Framework - Software Development Kit (SDK)

Administrator's Guide

Version 5.1.0

Table of Contents

	1
Table of Contents	2
Introduction	6
Strong Authentication	6
Strong Authentication Advantage	6
NetIQ SDK Overview	8
How Does It Work?	8
Integrates with NetIQ Advanced Authentication Framework Edition	8
Getting Started	9
Environment	9
Requirements	9
What is Included	9
Integration	9
Runtime Requirements	10
Using in Different Bit Depth	10
PasswordStore	12
API Reference	13
Terms	13
Class List	14
ISSOLogonClient	15
Function Logon	15
Parameters	15
Return value	15
Examples	15
ISSOSimpleLogonClient	17
Function Logon	17
Parameters	17
Return value	17
Examples	17
ILogonManager	19
Function Connect	19
Parameters	19
Examples	19
Function User	20
Return Value	20
Examples	20
Function UserEx	21
Parameters	21
Return Value	21
Examples	21
Function ProductVersion	22
Returns	22
Examples	22

Function IsBiometricUser	23
Parameters	23
Return Value	23
Examples	23
Function EnableBiometricUser	25
Parameters	25
Examples	25
Function DisableBiometricUser	27
Parameters	27
Examples	27
IManagedUser	28
Function SubsystemsConnector	28
Return Value	28
Function ImportBIR	28
Parameters	28
Function UserSettings	28
Parameters	29
Return Value	29
typedefUserProperties	29
Examples	29
ILogonedUser	30
Properties	30
Examples	30
ILogonedUser2	31
Function GetSelectedLogonMode	31
Return Value	31
Function UnitID	31
Returns	31
Function GetLogonAuthenticator	31
Return Value	31
IUserSubsystemData	33
Function EnumRecords	33
Return Value	33
Function Create	34
Parameters	34
Return Value	34
Function Get	34
Parameters	34
Return Value	34
Function Remove	35
Parameter	35
Function Commit	36
Examples	36
ISubsystemsConnector	38
Function CreateSubsystemData	38
Parameters	38

Return Value	38
Examples	38
Function GetSubsystemData	40
Parameters	40
Return Value	40
Examples	40
Function RemoveSubsystemData	41
Parameters	41
Examples	41
Function IsSubsystemUser	42
Parameters	42
Return Value	42
Examples	42
IUserCredentials	44
Function UpdateCredentials	44
Parameters	44
Properties	44
Examples	44
IUpdateCredentialSettings	46
Properties	46
typedefUserCredentialsProperties	46
IRecord	47
Function EnumValues	47
Parameter	47
Function WriteField	47
Parameter	47
Function ReadField	48
Parameter	48
Return Value	48
Function ClearField	48
Parameter	48
Function ClearField	48
Parameter	49
Properties	49
Examples	49
ILogonSettings	52
Properties	52
typedefLogonProperties	52
Examples	54
IStringEnum	55
Properties	55
IDeviceMonitor	56
Identifiers of Authentication Service Provider	56
Function IsUnitInserted	56
Parameters	56
Function AdviseAll	56

Parameter	57
Function AdviseMode	57
Parameters	57
Function AdviseUnit	57
Parameters	57
Function Unadvise	58
Parameter	58
IDeviceMonitorEvents	59
Function OnUnitInserted	59
Parameters	59
Function OnUnitRemoved	59
Parameters	59
Examples	59
Error Codes Description	62
RPC Server Errors	62
SrvWrapperErrors	68
Password Filter Errors	68
Password Manager Errors	69
EventLogErrors	69
BioAPIErrors	69
Authenticore Server Errors	70
Authentication Providers Errors	72
Cryptography Errors	72
Manager Errors	72
Plugins Errors	73
Licensing Errors	73
Backup Provider Errors	74
Administration Tools Errors	74
GINA Errors	75
Data Errors	75
Troubleshooting	77
Error "The user was authenticated by password"	77
Index	78

Introduction

Strong Authentication

Compliance is important in a world where organizations have to adhere to increasingly complex rules and regulations. And information security is already vitally important for every business in our connected global society. No real solution for compliance or information security is possible without proper authentication of users. But authentication by user name and passwords is not reliable anymore. There are lots of stronger authentication methods on the market, but regrettably there is not the best solution in respect to cost, reliability and user convenience in every situation.

Strong Authentication Advantage

Many organizations are discovering that traditional password-based authentication systems frustrate users and administrators, while remaining costly to the organization. A recent study cites that password-related calls account for more than 30% of all Helpdesk calls. Unlike passwords, strong authentication systems do not require the hassle of memorizing a series of letters, numbers, and symbols, nor do they require periodic changing.

NetIQ connects to all leading authentication methods. Although there still is a lot of good old username password combination in use for authentication in the virtual world, there is general consensus that passwords are too vulnerable and not secure anymore. New, stronger, authentication methods have been launched into the market. Authentication methods like contact cards, contactless cards, biometric technologies, one time password generators, hardware/software tokens and many more have all gained traction on the market. But regrettably none of these solutions is the best fit for every authentication requirement within a modern company. There is an urgent need for a more generic solution that enables companies to select a bundle of methods simultaneously, that serves specific and generic needs and is future proof. Adding new methods to NetIQ is a simple standardized and fast process.

NetIQ SDK Overview

NetIQ BV provides a software development kit (SDK) for third-party developers to integrate with NetIQ Advanced Authentication Framework. The NetIQ SDK allows you to add strong authentication functionality to your applications. With the NetIQ SDK it is possible to use any of the supported strong authentication methods to authenticate to your application.

How Does It Work?

Your application calls the SDK COM component, which is located in the *LogonSDK.tlb* file. The Logon method passed the user credentials into NetIQ, which performs authentication. Logon will by default ask the user to choose the authentication method (i.e. fingerprint, contactless card, smartcard, etc.) and return the *Authenticated User*.

NetIQ SDK components only interact with the client portion of NetIQ Advanced Authentication Framework, which will interact with the server when appropriate.

Benefits

- Replace password verification with any supported strong authentication method. The NetIQ SDK provides a flexible solution that can support any authentication method on the market like fingerprint, iris, contactless card, smartcard, etc.
- Operates in conjunction with NetIQ Advanced Authenticated Framework, an integrated authentication solution for Microsoft Active Directory.
- Simple development; add NetIQ SDK API calls to your application and you're done. NetIQ SDK, in conjunction with NetIQ Advanced Authentication Framework, will manage hardware and software interoperability.

Integrates with NetIQ Advanced Authentication Framework Edition

NetIQ Advanced Authentication Framework Edition is a strong authentication security solution that enables users to log on to their workstation and Windows domains. Designed and tested for enterprise-level deployments, NetIQ tightly integrates with Active Directory to allow administrators to secure network and workstation access. See NetIQ Advanced Authentication Framework Administrator Guide for detailed configuration information, or contact an NetIQ sales representative to learn more about this product.

Getting Started

The following chapters will provide the details on how to get started using the NetIQ SDK.

Environment

Requirements

You will need a development PC, which must have Microsoft Visual C++, Microsoft Visual Basic or another COM client development platform installed. For testing purposes, you may also wish to install the NetIQ Advanced Authentication Framework Client or RTE software and the authenticators you wish to use.

What is Included

The NetIQ SDK consists of one API including one COM interface: "LogonSDK". The interface is deployed in a single Type Library component: LogonSDK.tlb. You need import this file into your development project.

The distributive of NetIQ SDK includes next files:

bin

- Logonsdk.tlb – NetIQ Type Library component.

doc

- NAAF – SDK Administrator Guide.pdf – SDK documentation in pdf format.

samples

- VBS files which contains VBScript examples.

Integration

Once you have deployed the SDK COM component, you may start working with them in any COM client development environment, such as C++ ATL or Visual Basic. The general integration process is as follows:

1. Import type library for use in either C++ or VB.
2. Create an instance of the COM component: "LogonSDK".
3. Access the desired COM interface: *ISSOLogonClient* for AD authentication and *ISSOSimpleLogonClient* for third-party authentication.
4. Call the *Logon* interface method to perform the authentication. The SDK will then invoke underlying product functionality to perform a strong authentication.
5. Free the SDK COM component, if necessary.

