MyID MFA and PSM

Version 5.1

MyID Authentication Server Developers Guide

Lutterworth Hall, St Mary's Road, Lutterworth, Leicestershire, LE17 4PS, UK
www.intercede.com | info@intercede.com | @intercedemyid | +44 (0)1455 558111

Document reference: IMP2062-5.1.0

May 2025

Copyright

Conventions used in this document

- Lists:
  - Numbered lists are used to show the steps involved in completing a task when the order is important.
  - Bulleted lists are used when the order is unimportant or to show alternatives.
- **Bold** is used for menu items and for labels.

  For example:
  - Record a valid email address in **'From' email address**.
  - Select **Save** from the **File** menu.
- *Italic* is used for emphasis:

  For example:
  - Copy the file *before* starting the installation.
  - Do *not* remove the files before you have backed them up.
- ***Bold and italic*** hyperlinks are used to identify the titles of other documents.

  For example: "See the ***Release Notes*** for further information."

  Unless otherwise explicitly stated, all referenced documentation is available on the product installation media.
- A `fixed width` font is used where the identification of spaces is important, including filenames, example SQL queries and any entries made directly into configuration files or the database.
- **Notes** are used to provide further information, including any prerequisites or configuration additional to the standard specifications.

  For example:

  Note: This issue only occurs if updating from a previous version.
- Warnings are used to indicate where failure to follow a particular instruction may result in either loss of data or the need to manually configure elements of the system.

  For example:

  Warning: You must take a backup of your database before making any changes to it.

Contents

1 Introduction

Note: MyID MFA and MyID PSM were previously known as Authlogics products. Authlogics is now an Intercede Group company and the products have been rebranded accordingly. The term 'Authlogics' may still appear in certain areas of the product.

MyID Authentication Server is a multi-factor authentication system which provides:

- Token and tokenless, device and deviceless, Multi-Factor Authentication.
- Mobile Push Authentication.
- NIST 800-63B compliant Password Security Management solution.
- Self-service password reset and unlocking.
- Web Service API and RADIUS interfaces for connectivity.
- Multiple Authentication technologies.

MyID Authentication Server has been designed to work with the following directory services:

- Microsoft Active Directory (no schema extensions required).

1.1 Guidance

This developers guide provides detailed information about how to use the MyID Authentication Server Web Services Application Programming Interface (WSAPI). You are recommended to use this guide in conjunction with the ***MyID Authentication Server Installation and Configuration Guide***, which is designed to be an infrastructure document.

2 Web services communication

MyID Authentication Server supports authentication with a Web API using the following protocols:

- HTTPS GET
- HTTP POST

By default, the Web Services run on TCP:14443 for SSL (with encryption). You *must* install an SSL certificate onto the server and bind it to the IIS web site.

Both IPv4 and IPv6 are supported for communication with Web Services.

You can use the web API for managing and automating all server functions. A list of available methods is available in section *3.3*, *WSAPI function details*, and also through:

`https://<ServerName>:14443/Services/swagger`

Where `<ServerName>` is your server address.

This section contains details on communicating with the web service for the following purposes:

- Processing an authentication request and verifying a transaction.

  See section *2.1*, *Authentication and transaction verification*.
- Generating a Grid challenge.

  section *2.2*, *Challenge Grid generation*
- Generating a Phrase challenge.

  See section *2.3*, *Phrase Challenge generation*.
- Generating a One Time Code challenge.

  See section *2.4*, *One Time Code Challenge triggering*.
- Authenticating with an Identity Provider.

  See section *2.5*, *Identity Provider*.

2.1 Authentication and transaction verification

You can use the following web service operations to process an authentication request and verify a transaction:

- `AuthenticateUser`
- `VerifyTransaction`

2.1.1 AuthenticateUser

To process an authentication request through the API, supply the `accountName` and `passcode` to the `AuthenticateUser` function, and it returns a status code; see section *2.1.3*, *Return codes*.

Example of an HTTPS GET authentication validation request using PowerShell:

```
$uri = 'https://<ServerName>/Services/api/AuthenticateUser'
$body = 'accountName=$AccountName&passcode=$Passcode'
$contentType = 'application/x-www-form-urlencoded'
$headers = @{Accept = 'application/json'}
$response = Invoke-WebRequest -Uri $uri -Method Post -ContentType $contentType -Body $body
$response.Content
```

Which returns the following:

```
<int>2</int>
```

indicating an invalid login.

Alternatively, you can request a JSON response:

```
$uri = 'https://<ServerName>/Services/api/AuthenticateUser'
$body = 'accountName=$AccountName&passcode=$Passcode'
$contentType = 'application/x-www-form-urlencoded'
$headers = @{Accept = 'application/json'}
$response = Invoke-WebRequest -Uri $uri -Method Post -ContentType $contentType -Body $body
-Headers $headers
$response.Content
```

Which returns the following:

```
2
```

2.1.2 VerifyTransaction

To verify a transaction through the API, supply the `accountname`, `passcode` and transaction-specific data to the `VerifyTransactionfunction` function, and it returns a status code; see section *2.1.3*, *Return codes*.

Example of an HTTPS GET transaction verification request user PowerShell:

```
$uri = 'https://<ServerName>/Services/api/VerifyTransaction'
$body = 'accountName=$AccountName&passcode=$Passcode&transactionData=1234567890'
$contentType = 'application/x-www-form-urlencoded'
$headers = @{Accept = 'application/json'}
$response = Invoke-WebRequest -Uri $uri -Method Post -ContentType $contentType -Body $body
$response.Content
```

Which returns the following:

```
<int>2</int>
```

indicating an invalid transaction verification.

2.1.3 Return codes

Code	Message	Description
0	Access Granted.	Credentials are valid.
1	Access Denied.	Account name not found.
2	Access Denied.	Invalid passcode.
5	Access Denied.	Account expired.
7	Access Denied.	Account disabled, locked out, or not valid at this time.
8	Access Denied.	Authentication not available due to a licensing issue.
13	Access Granted.	A pattern change is required.
111	Access Denied.	Directory related error.

2.2 Challenge Grid generation

To integrate a Grid Deviceless challenge into a web page or application, call the `GetToken` endpoint with an HTTPS GET request, specifying the `type=pingrid`. The `GetToken` endpoint accepts the following parameters, all of which are optional, and you can combine as required:

- `accountname`
- `type`
- `resolution`
- `format`
- `background`

You can specify the theme of the generated image using the Global Settings in MMC.

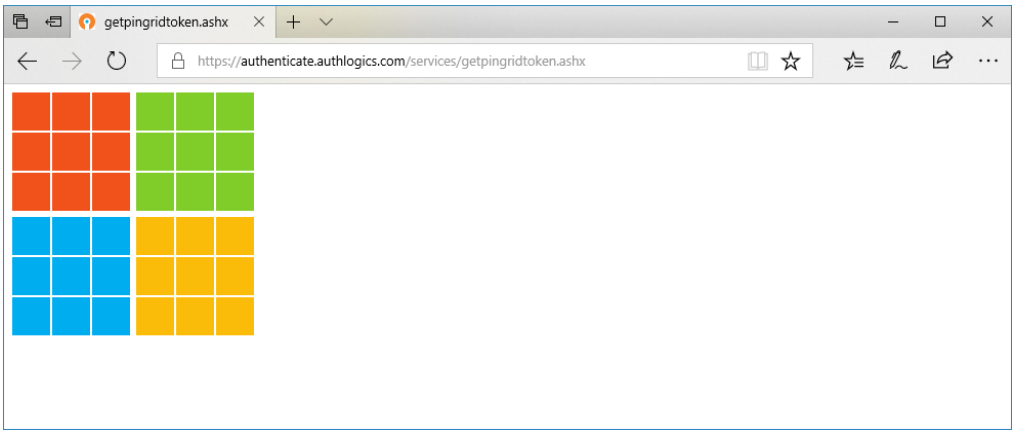
2.2.1 accountname

The `accountname` parameter generates a challenge specific for that user.

To generate a blank challenge grid, call the `GetToken` endpoint without the `accountname` parameter, or a blank `accountname` value. For example:

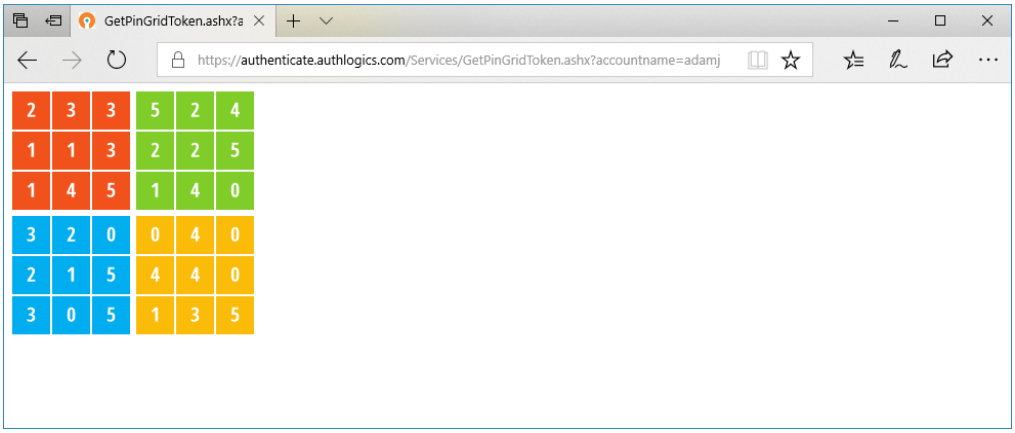
`https://<ServerName>:14443/services/api/gettoken`

The web service returns an image as follows, which is appropriate to show when the `accountname` is not specified:



If you specify a value for the `accountname` parameter, numbers are placed on the Grid, or Phrase challenge text is produced. Even if a user account does not match the name provided, a challenge is still generated to prevent the disclosure of an actual user account. For example:

```
https://<ServerName>:14443/services/api/gettoken?accountname=bogususer
```



2.2.2 format

The `format` parameter determines the graphical image type of the challenge grid, and accepts the following options:

- `PNG` (the default format when this parameter is not specified)
- `BMP`
- `JPG`
- `GIF`
- `TXT` (this returns the values for a grid in plain text and does not return a graphic image)

The image is always a square 1:1 ratio.

