Introduction to OPC

OPC (OLE for process control) has been around for some time. It is a set of standards maintained by the OPC Foundation that are used in the automation industry (i.e. factories) for integrating devices and control process' into the office environment (it is a piece of middleware for getting data from your robot arm to your spread sheet). It has become a de facto standard for the automation industry.

The OPC architecture is based on the client/server paradigm, the protocol allowing a client to access data (OPC items) on demand or though a publish/subscribe mechanism. A good introductory tutorial to OPC provided by Matrikon can be found here.

Design

The goal of this project is to provide a set of wrappers which hide a details of DCOM from the OPC client developer whilst providing a set of OPC abstractions that are useful and intuitive to that developer.

Download the OPC Client SDK

Version 0.4 of the OPC client SDK supports the following functionality,

- Flat namespace browsing.
- Synchronous/Asynchronous reads (at the item and group level).
- Synchronous/Asynchronous writes (item level only).
- Asynchronous notification of changes to OPC item values.
- Obtaining OPC item properties and their values.
- The group refresh operation.

The distribution is maintained on Sourceforge and can be obtained here. It contains:-

1. An OPC DA client library (static).
2. A small demonstration program showing how to use all the library features.
3. A program for measuring the performance of an OPC server.

Developing with the SDK

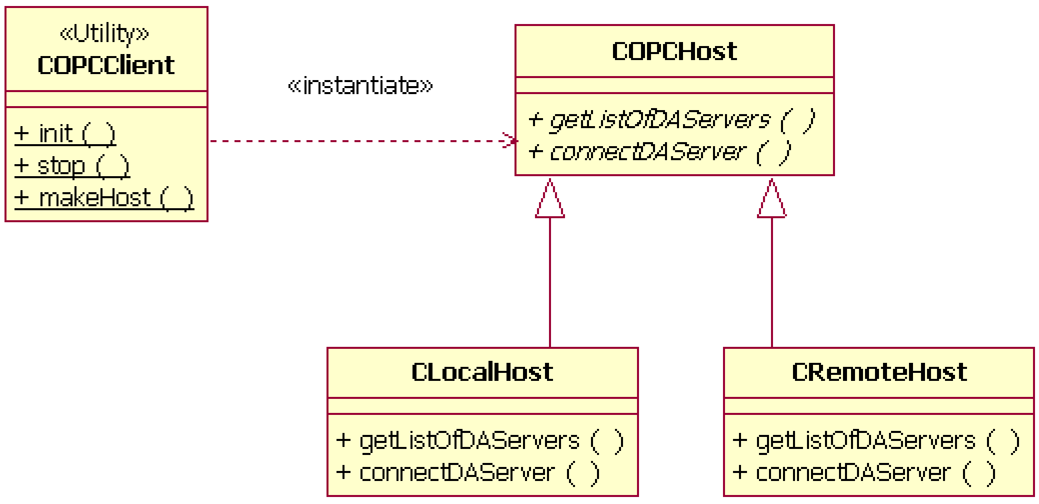
In my experience, its a good idea to start simple.

1. Download a OPC simulator, install it locally, get the client working with this (this should be simple).
2. Next move the simulator into a remote host, use the CRemoteHost object to connect to the server (this is harder, make sure you know what dcomcnfig.exe is).
3. Now modify the client to access you target server.

Personally, I can get the client tool kit working locally and remotely where both machines are in the same domain. At the moment I'm having trouble where both machines are in the same workgroup (yes - I have same ID's and passwords on both machines) - I will provide updates as I solve the problem.

Please subscribe to the OPC client users mailing list, to get information about new releases, bug fixes and so forth.

Key class'

