intercede

Developers Guide

MyID Multi Factor Authentication and Password Security Management

Product Version: 5.0.6942.0

Publication date: March 2024

Call us on: +44 (0)1455 558 111 (UK & EMEA)
+1 408 706 2866 (US)

Email us: info@intercede.com

Intercede Ltd, Lutterworth Hall, St. Mary's Road, Lutterworth, Leicestershire, LE17 4PS, United Kingdom
UK Tel: +44 (0)1455 558 111 US Tel: +1 888 646 6943 email: info@intercede.com web: www.intercede.com

Table of Contents

Introduction 3
Guidance 3
Web Services Communication 4
Authentication and Transaction Verification 4
Authenticate User 4
VerifyTransaction 5
Return Codes 5
Challenge Grid Generation 5
accountname 6
Format 6
Resolution 7
Background 7
Phrase Challenge Generation 8
accountname 8
One Time Code Challenge Triggering 9
Identity Provider (IdP) 10
Web Services API (WSAPI) 10
Authenticating to the WSAPI 10
WSAPI Function list 11
WSAPI Function Details 13
Data Types 40
FidoCredential 40
Example: Programmatically creating a user account 41
Process Flow 42
Explanation 43
Using the Web Services API with Visual Studio 44
AuthlogicsApiClient 44
Authentication 44
Examples 45
Advanced Configuration 46
Diagnostics Logging 46
Other settings 47
Web Service call changes in Version 5.0 from 4.2.1 48

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)

Guidance

This developers guide provides detailed information about how to leverage the MyID Authentication Server Web Services Application Programming Interface (WSAPI). It should be used in conjunction with the "MyID Authentication Server Installation and Configuration Guide" which is designed to be an infrastructure document.

Web Services Communication

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

- HTTPS GET
- HTTP POST

By default, the Web Services run on TCP:14443 for SSL (with encryption). An SSL certificate MUST be installed onto the server and bound to the IIS web site.

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

A single Web API interface is available for common logon processing capabilities a complete API set for managing and automating all server functions. A list of available methods is available later in this document and via:

```
https://{ServerName}:14443/Services/swagger
```

Authentication and Transaction Verification

The following 2 Web Service operations can be used to process an authentication request and verify a transaction:

- AuthenticateUser
- VerifyTransaction

Authenticate User

To process an authentication request via Web Services simply supply the accountName and passcode to the AuthenticateUser function and it will return a status code from the ***Return Codes*** table above.