For example:

```
https://<ServerName>:14443/services/api/gettoken?accountname=bobm&
format=TXT
```

2.2.3 resolution

The `resolution` parameter sets the resolution of the generated Grid image. This should match the size of the HTML image object where the image is being displayed to prevent browser rescaling, which may result in a fuzzy or blurred image. If you do not specify the `resolution` parameter, the resolution set in the MMC is used.

For example:

```
https://<ServerName>:14443/services/api/gettoken?accountname=bobm&
resolution=500
```

Note: If you specify a resolution greater than 2500, a resolution of 2500 is used. If you specify a resolution less than 50, the Global Settings resolution is used.

2.2.4 background

The optional `background` parameter sets the background color to use when a non-transparent image type is used (that is, `BMP` and `JPG`). If you do not set the `background` parameter, the resolution set in the MMC is used.

For example:

```
https://<ServerName>:14443/services/api/gettoken?accountname=bobm&
background=black
```

The `background` parameter accepts the following values:

- `Black`
- `White`
- `Transparent`

2.3 Phrase Challenge generation

To integrate a Phrase Deviceless challenge question into a web page or application, call the `GetToken` endpoint with an HTTPS GET request and specify the `type=pinphrase`. The MyID Authentication Server returns a challenge string.

The `GetToken` endpoint requires only the `type` and `accountname` parameters for Phrase.

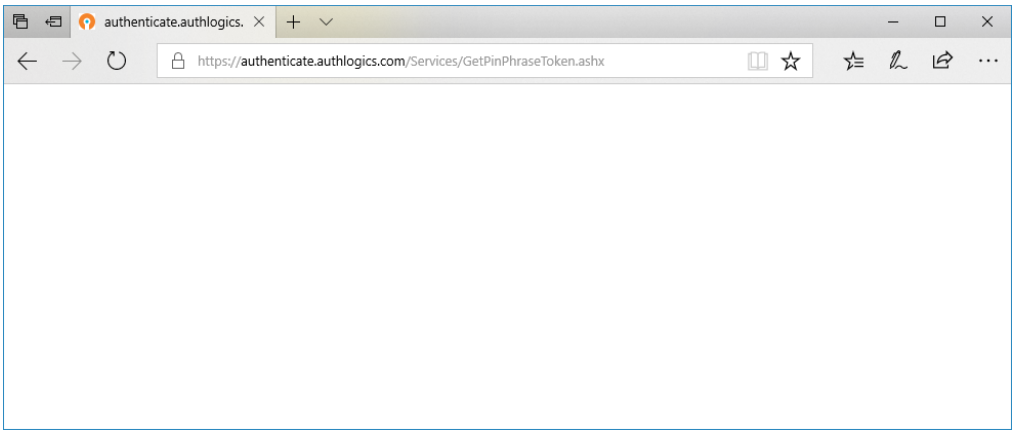
2.3.1 accountname

The `accountname` parameter generates a challenge specific for that user.

To generate a blank PINphrase challenge, call the `GetToken` endpoint without the `accountname` parameter, or with a blank `accountname` value; for example:

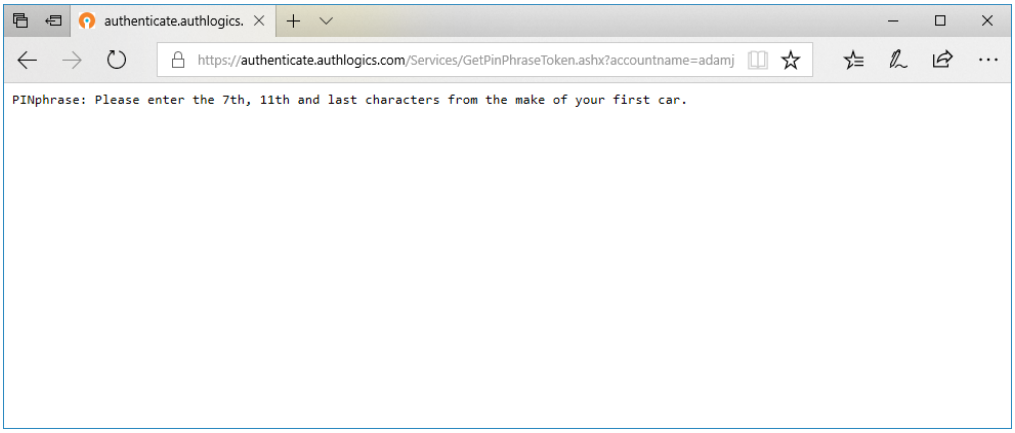
```
https://<ServerName>:14443/services/api/gettoken
```

The web service returns a blank string as follows, which is appropriate to show when the `accountname` is not specified.



When you specify a value for the `username` parameter, a challenge string is returned. Even if a user account does not match the name provided, a challenge string is still generated to prevent the disclosure of an actual user account. For example:

```
https://<ServerName>:14443/services/api/gettoken?accountname=bobm&
type=pinphrase
```



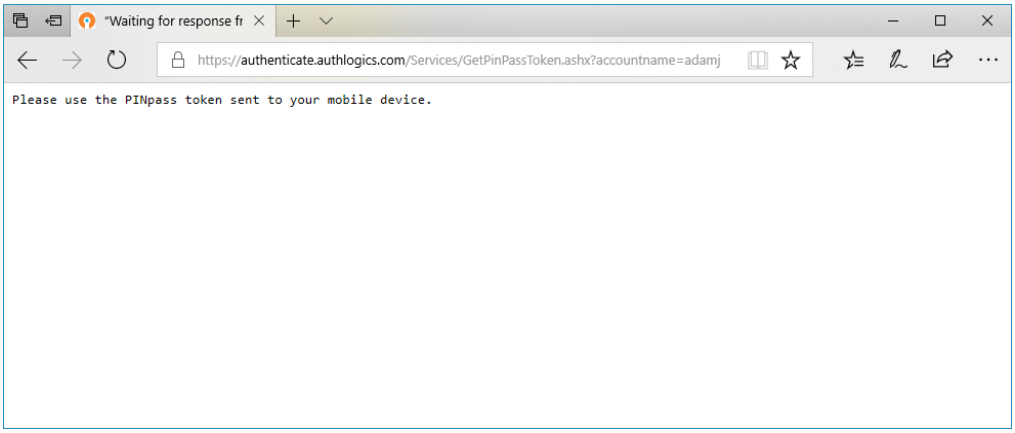
2.4 One Time Code Challenge triggering

While One Time Code cannot generate a Deviceless challenge, the web service call triggers the sending of a server-generated token that is appropriate for the user.

The `GetToken` endpoint requires only the `type` and `accountname` parameters for PINpass.

For example:

```
https://<ServerName>:14443/services/api/gettoken?accountname=bobm&
type=PINpass
```



2.5 Identity Provider

The Identity Provider (IdP) carries out all authentication in the MyID Authentication Server. Without additional configuration, it supports Windows Authentication (Kerberos or NTLM) and Mutual TLS.

In addition, you can use the MyID Management Console to configure it to use Client Credentials.

3 Web Services API

In addition to the MyID Management Console, the MyID Authentication Server includes a flexible Web Services Application Programming Interface (WSAPI) which allows you to carry out user account and server configuration management from remote systems. The WSAPI allows for integration with your workflow, change management processes, and automatic provisioning systems without manual intervention using the MyID Management Console.

Note: Some API functions make use of `enum`-based properties. These property values are case sensitive, and fail if you use the wrong case.

This section contains:

- How to authenticate to the WSAPI.

  See section *3.1*, *Authentication to the WSAPI*.
- A list of the WSAPI functions.

  See section *3.2*, *WSAPI function list*.
- Details of the WSAPI functions.

  See section *3.3*, *WSAPI function details*.
- The data types.

  See section *3.4*, *Data types*.
- An example of how to create a user programmatically using the WSAPI.

  See section *3.5*, *Example: programmatically creating a user account*.
- Using the WSAPI with Visual Studio.

  See section *3.6*, *Using the Web Services API with Visual Studio*.
- Changes to the WSAPI in version 5.0 from 4.2.1.

  See section *3.7*, *Web Service call changes in version 5.0 from 4.2.1*.

3.1 Authentication to the WSAPI

The WSAPI interface requires an authentication token from the IdP for certain function calls and to access certain property values. You can perform some operations without authentication, whereas others require authentication with an Administrator, an Operator, or the actual user.

You must install an SSL certificate on the web server, and make all IdP and WSAPI connections using HTTPS to secure the logon credentials to the web server.

You cannot authenticate with Windows or Basic authentication. For server-to-server authentication, you are recommended to create a client credential. For more information on creating a client credential, see the *Creating a client credential application* section in the ***MyID Authentication Server Installation and Configuration Guide***.

In addition to the credentials, you must provide a scope when you authenticate. The supported scopes are:

- `rest_api` – Provides access to all the API methods.
- `rest_api_external` – Provides access to a restricted set of API methods.

