SYSPRO WCF Services Client Libraries

Overview

The SYSPRO WCF Services Client Libraries provide an easy to use and configurable interface with which to call the SYSPRO WCF Service from managed code. There are separate libraries for use with the .Net Framework 2.0 and the .Net Framework 4.0. The Framework 2.0 library only supports HTTP REST calls to the service while the .Net Framework 4.0 library supports calls using both the HTTP Rest Endpoint along with the Soap endpoint which can be used with a number of communication methods (ie. HTTP, NetTcp etc.). To make use of the SYSPRO WCF Service Client Libraries simply make reference to either the SYSPROWCFServiceClientLibrary.dll (for Framework 2.0 projects) or the SYSPROWCFServiceClientLibrary40.dll (for Framework 4.0 projects) library in you .Net project. Both libraries are located in the SYSPRO Base directory.

Both libraries consist of the same classes and available methods described below however only the *SYSPROWCFServiceClientLibrary40.dll* library supports a number of communication bindings (For SOAP communication).

SYSPRO WCF Services Primitive Client Class

Description

The SYSPRO WCF Services Primitive Client Class provides a number of methods providing complete access to the SYSPRO E.Net Layer through the SYSPRO WCF Service. Each method can also be called asynchronously allowing the client to call the service and carry on performing other tasks until the call back from the service is received. The user must first create an instance of the SYSPRO WCF Services Primitive Client Class, specifying the Base Address of the service along with the communication Binding to use. It is noted that if a Soap binding is specified it must correspond with the binding setup for the SYSPRO WCF Service Install. It is also noted asynchronous operations are currently only available for use with the REST HTTP binding.

Methods

Constructor

```
SYSPROWCFServicesPrimativeClient(String BaseAddress, SYSPROWCFBinding binding)
```

- **BaseAddress:** The address of the SYSPRO WCF Service to call. E.g. http://localhost/SYSPROWCFService. It is noted that *localhost* should be replaced with the name of the server and the *http://* instruction should reflect the communication binding in use. I.e. *net.tcp://* for NetTcp binding.
- **SYSPROWCFBinding:**
  The Binding Selected must correspond with the binding selected on the server.
  - **Binding Types** *(For the Framework 2.0 Library only the RESTHttp binding is available.)*
    - **RESTHttp:** Calls are made to the REST endpoint of the service using

- **NetNamedPipe:** Calls are made to the SOAP endpoint of the service using
  Named Pipes. This is the fasted form of communication but is only
- suitable for single machine use.
  NetTcp: Calls are made to the SOAP endpoint of the service using Net
  Tcp. This is recommended binding to use and is best suited for intranet
- environments.
- **BasicHttp:** Calls are made to the SOAP endpoint of the service using
  HTTP. **WSHttp:** Calls are made to the SOAP endpoint of the service using

Logon Method

Before using SYSPRO WCF Services a logon must be performed to provide a userid with which to make subsequent calls. Logon must be called along with the user credentials that will be used for the logon.

There are two logon methods available. The first requires all parameters to be passed while the second uses defaults (or the value setup in the configuration file of the SYSPRO WCF Service itself) for the last four parameters.

String Logon(String OperatorName, String Password, String CompanyId, String CompanyPassword, String LanguageCode, String LogLevel, String EncoreInstance, String XmlIn)

String Logon(String OperatorName, String Password, String CompanyId, String CompanyPassword)

Logoff Method

After performing a logon and any operations using the userid, a logoff must be

String Logoff(String ENetGuid)

Query Methods

Browse, **Fetch** and **Query** calls can be made using SYSPRO WCF Services Client

String   QueryQuery(String UserId, String BusinessObject, String XmlIn)

String   QueryBrowse(String UserId, String XmlIn)

String   QueryFetch(String UserId, String XmlIn)

Setup Methods

Add, **Delete** and **Update** calls can be made using SYSPRO WCF Services Client

String   SetupAdd(String UserId, String BusinessObject, String XmlParameters, String XmlIn)

String   SetupDelete(String UserId, String BusinessObject, String XmlParameters, String XmlIn)

String   SetupUpdate(String UserId, String BusinessObject, String XmlParameters, String XmlIn)

Transaction Methods

Add, Delete and Update calls can be made using SYSPRO WCF Services Client

Post and **Build** calls can be made using SYSPRO WCF Services Client

```
String    TransactionBuild(String UserId, String BusinessObject, String XmlIn)

String    TransactionPost(String UserId, String BusinessObject, String XmlParameters,
String    XmlIn)
```

