4 Access to devices

4.3 OLE for Process Control (OPC)

4.3.2 Data Access Specification

Prof. Dr. H. Kirrmann
ABB Research Centre, Baden, Switzerland

OPC DA: Overview

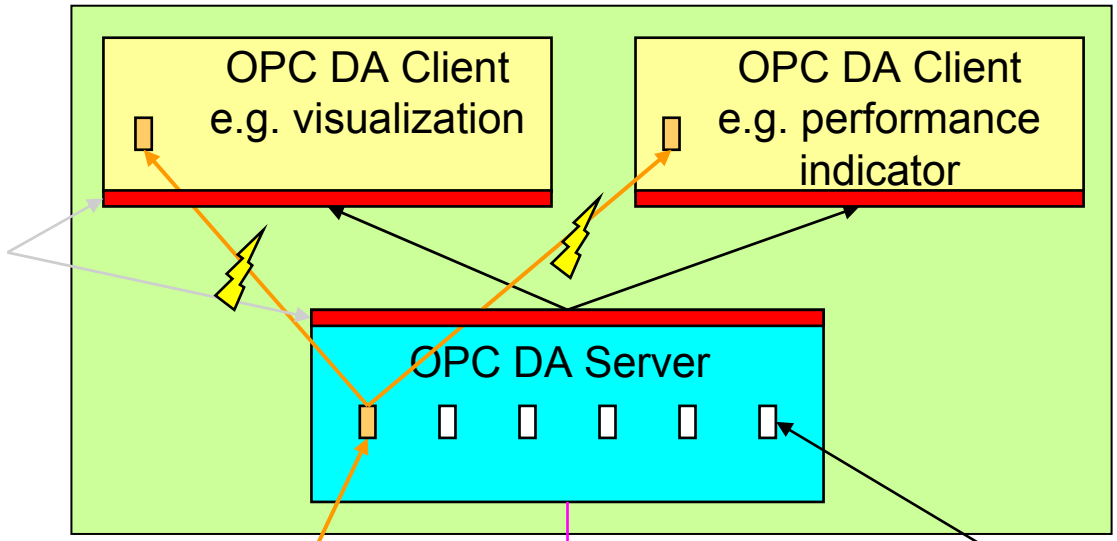
OPC Common
- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access
- **Overview: Browsing the server**
- Objects, Types and properties
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification
- Overview

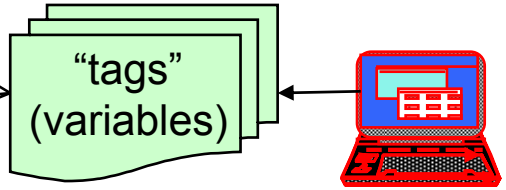
OPC DA: Scope of specification



OPC DA specified interface

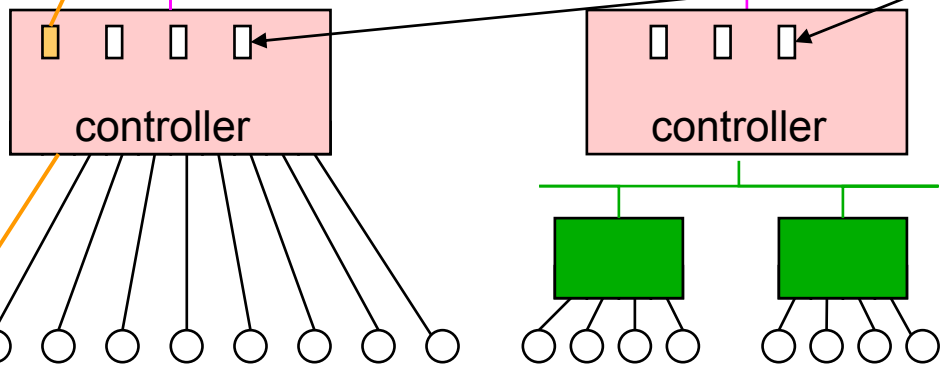
An OPC DA Server is configured using the information coming from the development tools for the controllers

Variables defined in the controllers are mirrored to the OPC DA server



e.g. Ethernet

controllers



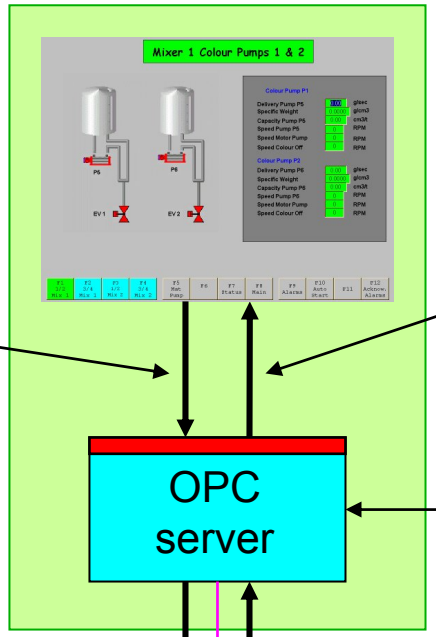
field bus

field devices

plant measurement points

OPC DA: Example of access to a variable

OPC application



ReadItem ("OPC:Reactor1: Program2.MotorSpeed")

Value: 112

OPC server

load symbol table

controller programming

Reactor_1.Program2	
MW%1003	MotorSpeed
MW%1004	Temperature
…	….

symbols

code

Get (192.162.0.2), MW%1003)

Return (MW%1003, 112)

Network

Program 2

Reactor_1

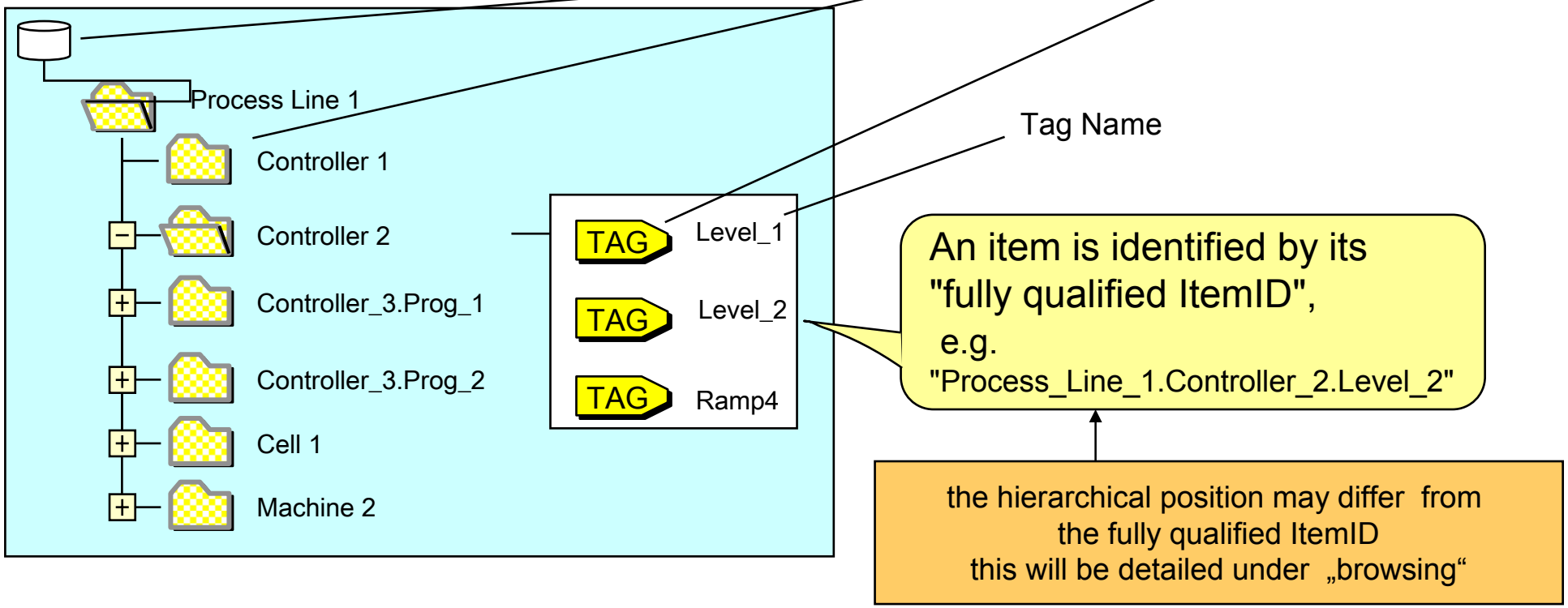
Marker: MW%1003

analog input to : IXD.11.2.1

Oven controller

OPC DA: Objects as viewed by the OPC server

An OPC server is structured as a directory with root, branches and leaves (items)



Branches may contain other branches and items
The structure may also be flat instead of hierarchical
This structure is defined during engineering of the attached devices and sensor/actors.
(Intelligent servers could configure themselves by reading the attached devices)

OPC DA: Browsing - methods

An OPC DA server presents an interface that allows the client to explore its structure, with the methods:

```
MoveDown
MoveUp
MoveToRoot


showBranches
showLeafes *
```

`GetItemID`: retrieves the fully qualified item ID (see later)

(*the English error is unfortunate)

OPC DA: Browsing: Fully Qualified ItemID and hierarchy

A server has internally two ways to access the items:

1) the path shown when exploring the tree, and
2) the „fully qualified ItemID“, which is the internal path name used by the server.

Example:
an item reached as: Root.SimulatedItems.UserDefined.Ramp.Ramp1
needs to be accessed internally as: UserDefined!Ramp.Ramp1

Clients usually search for an item though the hierarchical way.

They position the browser on the corresponding branch and retrieve the fully qualified item ID, which is the name of the item as the server understands it.

The fully qualified name is only used at configuration time, afterwards, objects are accessed over client handles and server handles (see later)

OPC DA: Object Types and properties

OPC Common
- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access
- Overview: browsing the server
- **Objects, types and properties**
- Communication model
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification
- Overview

OPC DA: Item properties

The process data are represented by three dynamic properties of an item:

value: numerical or text

time-stamp: the time at which this data was transmitted from the PLC to the server
this time is UTC (Greenwich Winter time), not local time.