Runtime Requirements

The next requirements are for host machine to run ready SDK integration:

1. Microsoft® Windows XP Professional x86 Service Pack 3, Microsoft® Windows XP Professional x64 Service Pack 2, Microsoft® Windows Vista SP2 x86/x64, Microsoft® Windows 7 SP1 x86/x64, Microsoft® Windows Server 2003 SP2 x86/x64, Microsoft® Windows Server 2003 R2 SP2 x86/x64, Microsoft® Windows Server 2008 SP2 x86/x64, Microsoft® Windows Server 2008 R2 SP1, Microsoft® Windows 8 x86/x64.
2. NetIQ Advanced Authentication Framework - Client or RTE software.
3. NetIQ Advanced Authentication Framework supported device and appropriate authentication service provider.

Using in Different Bit Depth

If the application that uses NetIQ SDK has the same bit depth as installed NetIQ, you shouldn't care about bit differences and use the constructions like this directly:

```
Set dlgSets = CreateObject("LogonSDK.LogonSettings")
```

If the bit depth is different, you should use extended PROGID like "LogonSDK.HostLogonSettings". In this case, NetIQ SDK will be executed in separated process (dllhost).

The next example (SDK_18.vbs) provides authentication for 32-bit third-party application which executing onto 64-bit operating system, and 64-bit NetIQ components are installed:

```
1. Const SubsName= "PasswordStore"  
    ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.HostSSOSimpleLogonClient")  
    ' Authentication in third-party application object  
3. Set dlgSets = CreateObject("LogonSDK.HostLogonSettings")  
    ' Authentication settings object  
4. Set Creds = logonCl.Logon(dlgSets, SubsName, "CredsSetName")  
    ' Request of authentication, than new credential registration request
```

PasswordStore

PasswordStore is a name of default subsystem that contains the data, which is necessary for NetIQ. The data are storing in Active Directory and contains records. Every record contains three fields: username, password, userdata.

Some facts about PasswordStore subsystem:

1. Only a user can create and delete records in this subsystem.
2. Only a user can write, read and clean fields of record.
3. The data of field "password" clears after resetting password.
4. The data of field "password" cannot be returned after logon by password.
5. Subsystem is not licensed.
6. The field "password" in PasswordStore subsystem is not available if user logged on by domain password.

Also you can see [example](#) of using PasswordStore subsystem.

API Reference

This chapter will describe the classes and functions included in the NetIQ SDK.

Terms

Extended mode – is a mode that is using for Active Directory authentication.

Record – is an object of Subsystem.

Simple mode – is a mode that is using for authentication in third-party applications and solutions.

Subsystem – is an object of Active Directory that contains the data such as usernames, passwords, records, custom data and etc. These data are used by NetIQ Advanced Authentication Framework. NetIQ SDK examples using default subsystem named "PasswordStore".

Class List

Here are interfaces with brief description:

<u>IDeviceMonitor</u>	Enables the device monitoring features
<u>ILogonManager</u>	Enables administrators to manage users
<u>ILogonedUser</u>	Exposes properties for currently logged in user management
<u>ILogonedUser2</u>	Exposes properties for currently logged in user management
<u>ILogonSettings</u>	Exposes the default property to manage the logon settings through automation
<u>IManagedUser</u>	Enables to manage user settings
<u>IRecord</u>	Represents a single Record object for a given subsystem
<u>ISSOLogonClient</u>	The main entry point when using the SDK in extended mode
<u>ISSOSimpleLogonClient</u>	The main entry point when using the SDK in simple mode
<u>IStringEnum</u>	Enables to enumerate strings in read only mode
<u>ISubsystemsConnector</u>	Exposes methods for user's custom data management
<u>IUpdateCredentialSettings</u>	Exposes the default property to manage the update credentials settings through automation
<u>IUserCredentials</u>	Exposes properties and methods to manage user credentials for custom systems
<u>IUserSubsystemData</u>	Represents a collection of records for a given subsystem

ISSOLogonClient

The main entry point when use the SDK in extended mode. The extended mode is using for Active Directory authentication.

Function Logon

The function provides Active Directory authentication.

Parameters

ILogonSettings

Type: *pLogonSettings

Authentication settings object.

subsystemName

Type: BSTR

Subsystem name.

Return value

ILogonedUser

Type: **ppLogonedUser

Authenticated user object.

Examples

The next example (SDK_01.vbs) provides request of authentication, than authentication in Active Directory.

```
1. Const SubsName= "PasswordStore"  
    ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.SSOLogonClient")  
    ' Authentication object  
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
    ' Authentication settings object  
4. Set lgnUser = logonCl.Logon(dlgSets, SubsName)  
    ' Request of authentication
```

ISSOSimpleLogonClient

The main entry point when use the SDK in simple mode. The simple mode is using for authentication in third-party applications and solutions.

Function Logon

The function provides authentication in third-party applications and solutions.

Parameters

ILogonSettings

Type: *pLogonSettings

Authentication settings object.

subsystemName

Type: BSTR

Subsystem name.

credentialsSetName

Type: BSTR

Name of credential's set.

Return value

ILogonedUser

Type: **ppLogonedUser

Authenticated user object.

Examples

The next example (SDK_02.vbs) provides authentication in third-party application.

```
1. Const SubsName= "PasswordStore"  
    ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.SSOSimpleLogonClient")  
    ' Authentication in third-party application object  
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
    ' Authentication settings object  
4. Set Creds = logonCl.Logon(dlgSets, SubsName, "CredsSetName")  
    ' Request of authentication, than new credential registration request
```

ILogonManager

The interface enables administrators to manage users.

Function Connect

The function provides connectivity to specified domain.

Parameters

domain

Type: BSTR

Domain name (DNS).

subsystem

Type: BSTR

Subsystem name.

Examples

The next example (SDK_03.vbs) provides connectivity to specified domain.

```
1. Set lgnManager = CreateObject("LogonSDK.LogonManager")      ' Logon
   manager object
2. lgnManager.Connect "authasas.local","PasswordStore"         ' Conne
   ct to specified domain
3.
```

Function User

The function allows administrators getting current user for managing.

Return Value

*IManagedUser***

Type: ppVal

Managed user object.

Examples

The next example (SDK_04.vbs) provides getting current user for managing.

```
1. Set lgnClient = CreateObject ("LogonSDK.SSOLogonClient")      ' Authentication object
2. Set dlgSets = CreateObject("LogonSDK.LogonSettings")          ' Authentication settings object
3. Set lgnManager = CreateObject("LogonSDK.LogonManager")        ' Logon manager object
4. Set lgnUser = lgnClient.Logon(dlgSets, SubsName)              ' Request of authentication
5.
6. lgnManager.Connect "authasas.local", "PasswordStore"          ' Connect to specified domain
7. Set ManagUser = lgnManager.User                               ' Get logged in user for managing
```

Function UserEx

The function allows administrators getting specified user for managing.

Parameters

UserName

Type: BSTR

User name (DNS).

Return Value

*IManagedUser***

Type: ppVal

Managed user.

Examples

The next example (SDK_05.vbs) provides getting specified user administrator@authasas.com for managing.

```
1. Set lgnClient = CreateObject ("LogonSDK.SSOLogonClient")      ' Authe
   ntication object
2. Set dlgSets = CreateObject("LogonSDK.LogonSettings")          ' Authe
   ntication settings object
3. Set lgnManager = CreateObject("LogonSDK.LogonManager")        ' Logon
   manager object
4. Set lgnUser = lgnClient.Logon(dlgSets, SubsName)              ' Reque
   st of authentication
5.
6. lgnManager.Connect "authasas.local", "PasswordStore"         ' Conne
   ct to specified domain
7. Set ManagUser = lgnManager.UserEx("CN=Administrator,CN=Users,dc=authasa
   s,dc=local") ' Get specified user for managing
```

Function ProductVersion

The function allows getting product version of NetIQ Advanced Authentication Framework.

Returns

String contains product version.

Examples

The next example (SDK_06.vbs) provides getting product version of NetIQ.

```
1. Set lgnManager = CreateObject("LogonSDK.LogonManager")  
   ' Logon manager object  
2. WScript.Echo "Product version: " & lgnManager.ProductVersion  
   ' Show product version
```

Function IsBiometricUser

The function allows checking whether the specified user is biometric user.

Parameters

UserName

Type: BSTR

User name (LDAP).

Return Value

Valid

If the specified user is biometric user a boolean value of VARIANT_TRUE is returned, else VARIANT_FALSE is returned.

