SecureAnyBox Client for PowerShell

SecureAnyBox Client for PowerShell is a tool that enables working with SecureAnyBox through the API. Using a client is suitable when making bulk operations - for example, uploading new files or changing the server address in records where the stored information is used.

SecureAnyBox is a cryptographic repository that allows you to store passwords, files and other sensitive information.

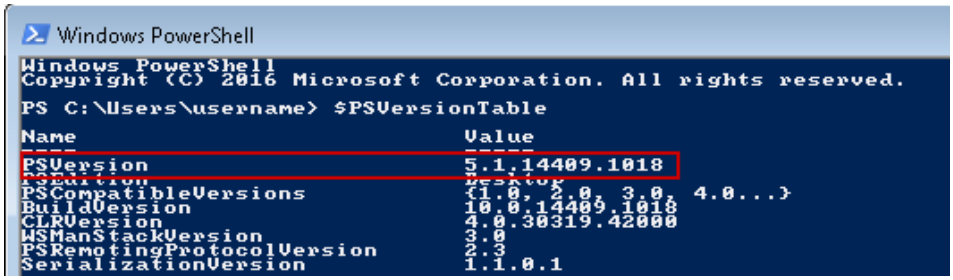
To better organize stored information, records can be stored in different Safe Boxes and Safe Box Groups.

Safe Box Group is intended for associating Safe Boxes to larger units. For example, the Safe Box Group can associate all Safe Boxes related to some project, server, etc.

Installing SecureAnyBox Client for PowerShell

In order to run SecureAnyBox Client for PowerShell, it is necessary to have PowerShell version 5.1 or higher. We recommend to use a version 5.1

To check which version of Powershell terminal is running, use command **$PSVersionTable** in the PowerShell Terminal. Version number is in the first row of the response table.

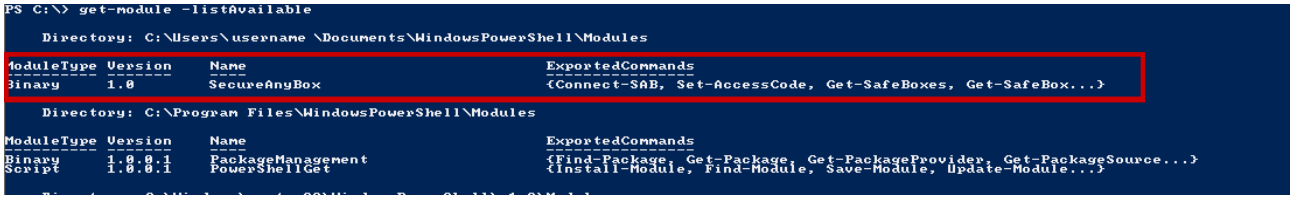


To install SecureAnyBox Client for PowerShell, run the msi file. You can run in by double-clicking the file in a folder or in the PowerShell terminal by running **.\<file_path>\SecureAnyBoxPowerShellModule.msi**

Default folder, where the PowershellModule installs, is „C:\Users\<your_username>\Documents\WindowsPowerShell\Modules\SecureAnyBox.

After installation completed, we recommend to check available modules by using command **Get-Module -ListAvaliable** in PowerShell terminal

If the installation is successful, SecureAnyBox and its exported commands should be displayed in the list.



If the SecureAnyBox Client for PowerShell is missing or commands are not exported, you can try to import them by using the command:

```
Import-Module -Name C:\Users\<your_username>\Documents\WindowsPowerShell\Modules\
SecureAnyBox - Verbose
```

In order to validate, that Module and its commands works, you can try to login to SecureAnyBox using **Connect-SAB** command. It should ask you for base Url of SecureAnyBox.