Synchronous Calls

All of the methods documented can be called synchronously. The synchronous methods call the service and wait for a response before allowing execution to continue. The Following C# code snippet shows a synchronous query to SYSPRO. When the query is completed its result is displayed in a message box.

```
private void btnQuery_Click(object sender, EventArgs e)
{
        try
        {
            //Instantiate the Primitive Client using the BaseAddress of the service and the binding to use
            SYSPROWCFServicesClientLibrary40.SYSPROWCFServicesPrimitiveClient clientPrim = new
            SYSPROWCFServicesClientLibrary40.SYSPROWCFServicesPrimitiveClient(BaseAddress,
            CurrentBinding);
            //Call the Synchronous method for the Query and assign the output to a variable String
            XmlOut = clientPrim.QueryQuery(ENetGuid, BusinessObject, XmlIn);
            //Finally display the query output in a message box.
            MessageBox.Show(XmlOut);
        }
        catch (Exception ex)
        {
            throw (ex);
        }
}
```

Asynchronous Calls

All of the methods documented can be called asynchronously. This means that a client application can initially call a service method, the client application can then continue executing other tasks while the service processes the method. When the method completes, a call back event on the client is fired and the client application can then proceed with processing the output result.

This can be useful when calls to the service are expected to take sizeable amount of time to execute, such as a detailed query. The client can also queue up a number of requests to the service and then deal with the responses when they return, without having to worry about the connection with the service timing out.

The Asynchronous methods in the Client Libraries are the same as the methods specified above but including the naming convention *Async* after the method name. E.g. *LogonAsync*. The callback events to subscribed to are named similarly and instead end with *CallBack.* E.g. *LogonCallBack.* So finally to summarize: In order to call the Logon method asynchronously, firstly call the LogonAsync method and subscribe to the event *LogonCallBack* to process the output of the call.

Calling the service synchronously however a callback event must be included. The Following C# code snippet shows an asynchronous post to SYSPRO. When the query is completed its result is displayed

```
private void btnPost_Click(object sender, EventArgs e)
{
        //Instantiate the Primitive Client using the BaseAddress of the service and the binding to use
        SYSPROWCFServicesClientLibrary.SYSPROWCFServicesPrimitiveClient clientPrim = new
        SYSPROWCFServicesClientLibrary.SYSPROWCFServicesPrimitiveClient(BaseAddress,
        CurrentBinding);
        //Subscribe to the Post Callback event clientPrim.TransactionPostCallBack
        += new
        SYSPROWCFServicesClientLibrary.SYSPROWCFServicesPrimitiveClient.CallbackDelegate(TransactionP
        o   stCallBack);
        //Finally call the Asynchronous method for Posting
        clientPrim.TransactionPostAsync(ENetGuid, BusinessObject, XmlParameters, XmlIn);
}
private void TransactionPostCallBack(string CallOutput)
{
        //This method will execute when the Post is completed. CallOutput contains the output of the
        //Post.
        MessageBox.Show(CallOutput);
}
```

SYSPRO WCF Services Client

The SYSPROWCFServicesClient class provides all of the functionality of the Primitive client in a slightly more easy to use fashion. It encapsulates logons and logoffs and also manages the current E.Net Guid, making it more developer friendly. It does not however include the asynchronous methods.

To you use the SYSPROWCFServicesClient, create an instance of the class passing through the BaseAddress of the service, the binding to use for communication and finally the SYSPRO credentials with which to logon. The instance can then be used to call of the all of the above methods in a similar fashion only without the need for the E.Net Guid, which is managed

The Following C# code snippet shows a synchronous query to SYSPRO. When the query is completed

```
private void btnQuery_Click(object sender, EventArgs e)
{
      try
      {
        //Instantiate the Full Client using the BaseAddress of the service and the binding to use
        //Also pass in the SYSPRO Credentials for a Logon
        //A using statement ensures the full client is disposed when it is no longer is in scope
        //The disposal performs a logoff
        using (SYSPROWCFServicesClientLibrary40.SYSPROWCFServicesClient fullClient = new
        SYSPROWCFServicesClientLibrary40.SYSPROWCFServicesClient(BaseAddress, CurrentBinding,
        Operator, Password, Company, CompanyPassword))
        {
              //Call the Synchronous method for the Query and assign the output to a variable
              String XmlOut = fullClient.QueryQuery(BusinessObject, XmlIn);
              //Finally display the query output in a message box.
              MessageBox.Show(XmlOut);
        }
     }
     catch (Exception ex)
     {
        throw (ex);
     }
}
```

Conclusion

The SYSPRO WCF Services Client Libraries provide a standard means with which to communicate with the SYSPRO WCF Service from any .Net application. The libraries can be used with any of the .Net Frameworks above and including 2.0, however the .Net Framework 4.0 is required to make use of Soap calls and configurable bindings. All methods can be called both synchronously and asynchronously with either the REST or Soap bindings.