Examples

The next example (SDK_07.vbs) allows checking whether the authenticated user is biometric.

```

1. Const TestUser="LDAP://CN=test2,CN=Users,DC=authasas,DC=local" ' Test
   user (LDAP)
2.
3. Set lgnClient = CreateObject ("LogonSDK.SSOLogonClient") ' Authe
   ntication object
4. Set dlgSets = CreateObject("LogonSDK.LogonSettings") ' Authe
   ntication settings object
5. Set lgnManager = CreateObject("LogonSDK.LogonManager") ' Logon
   manager object
6. Set lgnUser = lgnClient.Logon(dlgSets, SubsName) ' Reque
   st of authentication
7.
8. lgnManager.Connect "authasas.local","PasswordStore" ' Conne
   ct to specified domain
9. If lgnManager.IsBiometricUser(TestUser) = TRUE Then
   ' If authenticated user is a biometric user
10.     WScript.Echo TestUser & " is biometric user"
   ' show message
11. ElseIf lgnManager.IsBiometricUser(TestUser) = FALSE Then
   ' If authenticated user is not a biometric user
12.     WScript.Echo TestUser & " is NOT biometric user"
   ' show message
13. End If

```

Function EnableBiometricUser

The function provides enabling capability to use NetIQ Authentication Providers for specified user. Requires delegation rights "NetIQ Advanced Authentication Framework User/Computer settings management" for logged in user.

Parameters

IUserSettings

Type: BSTR

User name (LDAP).

*IUserSettings**

Type: pVal

User settings object.

Examples

The following example (SDK_08.vbs) demonstrates capability to use NetIQ Authentication Providers for specified user.

```

1. Set lgnManager = CreateObject("LogonSDK.LogonManager")           ' Logon
   manager object
2. Set usrSets = CreateObject("LogonSDK.UserSettings")             ' Authe
   ntication settings object
3.
4. usrSets(0) = "The user for SDK test"                             ' Notes for the use
   r
5. usrSets(1) = true                                                ' Allows to enroll new authenti
   cators
6. usrSets(2) = true                                                ' Allows to delete enrolled aut
   henticators
7. usrSets(3) = true                                                ' Allows to change existing aut
   henticators
8. usrSets(4) = true                                                ' Allows to generate random pas
   sword
9. usrSets(5) = true                                                ' Allows the password logon
10.     usrSets(6) = 10                                             ' Maximum number of authe
   nticators
11.
12.     lgnManager.Connect "authasas.local","PasswordStore"         '
   Connect to specified domain
13.     lgnManager.EnableBiometricUser TestUser,usrSets             ' Enable
   capability to use Authasas Authentication Providers for specified user
14.     Set ManagUser = lgnManager.UserEx(TestUser)                 ' Get spe
   cified user for managing

```

Function DisableBiometricUser

The function provides disabling capability to use NetIQ Authentication Providers for specified user. Requires delegation rights "NetIQ Advanced Authentication Framework User/Computer settings management" for logged in user.

Parameters

IUserSettings

Type: BSTR

User name (LDAP).

Examples

The following example (SDK_09.vbs) provides disabling capability to use NetIQ Authentication Providers for specified biometric user.

Returns 0xc1020000 "The user was not found" when the specified user is not biometric user.

```
1. Const TestUser="CN=test2,CN=Users,DC=authasas,DC=local" ' Test user (L
   DAP)
2.
3. Set lgnManager = CreateObject("LogonSDK.LogonManager")      ' Logon
   manager object
4. Set usrSets = CreateObject("LogonSDK.UserSettings")          ' Authe
   ntication settings object
5. lgnManager.Connect "authasas.local","PasswordStore"         ' Conne
   ct to specified domain
6. lgnManager.DisableBiometricUser TestUser                    ' Disable capabilit
   y to use Authasas Authentication Providers for specified user
```

IManagedUser

The interface enables to manage user settings.

Function SubsystemsConnector

The function allows connection to subsystems.

Return Value

*ISubsystemsConnector***
Type: ppVal

Function ImportBIR

The function provides wrapping the authenticator raw data in the BioAPI headers and biowrapper.

Parameters

bsp_id
Type: BSTR
Biometric Service Provider identifier.

comment
Type: BSTR
Comment.

bir
Type: SAFEARRAY (unsigned char)
Raw data obtained from authentication device.

Function UserSettings

The function provides getting or setting of user settings.

Parameters

*IUserSettings**

Type: ppVal

User settings object.

Return Value

*IUserSettings***

Type: ppVal

User settings object.

typedefUserProperties

The list of available properties for NetIQ users.

- Description = 0

Specifies the optional notes.

- AllowAddAuthenticators = 1

Allows to enroll new authenticators.

- AllowDeleteAuthenticators = 2

Allows to delete enrolled authenticators.

- AllowModifyAuthenticators = 3

Allows to change existing authenticators.

- GenerateRandomPassword = 4

Allows to generate random password.

- AllowPasswordLogon = 5

Allows the password logon.

- MaximumAuthenticatorsNumber = 6

Specifies the maximum number of authenticators.

Examples

Please check the example of using these properties ([SDK_08.vbs](#)).

ILogonedUser

Exposes properties for currently logged in user management.

Properties

Name

Type: BSTR

Active Directory account username of the currently logged in user.

Password

Type: BSTR

Active Directory account password of the currently logged in user.

Examples

The next example (SDK_10.vbs) gets login and password of authenticated Active Directory user.

```
1. Const SubsName= "PasswordStore"  
   ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.SSOLogonClient")  
   ' Authentication object  
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
   ' Authentication settings object  
4. Set lgnUser = logonCl.Logon(dlgSets, SubsName)  
   ' Request of authentication  
5. WScript.Echo "Username: " & lgnUser.Name & vbCrLf & "Password: " & lgnU  
   ser.Password      ' Show username and password of authenticated user
```

ILoggedUser2

Exposes properties for currently logged in user management.

Function GetSelectedLogonMode

The function allows getting selected logon mode.

Return Value

*char**

Type: pPasswordSelected

The flag determines whether the user is authenticated by password or by biometry.

*BioUUID**

Type: pSelectedBSP

If the user is authenticated by biometry, this parameter contains identifier of logon method.

Function UnitID

The function allows getting Biometric Service Provider authenticator ID.

Returns

String contains authenticator ID.

Function GetLogonAuthenticator

The function allows getting logon authenticator.

Return Value

*VARIANT**

Type: pRet

Wrapped authenticator template.

IUserSubsystemData

Represents a collection of records for a given subsystem.

Function EnumRecords

The function provides enumerating records for a given subsystem.

Return Value

IStringsEnum

Type: Object

Number of records.

Function Create

The function provides creating new record for a current subsystem.

Parameters

RecordId
Type: BSTR
The name of the record.

Return Value

IRecord
Type: Object
Record object.

Function Get

The function requests an existing record for a current subsystem.

Parameters

RecordId
Type: BSTR
The name of the record.

Return Value

IRecord
Type: Object
Record object.

Function Remove

The function allows removing record.

Parameter

RecordId

Type: BSTR

The name of the record.

Function Commit

The function provides committing changes.

Examples

The next example (SDK_11.vbs) creates a record in default subsystem "PasswordStore", gets existing record, enumerates existing records, commits changes and removes record.

```
1. Const SubsName= "PasswordStore"  
   ' Default subsystem name  
2. Const ExampleRecName="Record3"  
   ' Sample record name  
3. Set logonCl = CreateObject ("LogonSDK.SSOLogonClient")  
   ' Authentication object  
4. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
   ' Authentication settings object  
5. Set lgnUser = logonCl.Logon(dlgSets, SubsName)  
   ' Request of authentication  
6. Set sbsConnector = lgnUser.SubsystemsConnector  
   ' Subsystem connector object  
7. Set NewSubsystemData = sbsConnector.CreateSubsystemData(SubsName)  
   ' Create subsystem data  
8.  
9. Set NewRecord = NewSubsystemData.Create(ExampleRecName)  
   ' Create record "Record3"  
10.   WScript.Echo "Record " & ExampleRecName & " was successfully crea  
    ted"           ' Show log message  
11.  
12.   Set Record = NewSubsystemData.Get(ExampleRecName)  
   ' Get record "Record3"
```

```

13.      WScript.Echo "Record " & ExampleRecName & " was successfully got"
        ' Show log message
14.
15.      Set EnRecords = NewSubsystemData.EnumRecords()
        ' Enumerate records in subsystem
16.      For Each i In EnRecords
        ' For each record
17.          res = res & i & "; "
        ' Form result string
18.      Next
19.      WScript.Echo "Enumerating records result: " & res
        ' Show log message
20.
21.      NewSubsystemData.Commit()
        ' Commit changes
22.      WScript.Echo "All changes were committed"
        ' Show a log message
23.
24.      NewSubsystemData.Remove(ExampleRecName)
        ' Remove record "Record3"
25.      WScript.Echo "Record " & ExampleRecName & " was successfully removed"
        ' Show log message

```

ISubsystemsConnector

Exposes methods for user's custom data management.