Commands

- Connect-SAB
- Set-2FA
- Get-LoginStatus
- Disconnect-SAB
- Set-AccessCode
- Get-SafeBoxes
- Get-SafeBox
- Get-SafeBoxPermissions
- Add-SafeBox
- Get-Records
- Get-Record
- Add-Record
- New-Account
- New-Certificate
- New-CreditCard
- New-File
- New-SecretAccount
- Set-Record
- Get-Password
- Set-Password
- New-PasswordRequirements
- New-CustomPassword
- Get-File
- Set-File
- Remove-Record
- Remove-SafeBox
- Get-StationPassword

Connect-SAB

Connects with SecureAnyBox server. To establish a session, it is necessary to know the URL of SecureAnyBox and a user's credentials. If any of the parameters are not specified directly in the command, the user will be prompted to enter them manually.

It is possible to connect with the SecureAnyBox server without authentication by using the parameter **-NoAuth**. Without authentication, it is only possible to obtain the station password using a ticket.

If user is successfully connected to SecureAnyBox, the Connect-SAB command returns information about currently logged user. These information can be obtained anytime using **Get-LoginStatus** command.

Examples:

```
Connect-SAB
```

Users will be prompted to enter the URL in the PowerShell terminal and username and password into the credentials form.

```
Connect-SAB <URL_of_SecureAnyBox>
```

Users will be prompted to enter username and password into the credentials form.

```
Connect-SAB <URL_of_SecureAnyBox> -Username <username> -Password (Read-Host
"Enter the password" -AsSecureString)
```

Users will be prompted to enter a password into the PowerShell terminal.

```
$credential = Get-Credential
Connect-SAB https://172.22.14.2/sab/ -Credential $credential
```

Users will be prompted to enter username and password into the credentials form.

```
Connect-SAB <URL_of_SecureAnyBox> -NoAuth
```

Connecting with NoAuth parameter to obtain station password with ticket

Set-2FA

If two factor authentication required, use command Set-2FA to enter a code from the Authenticator. User has to be logged in, to authenticate.

Required parameter is a Code and optional parameter is Timeout.

Code have to be in SecureString format. If 2FA Code is not specified when calling the function, then user is prompted to enter the code in dialogue window.

If Timeout is greater than 0, entered 2FA Code will be stored on the SecureAnyBox server for the specified amount of seconds.

Examples:

```
Set-2FA
```

User is prompted to enter the 2FA code into dialogue.

```
$code = Read-Host "Enter 2FA code" -AsSecureString
Set-2FA -code $code
```

User is prompted to enter the 2FA code into PowerShell terminal. Using function AsSecureString, input is stored in $code variable in SecureString format.

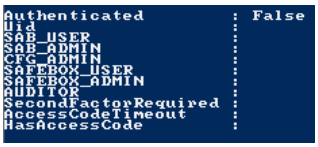
Get-LoginStatus

Returns information about currently logged user.

	Value	Description
Authenticated	True / False	Is user authenticated
Uid	<Domain>/<username>	Domain and username of currently logged user
SAB_USER	True / False	Has currently logged user a role SecureAnyBox user
SAB_ADMIN	True / False	Has currently logged user a role SecureAnyBox Administrator
CFG_ADMIN	True / False	Has currently logged user a role Administrator
SAFEBOX_USER	True / False	Has currently logged user a role Safe Box user
SAFEBOX_ADMIN	True / False	Has currently logged user an Inherited permission for the root level of Safe Boxes
AUDITOR	True / False	Has currently logged user a role Auditor
SecondFactorRequired	True / False	Is second factor authentication required
AccessCodeTimeout	<timeout in seconds>	How long the Access Code is stored on the SecureAnyBox Server
HasAccessCode	True / False	Has currently logged user the Access Code set

Disconnect-SAB

Disconnects from the SecureAnyBox server. Command has no parameters. If the command is successful, then the response is in the Login Status format.



Set-AccessCode

Command Set-AccessCode sends an Access Code to the SecureAnyBox server. A required parameter is the Access Code which has to be in SecureString format. If the Access Code is not specified when calling the command, then the user is prompted to enter it in the form.

An optional parameter is a Timeout. If Timeout is greater than 0, entered Access Code will be stored on the SecureAnyBox server for the specified amount of seconds.