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
```

VerifyTransaction

To verify a transaction via Web Services, supply the accountname, passcode and transaction-specific data to the VerifyTransaction function. This operation will return a status code from the ***Return Codes*** table above.

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.

Return Codes

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 licencing issue.
13	Access Granted.	A pattern change is required.
111	Access Denied.	Directory related error.

Challenge Grid Generation

Integrating a Grid Deviceless challenge into a web page or application is a simple as calling the GetToken Web Service with an HTTPS GET request and specifying the *type=pingrid*. The GetToken Web Service accepts the following parameters which are all optional and can be combined as required:

- accountname
- type
- resolution
- format
- background

The theme of the generated image is determined by the Global Settings in MMC.

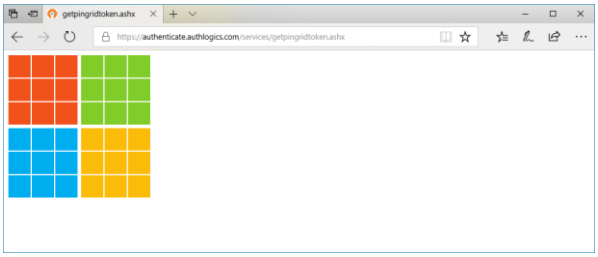
accountname

The "accountname" parameter will generate a challenge specific for the user defined in "accountname".

To generate a blank challenge grid call the GetToken Web Service without the "accountname" parameter, or a blank accountname value, e.g.:

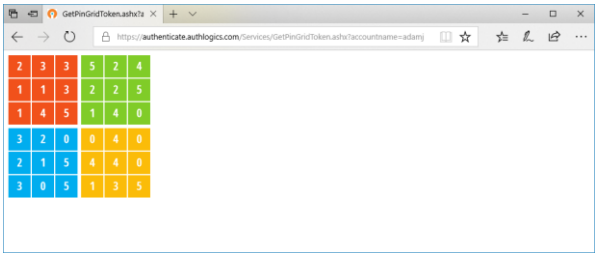
https://{servername}:14443/services/api/gettoken

The Web Service will return an image as follows which is appropriate to show when the accountname is not specified.



By adding a value for the accountname parameter, numbers are placed on the Grid or Phrase challenge text is produced. Even if a user account doesn't match the name provided a challenge will still be generated so not to disclose the existence of an actual user account. e.g.:

https://{servername}:14443/services/api/gettoken?accountname=bogususer



Format

The "format", and optional "background", parameters determines the graphical image type of the challenge grid and accepts the following options:

- PNG (default format when 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.

https://{servername}:14443/services/api/gettoken?accountname=bobm&format=TXT

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 the “resolution” parameter is not specified the resolution set in the MMC will be used.

https://{servername}:14443/services/api/gettoken?accountname=bobm&resolution=500

> **Note**
> If the specified resolution is greater than 2500 then a resolution of 2500 will be used. If the specified resolution is less than 50 then Global Settings resolution will be used.

Background

The optional “background” parameter sets the background HEX colour to use when a non-transparent image type is used, .i.e BMP and JPG. If the “background” parameter is not specified the resolution set in the MMC will be used.

https://{servername}:14443/services/api/gettoken?accountname=bobm&background=black

Background parameter accepts:

- Black
- White
- Transparent

Phrase Challenge Generation

Integrating a Phrase Deviceless challenge question into a web page or application is a simple as calling the GetToken Web Service with an HTTPS GET request and specifying the *type=pinphrase*. The MyID Authentication Server will return a challenge string.

The GetToken Web Service only requires the “type” and “accountname” parameters for Phrase .

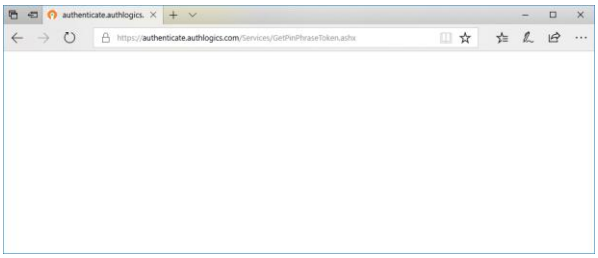
accountname

The “accountname” parameter will generate a challenge specific for the user defined in “accountname”.

To generate a blank PINphrase challenge call the GetToken.ashx Web Service without the “accountname” parameter, or a blank accountname value, e.g.:

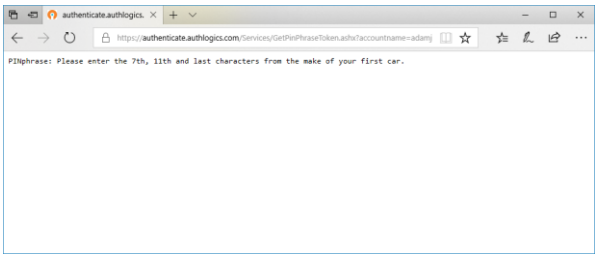
https://{servername}:14443/services/api/gettoken

The Web Service will return a blank string as follows which is appropriate to show when the accountname is not specified.



By adding a value for the username parameter, a challenge string is returned. Even if a user account doesn’t match the name provided, a challenge string will still be generated so as not to disclose the existence of an actual user account. e.g.:

https://{servername}:14443/services/api/gettoken?accountname=bobm&type=pinphrase

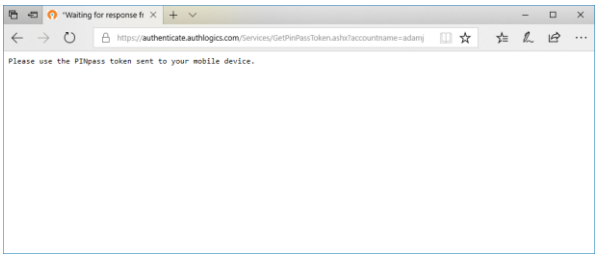


One Time Code Challenge Triggering

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

The GetToken. Web Service only requires the “type” and “accountname” parameters for PINpass.

https://{servername}:14443/services/api/gettoken?accountname=bobm&type=PINpass



Identity Provider (IdP)

All authentication in MyID Authentication Server is performed by the IdP. Without additional configuration it supports Windows Authentication (Kerberos or NTLM) and Mutual TLS.

In addition, it can be configured to use Client Credentials through the MyID Management Console.

Web Services API (WSAPI)

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

Note

Some API functions make use of enums based properties. Enum property values are case sensitive and will fail if the wrong case is used.

Authenticating to the WSAPI

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

An SSL certificate should be installed on the webserver and all IdP and WSAPI connections should be made using HTTPS to secure the logon credentials to the webserver.

As well as the credentials in order to authenticate you must supply a scope. The two scopes supported are:

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

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	✓	✓	✓
ChangeADpassword		✓	✓
CheckPasswordAgainstPolicy		✓	✓
CreateRealm			
CreateUser			
CreateUserExternal			
CreateUserPasswordReset	✓	✓	✓
DeleteRealm			
DeleteUser			
DisableEmergencyOverride			
DisablePinGrid			
DisablePinPass			
DisablePinPhrase			
DisablePush			
DomainExists			
EnableEmergencyOverride			
EnableFidoCredential			✓
EnablePinGrid			
EnablePinPass			
EnablePinPhrase			
EnablePush			
GenerateNewUserSeed			
GetAuthlogicsUsers			
GetDomains			
GetGlobalSettings	✓	✓	
GetOathUrl			✓
GetPairKeyParameters	✓		✓
GetPasswordPolicySettings	✓	✓	
GetRealms			
GetServerVersion		✓	
GetSettingsProperty			
GetUser	✓	✓	
GetUserProperty		[1]	✓
PairDevice	✓		✓
PasswordHashExists			
PinGridChangeMIP			✓
PinGridGenerateMIP			
PinGridProvision			
PinPassChangePin			✓
PinPassGeneratePIN			
PinPassProvision			

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

PinPhraseAddQuestion			
PinPhraseEditQuestion			
PinPhraseGenerateCodeword			
PinPhraseProvision			
PinPhraseRemoveQuestion			
RealmExists			
RemoveFidoCredential			✓
RenameRealm			
RenameUser			
ResetADpassword	✓	✓	✓
SendHTMLPinGridLetter			
SendHTMLPinPassLetter			
SendHTMLPinPhraseLetter			