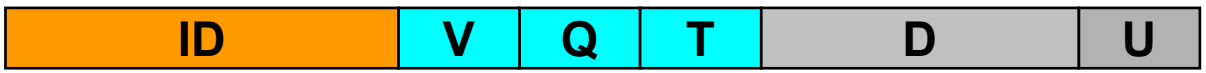
quality: validity of the reading (not readable, dubious data, o.k.)

and two optional static property:

description: a text string describing the use and of the variable (optional)

engineering unit: the unit in which the variable is expressed (optional)

(when writing, only the value is used)



OPC DA: Item types

Each item value has a type:

Boolean,
Character,
Byte, (1 byte)
Word, (2 bytes)
Double Word, (4 bytes)
Short Integer (2 bytes)
Integer (4 bytes)
Long Integer:
Long Unsigned Integer
Single Float (4 bytes)
Double Float (8 bytes)
Currency,
Date,
String,
Array of "the above"

When accessing an item, the client may request that it is returned with a specific type, which could be different from the server's type.

(The server's type is returned by browsing)

Type conversion is left to the server, there are no rules whether and how a server does the conversion. (use with caution)

Care must be taken that the data types in the programming language or in the database match those of the OPC Server.

Items also may have engineering units, but this option is not often used.

OPC DA: Objects as viewed by the OPC client

A client builds its own hierarchy, using the server's hierarchical view.

A client builds groups, populating them with items it is interested in.

Items of a group are expected to have similar real-time requirements

Items in the server are defined by the programmer of the PLC – a full fledged PLC may export some 10'000 items, a client needs only a subset.

For instance, the client can make a group for all time-critical variables to be displayed in each screen page.

Or, a client may create a group for each equipment part (and possible shut down that group when the equipment is inactive).

Each client may be interested in a different subset.

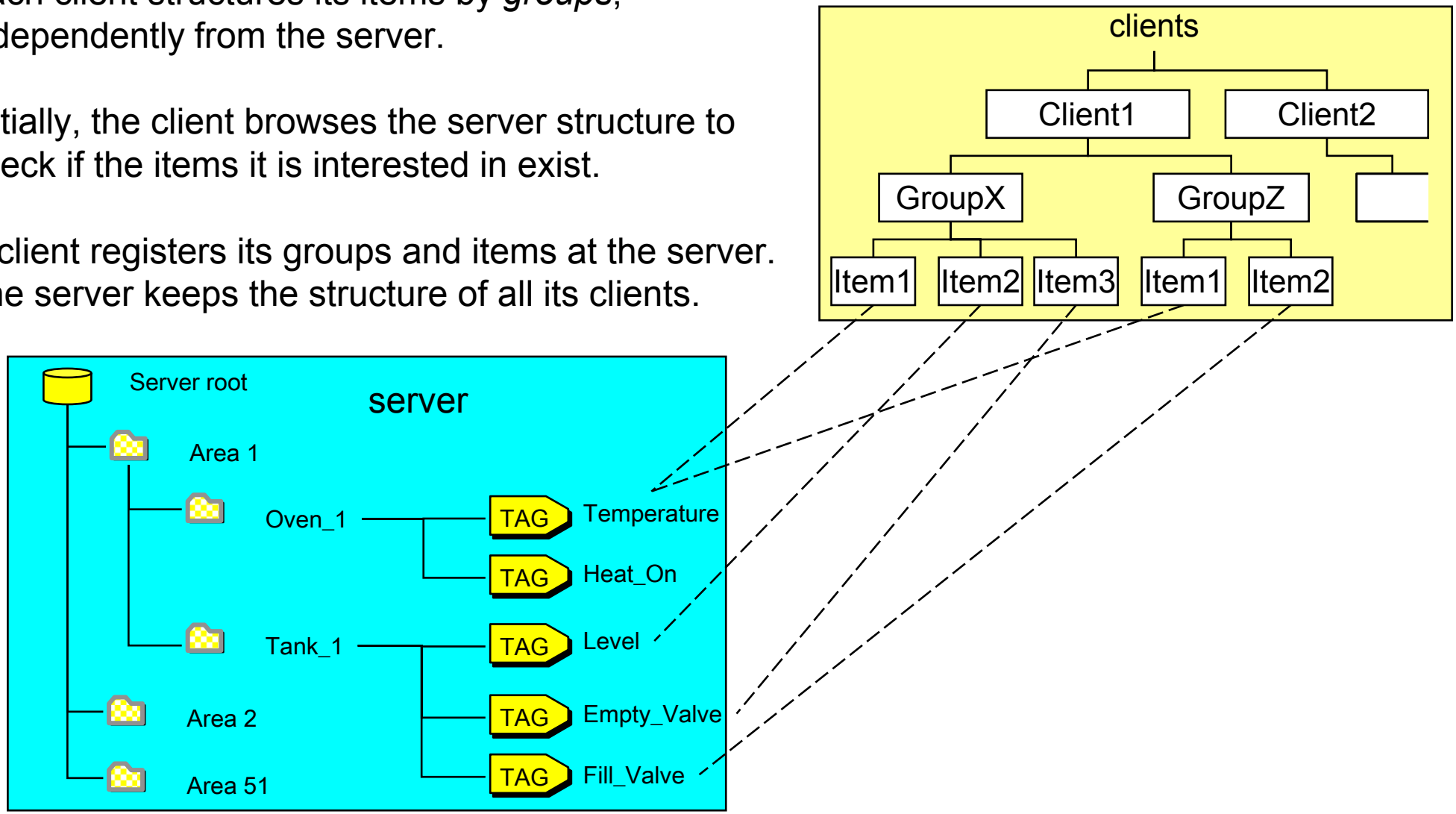
Groups are not hierarchical, but flat.

OPC DA: Mapping items to groups

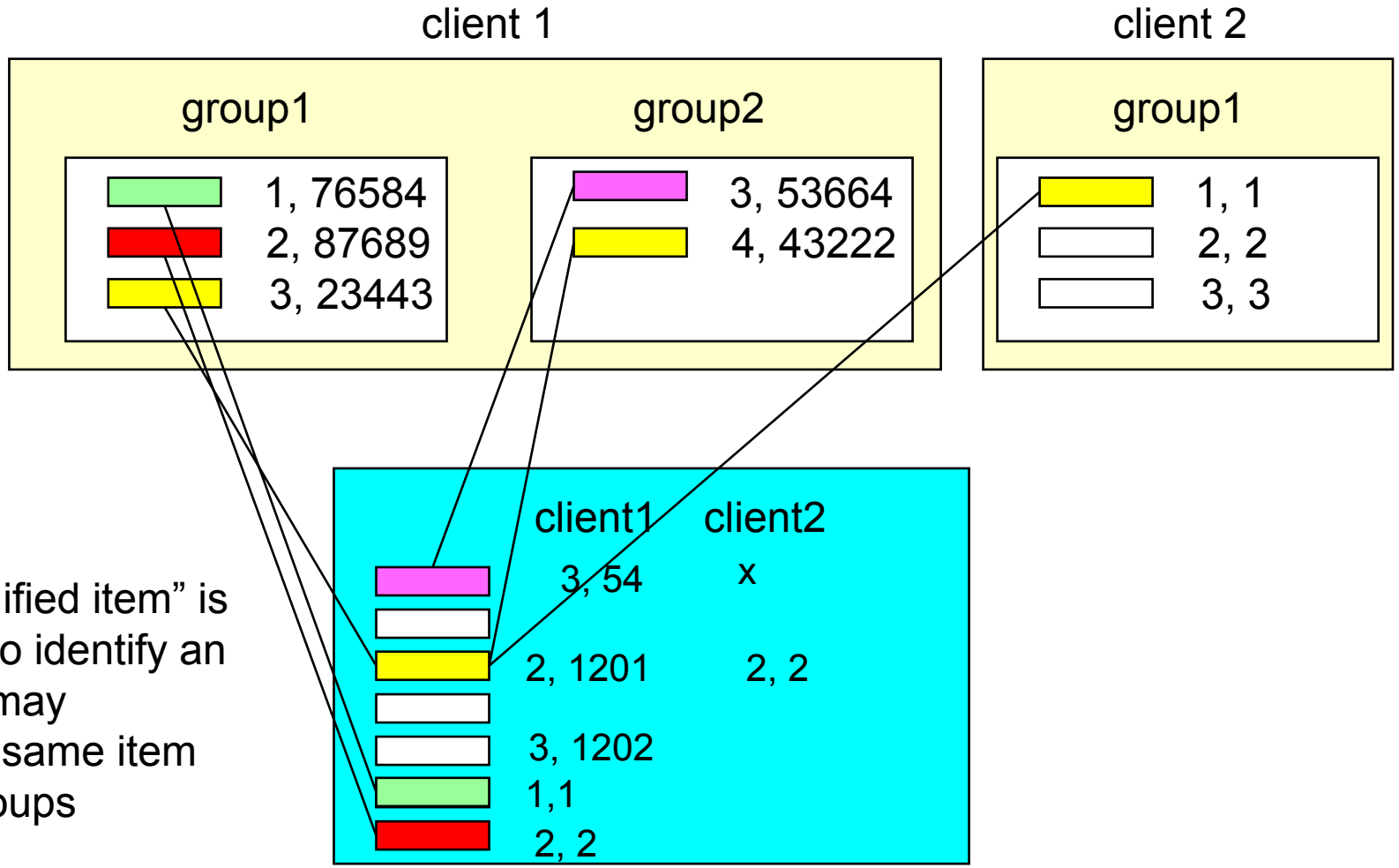
Each client structures its items by *groups*, independently from the server.

Initially, the client browses the server structure to check if the items it is interested in exist.

A client registers its groups and items at the server. The server keeps the structure of all its clients.



OPC DA: Client Handle and Server handle



The "fully qualified item" is not sufficient to identify an item, a client may subscribe the same item in different groups

The pair { ClientHandle, ServerHandle } uniquely identifies an item.