Examples:

```
Set-AccessCode -Timeout 60
```

User is prompt to enter Access Code into a form. After entering, the Access Code is stored for 60 seconds (1 minute) on the SecureAnyBox server.

```
$accessCode = Read-Host "Enter Access Code" -AsSecureString
Set-AccessCode -AccessCode $accesscode -Timeout 60
```

User is prompt to enter Access Code into a PowerShell terminal. After entering, the Access Code is stored for 60 seconds (1 minute) on the SecureAnyBox server.

Get- SafeBoxes

Command Get-SafeBoxes retrieves a list of Safe Boxes (and Safe Box Groups). An optional parameter is an id of Safe Box Group from which the Safe Boxes should be listed. If the id of Safe Box Group is not specified, the function response is a list of Safe Boxes and Safe Box Groups from the root level.

Examples:

```
Get-SafeBoxes
```

Retrieves a list of Safe Box Groups and Safe Boxes from the root level

```
Get-SafeBoxes <id>
```

Retrieves a list of Safe Boxes from the Safe Box Group with the specified id

```
Get-SafeBoxes | Get-SafeBoxes
```

Retrieves a list of Safe Box Groups and Safe Boxes from the root level and all Safe Box Groups

Get-SafeBox

Using the command Get-SafeBox, retrieve Safe Box (or Safe box Group) information. The required parameter is the id of Safe Box (or Safe Box Group). If id is not specified, then the command returns information about the root level.

Examples:

```
Get-SafeBox
```

Retrieves information about the root level.

```
Get-SafeBox 35741
```

Retrieves information about the Safe Box (or Safe Box Group) with specified id.

```
Get-SafeBox -Id 35741
```

Retrieves information about the Safe Box (or Safe Box Group) with specified id.

Get-SafeBoxPermissions

Using the command Get-SafeBoxPermissions, it is possible to obtain a list of users who have permissions for a specific Safe Box or Safe Box Group. The required parameter is an id of Safebox or Safe Box Group.

Example:

```
$resp = Get-SafeBoxPermissions -Id <safe_box_id>
$resp.SafeBoxLevel
```

Response is a list of users and their permissions for specified Safe Box or Sfae Box Group.

Add-SafeBox

By usage of command Add-Safebox, create Safe Box or Safe Box Group. Required parameters of the function are Name, Type (SAFEBOX or SAFEBOX_GROUP) and Access Code.

Access Code has to be in SecureString format. If Access Code is stored on the SecureAnyBox server, it is not necessary to specify the Access Code in command.

Optional parameter id of Safe Box Group to where the Safe Box will add. If id not specified, Safe Box will create to the root level.

Examples:

```
Set-AccessCode -Timeout 10
Add-SafeBox -Name <safe_box_name> -Type SAFEBOX
```

Creates a Safe Box into the root level. Access Code is stored on the SecureAnyBox server for 10 seconds.

```
Set-AccessCode -Timeout 10
Add-SafeBox -Id <Safe_Box_Group_id> -Name <safe_box_name> -Type SAFEBOX
```

Creates a Safe Box into the specified Safe Box Group. Access Code is stored on the SecureAnyBox server for 10 seconds.

```
$accessCode = Read-Host "Enter Access Code" -AsSecureString
Add-SafeBox -Name <safe_box_name> -Type SAFEBOX – AccessCode $accessCode
```

Creates a Safe Box into the root level. Access Code is entered into the PowerShell console before calling the command Add-SafeBox.

```
Set-AccessCode -Timeout 10
$box = [PsCustomObject]@{ Type='SAFEBOX_GROUP'; Name = '<safe_box_group_name>'}
$box | Add-SafeBox
```

It is also possible to specify parameters of Safe Box / Safe Box Group in PsCustomObject. Required Access Code is stored on the SecureAnyBox server for 10 seconds.

Get-Records

Command Get-Records retrieves a list of records. A required parameter is an id of Safe Box from which records should be listed.