SendHTMLPushLetter			
SendPresendToken			
SendRealTimeToken		✓	
SendRealTimeTokenbyProduct		✓	
SendToken			
SetADpassword			
SetOnlineVaultAdPassword	✓		✓
SetSettingsProperty			
SetUserProperty			✓
StartPushAuthentication	✓	✓	✓
SyncDevice			✓
TokenHardwareAdd			✓
TokenHardwareEnabled			✓
TokenHardwareRemove			✓
UpdateFidoCredential			✓
UpdateLicenceFile			
UpdateLicenceKey			
ValidateAdPassword			
VerifyEmergencyAccess		✓	✓
VerifyTransaction		✓	
YubiKeyOtpChangePin		✓	✓
YubiKeyOtpProvision			

WSAPI Function Details

Function: `AddFidoCredential`

The AddFidoCredential function will add 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 Data Types)	A representation of a FidoCredential

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.

An integer code is returned indicated 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 licencing issue.
13	Access Granted.	A pattern change is required.
111	Access Denied.	Directory related error.

Function: `ChangeADpassword`

The ChangeADpassword function changes the Active Directory Domain account password for a user account. This function can be called anonymously although the existing password must be supplied.

This function will result in a password change event on the AD 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 AD password for the user account specified in accountName.

newADpassword	string	P@ssWord	The new AD password to be written to the Windows Domain.

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 which 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

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 will create the entire hierarchy. The names should contain only alphanumeric, dot and underscore characters.

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.

Function: `CreateUserExternal`

The CreateUserExternal function creates an External Authlogics user account using default values and allows updating common properties during the creation. It does not configure the account for any authentication types.

Parameter	Type	Value Sample	Description
realm	string	Realm01	The Realm to create the External user account in.
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.

Function: `DeleteRealm`

The DeleteRealm function deletes an 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 comma separated list of parents and children, it will only attempt to to delete the last child but will fail if the realm isn't empty.

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 AD 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 which is resolvable in the directory.

Function: `DisableEmergencyOverride`

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

Parameter	Type	Value Sample	Description

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

Function: `DisablePinGrid`

The DisablePinGrid function disables the use of PINgrid on a user account. If PINgrid is not enabled on the account then this function has no effect.

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

Function: `DisablePinPass`

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

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

Function: `DisablePinPhrase`

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

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

Function: `DisablePush`

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

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

Function: `DomainExists`

The DomainExists function checks if the specified AD domain exists in the directory.

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

Parameter	Type	Value Sample	Description
domain	string	mydomain.com	An AD domain or external realm name which may or may not exist.

Function: `EnableEmergencyOverride`

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

Parameter	Type	Value Sample	Description
accountName	string	AndyP domain\andyp	A user account name which 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 AD password instead of a set code/password. Note: This feature is only available in Active Directory environments.
EmergencyOverride Code	string	5ecr3t	The code/password to be used for Emergency Override.

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 which is resolvable in the directory.
credentialId	string	INbV8ZJKjb6oa	The credentialId of the Fido token

Function: `EnablePinGrid`

The EnablePinGrid function enables the use of PINgrid on a user account. If PINgrid is enabled on the account then this function has no effect.

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

Function: `EnablePinPass`

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

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

Function: `EnablePinPhrase`

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

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

Function: `EnablePush`

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

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

Function: `GenerateNewUserSeed`

The GenerateNewUserSeed function generates a new 256bit seed on a user account. If a seed already exists then it will be replaced with the new one.

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

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 AD domain or external realm name which may or may not exist.

Function: `GetPasswordPolicySettings`

The GetPasswordPolicySettings returns a comma separated return of all Password Policies followed by the delimeter of a colon ":" and the setting value i.e True/ False or the numeric value for the control.

Parameter	Type	Value Sample	Description
n/a			

AllowUsername:False,DisableSharedPasswordProtection:False,DisableCloudPasswordBlacklist:False,DisableLocalPasswordBlacklist:False,DisallowMonthAndDay:False,DisallowSpaces:False,MaxAllowedUsernameCharacters:0,MaxLength:127,MaxRepeatingChars:8,MaxSequentialChars:3,MaxSequentialKeyBoardChars:0,EnablePasswordPolicy:True,MinLength:8,MinLowerCaseChars:0,MinNumericChars:0,MinSpecialChars:0,MinUnicodeChars:0,MinUpperCaseChars:0

Function: `GetDomains`

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

Parameter	Type	Value Sample	Description
n/a			

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.

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 will return all realms.

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

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			

Function: `GetSettingsProperty`

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

Multiple values can be queried 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,
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
```