3.2 WSAPI function list

The following is a list of all available WSAPI functions:

Method	Internal use only	Available without authentication	Available with external scope
AddFidoCredential			✓
AddUserPasswordHash	✓		
AuthenticateRadius	✓	✓	✓
AuthenticateUser		✓	✓
AuthenticateUserAdPasswordless	✓	✓	✓
AuthenticateUserOidc	✓		✓
ChangeADpassword		✓	✓
CheckPasswordAgainstPolicy		✓	✓
CreateRealm			
CreateUser			
CreateUserExternal			
CreateUserPasswordReset	✓	✓	✓
DeleteRealm			
DeleteUser			
DisableEmergencyOverride			
DisablePinGrid			
DisablePinPass			
DisablePinPhrase			
DisablePush			
DomainExists			
EnableEmergencyOverride			
EnableFidoCredential			✓
EnablePinGrid			
EnablePinPass			
EnablePinPhrase			
EnablePush			
GenerateNewUserSeed			
GetAuthlogicsUsers			
GetDomains			
GetFullProvisionedUsers			
GetGlobalSettings	✓	✓	✓
GetMinimumClientVersion		✓	✓

Method	Internal use only	Available without authentication	Available with external scope
GetOathUrl			✓
GetPairKeyParameters	✓		✓
GetPasswordPolicySettings	✓	✓	✓
GetRealms			
GetRealmsAt			
GetRealmsInfo			
GetSecurityKey		✓	✓
GetServerVersion		✓	✓
GetSettingsProperty			
GetUser	✓	✓	✓
GetUserProperty		1	✓
IsRealmEmpty			
PairDevice	✓		✓
PasswordHashExists			
PinGridChangeMIP			✓
PinGridGenerateMIP			
PinGridProvision			
PinPassChangePin			✓
PinPassGeneratePIN			
PinPassProvision			
PinPhraseAddQuestion			
PinPhraseEditQuestion			
PinPhraseGenerateCodeword			
PinPhraseProvision			
PinPhraseRemoveQuestion			
RealmExists			
RemoveFidoCredential			✓
RenameRealm			
RenameUser			
ResetADpassword	✓	✓	✓
SendHTMLPinGridLetter			
SendHTMLPinPassLetter			

[1] Authentication is performed on an individual property basis.

Method	Internal use only	Available without authentication	Available with external scope
SendHTMLPinPhraseLetter			
SendHTMLPushLetter			
SendPresendToken			
SendRealTimeToken		✓	✓
SendRealTimeTokenbyProduct		✓	✓
SendToken			
SetADpassword			
SetOnlineVaultAdPassword	✓		✓
SetSettingsProperty			
SetUserProperty			✓
StartPushAuthentication	✓	✓	✓
SyncDevice			✓
TokenHardwareAdd			✓
TokenHardwareEnabled			✓
TokenHardwareRemove			✓
UpdateFidoCredential			✓
UpdateLicenceFile			
UpdateLicenceKey			
ValidateAdPassword			
VerifyEmergencyAccess		✓	✓
VerifyTransaction		✓	✓
YubiKeyOtpChangePin		✓	✓
YubiKeyOtpProvision			

3.3 WSAPI function details

This section contains a reference to the WSAPI functions.

3.3.1 Function: AddFidoCredential

The `AddFidoCredential` function adds a Fido credential to the user account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`credential`	FidoCredential	See section *3.4, Data types*.	A representation of a FidoCredential.

3.3.2 Function: AuthenticateUser

The `AuthenticateUser` function processes a logon request for a user account.

Parameter	Type	Value Sample	Description

`accountName`	string	`AndyP`
		`domain\andyp`
		`andyp@sample.com`
			A user account name which is resolvable in the directory.

`passcode`	string	`123456`	A passcode to authenticate the account.

The function returns an integer code indicating the result of the authenticate request:

0	Access Granted.	Credentials are valid.
1	Access Denied.	Account name not found.
2	Access Denied.	Invalid passcode.
5	Access Denied.	Account expired.
7	Access Denied.	Account disabled, locked out, or not valid at this time.
8	Access Denied.	Authentication not available due to a licensing issue.
13	Access Granted.	A pattern change is required.
111	Access Denied.	Directory related error.

3.3.3 Function: ChangeADpassword

The `ChangeADpassword` function changes the Active Directory Domain account password for a user account. You can call this function anonymously, although you must supply the existing password.

This function results in a password change event on the Active Directory and not a password reset event.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`oldADpassword`	string	`Pa55w0rd`	The old Active Directory password for the user account specified in `accountName`.
`newADpassword`	string	`P@ssWord`	The new Active Directory password to be written to the Windows Domain.

3.3.4 Function: CheckPasswordAgainstPolicy

The `CheckPasswordAgainstPolicy` function checks a supplied password against the configured Password Policy on the MyID Server; it does *not* attempt to set the supplied password.

To configure the policy, apply a Group Policy containing the Password Policy Agent template to the authentication server computer account. This is typically the same policy that is applied to the Domain Controllers.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP`	A user account name in the directory.
`dnsDomain`	string	`DomainName`	The user's domain DNS name.
`plainTextpassword`	string	`Pa55w0rd`	A sample password to test in clear text.
`mode`	PasswordCheckMode {Enum}	See table below.	The mode to check the password against.

Accepted password check modes:

0	None	
1	Local	Check password against local checks.
2	Shared	Check if the password is a shared password with other accounts internally
3	Remote	Check if the password is a breached password from breach lists

3.3.5 Function: CreateRealm

The `CreateRealm` function creates a Realm for containing MyID External user accounts.

Parameter	Type	Value Sample	Description
`realm`	string	`Realm01` `ParentRealm01,ChildRealm01`	A Realm hierarchy. This can be a single name or comma separated list of parents and children; it creates the entire hierarchy. The names should contain only alphanumeric, dot, and underscore characters.

3.3.6 Function: CreateUser

The `CreateUser` function creates an Enabled MyID user account using default values. It does not configure the account for any authentication types.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account Domain\Realm and account name to store in the directory.

3.3.7 Function: CreateUserExternal

The `CreateUserExternal` function creates an External MyID user account using default values and allows you to update common properties during the creation.

Parameter	Type	Value Sample	Description
`realm`	string	`Realm01`	The Realm in which to create the External user account.
`accountName`	string	`AndyP`	The user account name to store in the directory.
`upn`	string	`AndyP@Realm01`	A UPN logon name for the user account.
`firstName`	string	`Andy`	User First name.
`lastName`	string	`Pearson`	User Last name.
`mailAddress`	string	`andyp@sample.com`	A valid email address for the user.
`mobilePhone`	string	`0123 456 789`	A mobile phone number for the user.
`enableFido`	boolean	`TRUE`	Enable FIDO authentication for the user.
`enablePush`	boolean	`TRUE`	Enable Push authentication for the user.
`requireBiometricSeedInApp`	boolean	`TRUE`	Require the user to provide a biometric seed in the mobile app.
`enableOtc`	boolean	`TRUE`	Enable OTC authentication for the user.
`pin`	string	`123456789`	The OTC PIN.
`userMustChangePinAtNextLogon`	boolean	`TRUE`	Require the user to change their OTC PIN at the next logon.
`entraId`	string	`e17fa4f0-c066-46d8-b040-75a37af8e2d1`	The EntraID for the user account.

3.3.8 Function: DeleteRealm

The `DeleteRealm` function deletes a MyID Realm. The Realm must be empty before it can be deleted.

Parameter	Type	Value Sample	Description
`realm`	string	`Realm01` `ParentRealm01,ChildRealm01`	A Realm hierarchy. This can be a single name, or a comma separated list of parents and children; it attempts to delete the last child, but fails if the realm is not empty.

3.3.9 Function: DeleteUser

The `DeleteUser` function deletes a MyID user account from the directory, including all its settings and attributes. When using Active Directory, it does *not* delete the actual Active Directory user account; only the MyID metadata is removed off of the account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.10 Function: DisableEmergencyOverride

The `DisableEmergencyOverride` function disables the Emergency Override setting on a user account. If Emergency Override is not enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.11 Function: DisablePinGrid

The `DisablePinGrid` function disables the use of a Grid pattern on a user account. If Grid is not enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.12 Function: DisablePinPass

The `DisablePinPass` function disables the use of PINpass on a user account. If PINpass is not enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.13 Function: DisablePinPhrase

The `DisablePinPhrase` function disables the use of PINphrase on a user account. If PINphrase is not enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.14 Function: DisablePush

The `DisablePush` function disables the use of Push on a user account. If Push is not enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.15 Function: DomainExists

The `DomainExists` function checks if the specified Active Directory domain exists in the directory.

The function returns a `True` or `False` Boolean value.

Parameter	Type	Value Sample	Description
`domain`	string	`mydomain.com`	An Active Directory domain or external realm name that may or may not exist.

3.3.16 Function: EnableEmergencyOverride

The `EnableEmergencyOverride` function enables the Emergency Override functionality on a user account. If `UseADpassword` is set to `True`, the `EmergencyOverrideCode` value is ignored.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.
`EmergencyOverride ExpiryMethod`	Enum	`nLogins`	The method used to expire the use of Emergency Override. This can be a combination of number of logins (`nLogins`), a period of time (`TimePeriod`) or both (`nLoginsorTimePeriod`).
`UseADpassword`	Boolean	`False`	Set the account to use the Active Directory password instead of a set code/password. **Note:** This feature is available only in Active Directory environments.
`EmergencyOverride Code`	string	`5ecr3t`	The code/password to be used for Emergency Override.

3.3.17 Function: EnableFidoCredential

The `EnableFidoCredential` function enables or disables a Fido credential for a user account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.
`credentialId`	string	`INbV8ZJKjb6oa`	The `credentialId` of the Fido token.