Function CreateSubsystemData

The function provides enabling the user to use the given subsystem and returns the User-SubsystemData object for the data management.

Parameters

SubsystemName

Type: BSTR

Name of the subsystem.

Return Value

IUserSubsystemData

Type: object

Object for the data management.

Examples

The following example (SDK_12.vbs) allows creating subsystem data.

```
1. Const SubsName= "PasswordStore"  
    ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.SSOLogonClient")  
    ' Authentication object  
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
    ' Authentication settings object  
4. Set lgnUser = logonCl.Logon(dlgSets, SubsName)  
    ' Request of authentication  
5. Set sbsConnector = lgnUser.SubsystemsConnector  
    ' Subsystem connector object  
6. Set NewSubsystemData = sbsConnector.CreateSubsystemData(SubsName)  
    ' Create subsystem data
```

Function GetSubsystemData

The function provides receiving the requested user data from server and returns the User-SubsystemData object for the data management.

Parameters

SubsystemName

Type: BSTR

Name of the subsystem.

Return Value

IUserSubsystemData

Type: object

Object for the data management.

Examples

The following example (SDK_13.vbs) allows getting subsystem data.

```
1. Const SubsName= "PasswordStore"  
   ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.SSOLogonClient")  
   ' Authentication object  
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
   ' Authentication settings object  
4. Set lgnUser = logonCl.Logon(dlgSets, SubsName)  
   ' Request of authentication  
5. Set sbsConnector = lgnUser.SubsystemsConnector  
   ' Subsystem connector object  
6. Set NewSubsystemData = sbsConnector.CreateSubsystemData(SubsName)  
   ' Create subsystem data  
7. Set SubsystemData = sbsConnector.GetSubsystemData(SubsName)  
   ' Get subsystem data
```

Function RemoveSubsystemData

The function provides removing all user data from the given subsystem from server.

Parameters

SubsystemName

Type: BSTR

Name of the subsystem.

Examples

```
1. Const SubsName= "PasswordStore"  
   ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.SSOLogonClient")  
   ' Authentication object  
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
   ' Authentication settings object  
4. Set lgnUser = logonCl.Logon(dlgSets, SubsName)  
   ' Request of authentication  
5. Set sbsConnector = lgnUser.SubsystemsConnector  
   ' Subsystem connector object  
6. Set NewSubsystemData = sbsConnector.CreateSubsystemData(SubsName)  
   ' Create subsystem data  
7. sbsConnector.RemoveSubsystemData(SubsName)  
   ' Remove subsystem data
```

Function IsSubsystemUser

The function allows checking whether the logged in user is the user of given subsystem.

Parameters

SubsystemName

Type: BSTR

Name of the subsystem.

Return Value

Value

If the logged in user is the user of given subsystem a boolean value of VARIANT_TRUE is returned, else VARIANT_FALSE is returned.

Examples

The next example (SDK_15.vbs) allows checking whether the logged in user is the user of given subsystem.

```

1. Const SubsName= "PasswordStore"
   ' Default subsystem name
2. Set logonCl = CreateObject ("LogonSDK.SSOLogonClient")
   ' Authentication object
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")
   ' Authentication settings object
4. Set lgnUser = logonCl.Logon(dlgSets, SubsName)
   ' Request of authentication
5. Set sbsConnector = lgnUser.SubsystemsConnector
   ' Subsystem connector object
6. Set NewSubsystemData = sbsConnector.CreateSubsystemData(SubsName)
   ' Create subsystem data
7. If sbsConnector.IsSubsystemUser(SubsName) = TRUE Then
   ' If authenticated user is subsystem user
8. WScript.Echo lgnUser.Name & " is a " & Chr(34) & SubsName & Chr(34) &
   " subsystem user"          ' Show log message
9. ElseIf sbsConnector.IsSubsystemUser(SubsName) = FALSE Then
   ' If authenticated user is not subsystem user
10. WScript.Echo lgnUser.Name & " is NOT a " & Chr(34) & SubsName &
    Chr(34) & " subsystem user"          ' Show log message
11. End If

```

IUserCredentials

The interface exposes properties and methods to manage user credentials for third-party applications and systems. A UserCredentials object is created with ISSOSimpleClient::Logon.

Function UpdateCredentials

The function allows updating credentials.

Parameters

IUpdateCredentialsSettings

Type: *pSettings

Settings that will be updated.

Properties

LogonName

Type: BSTR *

User logon name in third-party application.

Password

Type: BSTR *

User password in third-party application.

CustomData

Type: VARIANT

Optional data stored along with logon name and password.

Examples

The next example (SDK_16.vbs) gets Logon name and Password which were used in the third-party application.

```
1. Const SubsName= "PasswordStore"  
    ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.SSOSimpleLogonClient")  
    ' Custom system authentication object  
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
    ' Authentication settings object  
4. Set Creds = logonCl.Logon(dlgSets, SubsName, "ApplicationName")  
    ' Request of authentication, than new credential registration request (if it has not been yet)  
5. WScript.Echo "Logon name: " & Creds.LogonName & vbCrLf & "Password: " &  
    Creds.Password ' Show credentials
```

IUpdateCredentialSettings

The interface exposes the default property to manage the update credentials settings through automation.

Properties

UpdateCredentialsProperties

Type: VARIANT

Updating credential properties.

typedefUserCredentialsProperties

The list defines the set of the available Update Credentials dialog options

- DisableUserName = 1
Disables the UserName field.
- MainWindowCaption = 2
Specifies the window caption.
- MainWindowTip = 3
Specifies the window tip.
- UserName = 4
Specifies the username.
- ShowCancel = 5
Allows to hide Cancel button.
- ForceLanguageId = 6
Allows to specify the UI language.
- Max_UpdateCredentials_Property_Id = 6

IRecord

Represents a single Record object for a given subsystem.

Function EnumValues

The function performs the Values name enumeration for the given Field in the Record.

Parameter

Field

Type: BSTR

Field name for enumerating.

IStringEnum

Type: Object

Number of Values.

Function WriteField

The function allows writing new data in to the specified Value of the specified Field.

Parameter

Field

Type: BSTR

Field name.

ValueName

Type: BSTR

Value name for writing data.

Data

Type: VARIANT

Data to be written.

Function ReadField

The function allows reading existing data from the specified Value of the specified Field.

Parameter

Field

Type: BSTR

Field name.

ValueName

Type: BSTR

Value name for reading data.

Return Value

Data

Type: VARIANT

Read data.

Function ClearField

The function allows clearing data in the all Values of the specified Field.

Parameter

Field

Type: BSTR

Field name.

Function ClearField

The function allows clearing data in the specified Value of the specified Field.

Parameter

Field

Type: BSTR

Field name.

ValueName

Type: BSTR

Value name for clearing data.

Properties

RecordId

Type: BSTR

Identifier of record.