Function: `GetUserProperty`

This is a read-only function, to set the value of a property use the SetUserProperty function.

Multiple values can be queried at once by specifying all the required properties in a 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.

Valid values for the names parameter which 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 which require Admin or Operator rights:

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

Valid values for the names parameter which 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 TokenID DeviceTypes

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

10	SecurityKey
11	SyncedPasskey

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 which 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 calling domain.

Function: `PinGridChangeMIP`

The PinGridChangeMIP function sets a new PINgrid pattern on the user account. The pattern is specified in MIP comma separated notation of grid positions and is stored as a hash and once written, **CANNOT be retrieved in plain text**.

Note: This function may be called by any authenticated user so long as they are attempting to update their own MIP, otherwise MyID Admin rights are required.

If this function is called by an Administrator or Operator, the PinGrid Complexity Checks will NOT be performed. If the user's context initiates this call, the PinGrid Complexity Checks will be enforced and users will NOT be 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 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.

Function: `PinGridGenerateMIP`

The PinGridGenerateMIP function creates a new random pattern in MIP comma separated notation. Either a simple or a complex pattern can be generated.

The new MIP is returned by the function as a string and is not written to a user account. The new MIP can be used 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 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*.

Function: `PinGridProvision`

The PinGridProvision function provisions a user account for user with PINgrid. This must be done at least once to allow PINgrid to be used with the account. The inputs are very 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 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.

Function: `PinPassChangePin`

The PinPassChangePin function allow for the changing for 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 authorise the change to the new PIN.

Function: `PinPassGeneratePIN`

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

Parameter	Type	Value Sample	Description
n/a			

Function: `PinPassProvision`

The PinPassProvision function provisions a user account for user with PINpass. This must be done 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 then a random PIN will 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.

