Developers Guide

Multi Factor Authentication and Password Security Management

Product Version: 4.2.1052.0

Publication date: December 2023

Call us on: +44 1344 568 900 (UK/EMEA)
+1 408 706 2866 (US)

Email us: sales@authlogics.com

Table of Contents

Introduction

Authlogics 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:
    - PINgrid - Pattern Based Authentication
    - PINphrase - Random Character Authentication
    - PINpass - OATH (TOTP) Compliant Authentication
    - YubiKey – Yubico YubiKey hardware token support

Authlogics 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 Authlogics Authentication Server Web Services Application Programming Interface (WSAPI). It should be used in conjunction with the "Authlogics Authentication Server Installation and Configuration Guide" which is designed to be an infrastructure document.

Web Services Communication

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

- SOAP 1.1
- SOAP 1.2
- 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 complete Web Services Description Language (WSDL) listing for the Web Service can be accessed via:

https://{ServerName}:14443/Services/?wsdl

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…

https://{ServerName}:14443/Services/wsapi.asmx/AuthenticateUser?accountname=adamj&passcode=123456

…which returns the following…

```
<?xml version="1.0" encoding="utf-8" ?>
<int xmlns="https://{ServerName}:14443/Services/wsapi.asmx/">2</int>
```

…indicating an invalid login.

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…

https://{ServerName}:14443/Services/wsapi.asmx/VerifyTransaction?accountname=adamj&passcode=123456&transactionData=1234567890