Examples

The next example (SDK_17.vbs) writes data in the given Value of the given Field of the given Record, reads data from the given Value of the given Field of the given Record, clears data of the given Value of the given Field of the given Record, clears Field of the given Record.

```

1. Const SubsName= "PasswordStore"
   ' Default subsystem name
2. Const ExampleRecName="Record3"
   ' Sample record name
3. Const FieldName="UserName"
   ' Sample field name
4. Const ValueName="testing account"
   ' Sample value name
5.
6. Set logonCl = CreateObject ("LogonSDK.SSOLogonClient")
   ' Authentication object
7. Set dlgSets = CreateObject("LogonSDK.LogonSettings")
   ' Authentication settings object
8. Set lgnUser = logonCl.Logon(dlgSets, SubsName)
   ' Request of authentication
9. Set sbsConnector = lgnUser.SubsystemsConnector
   ' Subsystem connector object
10.   Set NewSubsystemData = sbsConnector.CreateSubsystemData(SubsName)
   ' Create subsystem data
11.
12.   Set Record = NewSubsystemData.Create(ExampleRecName)
   ' Create record "Record3"
13.
14.   Record.WriteField FieldName,ValueName,lgnUser.Name
   ' Write username of authenticated user in the value in sample fie
ld

```

```

15.      WScript.Echo "Successfully wrote " & Chr(34) & lgnUser.Name & Chr
      (34) & _
16.          " to value " & Chr(34) & ValueName & Chr(34) & _
17.          " in field " & Chr(34) & FieldName & Chr(34) & _
18.          " of record " & Chr(34) & Record.RecordId & Chr(34)
      ' Show log message
19.
20.      If Record.ReadField(FieldName,ValueName)<>"" Then
      ' If sample field is not empty
21.          WScript.Echo "Successfully read " & Chr(34) & Record.ReadFiel
      d(FieldName,ValueName) & Chr(34) & _
22.          " from value " & Chr(34) & ValueName & Chr(34) & _
23.          " in field " & Chr(34) & FieldName & Chr(34) & _
24.          " of record " & Chr(34) & Record.RecordId & Chr(34)
      ' Show log message
25.      Else
      ' Else (if sample field is empty)
26.          WScript.Echo "No data were read from value " & Chr(34) & Valu
      eName & Chr(34) & _
27.          " in field " & Chr(34) & FieldName & Chr(34) & _
28.          " of record " & Chr(34) & Record.RecordId & Chr(34)
29.      End If
30.
31.      Record.ClearFieldValue FieldName,ValueName
      ' Clear sample field value
32.      WScript.Echo "Successfully cleared value " & Chr(34) & ValueName
      & Chr(34) & _
33.          " in field " & Chr(34) & FieldName & Chr(34) & _
34.          " of record " & Chr(34) & Record.RecordId & Chr(34)
      ' Show log message
35.
36.      Record.ClearField FieldName
      ' Clear sample field
37.      WScript.Echo "Successfully cleared field " & Chr(34) & FieldName
      & Chr(34) & _
38.          " of record " & Chr(34) & Record.RecordId & Chr(34)
      ' Show log message

```

ILogonSettings

Exposes the default property to manage the logon settings through automation.

Properties

Property

Type: LogonProperties

Getting and setting logon properties.

typedefLogonProperties

- Main_DisableAutoUpdateCredentials = 0

In case of empty user credentials forces the SSOSimpleLogon object to not request users credentials.

- Main_RegistrySettingsName = 1

Specifies the registry subkey name to store last successful logon configuration in.

- Main_ForceBSPId = 2

Explicitly overrides the default logon method. The parameter should be in form of BSTR representing the ID of the BSP: {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}.

- AutoUpdateCredentials_Settings = 3

Settings for automatic UpdateCredentials dialog. The parameter should be in form of pointer to IUpdateCredentialsSettings interface.

- LogonDialog_MainWindowCaption = 5

Specifies the Caption text for the logon dialog.

- LogonDialog_MainWindowTip = 6

Specifies the Main tip text for the logon dialog.

- LogonDialog_ShowCancel = 7

Specifies whether or not the Cancel button on the Logon dialog should be visible.TRUE by default.

- LogonDialog_Domain = 8

Explicitly overrides the Domain field in the Logon dialog. By default currently logged in user's domain is used.

- LogonDialog_UserName = 9

Explicitly overrides the User name field in the Logon dialog. By default currently logged in user's name is used.

- LogonDialog_ShowOptions = 11

Specifies whether or not the Options button on the Logon dialog should be visible. TRUE by default.

- LogonDialog_OptionsExpanded = 12

Specifies whether or not options on the Logon dialog should be expanded. TRUE by default.

- Main_StoreSettingsInCurrentUser = 13

Specifies the root registry location to store logon parameters. If the option is set to TRUE (by default), the logon options are stored under the HKEY_CURRENT_USER and under the HKEY_LOCAL_MACHINE in the other case.

- LogonDialog_AllowPasswordLogon = 14

Specifies if the user is allowed to logon by password. If the parameter is set to FALSE (default), there will be no "Logon by password" option in the logon methods. The parameter is ignored and always FALSE when using with SSOSimpleLogonClient.

- LogonDialog_EnablePassthrough = 15

Specifies if the "Use current settings as defaults" check box on the Logon dialog should be enabled.

- LogonDialog_ParentWindowHandle = 16

Specifies the parent window handle for the Logon dialog.

- LogonDialog_EnableDomain = 17

Specifies whether or not user is allowed to change the logon Domain.

- LogonDialog_EnableUserName = 18

Specifies whether or not user is allowed to change the user's logon name.

- LogonDialog_ForcePassthrough = 19

Forces the logon process to automatically prompt user for the authenticator.

- LogonDialog_Image = 20

Overrides the default image in the top of the dialogs. Should be the BSTR with full path to the image, or the object implementing the standard IPicture interface. If path to the image is used as the parameter value, the image must be in BMP (bitmap), JPEG, WMF (metafile), or GIF format. The imagesize must be 452x85.

- Main_ForceLanguageId = 21

Forces language ID for UI. By default system language or language specified in Client Tray is used.

- LogonDialog_ForcePassword = 22

Specifies logon by password as a default logon mode.

- LogonDialog_ShowLangRect = 23

Specifies whether or not input language is displayed.

- Main_ForceWebServiceLogon = 24

NetIQ v4.7 and newer has a possibility to communicate through NetIQ WebService instead of NetIQ Authenticore Server. But in this case only some of SDK features are available (only obtaining of user credentials). The parameter specifies whether NetIQ Authenticore Server or NetIQ Web Service is used.

- Main_WebServerName = 25

NetIQ v4.7 and newer has a possibility to communicate through NetIQ WebService instead of NetIQ Authenticore Server. But in this case only some of SDK features are available (only obtaining of user credentials). This parameter specifies NetIQ Web Server name in DNS mode including port number. E.g. dc.authasas.local:8232.

- LogonDialog_SevenUI= 26

The parameter allows to activate the Windows 7 UI for authentication forms. This also works on the previous versions of operating systems.

Examples

The following example (SDK_19.vbs) shows the authentication dialog with specified caption and specified window tip text.

```
1. Const SubsName= "PasswordStore"  
   ' Default subsystem name  
2. Set logonCl = CreateObject ("LogonSDK.SSOSimpleLogonClient")  
   ' Custom system authentication object  
3. Set dlgSets = CreateObject("LogonSDK.LogonSettings")  
   ' Authentication settings object  
4.  
5. dlgSets(5) = "Logon window caption"  
   ' Logon window caption  
6. dlgSets(6) = "Logon window tip text"  
   ' Logon window tip text  
7.  
8. Set Creds = logonCl.Logon(dlgSets, SubsName, "ApplicationName")  
   ' Authentication, then new credential registration request
```

IStringEnum

The interface enables to enumerate strings in read only mode. You can use it in VBS with “For Each” construction.

Properties

IUnknown **	_NewEnum [get]
BSTR **	Item ([in] long Index) [get]
long *	Count [get]

IDeviceMonitor

The interface enables the device monitoring features. You can subscribe on events (for example, smart card was put on the reader, flash drive was disconnected from PC) and do whatever you want when these events occur.

The interface was implemented in NetIQ Advanced Authentication Framework version 4.7.

Identifiers of Authentication Service Provider

You can use the following authentication service providers GUIDs:

1. BIO-key Biometric Service Provider Version 1.9 – {EC4AC729-B969-6E46-BD2F-56B6055E18F8}.
2. Universal Card Authentication Provider – {ED2D1872-4DAC-A84B-AF7C-188642267D56}.
3. USB Flash Drive Authentication Provider – {1AF29AB5-0A30-0046-95DB-4FDA28989051}.
4. OATH OTP Authentication Provider – {C7D6704E-F66A-4EF0-93A3-C5EF13F0C7A2}.
5. RADIUS Authentication Provider – {E4828EC2-B520-46FC-9624-EB98487A7F2B}.

Function IsUnitInserted

The function allows to detect whether the smartcard/ flash drive was inserted or no.

Parameters

ModelId

Type: BSTR

The identifier of the authentication provider.

UnitId

Type: BSTR

The identifier of the smartcard or flash drive.

Function AdviseAll

The function allows to subscribe on all authentication providers events.

Parameter

AdviseCookie

Type: LONG*

The occurred events.

Function AdviseMode

The function allows to subscribe on events of all using smartcards/ flash drives of specified authentication providers.

Parameters

Modeld

Type: BSTR

The identifier of the authentication provider.