Function: `PinPhraseAddQuestion`

The PinPhraseAddQuestion function adds a new question to the list available for users to provide answers to.

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

Function: `PinPhraseEditQuestion`

The PinPhraseEditQuestion function alters an existing question in the list available for users to provide answers to.

Warning: This should only be used for minor changes and as existing answers will still be matched to the question. For brand new questions you should delete the old one and add a new one.

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.

Function: `PinPhraseGenerateCodeword`

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

Parameter	Type	Value Sample	Description
n/a			

Function: `PinPhraseProvision`

The PinPhraseProvision function provisions a user account for user with PINphrase. This must be done 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, which the default question is “your code word”. If no answer is specified then a random word will be 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 which will be requested from the user to create their OTP code. Recommended values are between 3 and 5.

Function: `PinPhraseRemoveQuestion`

The PinPhraseRemoveQuestion function removes an existing question from the list available for users to provide answers to. Any existing answers to the removed question will be removed from the user account only when the account is next updated. A bulk answer delete will NOT be triggered although the answer will be ignored during subsequent processing.

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

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 only return true if the entire hierarchy exists.

Function: `RemoveFidoCredential`

The RemoveFidoCredential function will remove 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 Data Types)	A representation of a FidoCredential

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 will rename the last child.
newRealmName	string	newrealm.com	The new Realm name.

Function: `RenameUser`

The RenameUser function renames an existing user account in the directory. Renaming a user account is only suppored with External Users and not with AD 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.

Function: `SendHTMLPinGridLetter`

Sends an HTML formatted "PINgrid welcome letter" to the user via 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.

The MIP must be specified here as it cannot be retrieved from the user account in plain text. This function should be called directly 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 which is resolvable in the directory.
templateName	string	PINgridUserTemplate	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 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.

Function: `SendHTMLPinPassLetter`

Sends an HTML formatted "PINpass welcome letter" to the user via 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	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 Authlogics Program Files folder.

Function: `SendHTMLPinPhraseLetter`

Sends an HTML formatted "PINphrase welcome letter" to the user via 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 Authlogics Program Files folder.

Function: `SendHTMLPushLetter`

Sends an HTML formatted "Push MFA welcome letter" to the user via 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 Authlogics Program Files folder.

Function: `SendRealtimeToken`

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 via 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.

Function: `SendRealtimeTokenbyProduct`

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 via 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`

Function: `SendToken`

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 via email or Text Messaging depending on the user settings.

This function is ideally called 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.

Function: `SetADpassword`

The SetADpassword function updates the Active Directory Domain account password for a user account. This function can be called by the actual user or a MyID Administrator.

This function will result in a password reset event on the AD 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 AD password to be written to the Windows Domain.

Function: `SetSettingsProperty`

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

Multiple values can be set at once by specifying all the required properties and values in the corresponding a CSV strings.

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 which 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,
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.
```

Function: `SetUserProperty`

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

Multiple values can be set at once by specifying all the required properties and values in the corresponding a CSV strings.

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 which 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 which 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

LockedOut can only be set to **False** to unlock an account. *LockedOut* cannot be set to **True** manually.

Function: `SyncDevice`

The SyncDevice function sets a device to the status that requires it to synchronise 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.

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. This would typically be set to True for a new device.
tokenType	enum	WindowsPhone	The plaform the soft token is installed on.
tokenID	string	2419216713157481	The hardware / device ID of as indicated by the Authenticator App.

Valid values for the tokenType parameter include:

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

Function: `TokenHardwareEnabled`

The TokenHardwareEnabled function allows for individual tokens to be enabled or disabled 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 as indicated by the soft token application.

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 as indicated by the soft token application.

Function: `UpdateFidoCredential`

The UpdateFidoCredential function will update 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 Data Types)	A representation of a FidoCredential

Function: `UpdateLicenceFile`

The UpdateLicenceFile function updates the licence file information on the Authlogics Server with the XML data provided. The XML must be baes64/URL encoded before submitting to the WebAPI. Online and offline licence files are supported and online licenses will be activated.

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

Function: `UpdateLicenceKey`

The UpdateLicenceKey function updates the licence key information on the Authlogics Server with the licence key provided. Only online licences are supported and will be activated.

Parameter	Type	Value Sample	Description
licenceKey	string		A license key.

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 AD user account name to test AD password against.
password	string	Pa55w0rd	The user's AD password to test.

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.

Function: `VerifyTransaction`

The VerifyTransaction function processes a 3-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, an integer code is returned indicated 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 licencing issue.
13	Access Granted.	A pattern change is required.
111	Access Denied.	Directory related error.

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

Function: `YubiKeyOtpProvision`

The YubiKeyOtpProvision function provisions a user account for user with YubiKey OTP. This must be done 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 no PIN is specified then a random PIN will 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.

Data Types

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",
```