Examples:

```
Get-Records -Id <id>
```

Retrieves a list of records from the Safe Box with specified id

```
Get-SafeBoxes | Get-SafeBoxes | Get-Records
```

Retrieves a list of all records from every Safe Box

Get-Record

Command Get-Record returns detailed information for record with the specified id. The required parameter is an id of the record. If the id is not specified when calling the function, a user is prompted to enter the id in the PowerShell terminal.

Optional parameter is the Access Code. Access Code must be in SecureString format.

Examples:

```
Get-Record -Id <id_of_record>
```

Retrieves a unecrypted information of record with specified id

```
$accessCode = Read-Host "Enter Access Code" -AsSecureString
Get-Record -Id <id_of_record> -AccessCode $accessCode
```

Retrieves all information of record with specified id. User is prompted to enter Access Code into the PowerShell terminal.

```
Set-AccessCode – Timeout 60
Get-Records – id <id of Safe Box> | Get-Record
```

Retrieves all information of records nested in Safe Box with specified id. Access Code is stored on the SecureAnyBox server for 1 minute.

Record values are retrieved as Properties.

Property	
Id	Id of the Record
Template	Type of the record – Account, Secret Account, Certificate, File, Credit Card
Name	Name of the record
FileName	Name of the stored file (in File or Certificate type of the record)
FileSizeApprox	Approximate size of stored file
ServerAddress	Value stored in the address field (in Account or Secret Account type of record)
Site	Value stored in the Login site field (in Account or Secret Account type of record)
Description	Description of record.
Tags	Tags of the record.
Launcher	Value stored in the Connection Type field.
Note	Value stored in the Note field.
LoginName	Value stored in the Login field.
Modified	Timestamp of the last modification in Unix time
Attributes	Values of another fields (secret note, credit card number, certificate alias)

To get a list of all stored attributes, it is necessary to enter the access code. Otherwise, attributes will not be decrypted and retrieved.

```
$accessCode = Read-Host "Enter Access Code" -AsSecureString
$record = Get-Record -id 8426 -AccessCode $accessCode
$record.Attributes
```

Add-Record

Using the command Add-record creates a record into a specified Safe Box. Required parameters to create the record are the id of Safe Box, Template (case sensitive!) and Name of record.

Record templates:

- Account - for Account type of record
- AccountSec - for Secret Account type of record
- Cert - for Certificate type of record
- File - for File type of record
- Ccard - for Credit Card type of record

Another required parameter is Access Code. Access Code has to be in SecureString format. If the Access Code is not stored on the SecureAnyBox server or not specified when the command is called, then user is prompted to enter the Access Code into a form.

If you want to specify more record’s values than the Name, create a record object by using one of the commands New-Account or New-Certificate or New-CreditCard or New-File or New-SecretAccount. After specifying all desired values, add the record to SecureAnyBox using the Add-Record command.

Examples:

```
Add-Record
```

By entering the command without any of the parameters, the user will have to enter the id of Safe Box, to which the record will create, Template and Name of the record to the Powershell terminal. After these values are entered, the user will have to enter the Access Code (if Access Code is not stored on the SecureAnyBox server).

```
Add-Record -Id <safe_box_id> -Template <template> -Name <record_name>
```

In the command are specified all required parameters except the Access Code. If Access Code not stored on the SecureAnyBox server, then user is prompted to enter the Access Code.

New-Account

Using the command New-Account, it is easier to create PsCustomObject, which can be used later to add account to SecureAnyBox. Required parameters are Name and LoginName. Optional parameters are Password, PasswordRe (repeated password) and Note.

If required parameters are not entered when calling the function, the user is prompted to enter these values into the PowerShell terminal.