OPC DA: Communication Model

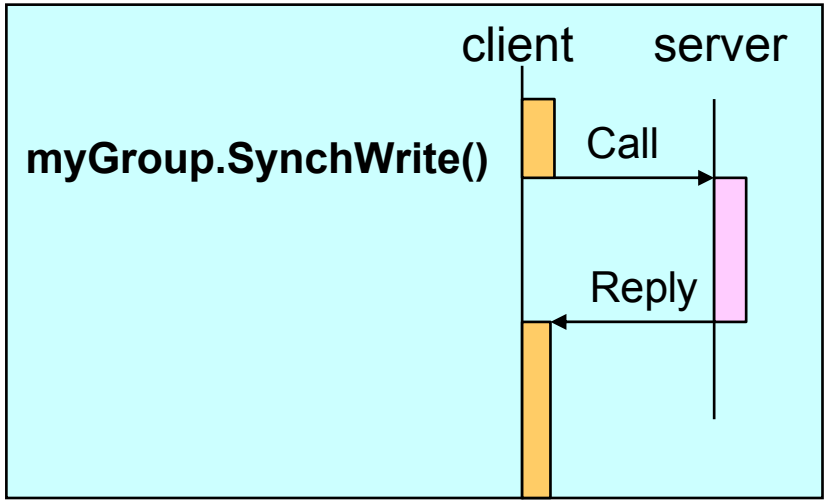
OPC Common
- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

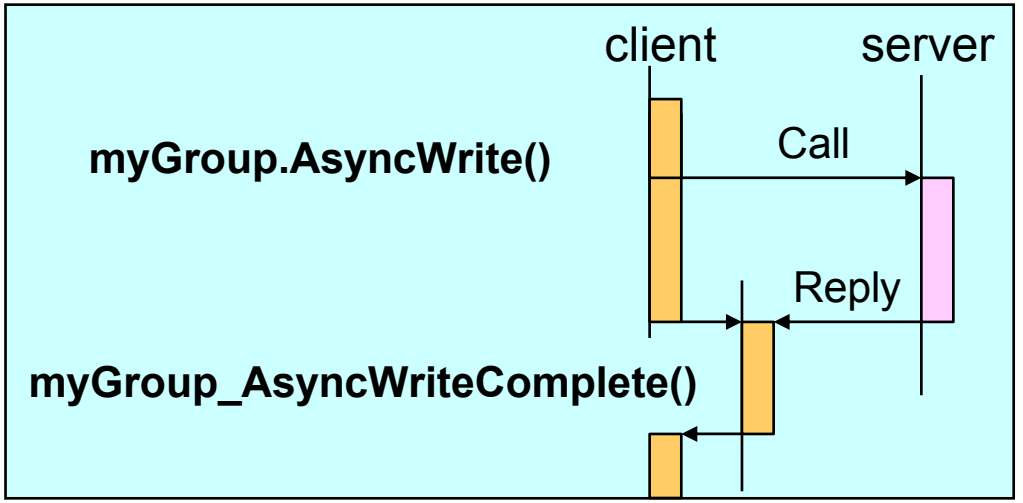
OPC Data Access
- Overview: browsing the server
- Objects, types and properties
- **Communication model**
- Simple Programming Example
- Standard and components

OPC Alarms and Events Specification
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification
- Overview

OPC DA: Write Communication Models (per group)



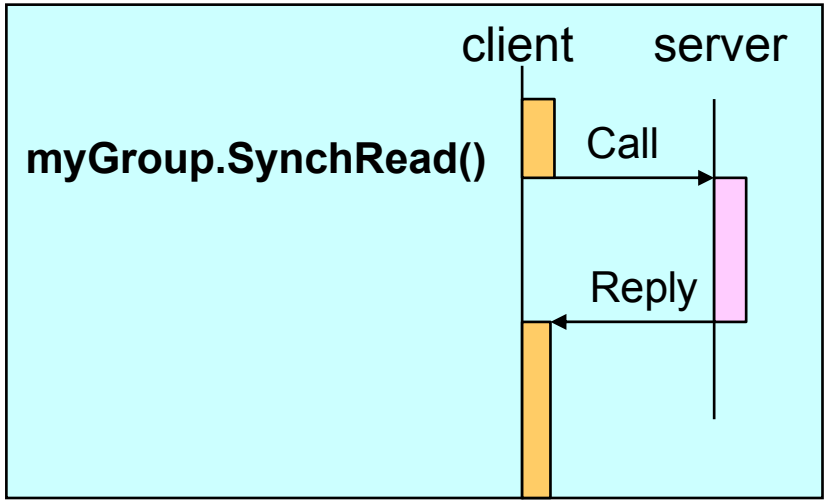


The OPC interface accesses only groups, not individual items.

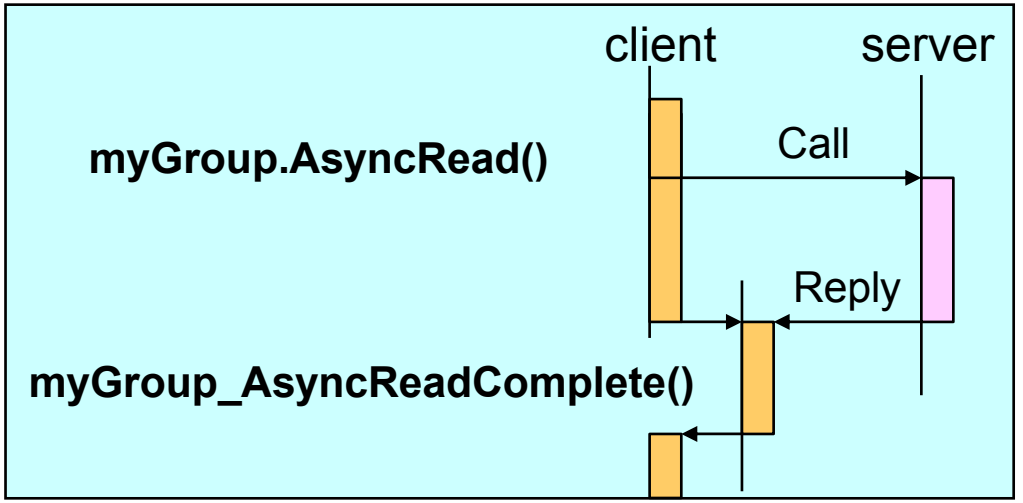
However, the "automation interface" allows to access individual items, but this does not give rise to a communication

OPC DA: Read Communication Models (per group)

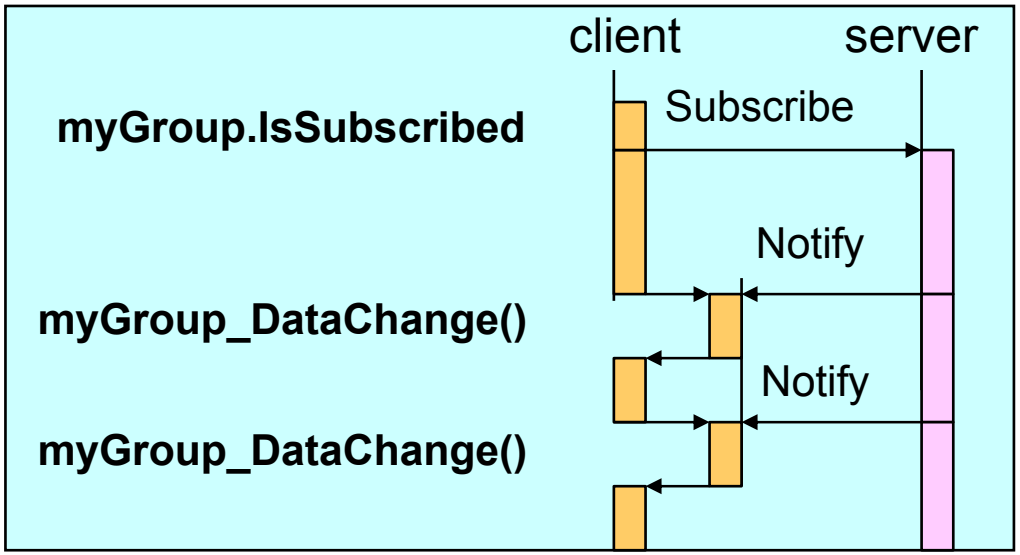
synchronous



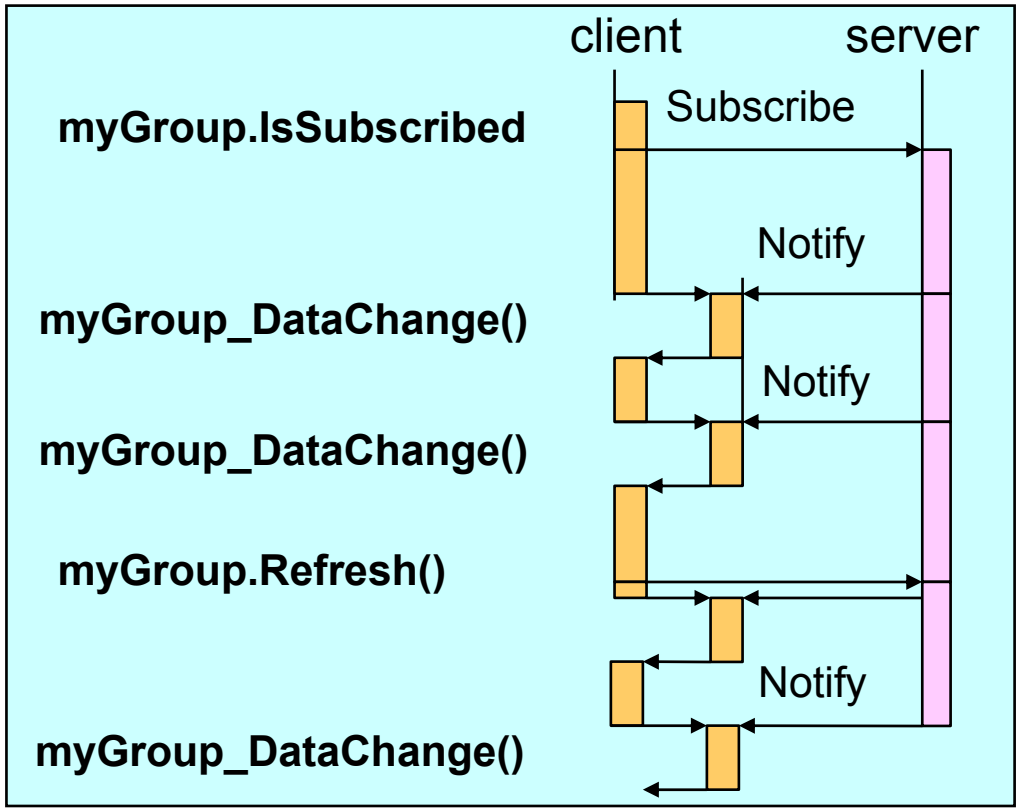
asynchronous



on change
("subscription-based")