3.3.18 Function: EnablePinGrid

The `EnablePinGrid` function enables the use of a Grid pattern on a user account. If Grid is enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.19 Function: EnablePinPass

The `EnablePinPass` function enables the use of PINpass on a user account. If PINpass is enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.20 Function: EnablePinPhrase

The `EnablePinPhrase` function enables the use of PINphrase on a user account. If PINphrase is enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.21 Function: EnablePush

The `EnablePush` function enables the use of Push on a user account. If Push is enabled on the account, this function has no effect.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.22 Function: GenerateNewUserSeed

The `GenerateNewUserSeed` function generates a new 256-bit seed on a user account. If a seed already exists, it is replaced with the new one.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.

3.3.23 Function: GetAuthlogicsUsers

The `GetAuthlogicsUsers` function returns a list of MyID provisioned users for the specified realm.

Parameter	Type	Value Sample	Description
`realm`	string	mydomain.com	An Active Directory domain or external realm name that may or may not exist.

3.3.24 Function: GetPasswordPolicySettings

The `GetPasswordPolicySettings` function returns a comma separated return of all Password Policies followed by the delimiter of a colon : and the setting value; that is, `True` or `False` or the numeric value for the control.

Parameter	Type	Value Sample	Description
n/a			

For example:

```
AllowUsername:False,DisableSharedPasswordProtection:False,DisableCloudPass
wordBlacklist:False,DisableLocalPasswordBlacklist:False,DisallowMonthAndDa
y:False,DisallowSpaces:False,MaxAllowedUsernameCharacters:0,MaxLength:127,
MaxRepeatingChars:8,MaxSequentialChars:3,MaxSequentialKeyBoardChars:0,Enab
lePasswordPolicy:True,MinLength:8,MinLowerCaseChars:0,MinNumericChars:0,Mi
nSpecialChars:0,MinUnicodeChars:0,MinUpperCaseChars:0
```

3.3.25 Function: GetDomains

The `GetDomains` function retrieves a string array of Active Directory domains that exist in the directory. This is a read-only function.

Parameter	Type	Value Sample	Description
n/a			

3.3.26 Function: GetFullProvisionedUsers

The `GetFullProvisionedUsers` function retrieves all users in a specified realm that have been fully provisioned; that is, they do not require knowledge factors to be changed for a specified MFA technology.

Parameter	Type	Value Sample	Description
`realm`	string	`Realm01`	The realm to query for users.
`apl`	Int32	`256`	The enum for the provisioned MFA technology.
`format`	string	`UPN`	Specify the format of the user account.

APL Technology enumeration:

`0`	None
`1`	PinGrid
`2`	PinPhrase
`4`	PinPass
`8`	YubiKey OTP
`16`	Push
`32`	FIDO
`256`	All

Format response options:

`Domain`	NETBIOS user name: `domain\username`
`UPN`	User's UPN: `username@domain`
`Email`	User's email address.

3.3.27 Function: GetOathUrl

The `GetOathUrl` function retrieves an OathAuthenticator URL for a user account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.

3.3.28 Function: GetRealms

The `GetRealms` function retrieves a string array of Realms that exist in the directory. A GET call or POST with an empty search returns all realms.

Parameter	Type	Value Sample	Description
`search`	string	`Realm01`	An optional search string.

3.3.29 Function: GetRealmsAt

The `GetRealmsAt` function retrieves a string array of sub realms without nesting; a GET call or POST with an empty search returns all root level realms.

Parameter	Type	Value Sample	Description
baseRealm	string	Realm01	The base realm to retrieve.

3.3.30 Function: GetRealmsInfo

The `GetRealmsInfo` function retrieves a string array of realms. A GET call or POST with an empty search returns all root level realms and their information.

Parameter	Type	Value Sample	Description
`baseRealm`	string	`Realm01`	The base realm to retrieve.
`daysSinceCreation`	Int32	`30`	Return only the realms that have been created more than the specified number of days ago.

3.3.31 Function: GetSecurityKey

The `GetSecurityKey` function retrieves a security key identified by a particular GUID.

Parameter	Type	Value Sample	Description
`aaguid`	string	`894cedf4-a9e3…`	The GUID of the security key to retrieve.

3.3.32 Function: GetServerVersion

The `GetServerVersion` function retrieves a string displaying the installed MyID version. This is a read-only function.

Parameter	Type	Value Sample	Description
n/a			

3.3.33 Function: GetMinimumClientVersion

The `GetMinimumClientVersion` function retrieves a string displaying the minimum client version required to use this version of the Authentication Server. This is a read-only function.

Parameter	Type	Value Sample	Description
n/a			

3.3.34 Function: GetSettingsProperty

The `GetSettingsProperty` function retrieves the values of various global settings properties. This is a read-only function; to set the value of a property, use the `SetSettingsProperty` function.

You can query multiple values at once by specifying all the required properties in a CSV string.

Parameter	Type	Value Sample	Description
`names`	string	`SMTPServer1, SMTPPort1`	The property names to access. **Note:** Leaving this value blank returns a list of supported property values.

Valid values for the `names` parameter which do not require authentication (Anonymous):

```
SchemaVersion, ToleranceLevel, TolerancePeriod, LockoutDuration,
LockoutThreshold, LockoutReset, SspAllowResetAdPassword,
SspUnlockMasterAccountOnPasswordReset, SspAllowUpdateMobilePhoneNumber,
SspAllowTokenDeviceChange, SspUrl, SspPasswordReset, AppLogoUrl,
AppLogoDescription, AppUseBiometrics, AppOtpCopyPaste,
```

AppTransactionValidation, AuthlogicsServerCertificate, AuthlogicsIdpSigningCertificate, AuthlogicsServerTrustedRootCertificate, ADUsernameCustomAttribute, RandomiseAdPasswordPeriod, RandomiseAdPasswordEnforced, GUIDAdministrators, GUIDOperators, GUIDAdministrators, GUIDServers, GUIDRadius, GUIDADPassthrough, SMTPServer1, SMTPServer2, SMTPPort1, SMTPPort2, SMTPFromAddress, SMTPEnableSSL, SMTPUseWindowsCredentials, SMSEnabled, SMSSendLimit, SMSDefaultCountryCode, RealTimeTokenLifespan, AllowEmergencyOverride, AllowTemporaryAccessCode, MaxOverrideTime, MaxOverrideUses, PasswordVaultEnabled, RequirePrivateMobile, EmailDomains, AccountBreachedAction, AccountSharedAction, AccountMfaDormantAction, AccountAdDormantAction, AlertAdDormant, AlertAdPasswordExpires, AlertAdPasswordExpiresDays, AlertBreachPassword, AlertDormantAdDays, AlertDormantMfaDays, AlertLicenceEvents, AlertMfaAccountLockedOut, AlertMfaDeviceChangeOnAccount, AlertMfaDormant, AlertSharedPassword, PsmScheduleStart, PsmScheduleRepeatCycle, PsmScheduleRecur, PinGridMatrixMinNumberOfSquares, PinGridMatrixTheme, PinGridMIPHistory, PinGridMIPMaxAge, PinGridMIPMinLength, PinGridMIPMinAge, PinGridMIPComplexity, PinGridMIPMaxAdjacencies, PinGridMIPMaxCellRepeats, PinGridHASHLevel, PinGridMessagePrefix, PinGridMIPMinNumberOfQuadrants, PinGridMatrixFontSize, PinGridMatrixColourQ1, PinGridMatrixColourQ2, PinGridMatrixColourQ3, PinGridMatrixColourQ4, PinGridMatrixBitmapSizeDPI, PinGridMatrixHTMLEmail, PinPhraseMinNumberOfQuestions, PinPhraseMinAnswerLength, PinPhraseQuestions, PinPhraseMessagePrefix, PinPhraseUseMultipleQuestionsPerLogin, PinPassMessagePrefix, PinPassMinLength, PinPassPINMinLength, PinPassPINPosition, PinPassPINEnforced, RADIUSFilterEnabled, ADPassthroughEnabled, YubiKeyOtpPinMinLength, YubiKeyOtpPinEnforced, YubiKeyOtpEnabled, YubiKeyOtpOnlineEnabled.

Valid values for the `names` parameter which require Admin rights:

`SMTPUsername, DirectoryID`

3.3.35 Function: GetUserProperty

The `GetUserProperty` function returns the properties for a user. This is a read-only function; to set the value of a property, use the `SetUserProperty` function.

You can query multiple values at once by specifying all the required properties in a comma-separated string.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`Names`	string	`FirstName,LastName`	The property names to access. **Note:** Leaving this value blank returns a list of supported property values.

