Best Practices Using CDS Client
This document will give some hints about how to use the CDS Client API efficiently.

The CDS Client API allows configuring SwyxWare. For that purpose, the CDS provides a set of web services. For each web service the CDS Client contains a client proxy class.

- **Minimize amount of CDS calls**:
  Each web service call causes a certain overhead. Therefore, try to minimize the amount of calls. For example, avoid invoking a method for each user to retrieve certain information. In most cases, there is another method to retrieve the data of several users by one invocation. See the difference between “Enum” proxies and “Facade” proxies below.

- **Reduce the set of used CDS Client proxies.**
  Try to identify the proxies that will be needed for you program:
  Recommendation:
  - Programs running in the client environment should use
    - PhoneClientFacade
    - FilesFacade
    - UserPhoneBookEnum
  - If you want to administrate SwyxWare you should use:
    - AdminFacade
    - UserEnum
    - GroupEnum
    - PortBundleEnum
    - LocationEnum
    - EditablePhoneBookEnum
    - FilesFacade
  - Following proxies should not be used by third party applications:
    - IpPbxServerFacade
    - PortManagerFacade

- **Understand the authentication system.**
  The CDS Client API does not use a classic login mechanism. The communication between CDS client and server is stateless. As a result, each web service call will be authenticated separately.
  The server address, type of authentication and the user credentials are properties of the central LibManager class. When creating a new proxy object, the LibManager attaches this information to the new proxy client object and an initial web service call will be made to contact the server.
  Each web service request (CDS calls) will be authorized by using this security information. The initial call checks if the service is available. In most cases the authorization will be checked by this call also. (Not in AdminFacade and PhoneClientFacade)

Consequences:

- If you want to change the server destination or the user credentials you have to change the information in the LibManager object and in all existing proxy objects.
- After a connection interruption there is nothing to do to re-establish the connection.

- **Understand the difference between "Enum" proxies and "Façade" proxies**
  The CDS Client API contains two different types of web service client proxies.
  - Each "Enum" proxy class is responsible for one primary configuration item (User, PortBundle etc.). The primary configuration item enumerations can contain sub enumerations.
    How to work with "Enum" proxies:
    - Receive a set of configuration data by specifying filter criteria. (Users, Groups, PortBundles etc.)
    - Modify the data on client side.
    - Write back the configuration to the server using update methods. If you call Update() on the enum objects all objects in that enumerator will be written back to the server and update the database.
  - "Facade" proxy classes are providing specialized, read-only views of the configuration data. A facade also provides specialized methods to manipulate the data directly, i.e. to perform specific tasks. Facade views need less performance (memory and cpu load).

- **Use locally collection classes for sorting and additional filtering**.
  Every collection class in the CDS Client API contains a method to create a sortable collection. This collection object can be easily sorted by any attribute; it can contain additional filter operations and can be bound directly to data grid controls.

- **Use the version check mechanism**
  The AdminFacade and the PhoneClientFacade provide a CheckVersion method.
  It checks if the used CDS Client API is compatible with the server. If that function fails, CDS client and server are not compatible. You should terminate the client application in that case. If it has a user interface, inform the user appropriately.

- **Use the GlobalConfigItem class to persist additional data used by your application**
  It is possible to persist third party data by using the GlobalConfigItem class. It offers a generic configuration data table to store named data items of typical simple types like integer, boolean or string.

- **Exception handling and localization of exception messages**
  all exception classes in namespace SWConfigDataSharedLib.Exceptions contain a property called 'Localized'. If true, the exception message text is localized in the languate defined in the client application config file:

```
<SWConfigDataSharedLib.Properties.Settings>
      <setting name="Culture" serializeAs="String">
            <value>en</value>
      </setting>
</SWConfigDataSharedLib.Properties.Settings>
```

Usually SwyxWare clients are single-language, so that only the satellite assembly (IpPbxCDSSharedLib.resource.dll) for one language is present. Make sure that your application configuration file setting matches that language or keep English which is available always.