OPC DA: Transmission by subscription (events)



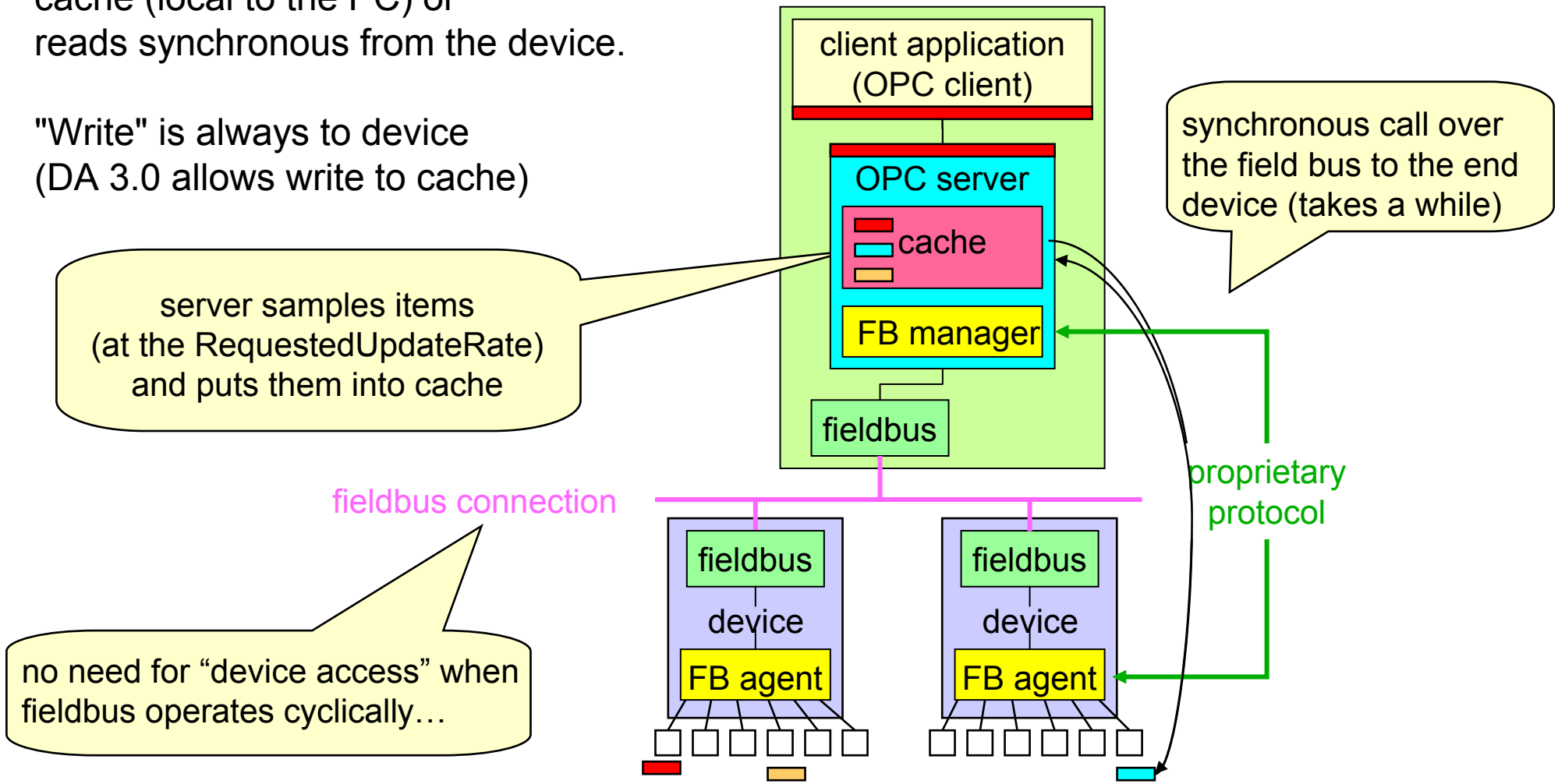
The server notifies the client if an item changed
- in a particular group (myGroup_DataChange) or
- in any of the groups (myGroups_GlobalDataChange)

In the second case, only the group in which the item changed will be sent.

OPC DA: "Cache" or "Device" ?

"SynchRead" reads the data either from cache (local to the PC) or reads synchronous from the device.

"Write" is always to device (DA 3.0 allows write to cache)



OPC DA: When are subscribed data transmitted ?

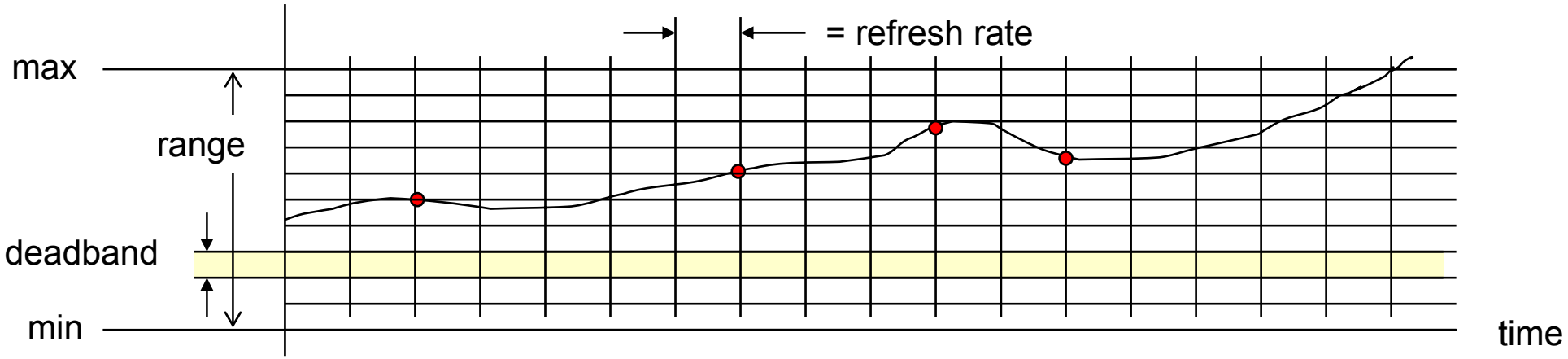
A group has two properties to control when a data change is to be transmitted:

myGroup.Refreshrate: (called UpdateRate in DA 3.0)

rate at which the server samples the process values, expressed in seconds ! (1/rate)
earliest interval between changes of value are communicated to the client, but minimum
rate at which the cache should be updated. (throttles changes, but may miss some)
The server never sends data to a client at a rate faster than the client requests.

myGroup.Deadband

applied only to analog values: deadband = % the range (in Engineering Units).
value is transmitted if the difference since last transmission exceeds deadband.
Applies to all items of a group, DA 3.0 allows settings per group and per item.



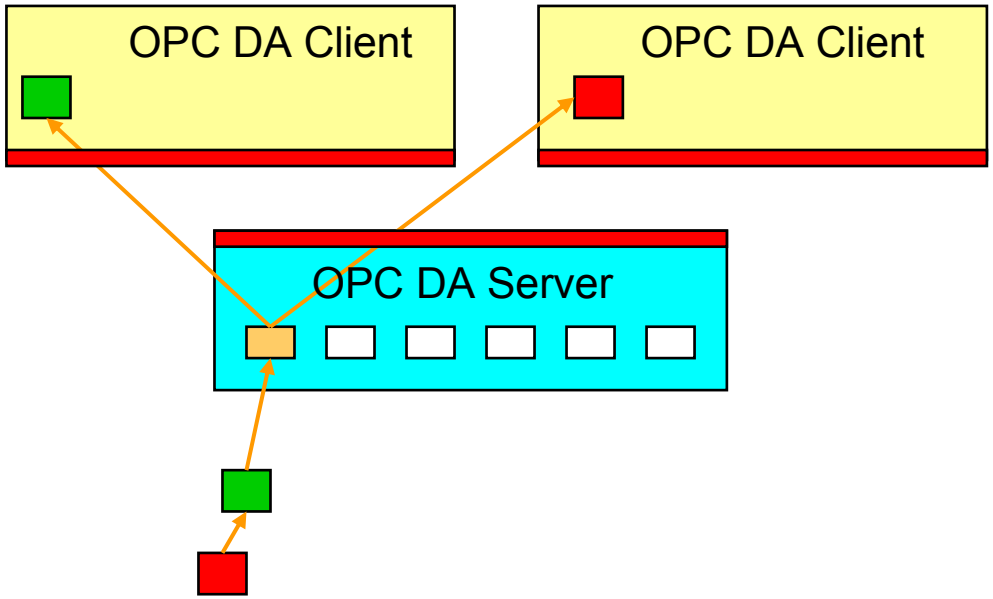
OPC DA: communication paradigm

OPC DA works according to the “shared memory” paradigm.

This means that a newer value overwrites the older one, no queues or history are kept.

The server does not guarantee that different clients see the same snapshot of the plant.