Valid values for the `Names` parameter that do not require authentication (Anonymous):

`AccountGuid, PushThrottled, HasPushDevices, AccountName, UPN, DirectPath, Devices, FirstName, LastName, Realm, Exists, ExistsAD, PsmOnly, ExternalUser, Enabled, APL, AccountExpiresAD, ValidFrom, ValidTo, PasswordExistsInVault, Description, RequireBiometrics, PinGridEnabled, PinGridProvisioned, PinGridMIPMustChange, PinGridMIPNeverExpires, PinGridRequire2FA, PinGridEnable2FA, PinGridDelivery, PinGridQueueType, PinPhraseEnabled, PinPhraseProvisioned, PinPhraseAnswersMustChange, PinPhraseRequire2FA, PinPhraseEnable2FA, PinPhraseDelivery, PinPhraseQueueType, PinPassEnabled, PinPassProvisioned, PinPassPINMustChange, PinPassDelivery, PinPassQueueType, PinPassTokensPerMessage, PushEnabled, LastLogonAd, LastLogonPinGrid, LastLogonPinPhrase, LastLogonPinPass, LastLogonYubiKeyOtp, YubiKeyOtpEnabled, YubiKeyOtpProvisioned, YubiKeyOtpPinMustChange.`

Valid values for the `Names` parameter that require Admin or Operator rights:

`PasswordLengthAD, LockedOut, EmergencyOverrideEnabled, PinGridMIPCreationDate, PinGridTokenLifespan, PinPhraseCodeLength, PinPhraseTokenLifespan, PinPassTokenLifespan, PinPassPIN, YubiKeyOtpPin.`

Valid values for the `Names` parameter that require Admin or Operator rights, or can be accessed by the actual user:

`UserCannotChangePasswordAD, PasswordNeverExpiresAD, PasswordLastSetAD, PasswordExpiryDateAD, PasswordExpiryZone, MobilePrivate, MobileNumber, MailAddress, LastLogin, BadLogins, PinGridMIPExpiryDate, PinGridMIPdaysSinceLastChanged, PinGridMatrixNumberOfSquares, PinPhraseAnswers, PinPassCodeLength, PinPassPINisADpassword, TokenIDs, YubiKeyOtpPinIsADpassword.`

The following table lists all the accepted device types for the `TokenIDs` property:

0	Unspecified
1	WindowsDesktop
2	WindowsStore
3	WindowsPhone
4	Android
5	AppleiOS
6	AppleMacOS
7	BlackBerry
8	YubiKey
9	OathAuthenticator
10	SecurityKey
11	SyncedPasskey

3.3.36 Function: IsRealmEmpty

The `IsRealmEmpty` function returns `False` if the specified realm contains sub realms or users.

Parameter	Type	Value Sample	Description
`realm`	string	`Realm01`	The realm to you want to check.

3.3.37 Function: PasswordHashExists

The `PasswordHashExists` function checks whether or not a MD4 hash password exists in the MyID Password breach database.

This is a read-only function that returns a `True` or `False` Boolean value.

Parameter	Type	Value Sample	Description
`md4Hash`	string	`ABC1543212BCD..`	The MD 4 hash of a clear text password.
`dnsDomain`	string	`Authlogics.com`	The DNS domain name of the calling domain.

3.3.38 Function: PinGridChangeMIP

The `PinGridChangeMIP` function sets a new Grid pattern on the user account.

Important: The pattern is specified in MIP comma-separated notation of grid positions and is stored as a hash; once written, it *cannot* be retrieved in plain text.

Note: Any authenticated user can call this function as long as they are attempting to update their own MIP, otherwise MyID Admin rights are required.

If an Administrator or Operator calls this function, the Pattern Complexity checks are *not* performed. If the user's context initiates this call, the Pattern Complexity checks are enforced, and users are *not* allowed to select non-complex patterns.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`gridSize`	string	`6`	The X or Y size of the grid to use for the new pattern. 6 or 8 are the only accepted values.
`MIP`	string	`23,29,35,24,30,36`	A comma-separated list of numbers denoting the positions in a grid for the pattern.

`currentMIP`	string	`1,2,3,9,8,7`
		or:
		`ABC1243E...`
			Provide the current user's pattern or the hash of the user's current pattern.

3.3.39 Function: PinGridGenerateMIP

The `PinGridGenerateMIP` function creates a new random pattern in MIP comma-separated notation. You can generate either a simple or a complex pattern.

The function returns the new MIP as a string and is not written to a user account. You can use the new MIP on a user account within the `PinGridChangeMIP`, `PinGridProvision` and `SendHTMLPinGridLetter` functions.

Parameter	Type	Value Sample	Description
`gridSize`	integer	`6`	The X or Y size of the grid to use for the new pattern. 6 or 8 are the only accepted values.
`complexPattern`	Boolean	`False`	To generate a complex pattern set this value to `True`. For a simple pattern set it to `False`.

3.3.40 Function: PinGridProvision

The `PinGridProvision` function provisions a user account for a user with a Grid pattern. You must do this at least once to allow Grid patterns to be used with the account. The inputs are similar to the `PinGridChangeMIP` function; however, this function does more than just setting the MIP.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`gridSize`	integer	`6`	The X or Y size of the grid to use for the new pattern. 6 or 8 are the only accepted values.
`MIP`	string	`23,29,35,24,30,36`	A comma-separated list of numbers denoting the positions in a grid for the pattern.
`OverrideRestrictions`	Boolean	`False`	Override the pattern complexity restriction checks when setting the pattern. It is not recommended to override the built-in complexity settings as this could lead to lower security.

3.3.41 Function: PinPassChangePin

The `PinPassChangePin` function allows you to change the PINpass PIN code.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`pin`	integer	`12345`	The new PIN code to be used for PINpass.
`currentPin`	string	`54321`	The current PIN code used for PINpass. This is used to authorize the change to the new PIN.

3.3.42 Function: PinPassGeneratePIN

The `PinPassGeneratePIN` function generates a random PIN which complies to the PINpass policy.

Parameter	Type	Value Sample	Description
n/a			

3.3.43 Function: PinPassProvision

The `PinPassProvision` function provisions a user account for a user with PINpass. You must do this at least once to allow PINpass to be used with the account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`PIN`	string	`7651`	A PIN (or password) to be used as the knowledge component for authentication. If no PIN is specified, a random PIN is generated and saved to the account.
`PINisADpassword`	Boolean	False	Use the Active Directory password instead of a PIN. If this value is set to True then the PIN value, if specified, will not be used. Note: This option is only available in an Active Directory Environment.
`OTPcodeLength`	integer	6	The length of the OTP code which will be sent to the user. As per OATH RFC specifications, accepted values are 6, 7 and 8 only.

3.3.44 Function: PinPhraseAddQuestion

The `PinPhraseAddQuestion` function adds a new question to the list of questions for which users can provide answers.

Parameter	Type	Value Sample	Description
`question`	string	`Your favourite color`	A string containing the actual question to add.

3.3.45 Function: PinPhraseEditQuestion

The `PinPhraseEditQuestion` function alters an existing question in the list of questions for which users can provide answers.

Warning: Use this function only for minor changes, as existing answers are still matched to the question. For brand new questions you are recommended to delete the old question and add a new question.

Parameter	Type	Value Sample	Description
`oldQuestion`	string	`Your best colour`	A string containing the old actual question.
`newQuestion`	string	`Your favourite colour`	A string containing the new actual question.

3.3.46 Function: PinPhraseGenerateCodeword

The `PinPhraseGenerateCodeword` function returns a random word from the PINphrase dictionary file.

Parameter	Type	Value Sample	Description
n/a			

3.3.47 Function: PinPhraseProvision

The `PinPhraseProvision` function provisions a user account for a user with PINphrase. You must do this at least once to allow PINphrase to be used with the account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`codeWord`	string	`England`	A string to be used as the answer to the first question; the default question is "your code word". If no answer is specified, a random word is selected from a dictionary file and saved to the account as the answer to the first question.
`OTPcodeLength`	integer	`3`	The number of letters from the answer are requested from the user to create their OTP code. Recommended values are between `3` and `5`.

3.3.48 Function: PinPhraseRemoveQuestion

The `PinPhraseRemoveQuestion` function removes an existing question from the list of questions for which users can provide answers. Any existing answers to the removed question are removed from the user account only when the account is next updated. A bulk answer delete is *not* triggered; however, the answer is ignored during subsequent processing.

Parameter	Type	Value Sample	Description
`question`	string	`Your favourite color`	A string containing the actual question.

3.3.49 Function: RealmExists

The `RealmExists` function checks if the specified Realm exists in the directory.

The function returns a `True` or `False` Boolean value.

Parameter	Type	Value Sample	Description
`realm`	string	`realm.com` `parentRealm,realm.com`	A Realm hierarchy which may or may not exist. This can be a single name or comma separated list of parents and children and will return true only if the entire hierarchy exists.

3.3.50 Function: RemoveFidoCredential

The `RemoveFidoCredential` function removes a Fido credential from the user account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`credential`	FidoCredential	See section *3.4, Data types*.	A representation of a FidoCredential.

3.3.51 Function: RenameRealm

The `RenameRealm` function renames an existing Realm in the directory.

Parameter	Type	Value Sample	Description
`oldRealm`	string	`oldrealm.com` `parentRealm,oldrealm.com`	A Realm hierarchy. This can be a single name or comma separated list of parents and children; it renames the last child.
`newRealmName`	string	`newrealm.com`	The new Realm name.

3.3.52 Function: RenameUser

The `RenameUser` function renames an existing user account in the directory. Renaming a user account is supported only with External Users, and not with Active Directory user accounts.

Parameter	Type	Value Sample	Description
`oldAccountName`	string	`oldname`	The existing user account name to be renamed.
`newAccountName`	string	`newname`	The new user account name.

3.3.53 Function: SendHTMLPinGridLetter

The `SendHTMLPinGridLetter` function sends an HTML formatted "PINgrid welcome letter" to the user through email from the MyID server providing details of how to use the account. The email is sent to the email address specified on the user account only.

You must specify the MIP here, as it cannot be retrieved from the user account in plain text. Call this function immediately after calling the `PinGridChangeMIP` or `PinGridProvision` functions while the MIP is still in memory as plain text.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.
`templateName`	string	`PINgridUserTemplate`	The name of the HTML template file to be used as a base for the email. You can specify the file name with or without the HTML file extension. The file must be available in the MyID MFA Program Files folder.
`MIP`	string	`23,29,35,24,30,36`	A comma separated list of numbers denoting the positions in a grid for the pattern.

