SwyxWare 11 CDS API Changes

In order to support bulk administration and to prepare the software for features available with SwyxWare 11.10, there will be some changes in the SwyxWare Database which have also an affect on the ConfigDataStore API

Affected functionality

- Using ConfigDataStore API to create SwyxWare users with SIP credentials

Not affected functionality

- SwyxIt! and 3rd-party applications using SwyxIt Client Line Manager Interface
- Call routing scripts
- SwyxWare PowerShell Module (but see the new, additional cmdlet below)

CDS API changes

Removed API functions

The following Functions were removed from the **UserEnum** class:

UserEnum

```
/// <summary>
/// Sets filter on user SipUserId
/// creates and executes SQL query on user table in configuration database
/// </summary>
/// <param name="sipUserIdFilter">filter string with optional SQL LIKE
wildcards.</param>
/// <param name="checkLoginPermissions">if true returns only users with
permission to login via H.323</param>
/// <remarks>This is a remote method.</remarks>
public void ExecuteSipUserIDFilter(String sipUserIdFilter, bool
checkLoginPermissions)


/// <summary>
/// Sets filter on user SipUserName
/// creates and executes SQL query on user table in configuration database
/// </summary>
/// <param name="sipUserIdFilter">filter string with optional SQL LIKE
wildcards.</param>
/// <param name="checkLoginPermissions">if true returns only users with
permission to login via H.323</param>
/// <remarks>This is a remote method.</remarks>
public void ExecuteSipUserNameFilter(String sipUserNameFilter, bool
checkPermission)


/// <summary>
```

```
/// Reads an user configration item from the database and adds it to the
local enumeration.
/// If the user configration item already exists in the local enumeration
the old entry will be removed
/// before the new one will be added. Any sortings of the local enumeration
will be ignored.
/// </summary>
/// <param name="SIPUserID">The name of the user to be added.</param>
public UserEntry AddUserToEnumBySIPUserID(String SIPUserID, bool
checkPermission)


/// <summary>
/// Reads an user configration item from the database and adds it to the
local enumeration.
/// If the user configration item already exists in the local enumeration
the old entry will be removed
/// before the new one will be added. Any sortings of the local enumeration
will be ignored.
/// </summary>
/// <param name="SIPUserID">The SipUserID of the user to be added. If there
is no user with this SipUserID a user with an equal name will be added.
</param>
public UserEntry AddUserToEnumBySIPUserIDOrUserName(String SIPUserID, bool
checkPermission)


/// <summary>
/// Reads an user configration item from the database and adds it to the
local enumeration.
/// If the user configration item already exists in the local enumeration
the old entry will be removed
/// before the new one will be added. Any sortings of the local enumeration
will be ignored.
/// </summary>
/// <param name="SIPUserName">The sip username the user to be
```

```
added.</param>
public UserEntry AddUserToEnumBySIPUserName(string SIPUserName, bool
checkPermission)
```

The following Properties have been removed from the **UserEntry** class:

UserEntry

`public string SIPUserID { get; set; }`
`public bool IsSIPUserIDNull { get; }`
`public void SetSIPUserIDNull()`
 	
`public string SIPUserName { get; set; }`
`public bool IsSIPUserNameNull { get; }`
`public void SetSIPUserNameNull()`
 	
`public string SIPPasswordDecrypted { get; set; }`
`public bool IsSIPPasswordNull { get; }`
`public void SetSIPPasswordNull()`

New API functions

To store the Sip credentials for a user it is necessary to have an existing user in the database. This means that a new user must be added in the database first before it's possible to add the Sip credentials for the user.

The Sip Credentials can be added/manipulated and received by the **UserEnum** of the **SWConfigDataClientLib**. The following functions were added to the UserEnum:

UserEnum

```
/// <summary>
/// Gets all stored sip credentials
/// </summary>
/// <returns>Collection of all sip credentials</returns>
public ICollection<UserSipCredentials> GetSipCredentials()

/// <summary>
/// Gets all sip credentials of a specific user
/// </summary>
/// <param name="userId"></param>
/// <returns>The third party Sip credentials of a specific user</returns>
public UserSipCredentials GetSipCredentials(int userId)

/// <summary>
/// Gets all sip credentials of a specific user
/// </summary>
/// <param name="userId"></param>
/// <returns></returns>
public UserSipCredentials GetSipCredentialsBySipUserId(string sipUserId)

/// <summary>
/// Set or add sip credentials to the database
/// </summary>
/// <param name="userId"></param>
/// <param name="sipUserId"></param>
/// <param name="sipUserName"></param>
/// <param name="sipPassword"></param>
public void SetSipCredentials(int userId, string sipUserId, string
sipUserName, string sipPassword)
```

It is also possible to receive the Sip Credentials by the **PhoneClientFacade**.