The server does not guarantee that all changes to variables are registered, changes may be missed if the polling period is too low.



OPC DA Part 2

Part 1 explains the concepts
Part 2 shows how they are implemented and programmed

OPC DA: Programming Example

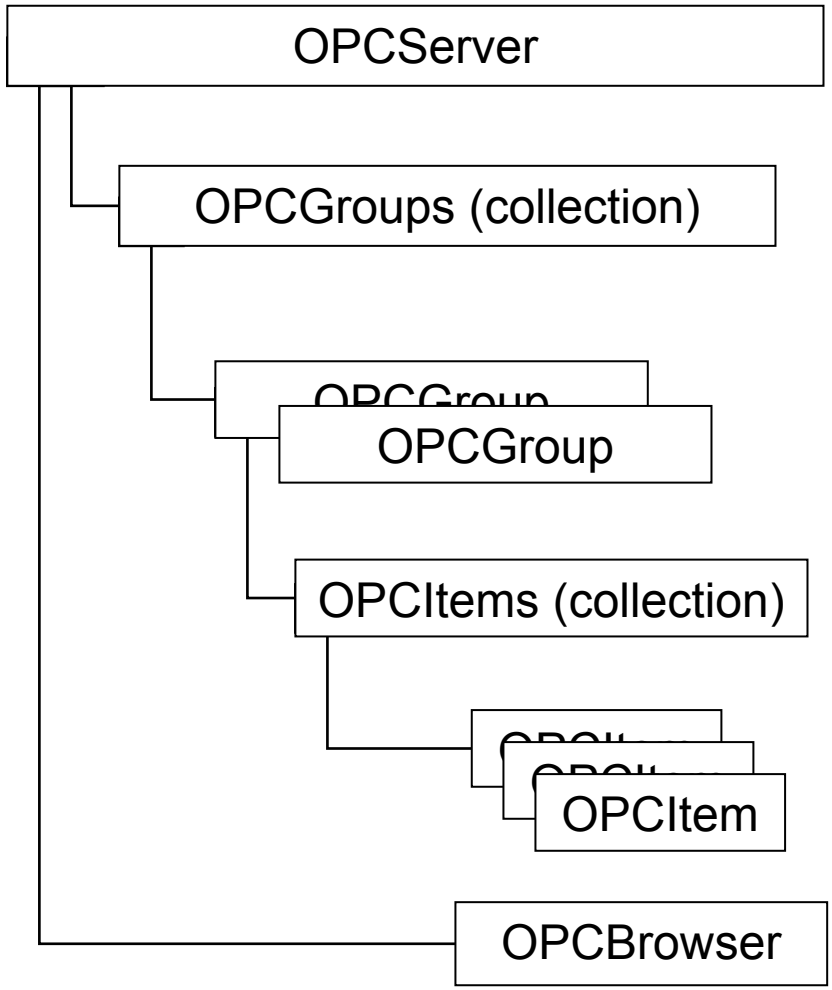
OPC Common
- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access
- Overview: browsing the server
- Objects, types and properties
- Communication model
- **Simple Programming Example**
- Standard and components

OPC Alarms and Events Specification
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

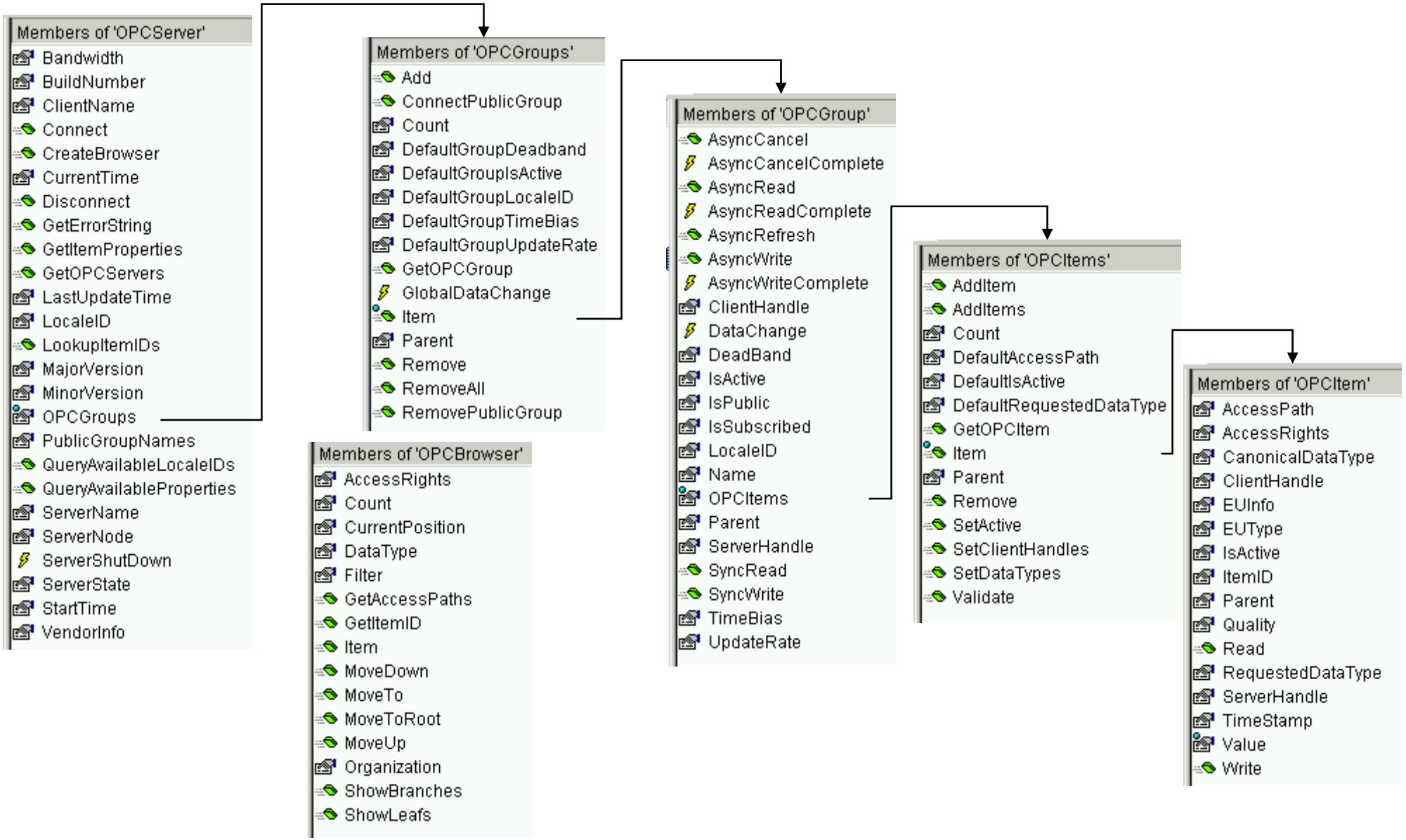
OPC Historical Data Specification
- Overview

OPC DA: Object hierarchy at the client

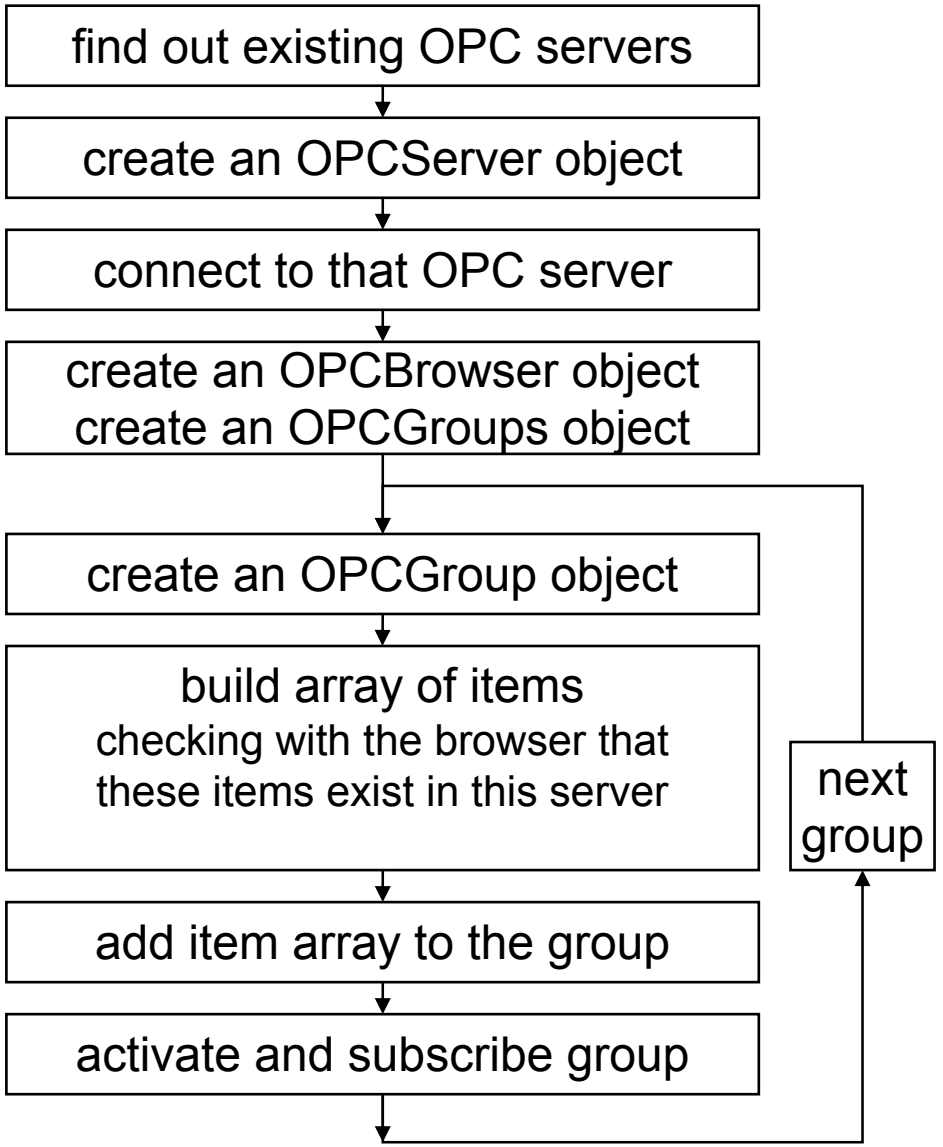


Description
An instance of an OPC Server. You must create an OPCServer object before you can get references to other objects. It contains the OPCGroups Collection and creates OPCBrowser objects.
A collection containing all of the OPCGroup objects this client has created within the scope of the OPCServer that the Automation Application has connected to via OPCServer.Connect()
An instance of an OPCGroup object. this object maintains state information and provides the mechanism to access data for the OPCItems Collection object that the OPCGroup object references.
A collection containing all of the OPCItem objects this client has created within the scope of the OPCServer, and corresponding OPCGroup object that the Automation Application has created.
An automation object that maintains the item's definition, current value, status information, last update time. Note the Custom Interface does not provide a separate Item Object.
An object that browses item names in the server's configuration. There exists only one instance of an OPCBrowser object per instance of an OPC Server object.