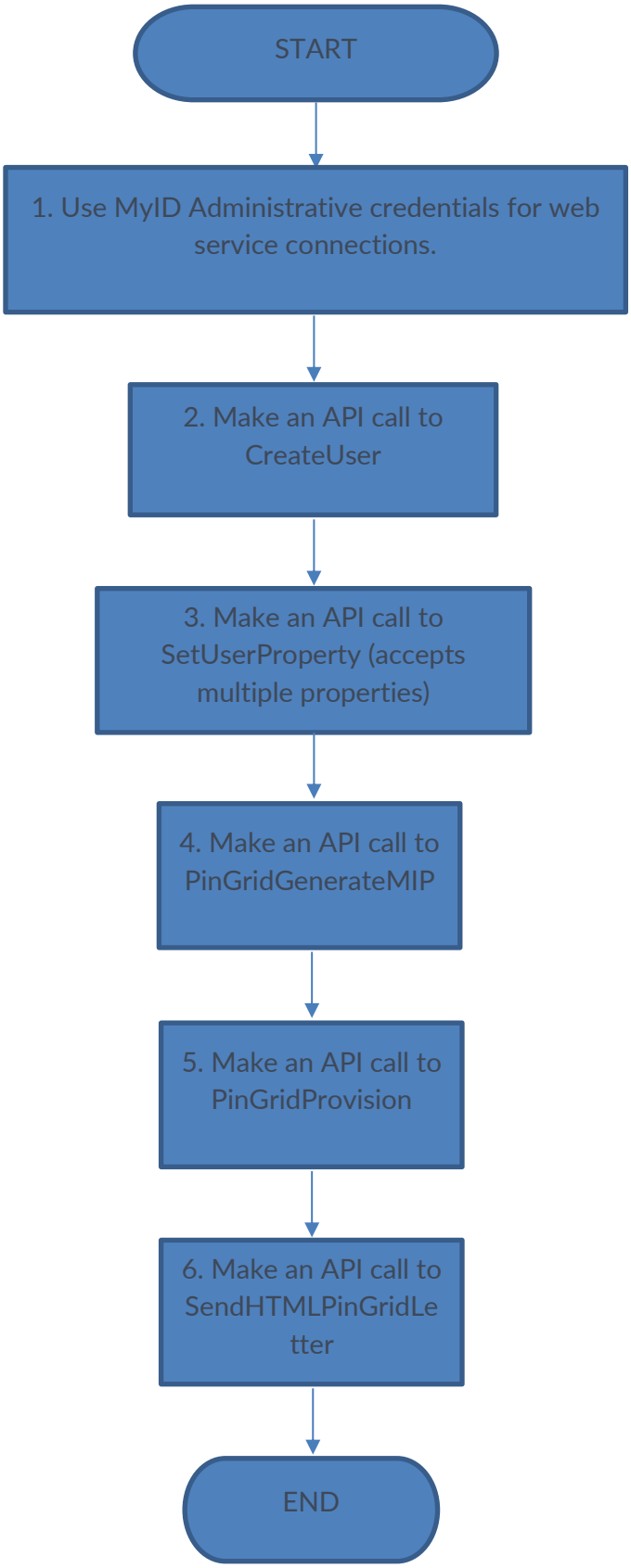
```
  "regDate": "2024-02-13T10:33:37.599Z",
  "signCount": 0,
  "transports": [
    0
  ],
  "type": 0,
  "userHandle": "string",
  "userId": "string"
}
```


Example: Programmatically creating a user account

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

Next a random PINgrid pattern (MIP) will be generated and use to provision the user for PINgrid. Lastly the user will be sent a welcome email containing their new pattern.