Examples:

```
New-Account | Add-Record -Id <safe_box_id>
```

The user is prompted to enter the required parameters of the New-Account command into PowerShell terminal. After these values are entered, a new account is added to the specified Safe Box. If Access Code not stored on the SecureAnyBox server, then user is prompted to enter the Access Code.

```
New-Account -Name "account_name" -LoginName "login_name" -Password "password" -
PasswordRe "password" -Note "note"| Add-Record -Id <safe_box_id>
```

Account variables are specified when calling the function. New account will be added to the SecureAnyBox server. If Access Code not stored on the SecureAnyBox server, then user is prompted to enter the Access Code.

```
$acc = New-Account -Name "new_account" -LoginName "loginname"

$acc.ServerAddress = "server_address"
$acc.Site = "site"
$acc.Description = "description"
$acc.Tags.Add('tag1')                    ## each tag must be added individually
$acc.Tags.Add('PS_import')
$acc.Note = "noteline1`nnoteline2"
$acc.Launcher = "NONE"
$acc.Password = "hyBE719lExma6w"
$acc.PasswordRe = "hyBE719lExma6w"

$rec | Add-Record -Id <safe_box_id>
```

If you want to store more information in the account, create a new account object $acc using the command Add-Account and specify them after the $acc object is created. After adding values, use the command Add-Record to create an account record on the SecureAnyBox server. If Access Code is not stored on the SecureAnyBox server, the user is prompted to enter the Access Code.

New-Certificate

Using the command New-Certificate, it is easier to create PsCustomObject, which can be used later to add certificate to SecureAnyBox. Required parameter is Name and optional parameter is Note.

If required parameter is not entered when calling the function, the user is prompted to enter it into the PowerShell terminal.

Examples:

```
New-Certificate | Add-Record -Id <safe_box_id>
```

The user is prompted to enter the required parameters of the New-Certificate command into PowerShell terminal. After these values are entered, a new certificate is added to the specified Safe Box. If Access Code is not stored on the SecureAnyBox server, user is prompted to enter the Access Code.

```
New-Certificate -Name "certificate_name" | Add-Record -Id <safe_box_id>
```

Certificate variables are specified when calling the function. New certificate will be added to the SecureAnyBox server. If Access Code is not stored on the SecureAnyBox server, user is prompted to enter the Access Code.

```
$cert = New-Certificate -Name "cert_name"

$cert.Description = "description"
$cert.Tags.Add('tag1')                    ## each tag must be added individually
$cert.Tags.Add('PS_import')
$cert.Note = "noteline1`nnoteline2"

$cert.Attributes.alias = "certificate alias"
$cert.Attributes.'sec-note' = "sec-note12`nsec-note22"
$cert.Attributes.filePassword = "WecO970ZAVKU7r"

$certPath = "full path to certificate file"

$newRec = $cert | Add-Record -Id $safeBoxId
Set-File -Id $newRec.id $certPath
```

If you want to store more information in the certificate, create a new certificate object $cert using the command Add-Certificate and specify them after the $cert object is created. After adding values, use the command Add-Record to create an certificate record on the SecureAnyBox server. If Access Code is not stored on the SecureAnyBox server, the user is prompted to enter the Access Code. After certificate record is created on the SecureAnyBox server, it is possible to upload the certificate file.

New-CreditCard

Using the command New-CreditCard, it is easier to create PsCustomObject, which can be used later to add credit card record to SecureAnyBox. The required parameter is a Name and the optional parameter is a Note.

If required parameter is not entered when calling the function, the user is prompted to enter it into the PowerShell terminal.

Examples:

```
New-CreditCard | Add-Record -Id <safe_box_id>
```

The user is prompted to enter name of the credit card when calling the command into PowerShell terminal. After these values are entered, a new credit card is added to the specified Safe Box. If Access Code is not stored on the SecureAnyBox server, user is prompted to enter the Access Code.

```
New-CreditCard -Name "certificate_name" -Note "credit_card_note" | Add-Record -Id
<safe_box_id>
```