3.3.54 Function: SendHTMLPinPassLetter

The `SendHTMLPinPassLetter` functions sends an HTML formatted "PINpass welcome letter" to the user through email from the MyID server providing details of how to use the account. The email is sent to the email address specified on the user account only.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name that is resolvable in the directory.
`templateName`	string	`PINpassUserTemplate`	The name of the HTML template file to be used as a base for the email. The file name can be specified with or without the HTML file extension. The file must be available in the MyID MFA Program Files folder.

3.3.55 Function: SendHTMLPinPhraseLetter

The `SendHTMLPinPhraseLetter`function sends an HTML formatted "PINphrase welcome letter" to the user through email from the MyID server providing details of how to use the account. The email is sent to the email address specified on the user account only.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`templateName`	string	`PINphraseUserTemplate`	The name of the HTML template file to be used as a base for the email. The file name can be specified with or without the HTML file extension. The file must be available in the MyID MFA Program Files folder.

3.3.56 Function: SendHTMLPushLetter

The `SendHTMLPushLetter` function sends an HTML formatted "Push MFA welcome letter" to the user through email from the MyID server providing details of how to use the account. The email is sent to the email address specified on the user account only.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`templateName`	string	`PINphraseUserTemplate`	The name of the HTML template file to be used as a base for the email. The file name can be specified with or without the HTML file extension. The file must be available in the MyID MFA Program Files folder.

3.3.57 Function: SendRealtimeToken

The `SendRealtimeToken` function sends a server-generated real-time token to the user based on the configured authentication provider only if they are configured for Real-Time token use. The token may be sent through email or text messaging depending on the user settings.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.

3.3.58 Function: SendRealtimeTokenbyProduct

The `SendRealtimeTokenbyProduct` function sends a server-generated real-time token to the user only if they are configured for Real-Time token use. The token may be sent through email or text messaging depending on the user settings.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`Product`	enum	`PinGrid`	Specify the type of authentication technology to send a token for.

Valid values for the product parameter include:

`PinGrid, PinPhrase, PinPass`

3.3.59 Function: SendToken

The `SendToken` function sends a server-generated token to the user based on the configured authentication provider and queue type (Real-Time / Pre-Send). The token may be sent through email or text messaging depending on the user settings.

Ideally, call this function when a user is configured to use a pre-send token and the initial tokens need to be delivered.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.

3.3.60 Function: SetADpassword

The `SetADpassword` function updates the Active Directory Domain account password for a user account. The actual user or a MyID Administrator can call this function.

This function results in a password reset event on the Active Directory and not a password change event.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`ADpassword`	string	`Pa55w0rd`	The new Active Directory password to be written to the Windows Domain.

3.3.61 Function: SetSettingsProperty

The `SetSettingsProperty` function commits the values of various global settings properties. This is a write-only function; to get the value of a property, use the `GetSettingsProperty` function.

You can set multiple values at the same time by specifying all the required properties and values in the corresponding CSV string.

Parameter	Type	Value Sample	Description
`Names`	string	`SMTPServer1, SMTPPort1`	The property names to access. Note: Leaving this value blank returns a list of supported property values.
`Values`	string	`mail.server.com, 25`	The values to write to the properties specified in `names`.

Valid values for the `Names` parameter that require Admin rights:

```
ToleranceLevel, TolerancePeriod, LockoutDuration, LockoutThreshold,
LockoutReset, SspAllowResetAdPassword,
SspUnlockMasterAccountOnPasswordReset, SspAllowUpdateMobilePhoneNumber,
SspAllowTokenDeviceChange, SspUrl, SspDevicelessMfa, SspPasswordReset,
SspLogonTechnology, WmpDevicelessMfa, WmpLogonTechnology, AppLogoUrl,
AppLogoDescription, AppEnableBiometrics, AppOtpCopyPaste,
AppTransactionValidation, AuthlogicsServerCertificate,
AuthlogicsIdpSigningCertificate, AuthlogicsServerTrustedRootCertificate,
ADUsernameCustomAttribute, RandomiseAdPasswordPeriod,
RandomiseAdPasswordEnforced, GUIDAdministrators, GUIDOperators,
GUIDAdministrators, GUIDServers, GUIDRadius, GUIDADPassthrough,
SMTPServer1, SMTPServer2, SMTPPort1, SMTPPort2, SMTPFromAddress,
SMTPEnableSSL, SMTPUseWindowsCredentials, SMTPUsername, SMTPPassword,
SMSSendLimit, SMSDefaultCountryCode, RealTimeTokenLifespan,
AllowEmergencyOverride, AllowTemporaryAccessCode, MaxOverrideTime,
MaxOverrideUses, PasswordVaultEnabled, RequirePrivateMobile, EmailDomains,
AccountBreachedAction, AccountSharedAction, AccountMfaDormantAction,
AccountAdDormantAction, AlertBreachPassword, AlertDormantAdDays,
AlertDormantMfaDays, AlertLicenceEvents, AlertMfaAccountLockedOut,
AlertMfaDeviceChangeOnAccount, AlertMfaDormant, AlertSharedPassword,
PsmScheduleRepeatCycle, PsmScheduleRecur, PinGridMatrixMinNumberOfSquares,
PinGridMatrixTheme, PinGridMIPHistory, PinGridMIPMaxAge,
PinGridMIPMinLength, PinGridMIPMinAge, PinGridMIPComplexity,
PinGridMIPMaxAdjacencies, PinGridMIPMaxCellRepeats, PinGridHASHLevel,
PinGridMessagePrefix, PinGridMatrixFontSize, PinGridMatrixColourQ1,
PinGridMatrixColourQ2, PinGridMatrixColourQ3, PinGridMatrixColourQ4,
PinGridMatrixBitmapSizeDPI, PinGridMatrixHTMLEmail,
PinPhraseMinNumberOfQuestions, PinPhraseMinAnswerLength,
PinPhraseMessagePrefix, PinPhraseUseMultipleQuestionsPerLogin,
PinPassMessagePrefix, PinPassMinLength, PinPassPINMinLength,
```

PinPassPINPosition, PinPassPINEnforced, RADIUSFilterEnabled, ADPassthroughEnabled , YubiKeyOtpPinMinLength, YubiKeyOtpPinEnforced, YubiKeyOtpEnabled, YubiKeyOtpOnlineEnabled.

3.3.62 Function: SetUserProperty

The SetUserProperty function commits the values of various user account properties. This is a write-only function; to get the value of a property, use the GetUserProperty function.

You can set multiple values at the same time by specifying all the required properties and values in the corresponding CSV string.

Parameter	Type	Value Sample	Description
accountName	string	AndyP domain\andyp	A user account name which is resolvable in the directory.
Names	string	FirstName,LastName	The property names to access. **Note:** Leaving this value blank returns a list of supported property values.
Values	string	John,Smith	The values to write to the properties specified in names.

Valid values for the Names parameter that require Admin or Operator rights:

LockedOut, Enabled, UserCannotChangePasswordAD, PasswordNeverExpiresAD, PasswordLastSetAD, MailAddress, ValidFrom, ValidTo, RequireBiometrics, PinGridMIPMustChange, PinGridMIPNeverExpires, PinGridRequire2FA, PinGridEnable2FA, PinGridTokenLifespan, PinGridDelivery, PinGridQueueType, PinPhraseAnswersMustChange, PinPhraseCodeLength, PinPhraseRequire2FA, PinPhraseEnable2FA, PinPhraseTokenLifespan, PinPhraseDelivery, PinPhraseQueueType, PinPassPINMustChange, PinPassCodeLength, PinPassPINisADpassword, PinPassTokenLifespan, PinPassDelivery, PinPassQueueType, PinPassTokensPerMessage, YubiKeyOtpPinMustChange, YubiKeyOtpPinisADpassword.

Valid values for the Names parameter that require Admin or Operator rights, or can be accessed by the actual user:

MobileNumber, MobilePrivate, PinGridMIP, PinPassPIN, PinPhraseAnswers, YubiKeyOtpPin.

Valid values for the Names parameter which require Admin rights:

FirstName, LastName, Description, UPN.

Valid values for the Names parameter which require Admin, Operator or Enterprise Domain Controller rights:

PasswordLengthAD.

Note: You can set the value of LockedOut to False to unlock an account; however, you cannot set LockedOut to True manually.

3.3.63 Function: SyncDevice

The `SyncDevice` function sets a device to the status that requires it to synchronize its settings.

Parameter	Type	Value Sample	Description
`accountName`	string	AndyP domain\andyp	A user account name which is resolvable in the directory.
`deviceID`	string	2419216713157481	The device ID of as indicated by the Authenticator App.

3.3.64 Function: TokenHardwareAdd

The `TokenHardwareAdd` function adds hardware token details to a user account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`Enabled`	Boolean	`True`	Set the enabled status of the new token. Typically set to True for a new device.
`tokenType`	enum	`WindowsPhone`	The platform on which the soft token is installed.
`tokenID`	string	`2419216713157481`	The hardware / device ID of the hardware token as indicated by the Authenticator App.

Valid values for the `tokenType` parameter include:

`WindowsDesktop, WindowsStore, WindowsPhone, Android, AppleiOS, AppleMacOS, BlackBerry, Yubikey.`

3.3.65 Function: TokenHardwareEnabled

The `TokenHardwareEnabled` function allows you to enable or disable individual tokens on a user account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`Enabled`	Boolean	`True`	The new enabled status of the token.
`tokenID`	string	`2419216713157481`	The hardware / device ID of the token as indicated by the soft token application.

3.3.66 Function: TokenHardwareRemove

The `TokenHardwareRemove` function removes individual tokens from a user account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`tokenID`	string	`2419216713157481`	The hardware / device ID of the token as indicated by the soft token application.