OPC DA: Automation interface summary



OPC DA: Program - initialising a connection




```
myDummyServer.GetOPCServers
```

```
myServer = new OPCServer
```

```
myServer.Connect
```

```
Set myBrowser = myServer.Browser
Set myGroups = myServer.Groups
```

```
Set myGroup1 = myGroups.Add
Set MyItems = MyGroup1.OPCItems
```

```
FQItems1[1] = "Device1.Temp1"
ClientHandle1[1] = 101
ReDim ServerHandle1(nrItems)
ReDim ServerErrors1(nrItems)
ReDim Value1(nrItems)

myGroup1.AddItems
```

```
myGroup1.IsActive
myGroup1.IsSubscribed
```

OPC DA: Program - Declarations

```
' Option Base 1                              'OPC arrays indices start with 1
Dim WithEvents MyServer As OPCAutomation.OPCServer   'OPC Server Object
optional)
Dim WithEvents MyGroups As OPCAutomation.OPCGroups   'OPC Group Collection
Dim WithEvents MyGroup  As OPCAutomation.OPCGroup    'OPC Group Object
' items
Dim nrItems           As Integer
Dim MyItems              As OPCAutomation.OPCItems    'OPC Item Collection
Dim MyItem               As OPCAutomation.OPCItem     'OPC Item Object
Dim ItemsID(2)           As String      'fully qualified items (see later)
Dim ClientHandles(2)     As Long
Dim ServerHandles()      As Long        ' must be a dynamic array
Dim ServerErrors()       As Long        ' must be a dynamic array
```

Reference: "OPC Automation 2.0" must be included into Visual Basic project

(if missing: copy `opcdaauto.dll` to `C:\WINNT\System32\opddaauto)`
and register it: `C:\>regsvr32 C:\WINNT\System32\opddaauto.`

A simple way to do it: install Software Toolbox's TopServer (freeware)

OPC DA: Program - Finding the OPC servers

The GetOPCServers function applied to a dummy Server object allow to list the existing servers on this node or on another node (over DCOM - security must be set correctly). The information about which OPC servers exist is taken from the registry, where it has been put by each server at its installation time

```
Private Sub ShowServers(netNodeName As String)
    Dim dummyServer As OPCAutomation.OPCServer
    Dim Servers As Variant                                  ' this is an array of strings
    Dim cntServers As Integer

    Set dummyServer = New OPCAutomation. OPCServer          ' create a dummy server object
    Servers = dummyServer.GetOPCServers(netNodeName)        ' returns all available servers

    For cntServers = LBound(Servers) To UBound(Servers)     ' display the names
        MsgBox Servers(cntServers)
    Next cntServers

    Set dummyServer = Nothing                               ' delete object (created by „New“)
    Exit Sub
End Sub
```

OPC DA: Program - Connecting to the OPC server

```
Set MyServer = New OPCAutomation.OPCServer    ' create server object
MyServer.Connect ("Matrikon.OPC.Simulation") ' connect to Matrikon server
```

Before connecting, it is safe to check the name of the server from the server's list.
Also, it is preferable to include the connection in a separate routine since it can fail:

```
Function ServerGetCare(Name As String, ServerNode As String) As
                                                    OPCAutomation.OPCServer
    On Error GoTo ServerGetCareErr
    Dim MyOPCServer As New OPCAutomation.OPCServer

    MyOPCServer.Connect ServerName, ServerNode ' connect risky
    Set ServerGetCare = MyOPCServer
    Exit Function

ServerGetCare_Err:                              ' error handler if connect fails
    Err.Clear
    MsgBox "Could not connect"
    Set MyServer = Nothing
    Exit Function
```

OPC DA: Program - Browsing the server

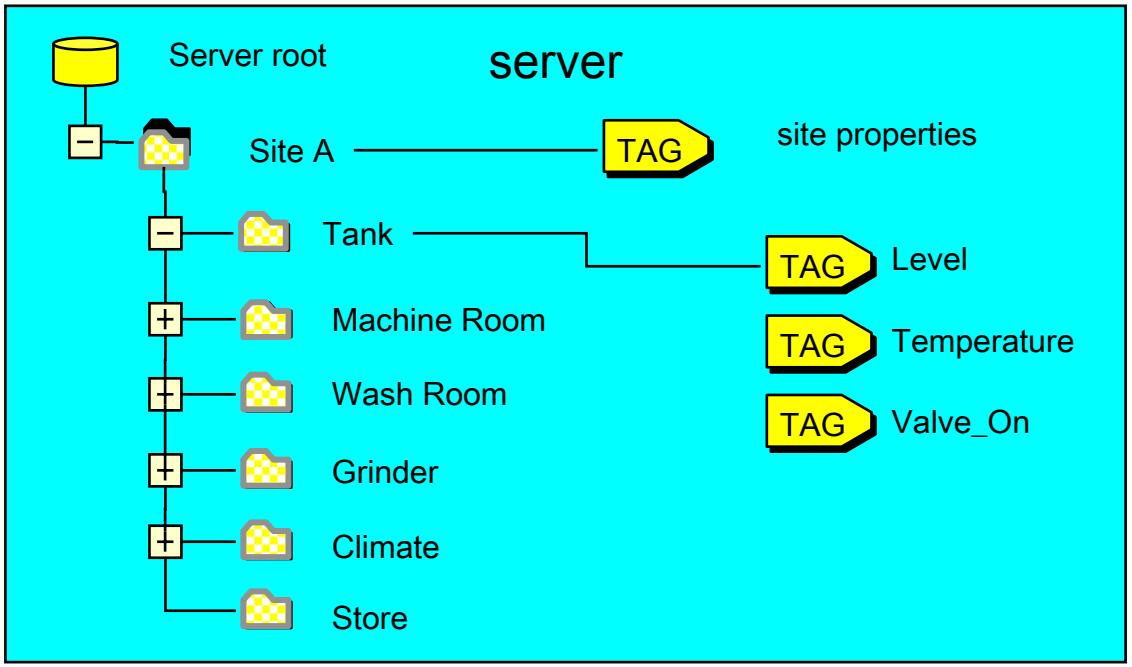
The object OPCBrowser (of type "collection") acts as a pointer to the server's tree:

```
Dim MyServer As OPCAutomation.OPCServer
Dim MyBrowser As OPCAutomation.OPCBrowser
Dim vName As Variant

MyServer.Connect "Matrikon.OPC.Simulation", "Orion"        'server and node name (DCOM)

Set MyBrowser = MyServer.CreateBrowser                     ' create an OPC browser

MyBrowser.ShowBranches                                      ' show the branches
For Each vName In MyBrowser
    MsgBox "Branch: " & vName                               ' display the branch name
Next vName

MyBrowser.ShowLeafs                                         ' explore the leaves
For Each vName In MyBrowser
    MsgBox "Leaf: " & vName                                 ' display the leaves's name
Next vName
```

OPC DA: Navigating

There may be leaves at every branch, since a branch may have properties

```
MyBrowser.MoveDown (strBranch)          ' go down the selected branch tree

MyBrowser.MoveUp                        ' go up the selected branch tree
```



Access path

The Access Path is an optional information that the client may provide regarding how to get to the data, where several possibilities exist. Its use is highly server specific. Do not confound with hierarchical path.

The optional function GetAccessPath retrieves the access path for items that can be accessed over different ways.

OPC DA: Program - get the Fully Qualified ItemID

To get the "fully qualified itemID", one positions the browser at the place where the leaf is attached to the branch and calls GetItemID

```
myOPCBrowser.MoveDown("TankArea")

myOPCBrowser.MoveDown("Tank1")

FQI = myOPCBrowser.GetItemId ("WaterLevel")
```

e.g. FQI could be "Controller1;Tanks!WaterLevel"

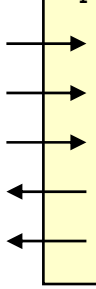
Of course, one can write an Item ID directly when defining a group, but it is safer to browse the server and get the FQI from there, since the delimiter depends on the server.