AdviseCookie

Type: LONG*

The occurred events.

Function AdviseUnit

The function allows to subscribe on events of specified smartcard/ flash drive of specified authentication providers.

Parameters

Modeld

Type: BSTR

The identifier of the authentication provider.

UnitId

Type: BSTR

The identifier of the smartcard or flash drive.

AdviseCookie

Type: LONG*

The occurred events.

Function Unadvise

The function allows to unsubscribe from all authentication providers events.

Parameter

AdviseCookie

Type: LONG*

The occurred events.

IDeviceMonitorEvents

The interface allows to configure actions when the specific events occur.

Function OnUnitInserted

The function allows to configure actions when the card was tapped in/put on/inserted in the smart card reader or flash drive was inserted.

Parameters

ModelId

Type: BSTR

The identifier of the authentication provider.

UnitId

Type: BSTR

The identifier of the smartcard or flash drive.

Function OnUnitRemoved

The function allows to configure actions when the card was took off/removed from the smart card reader or flash drive was unplugged.

Parameters

ModelId

Type: BSTR

The identifier of the authentication provider.

UnitId

Type: BSTR

The identifier of the smartcard or flash drive.

Examples

The following example (SDK_20.vbs) allows seeing the data of devices when they inserted or removed.

```
1. Function Test_OnUnitInserted(ModeId, UnitID)           ' Actions on inserting
2.     Select Case ModeId                                ' Translating the ModeId into
   the name of device
3.         Case "{1AF29AB5-0A30-0046-95DB-4FDA28989051}"
4.             AuthProv = "USB Flash Drive"
5.         Case "{ED2D1872-4DAC-A84B-AF7C-188642267D56}"
6.             AuthProv = "Card"
7.     End Select
8.
9.     WScript.Echo AuthProv + " with ID " + UnitId + " has been inserted" ' Output the re
   sult
10. End Function
11.
12. Function Test_OnUnitRemoved(ModeId, UnitID)           ' Actions on inserting
13.     Select Case ModeId                                ' Translating the ModeId into
   the name of device
14.         Case "{1AF29AB5-0A30-0046-95DB-4FDA28989051}"
15.             AuthProv = "USB Flash Drive"
16.         Case "{ED2D1872-4DAC-A84B-AF7C-188642267D56}"
17.             AuthProv = "Card"
18.     End Select
19.
20.     WScript.Echo AuthProv + " with ID " + UnitId + " has been removed" ' Output the re
   sult
21. End Function
22.
23. Set devMon = WScript.CreateObject("LogonSDK.DeviceMonitor","Test_") ' The Device Mo
   nitor object
24. Cookie = devMon.AdviseAll()           ' Subscribe to all events
25. MsgBox "Press Enter to quit"         ' Request to close the script
26. devMon.Unadvise Cookie                ' Unsubscribe from events
```


Error Codes Description

Here you can find the description of possible NetIQ errors. Also you can use [MSDN](#) website for detailed description of Microsoft and Windows Script Host errors (such as 0x80070005 "Access is denied").

RPC Server Errors

ID	Symbolic name	Description
0xC0FF0001L	RPCS_E_WAIT_FOR_INSTALL	Server installation was not completed. At the moment, server is awaiting for installation completion. Server is not able to work until the process is finished
0xC0FF0002L	RPCS_E_ALREADY_INSTALLED	Server is already installed. At the moment, it is working normally. Installation completion is not required
0xC0FF0003L	RPCS_E_CAN_NOT_IMPERSONATE	Could not impersonalize
0xC0FF0008L	RPCS_E_CREATE_CIPHER	Authenticore server could not create AstroSoft.Cipher COM-object. Either the object was not registered in the process of system installation or it could not get the Enterprise Key
0xC0FF0009L	RPCS_E_CREATE_DATA_PROVIDER	Server could not create AstroSoft.ADUserDataProvider object. Perhaps, the object was not registered while installing the system
0xC0FF000AL	RPCS_E_CREATE_KEYMANAGER	Authenticore server could not create AstroSoft.KeyManager COM-object. Perhaps, the object was not registered while installing the system

0xC0FF000BL	RPCS_E_CREATE_LOGON	Authenticore server could not create AstroSoft.Logon COM-object. Perhaps, the object was not registered while installing the system
0xC0FF000CL	RPCS_E_CREATE_MANAGER	Authenticore server could not create AstroSoft.Manager COM-object. Perhaps, the object was not registered while installing the system
0xC0FF000DL	RPCS_E_GENERATE_OR_WRITE_KEYS	Could not generate or save Enterprise Key. This computer may have problems either with the CryptoAPI or with keys storing infrastructure
0xC0FF000EL	RPCS_E_LISTEN_CALLS	Error calling RpcServerListen
0xC0FF000FL	RPCS_E_LOGON_USER	Could not log in as AuthenticoreService Possible error causes: - there is no AuthenticoreService account in the domain - account password and AuthenticoreService account unsynchronized - AuthenticoreService account was automatically blocked - AuthenticoreService account does not have "batch job" logon privileges on this computer
0xC0FF0010L	RPCS_E_READ_USER_NAME	Server could not read the name of user account under which the server must work
0xC0FF0011L	RPCS_E_REGISTER_INTERFACE	Server could not register RPC-interface
0xC0FF0012L	RPCS_E_WRITE_USER_NAME	Server could not save user account name under which it must work
0xC0FF0013L	RPCSKEY_E_WRONG_CLIENT	Server requested the Enterprise Key, is not the domain member or its request is incorrect

0xC0FF0014L	RPCSKEY_E_GET_TICKET	Could not get Kerberos Ticket of the Authenticore server which requested the Enterprise Key
0xC0FF0015L	RPCSKEY_E_NOT_LOCAL_CALL	This function is intended for the local call only
0xC0FF0016L	RPCSKEY_E_CONNECT_SERVER	Could not find Authenticore server or establish connection with it
0xC0FF0017L	RPCSKEY_E_REGISTER_SPN	Could not register Service Provider Name (SPN)
0xC0FF0018L	RPCSKEY_E_CREATE_TIKET	Could not get Kerberos Ticket using data received from Authenticore server
0xC0FF0019L	RPCSKEY_E_GET_TICKET_NO_SPN	Could not get Kerberos Ticket from Authenticore server, which had requested Enterprise Key: SPN is not registered. Most likely, the error occurred because Active Directory data replication had not been completed. In this case, please wait until replication is completed and then click Retry button

0xC0FF001AL	RPCSKEY_E_CLIENT_NOT_MEMBER _OF_GROUP	Authenticore server, which has requested Enterprise Key, is not included into the Authenticore Servers group. Most likely, the error occurred because Active Directory data replication had not been completed. In this case, please wait until replication is completed and then click Retry button
0xC0FF001BL	RPCS_E_NO_DELEGATE	The level of impersonalization, allowed by the requested side, is lower than "Delegate" level.
0xC0FF001CL	RPCS_E_WAIT_FOR_LICENSE	Server installation has not been completed. Currently the server is in progress of adding license.
0xC0FF001DL	RPCS_E_DELEGATION_DISABLED	Computer account is not trusted for delegation
0xC0FF001EL	RPCS_E_SENSITIVE_ACCOUNT	Cannot connect to the Authenticore server. Please, ensure that for your account the "Account is sensitive and cannot be delegated" option is turned off
0xC0FF0463L	RPCS_E_LOGON_LOGON_FAILED	Could not authenticate the user by provided authenticator

0xC0FF044FL	RPCS_E_LOGON_LOGON_BY_PASSWORD_FAILED	Could not authenticate the user by the entered password. The error could also occur if the entered account was invalid.
0xC0FF0451L	RPCS_E_ENUM_TEMPLATES_PUT_ITEM_FAILED	User could not re-enroll the authenticator
0xC0FF0453L	RPCS_E_ENUM_TEMPLATES_ADD_FAILED	User could not add new authenticator
0xC0FF0455L	RPCS_E_ENUM_TEMPLATES_REMOVE_FAILED	User could not remove the authenticator
0xC0FF045BL	RPCS_E_FIND_SERVER	Could not find Authenticore server
0xC0FF045EL	RPCS_E_FIND_LICENSED_SERVER	Could not find Authenticore server with active license.
0xC0FF0461L	RPCS_E_ADD_LICENSE	Could not add license
0xC0FF0463L	RPCS_E_LOGON_LOGON_FAILED_EX	Could not authenticate the user by provided authenticator
0xC0FF0465L	RPCS_E_ADD_LICENSE_EX	Could not add license
0xC0FF04BBL	RPCS_E_MANAGER_CREATE_FAILED	Could not permit User to use authenticators
0xC0FF04BDL	RPCS_E_MANAGER_REMOVE_FAILED	Could not forbid authenticators for User
0xC0FF04CCL	RPCS_E_USER_PUT_SETTINGS_FAILED	Could not initialize settings for User