3.3.67 Function: UpdateFidoCredential

The `UpdateFidoCredential` function updates a Fido credential for the user account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`credential`	FidoCredential	See section *3.4, Data types*.	A representation of a FidoCredential

3.3.68 Function: UpdateLicenceFile

The `UpdateLicenceFile` function updates the license file information on the MyID MFA Server with the XML data provided. The XML must be baes64/URL encoded before submitting to the Web API. Online and offline license files are supported, and online licenses are activated.

Parameter	Type	Value Sample	Description
`base64licenceXml`	string		A base64 encoded license file.

3.3.69 Function: UpdateLicenceKey

The `UpdateLicenceKey` function updates the license key information on the MyID MFA Server with the license key provided. Only online licenses are supported and are activated.

Parameter	Type	Value Sample	Description
`licenceKey`	string		A license key.

3.3.70 Function: ValidateAdPassword

The `ValidateAdPassword` function tests a user's Active Directory password against the supplied user's `accountname`.

The function returns a `True` or `False` Boolean value.

Parameter	Type	Value Sample	Description

`accountName`	string	`AndyP`
		`domain\andyp`
		`andyp@sample.com`
			An Active Directory user account name against which to test the password.

`password`	string	`Pa55w0rd`	The user's Active Directory password to test.

3.3.71 Function: VerifyEmergencyAccess

The `VerifyEmergencyAccess` function verifies an emergency access pass code is valid.

Parameter	Type	Value Sample	Description

`accountName`	string	`AndyP`
		`domain\andyp`
		`andyp@sample.com`
			A user account name which is resolvable in the directory.

`passcode`	string	`123456`	A passcode to authenticate the account.

3.3.72 Function: VerifyTransaction

The `VerifyTransaction` function processes a three-factor logon request for a user account.

Parameter	Type	Value Sample	Description

`accountName`	string	`AndyP`
		`domain\andyp`
		`andyp@sample.com`
			A user account name which is resolvable in the directory.

`passcode`	string	`123456`	A passcode to authenticate the account.
`transactionData`	string	`Abcd1234`	Transaction string being used as part of the signing process.

Much like the `AuthenticateUser` function, this function returns an integer code indicating the result of the `VerifyTransaction` request.

0	Access Granted.	Credentials are valid.
1	Access Denied.	Account name not found.
2	Access Denied.	Invalid passcode.
5	Access Denied.	Account Expired.
7	Access Denied.	Account disabled, locked out, or not valid at this time.
8	Access Denied.	Authentication not available due to a licensing issue.
13	Access Granted.	A pattern change is required.
111	Access Denied.	Directory related error.

3.3.73 Function: YubiKeyOtpChangePin

The `YubiKeyOtpChangePin` function changes the PIN on a YubiKey.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`pin`	string	`7651`	A PIN (or password) to be used as the knowledge component for authentication.
`currentPin`	string	`5976`	The current PIN for the YubiKey

3.3.74 Function: YubiKeyOtpProvision

The `YubiKeyOtpProvision` function provisions a user account for a user with YubiKey OTP. You must do this at least once to allow YubiKey OTP to be used with the account.

Parameter	Type	Value Sample	Description
`accountName`	string	`AndyP` `domain\andyp`	A user account name which is resolvable in the directory.
`PIN`	string	`7651`	A PIN (or password) to be used as the knowledge component for authentication. If you do not specify a PIN, a random PIN is generated and saved to the account.
`PINisADpassword`	Boolean	`False`	Use the Active Directory password instead of a PIN. If this value is set to `True`, the PIN value, if specified, is not used. **Note:** This option is available only in an Active Directory environment.

3.4 Data types

3.4.1 FidoCredential


```
{
  "aaGuid": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "attestationClientDataJson": "string",
  "attestationObject": "string",
  "be": true,
  "bs": true,
  "credentialType": 0,
  "credType": "string",
  "descriptor": {
    "id": "string",
    "transports": [
      0
    ],
    "type": 0
  },
  "devicePublicKeys": [
    "string"
  ],
  "enabled": true,
  "encryptedPassword": "string",
  "hmacSalt": "string",
  "id": "string",
  "lastUsed": "2024-02-13T10:33:37.599Z",
  "name": "string",
  "publicKey": "string",
  "regDate": "2024-02-13T10:33:37.599Z",
  "signCount": 0,
  "transports": [
    0
  ],
  "type": 0,
  "userHandle": "string",
  "userId": "string"
}
```

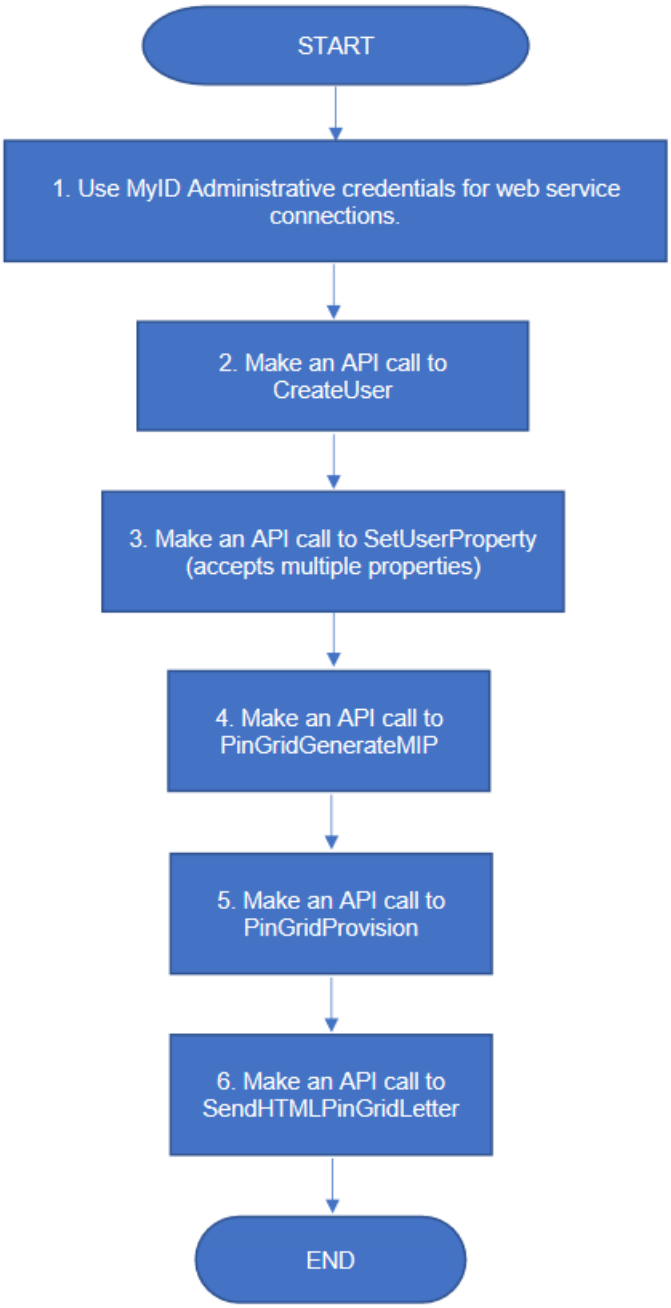
3.5 Example: programmatically creating a user account

The following example uses a user's email address as the basis for creating a new account. It uses the email address suffix as a realm name in which to create the account.

Next, a random Grid pattern (MIP) is generated and used to provision the user for Grid patterns.

Finally, the user is sent a welcome email containing their new pattern.

3.5.1 Process flow



3.5.2 Explanation

Note: All API requests expect their relevant parameters to be passed in the request for them to function correctly – refer to the API reference for this if necessary.

1. Retrieve the credentials using the method described before, and then assign them accordingly.

   If you are using a service reference, you can assign this to this object instance credentials, and it applies for each request. If you are using a standard web request, this is used for each request, setting them with the `NetworkCredential` object as previously described.

2. Make a request to `CreateUser` to create the user account.

3. Set any additional user properties as needed by calling the `SetUserProperty` function.

   This function accepts a comma-separated list of properties, and a comma-separated list of values, so you can apply many attributes to a user at any one time.

4. When calling the `PinGridGenerateMIP` function, you are recommended to store the result in a variable.

   You can then use this in the next API call to provision the user. You can make the MIP as complicated as you need it to be, and this depends on your requirements for pattern complexity.

5. Call the `PinGridProvision` function to provision the user for a Grid pattern.

   Use the stored value for the MIP. When adding parameters for the provision API call, the default value for a grid size is 6 – that is, a 6x6 grid.

6. Call the `SendHTMLPinGridLetter` to send the welcome email to the user.

   Use the stored value for the MIP. When adding parameters for sending the HTML letter to the user, the `templateName` string value is the default value of the Grid template; that is, `PINgridUserTemplate`.

   For a PINpass letter, this is instead `PINpassUserTemplate`; for PINphrase, the value `PINphraseUserTemplate`.

3.6 Using the Web Services API with Visual Studio

In addition to using HTTPS GET and POST to communicate with the API, you can use the `Authlogics.ApiClient` .NET library.

The library is .NET Standard, so works with both .NET Framework and .NET Core.