...which returns the following...

```
<?xml version="1.0" encoding="utf-8" ?>
<int xmlns="https://{ServerName}:14443/Services/wsapi.asmx/">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.

PINgrid Challenge Grid Generation

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

- accountname
- type
- resolution
- format
- background
- q1, q2, q3, q4

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

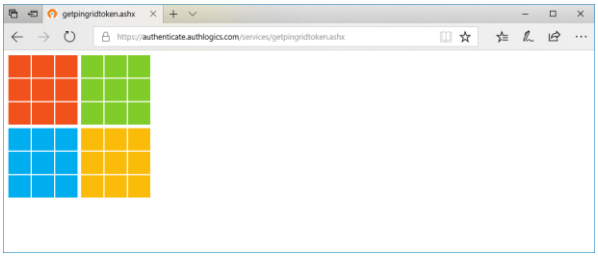
accountname

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

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

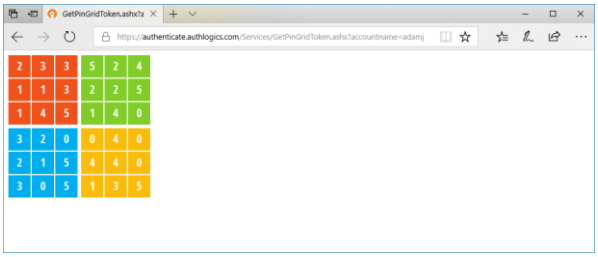
https://{ServerName}:14443/Services/GetToken.ashx?type=pingrid

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 PINgrid grid or PINphrase 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/GetToken.ashx?type=pingrid&accountname=adamj



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/GetToken.ashx?type=pingrid&format=gif

Resolution

The "resolution" parameter sets the resolution of the generated PINgrid 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/GetToken.ashx?type=pingrid&resolution=150

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/GetToken.ashx?type=pingrid&format=jpg&background=FFFFFF

Custom HEX colour codes can be found here:

- http://www.colorpicker.com/
- http://www.w3schools.com/tags/ref_colorpicker.asp

Quadrant Colours

The HEX colour of each PINgrid quadrant can be specified via the "q1", "q2", "q3" and "q4" parameters. The values specified override the global settings colours set in the MMC.

https://{ServerName}:14443/Services/GetToken.ashx?type=pingrid&1=dd4120&q2=31dd20&q3=2090dd&q4=ddc320

PINphrase Challenge Generation

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

The GetToken.ashx Web Service only requires the "type" and "accountname" parameters for PINphrase.

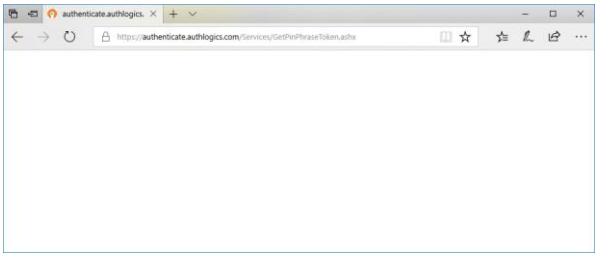
accountname

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.:

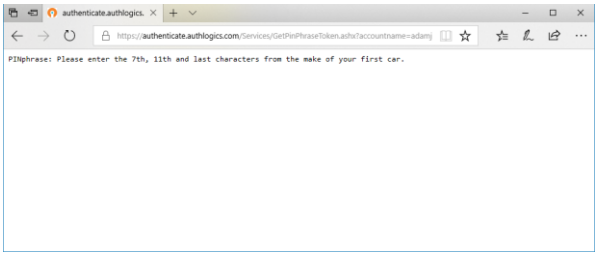
https://{ServerName}:14443/Services/GetToken.ashx?type=pinphrase

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/GetToken.ashx?type=pinphrase&accountname=adamj

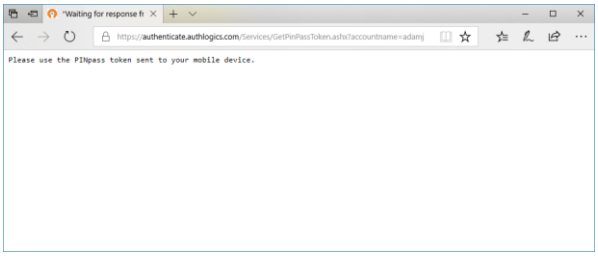


PINpass Challenge Triggering

While PINpass 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.ashx Web Service only requires the “type” and “accountname” parameters for PINpass.

https://{ServerName}:14443/Services/GetToken.ashx?type=pinpass&accountname=adamj



Web Services API (wsapi.asmx)

In addition to the Authlogics Management Console, Authlogics 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 Authlogics 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 HTTP Challenge authentication 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.

The web server used by Authlogics is IIS running on Windows. By default, the Authlogics IIS web site only allows Windows Authentication for HTTP 401 logons to ensure that passwords cannot be intercepted.

If the WSAPI client is not able to authenticate using Windows Authentication (Kerberos or NTLM) then either Basic or Digest authentication may be enabled in the IIS web server. An SSL certificate should be installed on the webserver and all WSAPI connections should be made using HTTPS to secure the logon credentials to the webserver.

WSAPI Function list

The following is a list of all available WSAPI functions:

1. AddUserPasswordHash (**Internal use only)**
2. AuthenticateRadiusUser (**Internal use only)**
3. AuthenticateUser
4. AuthenticateUserAdPasswordless **(Internal use only)**
5. ChangeADpassword
6. CheckPasswordAgainstPolicy
7. CreateRealm
8. CreateUser
9. CreateUserExternal
10. CreateUserPasswordReset (**Internal use only**)
11. DeleteRealm
12. DeleteUser
13. DisableEmergencyOverride
14. DisablePinGrid
15. DisablePinPass
16. DisablePinPhrase
17. DisablePush
18. DomainExists
19. EnableEmergencyOverride
20. EnablePinGrid
21. EnablePinPass
22. EnablePinPhrase
23. EnablePush
24. GenerateNewUserSeed
25. GetAuthlogicsUsers
26. GetDomains
27. GetGlobalSettings (**Internal use only**)
28. GetPairKeyParameters (**Internal use only**)
29. GetPasswordPolicySettings (**Internal use only**)
30. GetRealms
31. GetServerVersion
32. GetSettingsProperty
33. GetUser **(Internal use only)**
34. GetUserProperty
35. PairDevice (**Internal use only**)
36. PasswordHashExists
37. PinGridChangeMIP
38. PinGridGenerateMIP
39. PinGridProvision
40. PinPassChangePin
41. PinPassGeneratePIN
42. PinPassProvision

43. PinPhraseAddQuestion
44. PinPhraseEditQuestion
45. PinPhraseGenerateCodeword
46. PinPhraseProvision
47. PinPhraseRemoveQuestion
48. RealmExists
49. RenameRealm
50. RenameUser
51. ResetADpassword (**Internal use only**)
52. SendHTMLPinGridLetter
53. SendHTMLPinPassLetter
54. SendHTMLPinPhraseLetter
55. SendHTMLPushLetter
56. SendPresendToken
57. SendRealTimeToken
58. SendRealTimeTokenbyProduct
59. SendToken
60. SetADpassword
61. SetOnlineVaultPassword **(Internal use only)**
62. SetSettingsProperty
63. SetUserProperty
64. StartPushAuthentication (**Internal use only**)
65. SyncDevice
66. TokenHardwareAdd
67. TokenHardwareEnabled
68. TokenHardwareRemove
69. UpdateLicenceFile
70. UpdateLicenceKey
71. ValidateAdPassword
72. VerifyTransaction
73. WebPortalVerifyOTP **(Internal use only)**

WSAPI Function Details

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 Authlogics 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 Authlogics External user accounts.

Parameter	Type	Value Sample	Description
realmName	string	Realm01	A Realm name containing only alphanumeric, dot and underscore characters.

Function: `CreateUser`

The CreateUser function creates an Enabled Authlogics 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.com	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 Authlogics Realm. The Realm must be empty before it can be deleted.

Parameter	Type	Value Sample	Description
realmName	string	Realm01	An existing Realm name.

Function: `DeleteUser`

The DeleteUser function deletes an Authlogics 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 Authlogics 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
Realm	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: `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 Authlogics 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: `GetRealms`

The GetRealms function retrieves a string array of Realms that exist in the directory. This is a read-only function.

Parameter	Type	Value Sample	Description
n/a			

Function: `GetServerVersion`

The GetServerVersion function retrieves a string displaying the installed Authlogics 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,
SspDevicelessMfa, SspPasswordReset, SspLogonTechnology, SspPasswordlessPush,
WmpDevicelessMfa, WmpLogonTechnology, AppLogoUrl, AppLogoDescription,
AppUseBiometrics, AppOtpCopyPaste, AppTransactionValidation,
AuthlogicsServerCertificate, 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, CmsCallbackEnabled,
CmsAuthServerUrl, CmsCallbackServerUrl, CmsClientId, CmsClientScope.
```