0xC0FF04CEL	RPCS_E_USER_CLEAN_AUTHENTICATORS_FAILED	Could not clear the list of enrolled authenticators of user
0xC00004CFL	RPCS_E_COMPUTER_CANTWRITEOBJECT	Could not initialize settings for computer
0xC0FF04D2L	RPCS_E_USER_GET_SETTINGS_FAILED	Could not obtain settings for User
0xC0FF04D3L	RPCS_E_USER_GET_TEMPLATES_FAILED	Could not get the list of enrolled authenticators of user
0xC0FF04D4L	RPCS_E_USER_CHANGE_PASSWORD_FAILED	Could not change password for user
0xC0FF04D5L	RPCS_E_USER_PUT_PASSWORD_FAILED	Could not set password for user
0xC0FF0516L	RPCS_E_SERVER_CAN_NOT_START	Could not start Authenticore Server service
0xC0FF0517L	RPCS_E_SERVER_CAN_NOT_READ	Authenticore Server service could not read data from Active Directory
0xC0FF0518L	RPCS_E_SERVER_CAN_NOT_WRITE	Authenticore Server service could not write data into Active Directory
0xC0FF0519L	RPCS_E_SERVER_CAN_NOT_DECODE	Authenticore Server service could not decrypt data retrieved from Active Directory. Either data was corrupted or the Enterprise Key has been changed
0xC0FF051BL	RPCS_E_GETKEYS_FAILED	Could not transfer Enterprise Key to server
0xC0FF051CL	RPCS_E_GETKEYS_FROM_FAILED	Could not get Enterprise Key from server
0xC0FF051FL	RPCS_E_EXPORT_KEYS_FAILED	Could not export Enterprise Key
0xC0FF0521L	RPCS_E_IMPORT_KEYS_FAILED	Could not import Enterprise Key
0xC0FF0523L	RPCS_E_GENERATION_KEYS_FAILED	Could not generate Enterprise Key
0xC0FF0524L	RPCS_E_AD_IS_OFFLINE	Active Directory is offline
0xC0FF06BCL	RPCS_E_LOGON_REFUSED_BY_RULES	Logon refused by security rules.
0xC0FF06BDL	RPCS_E_RULESERVER_CALL_FAILED	Error occurred while checking security rules

SrvWrapperErrors

0xC1050457L	SRVWRAPPER_E_SERVER_NOT_FOUND	The user could not be authenticated. The error could occur due to 1. Authenticore server was not found 2. The authentication method is not supported by available Authenticore servers (required BSP module is missing on server) 3. Lost communication with Domain Controller 4. The required subsystem was not installed
0xC1050458L	SRVWRAPPER_LOG_E_SERVER_NOT_FOUND	The user could not be authenticated. The error could occur due to: 1. Authenticore server was not found 2. The authentication method is not supported by available Authenticore servers (there is no required BSP module on server) 3. Lost communication with Domain Controller. 4. The required subsystem was not installed.
0xC105045CL	SRVWRAPPER_E_LOCAL_USER	Either user account or authenticator is invalid.
0xC105045DL	SRVWRAPPER_E_NOT_BIOUSER	Authentication Failed. Press OK to try again.
0xC1050466L	SRVWRAPPER_E_CACHE_USED	Authenticore server not found. User could not be logged in using authenticator from cache

Password Filter Errors

0xC104057AL	PWDFILT_E_PASSWORD_SET_FAILED	Error while resetting password for user
0xC104058AL	PWDFILT_E_PASSWORD_CHANGE_FAILED	Error while changing password for user

Password Manager Errors

0xC1080585L	PWDMGR_E_ERROR_OCCURED	An error occurred during Password Manager work
0xC1080586L	PWDMGR_E_CHANGE_PASSWORD_FAILED	Could not change password for user It is recommended to check "Minimal password age" domain setting. In case its value differs from 0, it is possible that password change can be denied because the password has been already changed within the specified time interval. Also, password cannot be changed in case "User cannot change password" account setting is enabled
0xC1080587L	PWDMGR_E_BAD_START_TIME	The time period specified using command prompt had expired before Password Manager was started. The service has been stopped

EventLogErrors

0xC10705DCL	LOG_E_CANT_WRITE_REMOTE_LOG	Could not get access to remote Log Server There is either no Log Server, it was turned off, or being reloaded. In case the error persists, it is recommended to check Firewall settings and the correctness of the domain names permission.
-------------	-----------------------------	--

BioAPIErrors

0xC1010000L	BIO_E_INITIALIZE	Could not initialize BioAPI framework
0xC1010001L	BIO_E_LOAD_MODULE	Could not load the required BioAPI BSP module
0xC1010002L	BIO_E_ENROLL	Could not get enrolled authenticator
0xC1010003L	BIO_E_IDENTIFY	Could not get authenticator
0xC1010004L	BIO_E_VERIFY	Could not compare user's authenticators
0xC1010005L	BIO_E_DATA_CORRUPTED	Could not load authenticators from the memory. Data is corrupt
0xC1010006L	BIO_E_COMPARE_BSP_MISMATCH	The type of enrolled authenticator does not correspond to the type of the given authenticator
0xC1010007L	BIO_E_COMPARE_DATA_MISMATCH	Authenticator does not correspond to the enrolled authenticator

Authenticore Server Errors

0xC1000000L	LOGON_E_CREATE_TEMPLATE	Could not create authenticator. The list of user authenticators may be corrupt
0xC1000001L	LOGON_E_LOAD_TEMPLATE	Could not load the authenticator. The list of user authenticators may be corrupt
0xC1000002L	LOGON_E_READ_COLLECTION	Could not read user authenticators list
0xC1000003L	LOGON_E_WRONG_PASSWORD	Either user account or password value is invalid
0xC1000004L	LOGON_E_WRONG_AUTHENTICATOR	Authentication Failed. Press OK to try again.
0xC1000005L	LOGON_E_CANNOT_LOGON	Authentication Failed. Press OK to try again.
0xC1000006L	LOGON_E_OPERATION_DENIED	This operation is forbidden by administrator
0xC1000007L	LOGON_E_TOO_MANY_AUTHENTICATORS	The allowed amount of authenticators is exceeded
0xC1000008L	LOGON_E_SERVER_NOT_FOUND	Could not set connection with the Authenticore server. Check network connection and try again. If the error persists please contact your system administrator

0xC1000009L	USER_E_CHANGE_PASSWORD_INVALID	The passwords were unsynchronized
0xC100000AL	USER_E_CHANGE_PASSWORD_POLICY	Could not change password for the user. The generated value does not satisfy the security policies. It is recommended to check "Minimal password age" domain setting. In case its value differs from 0, the password change can be denied because the password has been already changed within the specified time interval
0xC100000BL	USER_E_CHANGE_PASSWORD_ACCESS_DENIED	Could not change user password. The current security settings forbid the user to change his/her password
0xC100000CL	USER_E_CHANGE_PASSWORD	Could not change password for the user. The reason is unknown
0xC100000DL	LOGON_E_WRONG_DATE	Time interval from the moment the user authenticator was obtained and the moment it was delivered to the Authenticore server exceeds the value of the settings, which regulates authenticator validity period (5 minutes by default). This error can occur as a result of either system time desynchronization between user computer and Authenticore server or criminal attempt to use authenticator intercepted over network
0xC100000EL	LOGON_E_LOAD_BSP	Could not load BioAPI BSP module. Either the required BSP module is not installed on the Authenticore server or it failed to load. The system will attempt to authenticate on another Authenticore server.
0xC100000FL	CHANGEPWD_OUT_OF_RESOURCES	System resources are not enough to change password for the user
0xC10006BEL	LOGON_E_LOGON_REFUSED_BY_RULES	Logon refused by security rules.
0xC10006BFL	LOGON_E_DENY_LOGON_BY_PASSWORD	Logon by password was denied.