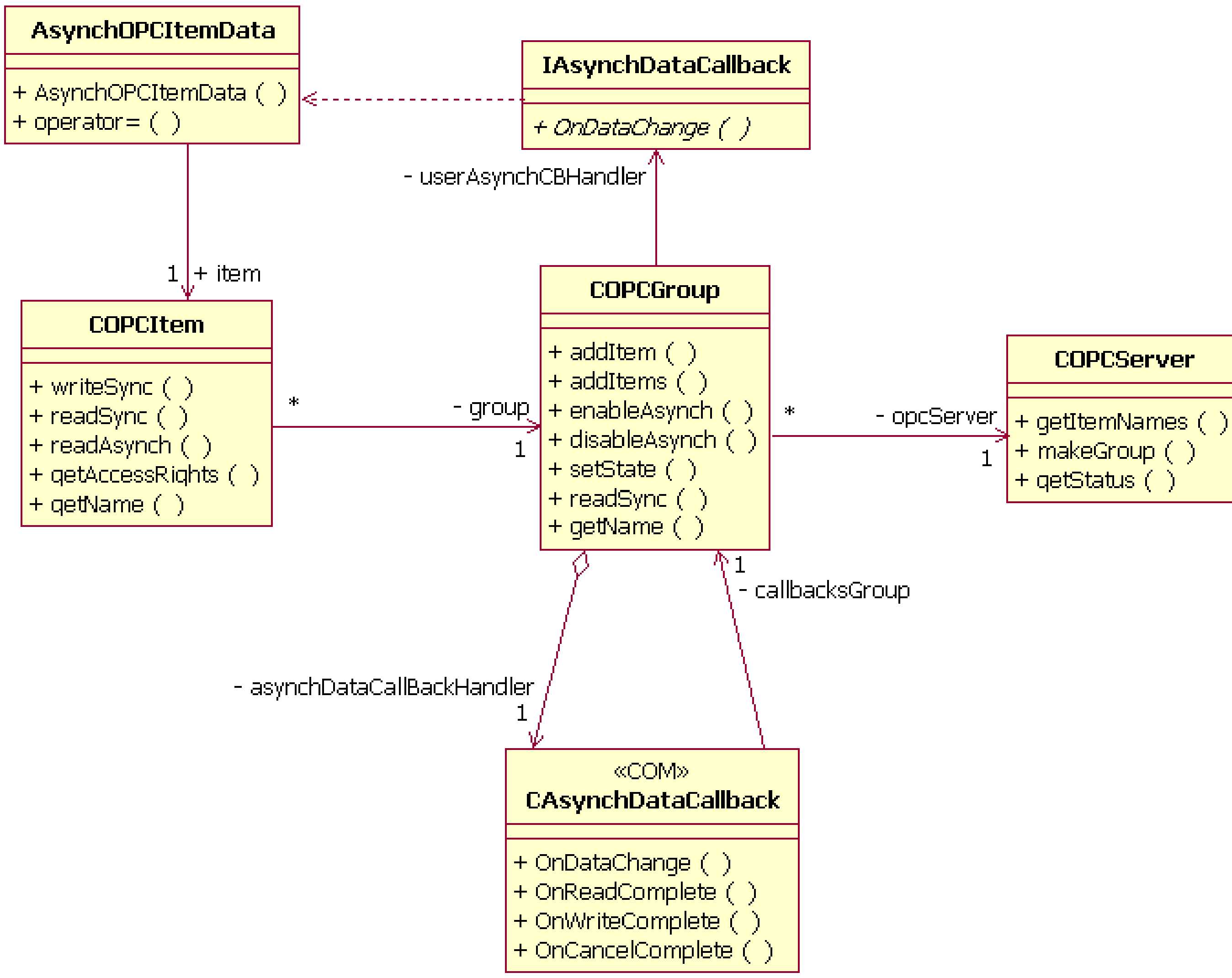


COPCClient - Starting point for 'everything'. Utility class that creates host objects and handles COM memory management. Effectively a singleton.

COPCHost - Abstract class that represents a PC which may host one or more OPC servers. Provides means of getting a list of OPC servers on the host and creating connections to OPC servers.

CLocalHost - Used for accessing OPC servers on a local (this) host. Uses the Component Categories manger to enumerate the OPC servers on the local machine.

CRemoteHost - Used for accessing OPC servers on a remote host. Make use of OPCenum to browse servers.



COPCServer - Local representation of a local or remote OPC server. Wrapper for the COM interfaces to the server.

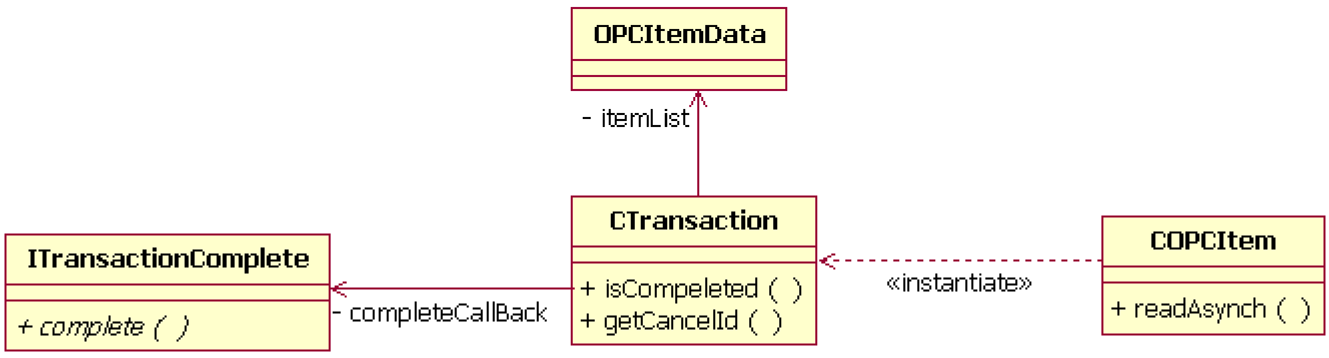
COPCGroup - Client sided abstraction of an OPC group, wrapping the COM interfaces to the group within the OPC server.