Credit card name and note is specified when calling the command. New credit card will be added to the SecureAnyBox server. If Access Code is not stored on the SecureAnyBox server, user is prompted to enter the Access Code.

```
$ccard = New-CreditCard -Name "new_credit_card"

$ccard.Description = "description"
$ccard.Tags.Add('tag1')                  ## each tag must be added individually
$ccard.Tags.Add('PS_import')
$ccard.Note = "noteline1`nnoteline2"

$ccard.Attributes.number = "8745 5896 1752 5963 4812"
$ccard.Attributes.expiration = "2024-10-31T00:00:00.000Z"
$ccard.Attributes.'cvv' = "111"
$ccard.Attributes.pin = "1234"
$ccard.Attributes.'sec-note' = "secretnoteline1`nline2"

$ccard | Add-Record -Id <safe_box_id>
```

If you want to store more information in the credit card, create a new credit card object $ccard using the command Add-CreditCard and specify them after the $ccard object is created. After adding values, use the command Add-Record to create an credit card record on the SecureAnyBox server. If Access Code is not stored on the SecureAnyBox server, the user is prompted to enter the Access Code.

New-File

Using the command New-File, it is easier to create PsCustomObject, which can be used later to add file record to SecureAnyBox. The required parameter is a Name and the optional parameter is a Note.

If required parameter is not entered when calling the function, the user is prompted to enter it into the PowerShell terminal.

Examples:

```
New- File | Add-Record -Id <safe_box_id>
```

The user is prompted to enter the required parameter of the New-File command. After these values are entered, a new file record is added to the specified Safe Box. If Access Code is not stored on the SecureAnyBox server, user is prompted to enter the Access Code.

```
New- File -Name "file_name" -Note "note" | Add-Record -Id <safe_box_id>
```

File variables are specified when calling the function. New file record will be added to the SecureAnyBox server. If Access Code is not stored on the SecureAnyBox server, user is prompted to enter the Access Code.

```
$file = New-File -Name "file_name"

$file.Description = "description"
$file.Tags.Add('tag1')                    ## each tag must be added individually
$file.Tags.Add('PS_import')
$file.Note = "noteline1`nnoteline2"

$file.FileName = "name of file to upload"
$file.Attributes."sec-note" =  "sec-note12`nsec-note22"

$filePath = "full file path"

$newRec = $file | Add-Record -Id $safeBoxId
Set-File -Id $newRec.id $filePath
```

If you want to store more information in the file, create a new file object $file using the command Add-File and specify them after the $file object is created. After adding values, use the command Add-Record to create an file record on the SecureAnyBox server. If Access Code is not stored on the SecureAnyBox server, the user is prompted to enter the Access Code. After file record is created on the SecureAnyBox server, it is possible to upload the file.

New-SecretAccount

Using the command New-Account, it is easier to create PsCustomObject, which can be used later to add secret account record to SecureAnyBox. Required parameters are Name and LoginName. Optional parameters are Password, PasswordRe (repeated password) and Note.

If required parameters are not entered when calling the function, the user is prompted to enter these values into the PowerShell terminal.

Examples:

```
New-SecretAccount | Add-Record -Id <safe_box_id>
```

The user is prompted to enter the required parameters of the New-SecretAccount command. After these values are entered, a new account is added to the specified Safe Box. If Access Code not stored on the SecureAnyBox server, then user is prompted to enter the Access Code.

```
New-SecretAccount -Name "account_name" -LoginName "login_name" -Password
"password" -PasswordRe "password" -Note "note"| Add-Record -Id <safe_box_id>
```

Secret account variables are specified when calling the function. New secret account will be added to the SecureAnyBox server. If Access Code not stored on the SecureAnyBox server, then user is prompted to enter the Access Code.

```
$secAcc = New-SecretAccount -Name "new_secret_account" -LoginName "loginame"

$secAcc.Site = "site"
$secAcc.Description = "description"
$secAcc.Tags.Add('tag1')                ## each tag must be added individually
$secAcc.Tags.Add('PS_import')

$secAcc.Note = "noteline1`nnoteline2"
$secAcc.Launcher = "NONE"

$secAcc.Attributes.'sec-serverAddress' = "serverAddress"
$secAcc.Attributes.'sec-note' = "secretnoteline1`nline2"

$secAcc.Password = "hyBE719DslExma6w"
$secAcc.PasswordRe = "hyBE719DslExma6w"

$secAcc | Add-Record -Id <safe_box_id>
```

If you want to store more information in the account, create a new secret account object $secAcc using the command Add-SecretAccount and specify them after the $secAcc object is created. After adding values, use the command Add-Record to create a secret account record on the SecureAnyBox server. If Access Code is not stored on the SecureAnyBox server, the user is prompted to enter the Access Code.