OPC DA: Program - Creating OPCGroups and OPCItems

```
Set MyGroups = MyServer.OPCGroups          ' create groups collection
Set MyGroup1 = MyGroups.Add("GRP1")        ' add group, name private
Set MyItems = MyGroup1.OPCItems            ' define the OPCItems of group
```

```
FQItemIDs(1) = "Area2.Tank1.WaterLevel"    ' fully qualified itemID
ClientHandles(1) = 5                       ' arbitrary
FQItemIDs(2) = "Area2.Tank1.Temperature"   ' fully qualified itemID
ClientHandles(2) = 6                       ' arbitrary (but different)
nrItems = 2
```

```
MyItems.AddItems _                         ' adds the items to collection
  nrItems, _                               ' input parameter
  FQItemIDs, _                             ' input fully qualified ID
  ClientHandles, _                         ' input ClientHandles
  ServerHandles, _                         ' return parameter ServerHandles
  ServerErrors                             ' return parameter ServerErrors
```



```
MyGroup1.ClientHandle = 1                  ' handle of the group (no s) !
MyGroup1.IsActive = True                   ' now ready to send and receive
MyGroup1.IsSubscribed = True               ' and to generate events
```

The role of the ServerHandles and ClientHandles will be explained later…

OPC DA: Data structures at the client

The client prepares data structures for its items and gives the server the corresponding pointers so the server can update them.
Items to be written and read can be mixed in the same group.
The type of the item (Boolean, Float,…) is implicit, but known at the server

communicated to server by registering group

returned by server when registering

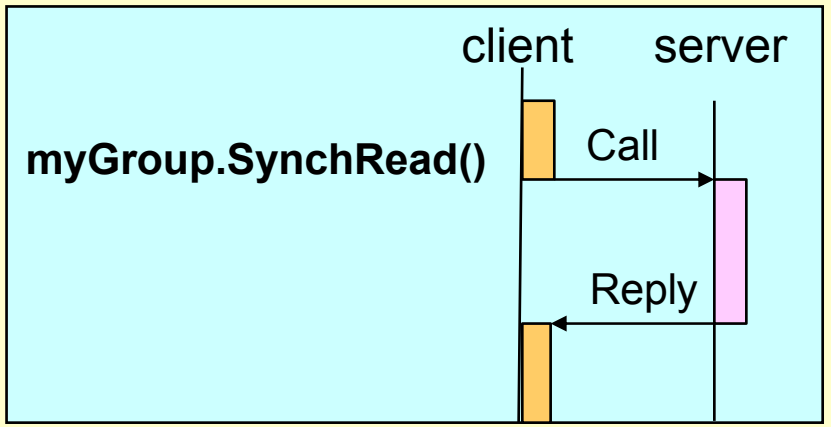
dynamic changes (refreshed on change)

FullyQualifiedItemID	ClientHandle	ServerHandle	ServerError	Value	Quality	TimeStamp
"Channel1.Device1.Temp1"	100	34543	0	123.4	OK	12:09.234
"Channel1.Device1.Speed1"	102	22532	0	999.8	OK	12:02.214
"Channel1.PLC2.Door"	203	534676	0	0	OK	12:03.002
"Channel1.PLC2.Valve3"	204	787234	0	1	OK	12:02.345
"Channel1.PLC2.CloseDoor"	205	58432	0	0	BAD	12:02.345
..		..	..	..	..	..

Note: OPC indices start with 1 !

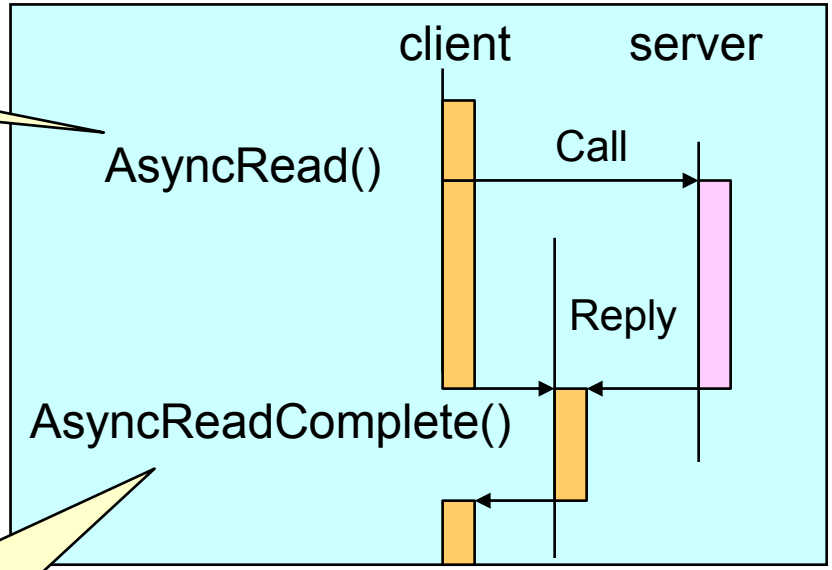
OPC DA: Synchronous Read of a group

```
Dim thisGroup As OPCGroup
Dim cntItems As Integer
Dim source As Integer
Dim serverHandles(2) As Long
Dim values() As Variant
Dim errors() As Long

serverHandles(1) =  ServerHandle(11) '
serverHandles(2) =  ServerHandle(14)

source = OPCcache                    ' could also be OPCDevice
thisGroup.SyncRead
    source,
    nrItems,
    serverHandles,                   ' identifies the items to be read !
    values,                          ' returns be a dynamic array
    errors                           ' returns a dynamic array

For cntItems = LBound(serverHandles) To UBound(serverHandles) ' 1..n
    MsgBox CStr(cntItems) & " : " & values(cntItems)
Next cntItems
```



OPC DA: Asynchronous read of single Items

```
   Dim WithEvents MyGroup
   ...
   MyGroup.AsyncRead
 →   nrItems,
 →   ServerHandles,
 ←   ServerErrors,
 →   TransactionID,
 →   CancelID
```



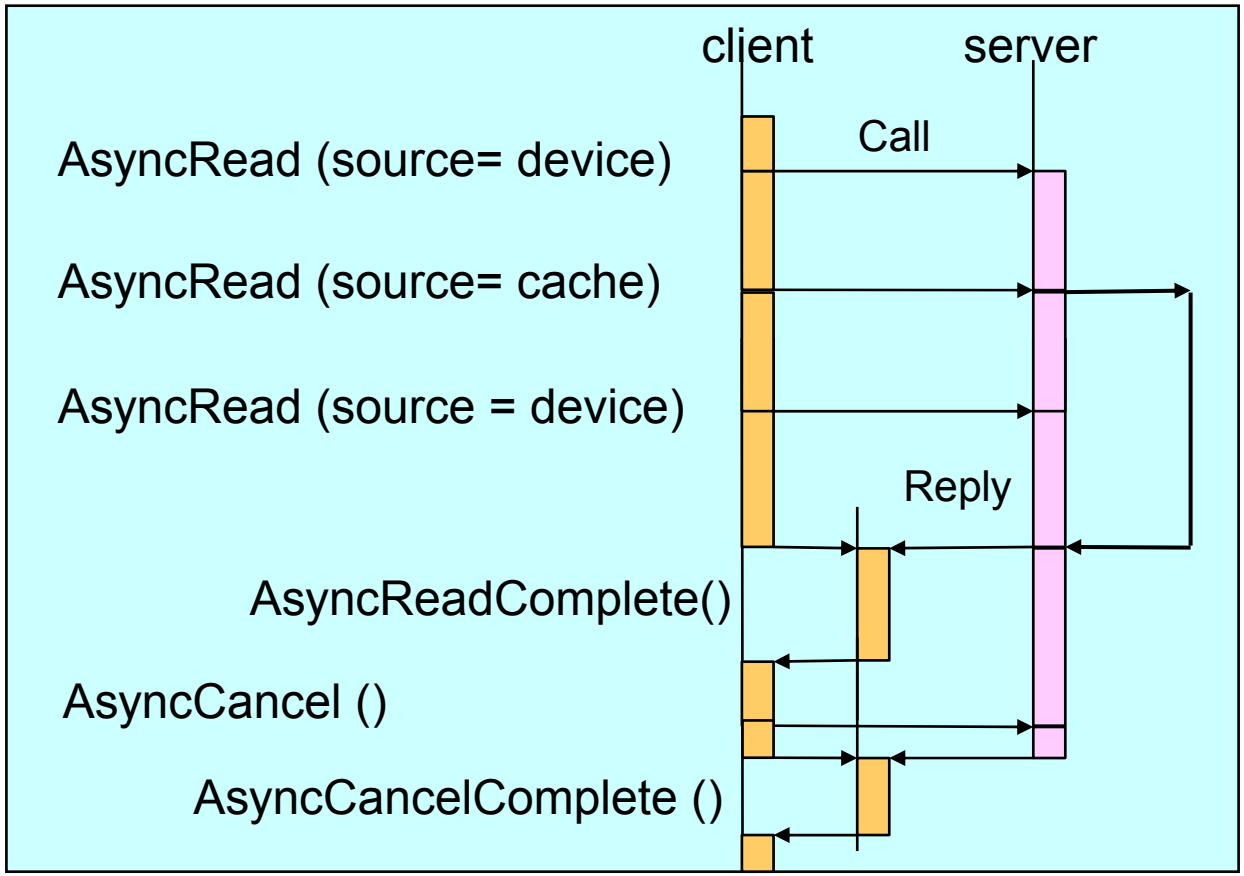

```
 Private Sub Mygroup_AsyncReadComplete (
← ByVal TransactionID As Long,
← ByVal NumItems As Long,
← ClientHandles() As Long,
← ItemValues() As Variant,
← Qualities() As Long,
← TimeStamps() As Date,
← Errors() As Long)

   MsgBox ("Async Read Complete")
End Sub
```

Asynchronous read separates Call and Reply.

Call supplies the ServerHandles

Reply returns the corresponding ClientHandles

OPC DA: Transaction ID



Although the AsynchReadComplete carries the ClientHandle of each item,
it does not tell which AsynchRead caused the AsynchReadComplete event to fire.

Call and Reply are linked by the TransactionID: this ID is returned in AsynchReadComplete
It can also be used to cancel the operation

OPC DA: Reading (by events) the OPC group

```
Dim WithEvents MyGroup
...
Private Sub MyGroup_DataChange( _
    ByVal TransactionID As Long, _
    ByVal NrItems As Long, _
    ClientHandles() As Long, _          ' returned by the server to the client
    ItemValues() As Variant, _
    Qualities() As Long, _
    TimeStamps() As Date)

    Dim cntItems As Integer
    For cntItems = LBound(ClientHandles) To UBound(ClientHandles)     ' index 1..n
        TextValue(cntItems - 1).Text     = ItemValues(cntItems)       ' display
        TextTimeStamp(cntItems - 1).Text = DateAdd("h", 9, TimeStamps(cntItems))
        TextQuality(cntItems - 1).Text   = Qualities(cntItems)
    Next cntItems
End Sub
```

This function is called each time an item in the group changes
The ClientHandles (here: 5 and 6) identifies the variables, not the “fully qualified itemID”
The values are displayed in the TextValue, TextTimeStamp and TextQuality fields.
The refresh rate is given in the group definition.