Valid values for the names parameter which require Admin rights:

```
SMTPUsername, DirectoryID
```

Function: `GetUserProperty`

The GetUserProperty function retrieves the value/values of various user account properties. 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,
LastLogonYubiKey.
```

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

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

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.
```

Function: `PasswordHashExists`

The PasswordHashExists function checks whether or not a MD4 hash password exists in the Authlogics 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 Authlogics Admin rights are required.

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	A2F43cCcbED4…	The HASH of the user's existing MIP for authorisation purposes

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	A Realm name which may or may not exist.

Function: `RenameRealm`

The RenameRealm function renames an existing Realm in the directory.

Parameter	Type	Value Sample	Description
oldRealmName	string	oldrealm.com	The existing Realm name to be renamed.
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 Authlogics 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 Authlogics 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 Authlogics 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 Authlogics 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 Authlogics 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 an Authlogics 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,
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, CmsCallbackEnabled,
CmsAuthServerUrl, CmsCallbackServerUrl, CmsClientId, CmsClientSecret,
CmsClientScope.
```

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,
PinGridMIP, PinPassPIN.
```

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

```
MobileNumber, MobilePrivate, PinPhraseAnswers.
```

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: `UpdateLicenceFile`

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.

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

Parameter	Type	Value Sample	Description
base64licenceXml	string	%3CLicenceFileModel %20xmlns%3Ai%3D %22http%3A%2F %2Fwww.w3.org ….	Base64 encoded licence file content

Function: `UpdateLicenceKey`

Updates the licence key information on the Authlogics Server with the licence key provided. Only online licences are supported and will be activated.

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

Parameter	Type	Value Sample	Description
licenceKey	string	XXXXX-XXXXX-XXX XXXXX-XXXXXX	Licence key string provided by Authlogics

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: `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.