3.6.1 AuthlogicsApiClient

To create the client, you can either explicitly pass it the server URI; for example:

```
new AuthlogicsApiClient(new Uri("https://myserver.com:14443"));
```

Or you can pass in an `IConfiguration` instance; for example:

```
var client = new AuthlogicsApiClient(configuration);
```

Which you can also configure through DI; for example:

```
using IHost host = Host
    .CreateDefaultBuilder(args)
    .ConfigureServices(services => services.AddTransient<IAuthlogicsApiClient,
AuthlogicsApiClient>())
    .Build();

var client = host.Services.GetRequiredService<IAuthlogicsApiClient>();
```

With the `appsettings.json` in the format:

```
{
  "Authlogics": {
    "ApiClient": {
      "ServerUrl": "https://myserver.com:14443"
    }
  }
}
```

You can also optionally set a an `ILogger` instance, which, if set, logs the details of each method call; for example:

```
client.Logger = logger;
```

Once you have created the client instance, you can call `TestConnection` to verify that the client can access the server using the URI provided. This calls an un-authenticated method on the server so it can verify the URI independently of any authentication issues that may arise. There is no return value, the test throws an exception if the connection fails; for example:

```
await client.TestConnection(cancellationToken);
```

3.6.2 Authentication

You require a JWT bearer token to access the API. If you have already generated a bearer token, you can pass it directly into the client; for example:

```
await client.AuthenticateWithBearerToken("...", cancellationToken);
```

You can also authenticate with a client credential along with the scope; for example:

```
await client.AuthenticateWithClientCredentials("my_client_credential", " my_client_
credential_secret", "rest_api", cancellationToken);
```

For more information on creating a client credential, see the *Creating a client credential application* section in the ***MyID Authentication Server Installation and Configuration Guide***.

In each case, the bearer token received or generated is stored in the client and automatically included with every method call that requires authentication.

3.6.3 Example

The following example checks for an existing realm, creates it if it does not exist, then returns all realms.

```
if (!await client.DoesRealmExists("MyTestRealm", cancellationToken))
{
    await client.CreateRealm("MyTestRealm", cancellationToken);
}

var realms = await client.GetRealms("", cancellationToken);
```

3.7 Web Service call changes in version 5.0 from 4.2.1

The following table lists the Web Service API calls that have been added and removed in MyID MFA Version 5.

Added	Removed
AddFidoCredential	WebPortalVerifyOTP
EnableFidoCredential	
GetOathUrl	
RemoveFidoCredential	
UpdateFidoCredential	
VerifyEmergencyAccess	
YubiKeyOtpChangePin	
YubiKeyOtpProvision	

4 Advanced configuration

You can control the advanced configuration options for MyID MFA through the Windows registry or the IIS `web.config` file. The entries described in this chapter are created during the installation of MyID server components; typically, you should change them only if instructed by an Intercede support engineer.

Note: After changing a registry key on the MyID Server, you must restart the IIS components; to do so, open a Windows command prompt with administrative permissions and run the following command:

```
iisreset
```

You can:

- Specify Active Directory Domain Controllers.

  See section *4.1*, *Specifying Active Directory Domain Controllers*.
- Configure the connection timeout for Active Directory.

  See section *4.2*, *Active Directory timing*.
- Configure diagnostics logging.

  See section *4.3*, *Diagnostics logging*.
- View other settings.

  See section *4.4*, *Other settings*.

4.1 Specifying Active Directory Domain Controllers

The MyID Authentication Server automatically locates Domain Controllers as needed. In environments where network segmentation exists, the MyID Authentication Server may not be able to contact all Domain Controllers. This can cause connectivity problems and logon delays.

In these environments, you can specify which Domain Controllers and Global Catalog Servers should be used using registry keys. Each key can contain one or many server names (FQDN recommended) separated by commas.

4.1.1 Specifying Global Catalog Servers

To specify the global catalog server to access from the MyID Authentication Server, set the following registry value:

```
HKLM\SOFTWARE\Authlogics\Authentication Server\DomainGCs
```

By default, this is blank.

Accepted values:

- One or more server names (FQDN recommended), separated by commas.

Used by components: MyID Authentication Server; Management Console

The MyID Authentication Server attempts to connect to each specified global catalog server and then remains connected to the server that responds to LDAP queries the quickest.

Note: This setting disables the auto-detect global catalog servers functionality within MyID.

4.1.2 Specifying Domain Controllers

To specify the Domain Controllers to access from the MyID Authentication Server, set the following registry value:

```
HKLM\SOFTWARE\Authlogics\Authentication Server\DomainDCs
```

By default, this is blank.

Accepted values:

- One or more Domain Controller names (FQDN recommended), separated by commas.

Used by components: MyID Authentication Server; Management Console

The MyID Authentication Server attempts to connect to each specified Domain Controller and then remains connected to the server that responds to LDAP queries the quickest. The MyID Authentication Server initially finds the names of all the Domains in the Forest, and the Domain Controllers in each Domain by querying the Global Catalog. It then maps the results against the Domain Controllers list in the registry to calculate which server to use for each Domain. If a Domain does not have a Domain Controllers specified, one is selected automatically.

Note: This setting disables the auto-detect Domain Controller functionality within MyID.

4.2 Active Directory timing

You can set the following values in the registry:

- Domain access timeout.
- Domain controller refresh.

4.2.1 Domain access timeout

`HKLM\SOFTWARE\Authlogics\Authentication Server\DomainAccessTimeout`

Default value: `60`

Accepted values:

- `0` – disabled, indefinite timeout.
- `1` to `120` – timeout in seconds.

The time taken in seconds before a connection established by a MyID component to a Domain Controller times out.

4.2.2 Domain Controller refresh

`HKLM\SOFTWARE\Authlogics\Authentication Server\DomainControllerRefeshTime`

Default value: `15`

Accepted values:

- `1` to `9999` – timeout in minutes.

The time taken in minutes before a new search is done to locate the quickest Global Catalog Server and Domain Controller.

4.3 Diagnostics logging

You can control the diagnostics logging using the Windows registry.

4.3.1 Enabling logging

To enable or disable diagnostics logging, set the following registry value:

`HKLM\SOFTWARE\Authlogics\Authentication Server\LoggingEnabled`

The default value is `0`.

Accepted values:

- `0` – disabled.
- `1` – enabled.

When you enable this value, various log files are created in the logging folder. Intercede support may request these logs from you.

4.3.2 Setting the logging location

To control the location of the log files, set the following registry value:

`HKLM\SOFTWARE\Authlogics\Authentication Server\LoggingFolder`

The default value is:

`C:\Program Files\Authlogics Authentication Server\Log\`

Accepted values:

- Any valid local folder with the same NTFS permissions as the default folder.

4.3.3 Setting the retention time for rolling logs

Old logs are deleted after a specified interval has passed; for example, after three days (which is the default), or two months. You specify this retention time using the interval type (`LoggingRollingIntervalType`) – for example, days or months, and the number of intervals (`LoggingFileCountLimit`) – for example, three (days) or two (months).

To set the interval type, set the following registry value:

```
HKLM\SOFTWARE\Authlogics\Authentication Server\LoggingRollingIntervalType
```

The default value is `3` (days).

Accepted values:

- `0` – Infinite time between rolling logs – this means that old logs are never deleted.
- `1` – Years.
- `2` – Months.
- `3` – Days.
- `4` – Hours.
- `5` – Minutes.

This setting also determines when new logs are created; for example, new logs are created every day, or every year. Multiple logs may be created within each interval depending on the size limit you have set for the logs; see section *4.3.4*, *Size limit of rolling log files*.

To set the number of intervals of logs stored, for example, three (days) or two (months), set the following registry value:

```
HKLM\SOFTWARE\Authlogics\Authentication Server\LoggingFileCountLimit
```

The default value is `3` – after three intervals, the logs from the first interval are deleted.

Accepted values:

- A number of intervals.

4.3.4 Size limit of rolling log files

New log files are created every interval (for example, every day, or every month). To prevent these files from becoming too large, you can set the maximum size of each log file. When this size is reached, a new log file is created within the same interval; for example, if you are using day interval logs:

```
AuthlogicsIdentityServer-20250325-0001.log
```

```
AuthlogicsIdentityServer-20250325-0002.log
```

or for year interval logs:

```
AuthlogicsIdentityServer-2025-0001.log
```

```
AuthlogicsIdentityServer-2025-0002.log
```

To set the maximum size of each log file, set the following registry value:

```
HKLM\SOFTWARE\Authlogics\Authentication Server\LoggingRollingSizeLimit
```

The default value is 20 megabytes.

Accepted values:

- A number in megabytes.

Note: This setting does not reduce the total size of the logs; by limiting the size of the individual files, it increases the number of files.

4.3.5 Example of rolling logs

With the default values of:

- `LoggingRollingIntervalType` – 3 (day intervals)
- `LoggingFileCountLimit` – 3 (three days)
- `LoggingRollingSizeLimit` – 20 (MB)

Old log files are deleted after three days.

An example of rolling log files produced starting on the March 25th 2025 is:

```
AuthlogicsIdentityServer-20250325-0001.log
AuthlogicsIdentityServer-20250325-0002.log
AuthlogicsIdentityServer-20250326-0001.log
AuthlogicsIdentityServer-20250326-0002.log
AuthlogicsIdentityServer-20250326-0003.log
AuthlogicsIdentityServer-20250327-0001.log
AuthlogicsIdentityServer-20250327-0002.log
AuthlogicsRestApi-20250325-0001.log
AuthlogicsRestApi-20250325-0002.log
AuthlogicsRestApi-20250326-0001.log
AuthlogicsRestApi-20250326-0002.log
AuthlogicsRestApi-20250326-0003.log
AuthlogicsRestApi-20250327-0001.log
AuthlogicsRestApi-20250327-0002.log
```

Each day has several files, each with a maximum size of 20 megabytes. When the logger starts writing to the first file of March 28th, the cleanup process is triggered, deleting the files from March 25th, as those are then more than three days old.

4.4 Other settings

The following setting is provided for information:

4.4.1 ProgramFolder

The program folder is specified in the following registry value:

```
HKLM\SOFTWARE\Authlogics\Authentication Server\ProgramFolder
```

The default value is:

```
C:\Program Files\Authlogics Authentication Server
```

Important: Changing this value is *not* supported.