OPC DA: Groups Events

Although transmission by groups is more efficient than AsyncRead, it can be improved by using Groups Events (Global Data Change)

This event is fired whenever a variable of a group changes.

If the group is subscribed also to a Group Event (DataChange), I.e. if the group is declared WithEvents, then both Events will be fired.

The application must sort out the groups and the items.

OPC DA: GlobalDataChange

```
Dim WithEvents MyGroups As OPCGroups
...
Private Sub MyGroups_GlobalDataChange(
    ByVal TransactionID As Long,      ' =0 if called by Refresh
    ByVal GroupHandle As Long,
    ByVal NumItems As Long,
    ClientHandles() As Long,          ' identifies the items
    ItemValues() As Variant,          ' value of the items
    Qualities() As Long,              ' value of the items
    TimeStamps() As Date)             ' timestamps of the items

Select Case GroupHandle               ' depending on the group ...
    Case 1
       ' treat group 1
    Case 2
       ' treat group 2
```

The GlobalDataChange event is fired when any item in a group changed.
(if Groups is also with events, the corresponding Group_DataChange will also be called)

OPC DA: Server Events

```
Dim WithEvents MyServer As OPCServer          ' define the event
.. ..
Private Sub MyServer_ServerShutDown(ByVal Reason As String)
        MsgBox "my OPC Server " & MyServer.ServerName & " quit"
End Sub
```

This event signals to the client that the server shut down.

The client must declare its server „WithEvents“ and provide the corresponding event Subroutine

This should stop all actions, otherwise exceptions will occur.

OPC DA: Do not forget cleanup !

To speed up connection/disconnection, an OPC server remembers its groups and clients when a client disconnects.

To do this, an OPC server initialises its structures with a client counter of 2, instead of 1. Therefore, it is imperative to shut down explicitly the server, otherwise links will subside (and you will have kill the server to clear them).

```
Private Sub ServerShutdown
    Dim dummyServer As OPCAutomation.OPCServer
    Dim Servers As Variant                          ' this is an array of strings
    Dim cntServers As Integer

    Set myGroup1 = Nothing                          ' create a dummy server object
    Set myGroups = Nothing                          ' returns all available servers
    MyServer.Remove
    MyServer.RemoveAllGroups
    MyServer.Disconnect                     ' delete this object (was created by New)
    Set MyServer = Nothing
```

OPC DA: Standard and components

OPC Common
- Overview: usage and specifications
- OPC as an integration tool
- Clients and Servers: configuration
- OPC Technology, client and custom interface

OPC Data Access
- Overview: browsing the server
- Objects, types and properties
- Communication model
- Simple programming example
- **Standard and components**

OPC Alarms and Events Specification
- Overview: definitions and objects
- Events
- Alarm Conditions
- Automation Interface

OPC Historical Data Specification
- Overview

OPC DA: Libraries

The OPC DA specification is not formal, conformance can hardly be checked against this document.

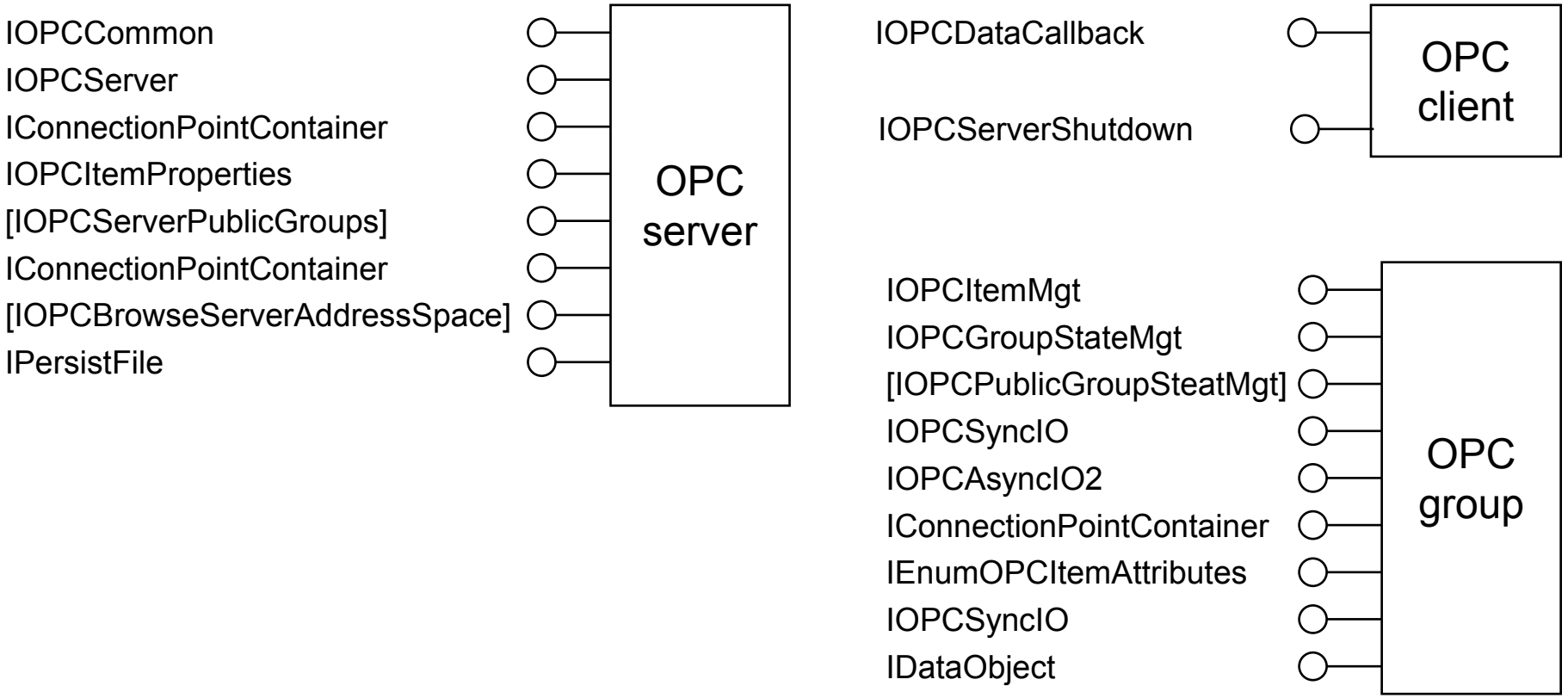
To ensure that the standard is observed, the OPC foundation distributes on its website the DLLs (opcdaauto.dll, opccomn_ps,…) that contain the type libraries to access the OPC server.

The vendors are not compelled to implement all features. For instance, the description of the variables is seldom used. Calling unimplemented functions causes exceptions that must be caught in Visual Basic with "On Error …" statements.

There exist three versions of DA, 1.0, 2.0 and 3.0, that behave differently, however, older servers do not have a property indicating which version they support.

OPC DA: Custom Interface

While the Automation Interface is easy to use and quite powerful, some OPC functions are missing and special operations can only be done in Visual C++ using the custom COM interface.
This is only recommended for experienced programmers.



OPC DA: Assessment

What is OPC ?

Which are the read and write operations ?

Is communication done by items, by groups or by collection of groups ?

What is the difference between cache and device reading ?

Can a change of an OPC variable be notified as an event, or shall the client poll ?

How is browsing done ?

Why is browsing necessary, even when one knows the variable's location in the server ?

To probe further….

OPC Foundation:
- Specifications http://www.opcfoundation.org

SoftwareToolbox
- Examples in Visual Basic
  http://www.softwaretoolbox.com/Tech_Support/TechExpertiseCenter/OPC/opc.html

The Code Project
- OPC and .NET
  http://www.codeproject.com/useritems/opcdotnet.asp

Matrikon
- Free client and server:
  http://www.matrikon.com

WinTech
- Toolkit for an OPC server
  http://www.win-tech.com/html/opcstk.htm

NewAge Automation
- Toolkit for an OPC server
  http://www.newageautomation.com


EPFL

OPC DAOPCGroup Custom Interface: comparison (1)

This checklist for experienced programmers (custom interface) shows the differences between the DA versions

Required Interfaces	DA 1.0	DA 2.0	DA 3.0
OPCGroup			
IUnknown	Required	Required	Required
IOPCItemMgt	Required	Required	Required
IOPCGroupStateMgt	Required	Required	Required
IOPCGroupStateMgt2	N/A	N/A	Required
IOPCPublicGroupStateMgt	Optional	Optional	N/A
IOPCSyncIO	Required	Required	Required
IOPCSyncIO2	N/A	N/A	Required
IOPCAsyncIO2	N/A Required	Required	
IOPCAsyncIO3	N/A	N/A	Required
IOPCItemDeadbandMgt	N/A	N/A	Required
IOPCItemSamplingMgt	N/A	N/A	Optional
IConnectionPointContainer	N/A	Required	Required
IOPCAsyncIO	Required	Optional	N/A
IDataObject	Required	Optional	N/A

OPC DA OPCServer 1.0, 2.0 & 3.0 comparison (2)

This checklist for experienced programmers (custom interface) shows the differences between the DA versions

Required Interfaces	**1.0**	**2.0**	**3.0**
OPCServer			
IOPCServer	Required	Required	Required
IOPCCommon	N/A Required	Required	
IConnectionPointContainer	N/A	Required	Required
IOPCItemProperties	N/A Required	N/A	
IOPCBrowse	N/A	N/A	Required
IOPCServerPublicGroups	Optional	Optional	N/A
IOPCBrowseServerAddressSpace	Optional	Optional	N/A
IOPCItemIO	N/A	N/A	Required

The differences do not yet appear in the automation interface