COPCItem - Provides wrapper for operations that typically exist at the group level (e.g. reads) (it is at this level that OPC supports the operation) however, we provide the operation at this level for ease of use.

CAsynchDataCallback - Handles OPC (DCOM) callbacks at the group level. It deals with the receipt of data from asynchronous operations. This is a fake COM object.

IAsynchDataCallback - Data received from the OnDataChange() method of the CAsynchDataCallback instance is delegated to an instance of a child class implementing this interface. The Child class must obviously provide the desired behavior in the overridden OnDataChange() method. This interface is active only when the corresponding group is active (achieved by the groups enableSynch() method.)

AsynchOPCItemData - Used to represent OPC data passed via the asynch route. Same as OPCItemData, except contains a pointer to the OPC item the data relates to.

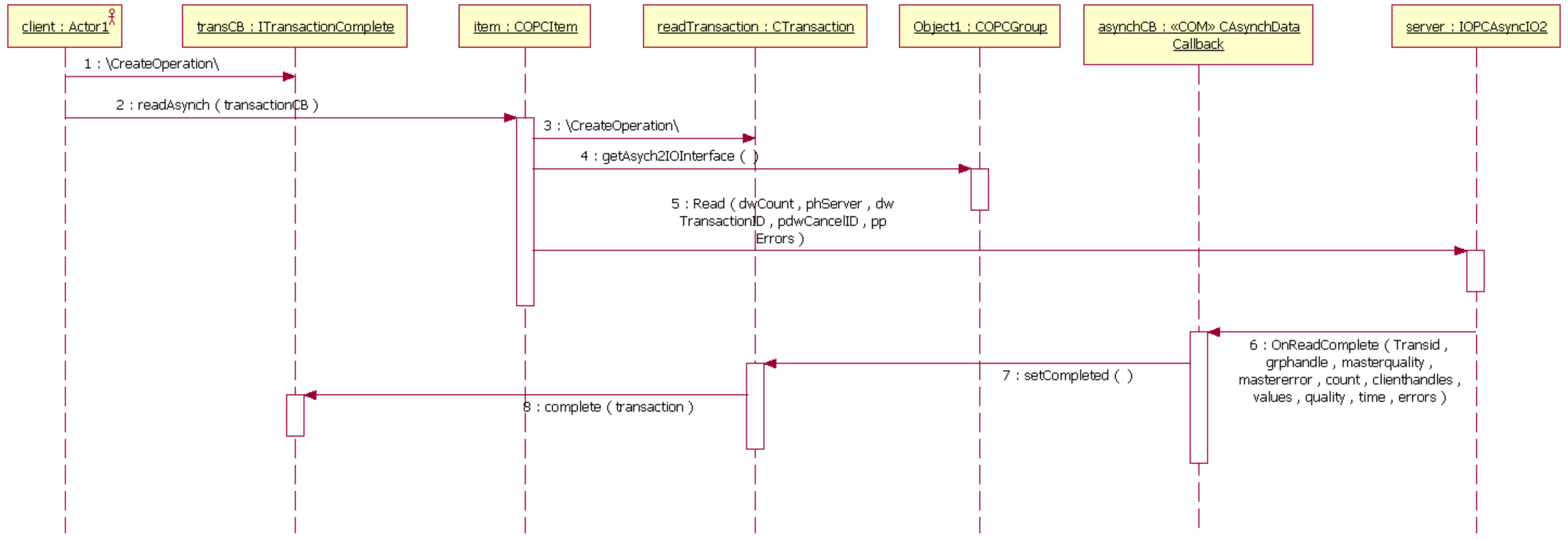


OPCItemData - Wrapper for OPC data. Hides COM memory management.

CTransaction - Used to indicate completion of an asynchronous operation. Will contain the results of that operation.

ITransactionComplete - Interface which provides means by which the client can be notified when an asynchronous operation is completed. The implementer must implement this interface overriding the complete method to provide the desired behavior. An instance of the implementation can be passed as a parameter to an asynchronous operation and used as a means of completion notification.

Often, it is hard to understand how objects interact with one another (particularly so when using an event driven approach). To give you a feel for whats going on, I've included below a basic sequence diagram for perhaps one of the most complex interactions that the OPC client SDK provides, this being an asynchronous read (in this instance of a single OPC item).


client : Actor1
transCB : ITransactionComplete
item : COPCItem
readTransaction : CTransaction
Object1 : COPCGroup
asynchCB : «COM» CAsynchData Callback
server : IOPCAsyncIO2
1 : \CreateOperation\
2 : readAsynch (transactionCB)
3 : \CreateOperation\
4 : getAsych2IOInterface ( )
5 : Read (dwCount , phServer , dw TransactionID , pdwCancelID , pp Errors)
6 : OnReadComplete (Transid , grphandle , masterquality , mastererror , count , clienthandles , values , quality , time , errors)
7 : setCompleted ( )
8 : complete (transaction)