PhoneClientFacade

```
/// <summary>
/// Gets all sip credentials of a specific user
/// </summary>
/// <param name="userId"></param>
/// <returns>The third party Sip credentials of a specific user</returns>
public UserSipCredentials GetSipCredentials(int userId)
```

Examples

Because the Sip credential specific properties were removed from the User table it is no longer possible to set those in the **UserEntry** classes.

This is how to insert Sip credentials for a User now.

Add user and sip credentials

```
public void AddUserAndSipCredentials()
{
 var userEnum = this.libManager.GetUserEnum();
 var userEntry = new SWConfigDataClientLib.Proxies.Users.UserEntry();

 // Insert user specific data
 userEntry.Name = "Bob";
 ...

 // Add user to the database
 userEnum.PrimaryCollection.Add(userEntry)
 userEnum.Update();


 // Add sip credentials for Bob
 var sipUserId = "BobsSipUserId";
 var sipUsername = "BobsSipUsername";
 var sipPassword = "BobsSecretPassword";
 userEnum.SetSipCredentials(userEntry.UserId, sipUserId, sipUsername,
sipPassword); // The Sip credentials will be updated in the database
instantly
}
```

SwyxWare PowerShell Module

It is still possible to use the Powershell Module to get or set the SIP credentials for a specific user.

Get SIP credentials

With the following new commandlet **Get-IpPbxUserSipLogin** it is possible to receive the SIP credentials of a specific user. The SIP credentials will be returned as an object type of **SWConfigDataClientLib.Proxies.Users.UserSipCredentials** which contains the following properties:

- SIP User Id
- SIP Username
- SIP Realm
- SIP Login Allowed

Get-IpPbxUserSipLogin

```
<#
.SYNOPSIS
    Gets the SIP login credentials for an ippbx user.
.DESCRIPTION
    The Get-IpPbxUserSipLogin cmdlet returns the SIP login credentials for
a user.
.PARAMETER UserId
    Id of the user
.INPUTS
    [UserEntry] $UserEntry
    [UserAdminView1Entry] $UserEntry
    [string] $UserName
.OUTPUTS
    [UserSipCredentials] object
.EXAMPLE
    Get-IpPbxUserSipLogin -UserName "Sebastian"

    Returns the SIP credentials of user "Sebastian".
.EXAMPLE
    Get-IpPbxUser | Get-IpPbxUserSipLogin

    Returns all SIP credentials of all users.
#>
function Get-IpPbxUserSipLogin
{
 [CmdletBinding()]
    param([parameter(ParameterSetName = 'Entry',
ValueFromPipeline=$true)]$UserEntry,
          [parameter(ParameterSetName = 'Name')][string]$UserName)

    #...
}
```

Set SIP credentials

The commandlet **Set-IpPbxUserSIPLogin** has been modified unterally, but its interface is unchanged, except the parameters are mandatory now.

Set-IpPbxUserSipLogin

```
<#
.SYNOPSIS
    Sets the SIP login credentials for a user.
.DESCRIPTION
    The Set-IpPbxUserPhoneLogin cmdlet sets the SIP login credentials for a
user.
.PARAMETER UserEntry
    [UserEntry] or [UserAdminView1Entry] object of the user.
.PARAMETER UserName
    Name of the user.
.PARAMETER SIPUserId
    New SIP UserId.
.PARAMETER SIPUserName
    New SIP Username.
.PARAMETER SIPPassword
    New SIP Password.
.PARAMETER EnableLogin
    Enable or disable the SIP Login.
    If parameter is not provided the current state will not be changed.
.INPUTS
    [UserEntry] $UserEntry
    [UserAdminView1Entry] $UserEntry
    [string] $UserName
.OUTPUTS
    None
.EXAMPLE
    Set-IpPbxUserSipLogin -UserName "Sebastian" -SIPUserId "Sebastian"
-SIPUserName "Sebastian" -SIPPassword "mypass" -EnableLogin

    Sets and enables the SIP login for user "Sebastian".
.EXAMPLE
    Set-IpPbxUserSipLogin -UserName "Sebastian" -EnableLogin:$false

    Disables the SIP login for user "Sebastian".
.EXAMPLE
    Get-IpPbxUser | Set-IpPbxUserSipLogin -EnableLogin:$false

    Disables the SIP login for all users.
#>
function Set-IpPbxUserSipLogin
{
    [CmdletBinding(SupportsShouldProcess=$True)]
    param([parameter(ParameterSetName = 'Entry',
ValueFromPipeline=$true)]$UserEntry,
          [parameter(ParameterSetName = 'Name')][string]$UserName,
          [Parameter(Mandatory=$true)][string]$SIPUserId,
          [Parameter(Mandatory=$true)][string]$SIPUserName,
          [Parameter(Mandatory=$true)][string]$SIPPassword,
          [switch]$EnableLogin)

    #...
}
```