Set-Record

The command Set-Record enables modification of SecureAnyBox records. The required parameter is an id of the record. The optional parameters are Name, Note, Attributes, Record and Access Code. If you want to change encrypted information in record, it is necessary to specify the Access Code when calling the command or have the Access Code stored on server.

Using the Set-Record command, it is not possible to modify a file in the record. For that operation use, the Set-File command instead.

Example:

```
Set-Record -Id <record_id> -Name "record_name" -Note "changed note"
```

Using only command Set-Record, record's name and note will be changed.

```
$rec = Get-Record -Id <record_id>
$rec.Description = "changed_description"
$rec.Attributes.alias = "changed_alias"
$rec | Set-Record –Id <record_id>
```

By command Get-Record user obtains existing record from the SecureAnyBox into $rec variable. After changing values in the $rec variable, changes are sent to the SecureAnyBox server using the command Set-Record. If Access Code is not stored on the SecureAnyBox server, the user is prompted to enter the Access Code.

```
$attr = New-Object System.Collections.Generic.Dictionary"[String,String]"

$attr.Add("number" , "5468 7185 2453 8752 6985")
$attr.Add("expiration" , "2022-11-30T00:00:00.000Z")
$attr.Add("cvv" , "135")
$attr.Add("pin" , "1111")
$attr.add("sec-note", "secretline1`nsecretline2")

Set-Record -Id 44500 -Attributes $attr
```

By using an $attr variable, user can change record's encrypted attributes. If Access Code is not stored on the SecureAnyBox server, the user is prompted to enter the Access Code.

```
$rec = $rec = Get-Record -Id <record_id>
$rec.Tags.Remove("tag1")
$rec.Tags.Add("tag2")
Set-Record –Id <record_id> -Record $rec
```

Using the command Get-Record user obtains existing record from the SecureAnyBox into the $rec variable. The changed $rec variable is later used as a parameter for the Set-Record command. If Access Code is not stored on the SecureAnyBox server, the user is prompted to enter the Access Code.

Get-Password

The command Get-Password returns a password as Secure String. If the Access Code is not stored on SecureAnyBox server, user is prompt to enter the Access Code. To show the password in a readable format, use the parameter **-AsPlainText.**
If you want to obtain the password for a specific record, call the function as:

```
Get-Password <id_of_record> -AsPlainText
```

Set-Password

Command will set password to record with specified id. Required parameters are id of record, password and the Access Code.

Password has to be in the Secure String format. To set a plain text password as a SecureString variable, you can use the ConvertTo-SecureString function.

Also Access Code has to be in SecureString format. If the Access Code is not stored on the SecureAnyBox server or not specified when the command is called, then user is prompted to enter the Access Code into a form.

Examples:

```
$password = ConvertTo-SecureString "P@5Hbikb&lk!aK2" -AsPlainText -Force
Set-Password <record_id> $password
```

Password is converted to SecureString format and set to the specified record. If Access Code is not stored on the SecureAnyBox server, a user is prompted to enter the Access Code into a form.

```
$accessCode = Read-Host "Enter Access Code" -AsSecureString
$password = ConvertTo-SecureString "P@5Hbikb&lk!aK2" -AsPlainText -Force
Set-Password <record_id> $password -AccessCode $accessCode
```

Password is converted to SecureString format and set to the specified record. Access Code is entered into PowerShell terminal before setting the password.

New-PasswordRequirements

Command New-PasswordRequirements creates a new set of password requirements that can be modified and later used when generating a new custom password.