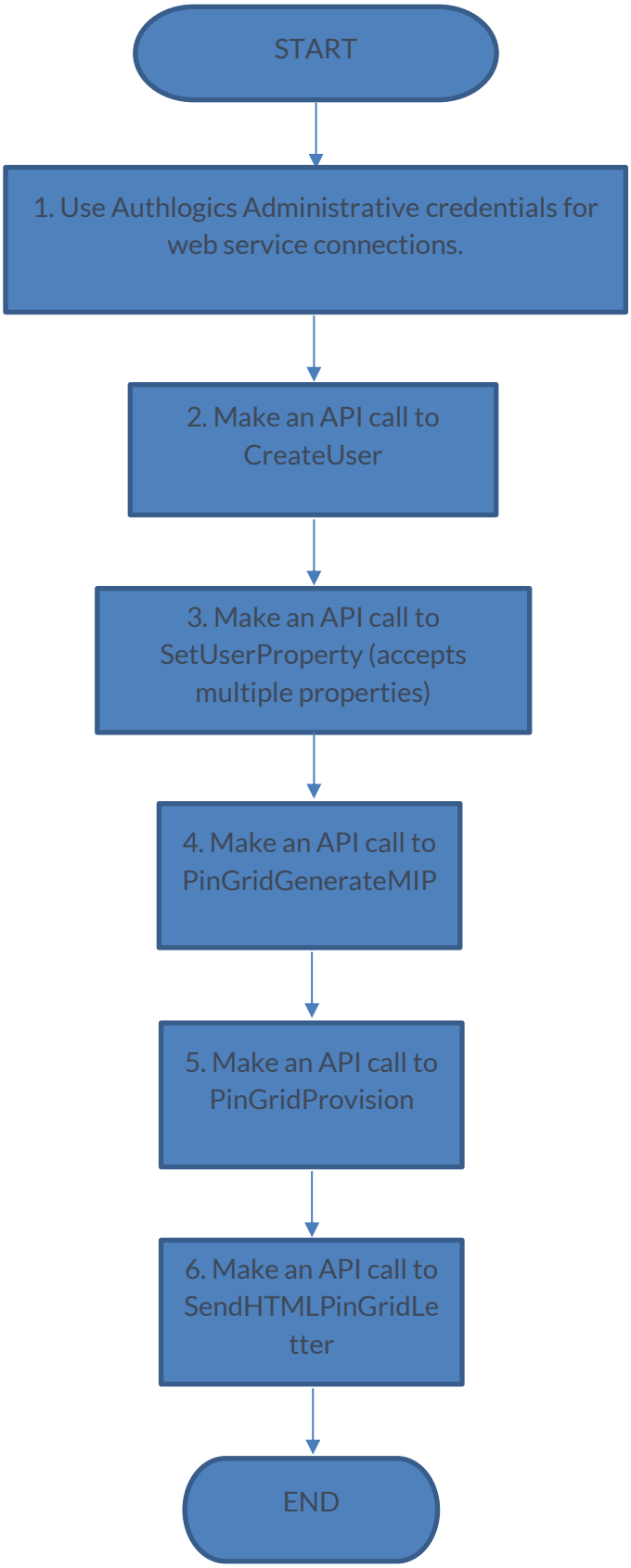
0	Access Granted.	Credentials are valid.
1	Access Denied.	Account name not found.
2	Access Denied.	Invalid passcode.
5	Access Denied.	Account Expired.
7	Access Denied.	Account Disabled, Locked out or not valid at this time.
8	Access Denied.	Authentication not available due to a licencing issue.
13	Access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

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* or *CreateUserExternal* 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

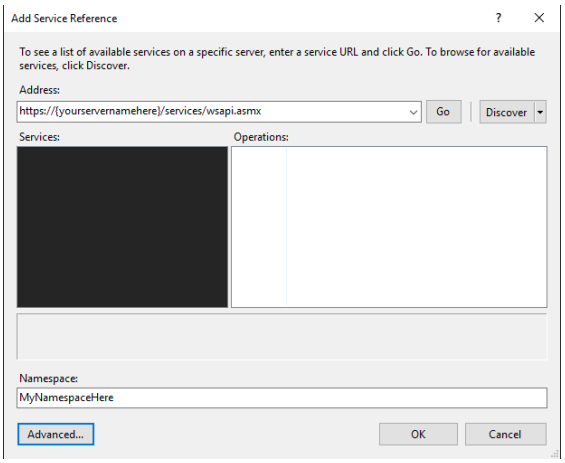
In addition to HTTPs GET, Post and SOAP you are also able to communicate with the API via code based references, as well as invoking parts of it from a URL. To do this, we can use a Service Reference or a straightforward web request.

Configuring a Service Reference

For a Service Reference, we are assuming the use of Visual Studio 2012 or higher. **The following examples are written with C# using .NET 4.6.2 – however lower versions can be used, although the code may vary.**

To create a service reference:

(1) Right click on the references section of your project and select Add Service Reference.



(2) Fill in the address of the server, as well as the namespace that you wish to work with. Click *OK*.

   a. The advanced options should be left as their default values.

Configuring the web.config for a Service Reference

Adding a service reference (as above) will place additional values into the web.config of your project automatically, such as the binding and the endpoint. There may be extra information, such as a custom binding (not required for this example).

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="YourBindingName">
        <security mode="Transport">
          <transport clientCredentialType="Basic" />
        </security>
      </binding>
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint address="https://yourservername/services/wsapi.asmx"
      binding="basicHttpBinding" bindingConfiguration="YourBindingName"
      contract="YourNamespace.YourBindingName" name="YourBindingName" />
  </client>
</system.serviceModel>
```

Within the *system.serviceModel* section, you will see all of the information required. Any of the extra information should be removed at this point, unless you need to add extra values that are specific to your configuration.

Fill in a suitable *binding name* (defaults such as WSAPISoap may be added, as an example), and make sure the specified value is consistent thought the configuration. Check the *endpoint address* to ensure it is as originally specified when creating the Service Reference.

The most important section of this configuration is *security mode*. This example uses the mode of *Transport*; i.e. that information is requested/sent over HTTPS. The credential type of *Basic* is being used for sending the username and password so that WSAPI methods that require authentication can authenticate.

> **Note**
> To use Basic authentication to the WSAPI you must enable Basic Authentication in IIS on the *Services* sub site as only Windows Authentication is enabled by default.

Making an anonymous request with a Service Reference

When making an anonymous/unauthenticated request, we instantiate the webservice so that we may use its methods like a normal class:

```
authenticate.WSAPISoapClient auth = new authenticate.WSAPISoapClient();
```

In this example, the namespace is *authenticate*, and we are using the WSAPISoapClient. We will also be looking at how to use the login validation method.

Next, we will use an asynchronous task to validate our login credentials. These can be passed from a front end screen such as an html form or similar.

```
var logon = await Task.Run(() => auth.AuthenticateUserAsync(context.UserName,
context.Password));
```

To run this asynchronous request, the method it is enclosed in should be marked as asynchronous with the keyword 'async' before the method name.

For example:

```
public async Task
```

This allows us to use the 'await' operator, which essentially means we will wait for the task to complete.

The *Task.Run* section of this is to run the AuthenticateUserAsync task that has been generated by the Service Reference when we added it earlier. The lambda expression it is encased in creates a delegate using LINQ – just so we can use it here without having to use another function to return a result.

NB - *Task.Run* is a part of .NET 4.5 onwards, and will need to include *System.Threading*

The result is then returned to the variable *logon*, with which we can then check against to confirm a success or failed logon, and then either log the user in, or give them an error message.

For example:

```
if (logon != 0)
{
	//This means that the logon has failed - handle it here
	return;
}
else
{
	//0 is a successful logon - continue as you wish to
}
```

For more information on using async operators and how they work visit:
https://msdn.microsoft.com/en-gb/library/mt674882.aspx

For information on using this code before .NET 4.5 visit:
https://msdn.microsoft.com/en-us/library/system.threading.tasks.task(v=vs.100).aspx

Making an authenticated request with a Service Reference

This is very similar to the above. The extra code needed is the credentials that will be required to authenticate against the method requiring authentication.

Where we instantiate the Service Reference, we can add in the credentials to the request:

```
auth.ClientCredentials.UserName.UserName = "YourUserName";
auth.ClientCredentials.UserName.Password = @"YourPassWord";
```

The credentials do not necessarily have to be specified in plain text within the code. If you are using keys within your web.config file, then they can be applied as well. These should be defined within the appSettings section.

For example:

```
    <add key="ApiUserName" value="YourUserName" />
    <add key="ApiPassWord" value="YourPassWord" />
```

These can then be utilised in code via the following means:

```
private string apiUserName = WebConfigurationManager.AppSettings["ApiUserName"];
private string apiPassWord = WebConfigurationManager.AppSettings["ApiPassword"];
auth.ClientCredentials.UserName.UserName = apiUserName;
```

So now we have the credentials, and they are assigned to the request. The request is then called in either the manner above, or another example such as the one below:

```
var realmsList = Task.Run(async () => await auth.GetRealmsAsync());

if (realmsList.Result != null)
{
	if (!realmsList.Result.Contains("mytestrealm.com"))
	{
		//Here we can check the result, and see if it contains a value
	}
}
```

The slight difference in this version, is that we are using an async LAMBDA expression, rather than asynchronously calling the task straight out.

This is because the parent method is not, or cannot be marked as async – Therefore we can mark the delegate as asynchronous, and then await the GetRealmsAsync task within that.

The result can then be checked by looking at the result attribute of the variable we created. If it is not null, then it has executed and returned something. We can then do more with that afterwards, such as calling another task.

In the above example, you might call the task to create the realm if it does not already exist.

Advanced Configuration

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

> **Note**
> After changing a registry key on the Authlogics 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: Authlogics Server

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

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

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

Used by components: Authlogics 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 an Authlogics 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 an Authlogics components refresh with Domain Controllers.

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

Default Value: *empty* `(string)`

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

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

Default Value: *empty* `(string)`

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

```
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 4.2 from 3.3

The following table lists the Web Service API calls that have been added and removed in Authlogics Version 4.2.

Added	Removed
AddUserPasswordHash	AuthenticateRadiusUser
GetServerVersion	CreateRealm
PasswordHashExists	DeleteRealm
ValidateAdPassword	DeliverMessage
ChangeADpassword	GetUserStatus
CreateUserExternal	RenameRealm
	RenameUser
	CreateUserEx