Authentication Providers Errors

0xC1020000L	PROV_E_NO_USER	The user was not found
0xC1020001L	PROV_E_ACCESS_DATA	Could not get access to user data
0xC1020002L	PROV_E_PROPERTY_NOT_FOUND	The property was not found. Perhaps the Active Directory scheme is not extended by additional attributes
0xC1020003L	PROV_E_ALREADY_CREATED	User is already allowed to use authenticators
0xC1020004L	PROV_E_CREATE_ENUMERATOR	Could not create users sorting object
0xC1020005L	PROV_E_SEARCH_USER	Could not start user search
0xC1020006L	PROV_E_ACCESS_DENIED	Access is denied. Not enough permissions
0xC1020009L	PROV_E_AD_OBJECT_NOT_BIND	Unable to get object data in AD.
0xC102000AL	PROV_E_ADAM_OBJECT_NOT_BIND	Unable to get object data in ADAM.
0xC102000BL	PROV_E_ADAM_NOT_OPERATIONAL	Could not get access to ADAM server

Cryptography Errors

0xC1030001L	CRYPT_E_USER_DATA_CORRUPTED	User data corrupted
0xC1030002L	CRYPT_E_VERIFY_SIGNATURE	Either user data or the Enterprise Key is corrupt
0xC1030003L	CRYPT_E_INIT_PROVIDER	Could not initialize required Crypto Service Provider (CSP)
0xC1030004L	CRYPT_E_GENERATE_OR_EXPORT_KEYS	Could not generate or export cryptographic keys
0xC1030005L	CRYPT_E_IMPORT_KEYS	Could not import cryptographic keys
0xC1030006L	CRYPT_E_DATA_CORRUPTED	Data is corrupted

Manager Errors

0x01060001L	MGR_S_LAST_TEMPLATES_REMOVED	Several authenticators were deleted because the allowed amount of authenticators was reduced
0xC1060002L	MGR_E_LOGON_DOMAIN_REDIRECTION_OP_UNSUPPORTED	The operation is not supported while the domain redirection policy is enabled

Plugins Errors

0xC1090000L	PLUGIN_E_NOT_REGISTERED	The specified Plug-in is not registered on the server.
0xC1090001L	PLUGIN_E_CANNOT_CREATE	Could not create registered Addon.
0xC1090002L	PLUGIN_E_USER_NOT_TRUSTED	The user was authenticated by password.
0xC1090003L	PLUGIN_E_OPERATION_DENIED	The operation is forbidden.

Licensing Errors

0xC10A0001L	LIC_E_INVALID_FORMAT	Invalid format of license data.
0xC10A0002L	LIC_E_LICENSE_NOT_FOUND	License not found.
0xC10A0003L	LIC_E_LICENSE_STORAGE_CORRUPTED	License storage data is corrupted.
0xC10A0004L	LIC_E_LICENSE_CORRUPTED	License data was changed or corrupted.
0xC10A0005L	LIC_E_RESTRICTIONS_ERROR	Your license does not match the time period restriction, the product version restriction or the domain name is wrong
0xC10A0006L	LIC_E_PUBLICKEY_CORRUPTED	Cannot validate digital signature of the license. Certificate may be missing or corrupt.
0xC10A0007L	LIC_E_PLUGIN_DOESNT_SUPPORT_LICENSING	This Addon does not support licensing.
0xC10A0008L	LIC_E_START_LIMIT_ERROR	The actual number of installed Authenticore Servers exceeds the number allowed by the License.
0xC10A0009L	LIC_E_USERS_LIMIT_ERROR	Actual number of Authasas-enabled accounts exceeds the number allowed by the License.
0xC10A000AL	LIC_E_DOWNGRADE	The license you are trying to add allows fewer number of licensed objects than you have now

Backup Provider Errors

0xC10C0001L	BACKUPPROV_E_BAD_PASSWORD_OR_DATA	Bad password or data corrupted.
-------------	-----------------------------------	---------------------------------

Administration Tools Errors

0xC10D0001L	ADMTOOLS_E_NOT_MLADMIN	You don't have rights for changing settings on this page. Please, ensure that you are the member of the NetIQ Admins group and these rights are delegated to the NetIQ Admins group.
0xC10D0002L	ADMTOOLS_E_NO_RIGHTS	You don't have rights for changing settings on this page. Please, ensure that these rights are delegated to you.

GINA Errors

0xC10B0645L	GINA_E_LOGON_BY_PASSWORD_FAILED	Could not authenticate the user by the entered password. The error could also occur if the entered account was invalid.
-------------	---------------------------------	--

Data Errors

0xC10E0001L	DATA_E_FIELD_NOT_SET	The field value is not set.
0xC10E0002L	DATA_E_VALUE_NOT_SET	The subfield value is not set.
0xC10E0003L	DATA_E_SUBSYSTEM_NOT_FOUND	Subsystem is not found.
0xC10E0004L	DATA_E_ACCESS_DENIED	Data access denied.
0xC10E0005L	DATA_E_RECORD_NOT_FOUND	Record is not found.
0xC10E0006L	DATA_E_USER_NOT_TRUSTED	The user was authenticated by password.
0xC10E0007L	DATA_E_INVALID_FIELD_NAME	Invalid field name.
0xC10E0008L	DATA_E_BAD_SCHEME_SIGNATURE	Bad schema signature
0xC10E0009L	DATA_E_USERS_LICENSE_NOT_FOUND	Subsystem users license is not found.
0xC10E000AL	DATA_E_BASE_LICENSE_NOT_FOUND	Subsystem servers license is not found.
0xC10E000BL	DATA_E_NOT_SUBSYSTEM_USER	User is not using given subsystem.
0xC10E000CL	DATA_E_USERS_LICENSE_LIMIT_ERROR	Actual number of the subsystem-enabled accounts exceeds the number allowed by the License.
0xC10E06A5L	DATA_E_ADMIN_GET_DATA_FAILED	Unable to get the subsystem data for the user
0xC10E06A6L	DATA_E_USER_GET_DATA_FAILED	User is unable to get the subsystem data
0xC10E06A7L	DATA_E_ADMIN_FAILED_TO_ALLOW_TO_USE_SS	Unable to make user the client of the subsystem
0xC10E06A9L	DATA_E_USER_FAILED_TO_BE_SS_CLIENT	User failed to be a client of the subsystem

0xC10E06ADL	DATA_E_ADMIN_CHANGE_DATA_FAILED	Unable to change the subsystem data for the user
0xC10E06AFL	DATA_E_USER_CHANGE_DATA_FAILED	User is unable to change the subsystem data
0xC10E06B1L	DATA_E_RESET_PASSWORD	The password was reset for user. Could not reset special data for subsystem
0xC10E06B3L	DATA_E_ADMIN_REMOVE_SS_DATA	Unable to deny user to use the subsystem
0xC10E06B5L	DATA_E_USER_REMOVE_SS_DATA	User was unable to quite using the subsystem
0xC10E06B7L	DATA_E_RESET_DATA	The password was reset for user. Could not reset special data for subsystem
0xC10E06B8L	DATA_E_RESET_DATA_FULL_RESET	The password was reset for user. Could not reset special data for subsystem. The subsystem data was reset completely
0xC10E06BAL	DATA_E_SUBSYSTEM_LIST_INVALID_COMMON	The subsystems list for user is invalid and was cleared.
0xC10E06BBL	DATA_E_SUBSYSTEM_LIST_INVALID_SS	The subsystems list for user is invalid and was cleared.
0xC10E06C0L	DATA_E_CONTAINER_NOT_FOUND	Data container is not defined in the schema.

Troubleshooting

Error “The user was authenticated by password”

You can get the error “The user was authenticated by password” during reading information from a record for user without enrolled authenticators. This error is by design. The field “password” in PasswordStore subsystem is not available if user logged on by domain password.

Index

A

Active Directory 8, 12-13, 15, 30
Administrator 1, 8-9
Authentication 1, 6, 8-10, 13, 22, 25, 27, 56, 72

B

BIO-key 56

C

Card 56
Client 9-10
Client Tray 53
Create 10, 34

D

Data 75
Domain 52

E

Error 62, 77

G

GINA 75

L

List 14
Logon 15, 17, 44

O

OATH 56

P

Password 68-69
Properties 30, 44, 46, 49, 52, 55

R

RADIUS 56

Record 13, 47, 49

Remove 35

S

Server 53, 62, 70

Software 1

U

User 8, 20

W

Windows 8, 62

Windows 7 54

Windows XP 10