# Swyx ConfigDataStore (CDS) Client SDK

## *Introduction*

The **CDSClientLib** is the client component of the SwyxWare ConfigDataStore. The CDS uses the Windows Communication Foundation (WCF) as middleware to provide access to the CDS Service and to the underlying SQL database.

The ConfigDataStore component is part of the SwyxWare and manages all configuration data of the PBX like users, groups, trunks, locations, etc.

The CDSClientLib consists of a couple of NET assemblies written in C#. Additionally, there is a native C++ wrapper around the API, which can be used by unmanaged applications.

The interface is provided as it is. With using the interface, the user accepts that Swyx does not take any warranty nor is responsible for any damage being caused by using it. Swyx provides technical support through the Technology Partner Program.

See the API documentation for more details: documentation\CDSClientAPI.chm.

## *Changes with SwyxWare v12.00*

The references folder is not a part of this SDK anymore. The referenced assemblies are included in the NuGet packages which are now consumed by the sample application. So, it is necessary to add the following NuGet packages to your solution/project:

| | |
|---|---|
| Swyx.ConfigDataStoreClient.Managed<br>*Current Version: 12.3.8* | Contains the IpPbxCDSClient.dll library<br>Contains the CDS Client API |
| Swyx.ConfigDataStoreClient.Blob<br>*Current Version: 12.3.8* | Contains the IpPbxBLOB32.dll<br>(De-)Serializer |
| Swyx.ConfigDataStoreClient.Common<br>*Current Version: 12.3.8* | Contains the following DLL's:<br>IpPbx.Configuration.Model.dll<br>IpPbxCDSSharedLib.dll<br>IpPbxCDSSharedLib.resources.dll<br>IpPbxTracing.dll<br>IpPbxWin32.dd<br>Xceed.Compression.dll |

**There are temporarily the following NuGet packages with fix versions added:**

- **Microsoft.IdentityModel.Tokens v5.5.0.0**
- **Newtonsoft.Json v12.0.1**
- **System.IdentityModel.Tokens.Jwt v5.5.0.0**

**These additionally added NuGet packages will be removed with the next SwyxWare version.**

## *Contents*

| | | |
|---|---|---|
| Documentation | CDSClient.chm<br>BestPractice.pdf<br>Readme.pdf | The API documentation<br>Best Practices to use the CDS Client<br>This readme document |
| Policy | ApplicationName.exe.config | Template of the client application config file. Copy the content of this file into the app.config file in your project. |
| Samples | | Sample projects showing CDS client usage |
| IpPbx.cer | | X509 Certificate Add this file to your project and set the “Copy to Output Directory” property to “Copy if newer”. |

## *Prerequisites*

| | |
|---|---|
| **Operating System** | Windows 7<br>Windows 8.1<br>Windows 10<br>Windows Server 2012 R2<br>Windows Server 2016<br>Windows Server 2019 |
| **.NET** | .NET Framework 4.7.2 |

## *Main Points of Interest*

### LibManager

The CDS Client LibManager is the central class of the CDS Client API. It contains methods to address the server and to authenticate users. You use LibManager to create instances of classes for retrieval and manipulation of PBX configuration items.
The LibManager should be the first class to instantiate. It checks the configuration and initiates the tracing of the client application.

### Configuration Item Enumerations

These enumeration objects allow retrieval and enumeration of configuration objects. These objects can be modified and update the database. Configuration item enumerations may contain sub enumerations. The top most enumerations are called *Primary Configuration Item* Enumerations. For each Primary Configuration Item Enumeration exists a proxy class on the client side which is created by using LibManger. The name of these proxy classes end with 'Enum' always. The following Primary Configuration Items Enumerations exist:

- **UserEnum**
  WebService Proxy for managing PBX users, internal numbers,
  phonebooks, call lists etc.
- **InternalNumberEnum**
  WebService Proxy for managing internal and public numbers and their mappings.

- **GroupEnum**
  WebService Proxy for managing PBX groups and their members
- **PortBundleEnum**
  WebService Proxy for managing trunks, trunk groups and the routing configuration
- **LocationEnum**
  WebService Proxy for managing the locations of the PBX
- **EditablePhoneBookEnum**
  Used to retrieve and manipulate the SwyxServer's phonebook.
- **UserPhoneBookEnum**
  Used to retrieve and manipulate a SwyxWare user's phonebook.

## Facades

Facades are web service proxies similar to the primary configuration enumerations. They contain specialized views of the configuration data, which are read only. A facade provides specialized methods to manipulate the data directly. Views in facades are optimized for their specific use and usually much faster and need less memory than using the enumerators.

- **IppbxServerFacade**
  Web service proxy for SwyxServer service (IpPbxSrv). This facade contains a specialized trunk view, methods to get routing information and methods to change dynamic data directly (for example logged in devices) etc.
  **It is meant for usage from SwyxServer service only**

- **PortManagerFacade**
  Web service proxy for Gateway and LinkManager. This facade contains a specialized trunk view.

- **AdminFacade**
  Web service proxy for administration clients. It contains specialized views analogue to the views used in the SwyxWare Administration. **If your application does any administrative tasks, look at this object first.**

- **PhoneClientFacade**
  Web Service proxy for SwyxWare clients. It contains specialized views and methods to be used from client application. A PhoneClientFacade instance is implicitly associated to the SwyxWare user specified in the lib manager.

- **FilesFacade**
  Web service proxy for managing all files stored in SwyxServer's database.

## Authentication

The Authentication is managed by the LibManager class. There are three ways of user identification: Kerberos, Username Password and Trusted. SwyxWare currently uses username/password and Trusted only. The latter one automatically uses the Windows account the user is currently logged in at his system. Both authentication mechanisms can be switched on in each user's properties via SwyxWare Administration.

For administrative tasks the user needs to have assigned an appropriate administrative profile. This can be done via SwyxWare Administration.

## *Samples*

There are some simple sample projects in this package.