Password Requirements are:

	Value (Format)	Description
UseLower	True (Boolean)	Enable/disable use of lowercase characters in generated password.
UseUpper	True (Boolean)	Enable/disable use of uppercase characters in generated password.
UseDigits	True (Boolean)	Enable/disable use of digits in generated password.
UseSpecial	False (Boolean)	Enable/disable use of special characters in generated password.
MinLower	0 (Integer)	Minimal count of lowercase characters in generated password.

MinUpper	0 (Integer)	Minimal count of uppercase characters in generated password.
MinDigits	0 (Integer)	Minimal count of digits in generated password.
MinSpecial	0 (Integer)	Minimal count of special characters in generated password.
MinLength	0 (Integer)	Minimal length of generated password.
MinEntropy	50 (Integer)	Minimal entropy of generated password.
IncludeCharacters	(String)	Characters to include in generated password
ExcludeCharacters	B8G6iI1l0oOQDS5Z2 (String)	Characters to exclude in generated password. (e.g. for better readability of password)

Example:

```
$req = New-PasswordRequierements
$req.UseSpecial = 1
$req.UseDigits = 0
$req.MinSpecial = 2
$req.MinUpper = 1
$req.IncludeCharacters = "Fj"
```

New Password requirements are created into $req variable and later changed according to requirements for password which should be generated. To change Boolean value use 1 for True and 0 for False.

New-CustomPassword

Command New-CustomPassword generates password according to custom password requirements. Required parameter is Requirements (which can be created using command New-Password requirements and edited).

Response of the command is in Generated Password format.

Example:

```
$req = New-PasswordRequirements
$req.UseSpecial = 1
$req.MinDigits = 2
$req.MinUpper = 1

$pswd = New-CustomPassword -Requirements $req

$password = ConvertTo-SecureString $pswd.Password -AsPlainText -Force

Set-Password <record_id> $password
```

At the first step are created Password requirements which are modified in the following steps. A new password is generated into the $pswd variable using modified password requirements. Because the $pswd variable is in Generated password format, the next step is converting generated password into SecureString format ($password variable). Password in the SecureString format can be finally set as a record's password using the Set-Password command.

Get-File

Command Get-File downloads a file from the record to the current folder. Required parameters are the id of record and Access Code.
If Access Code is not specified or stored on the SecureAnyBox server, a user is prompted to enter the Access Code in a form.

```
Get-File <id_of_record>
```

Set-File

Command Set-File uploads a file to specified record. Required parameters are the id of record, path to file to upload and Access Code.
If Access Code is not specified or stored on the SecureAnyBox server, a user is prompted to enter the Access Code in a form.

```
Set-File <id_of_record> "<filePath>"
```

Remove-Record

Command removes record from SecureAnyBox. Required parameters are the id of record and Access Code. If Access Code is not specified or stored on the SecureAnyBox server, a user is prompted to enter the Access Code in a form.
It is necessary to have REMOVE permission for the Safe Box from which the record will remove.

```
Remove-Record -id <id_of_record>
```

Remove-SafeBox

Command removes Safe Box or Safe Box Group from SecureAnyBox. Required parameters are the id of record and Access Code. If Access Code is not specified or stored on the SecureAnyBox server, a user is prompted to enter the Access Code in a form.
With removed Safe Box, all nested records are also removed. With removed Safe Box Group all nested Safe Boxes and records are also removed.
It is necessary to have REMOVE permission for the Safe Box Group or the root level, from which the Safe Box (Safe Box Group) will remove.

```
Remove- SafeBox -id <id_of_safebox>
```

Get-StationPassword

Command Get-StationPassword returns a password of the station via using a ticket. Required parameters are a ticket id and a station name, and an optional parameter is a username. Obtaining a station password via ticket can be done without authentication.
The command returns the station password as SecureString. To get the station password in a readable format, add -**AsPlainText** parameter.

```
connect-SAB https://sab.company.com -NoAuth
get-StationPassword b4447555fff74f65bceeea2f1408871c25c5d3feb Station1 Administrator
```