Process Flow



Explanation

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

1. Retrieve the credentials using the method described before, and then assign them accordingly. If using a service reference then you can assign this to this object instance credentials, and it will apply per request. If using a standard web request, then it is on a per request basis, 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 separates list of values, so many attributes can be applied to a user at any one time.
4. When calling the *PinGridGenerateMIP* function it is a good idea to store the result in a variable. This can then be utilised further in the next API call to provision the user. This can be made as complicated as you need it to be, and this will depend on your requirements for pattern complexity.
5. Call the *PinGridProvision* function to provision the user for PINgrid. Use the stored value from step 7 for the MIP. When adding parameters for the provision API call, the default value for a grid size is 6 – as in a 6x6 grid.
6. Call the *SendHTMLPinGridLetter* to send the welcome email to the user. Use the stored value from step 7 for the MIP. When adding parameters for the sending of the HTML letter to the user, the *templateName* string value is the default value of the PINgrid template, namely – 'PINgridUserTemplate', if this was a PINpass letter being sent then it would be 'PINpassUserTemplate', and if the letter was PINphrase then the value would be 'PINphraseUserTemplate'.

Using the Web Services API with Visual Studio

In addition to HTTPs GET, Post to communicate with the API via a .NET library, Authlogics.ApiClient.

The library is .Net Standard so will work with both .NET Framework and .NET Core.

AuthlogicsApiClient

To create the client you can either explicitly pass it the server uri, e.g.

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

Or you can pass in an IConfiguration instance, e.g.

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

Which can also be configured via DI, e.g.

```
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 will log the details of each method call, e.g.

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

Once the client instance has been created a call to TestConnection can be used 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. It has no return value but will throw an exception if the connection fails, e.g.

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

Authentication

The API requires a JWT bearer token to access and if you have already generated one you can pass that directly into the client, e.g.

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

You can also use Windows authentication, by using either the current logged on user credentials along with the scope, e.g.

```
await AuthenticateWithDefaultCredentials("rest_api", cancellationToken);
```

Or with explicit credentials along with the scope, e.g.

```
var credentials = new NetworkCredential
{
    UserName = "myserver.com\\Administrator",
    Password = "********"
};

await AuthenticateWithCredentials(credentials, "rest_api", cancellationToken);
```

You can also authenticate with a client credential along with the scope, e.g.

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

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

Examples

The following example will check for an existing realm and create it if it doesn't exist then return all realms.

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

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

Advanced Configuration

Advanced configuration options for MyID are controlled via the Windows registry and/or the IIS web.config file. The following entries are created during the installation of MyID server components and typically most of them should only be changed if instructed by an Intercede support engineer.

> **Note**
> After changing a registry key on the MyiD Server the IIS components must be restarted by running `IISRESET` from an elevated admin command prompt.

Diagnostics Logging

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

Default Value: `0`

Accepted Values:

`0 = Disabled`

`1 = Enabled`

Used by components: MyID Server

Notes: When this value is enabled various log files will be created in the logging folder. These logs may be requested by an Intercede support engineer.

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

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

Used by components: MyID Server

Notes: This Value may be changed to an alternative valid local folder with the same NTFS permissions as the default folder.

Other settings

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

Default Value: `60 (Decimal)`

Notes: This value sets the timeout in seconds to be used when a MyID component establishes a connection to the Domain.

```
HKLM\SOFTWARE\Authlogics\Authentication Server\DomainControllerRefreshTimeout
```

Default Value: `15 (Decimal)`

Notes: This value sets the timeout in minutes to be used when a MyID components refresh with Domain Controllers.

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

Default Value: *empty* `(string)`

Notes: Specify which Domain Controllers to access from the MyID Server. This control disables the auto-detect domain controller functionality within MyID.

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

Default Value: *empty* `(string)`

Notes: Specify which Global Catalogue Server to access from the MyID Server. This control disables the auto-detect global catalogue servers functionality within MyID.

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

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

Notes: Changing this value is NOT supported.

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 Authlogics MFA Version 5.

Added	Removed
AddFidoCredential	WebPortalVerifyOTP
EnableFidoCredential	
GetOathUrl	
RemoveFidoCredential	
UpdateFidoCredential	
VerifyEmergencyAccess	
YubiKeyOtpChangePin	
YubiKeyOtpProvision	