Tracing OPC DA in System 800xA using AfwApplogViewer

Stefan Strömqvist, SE L3 Operations Support Team

December 13th, 2019

Introduction

Each OPC data sample travelling between an OPC server and a client, e.g. a faceplate or history log in System 800xA can be traced using applog. The first key to success is identification of points to apply the logging at.

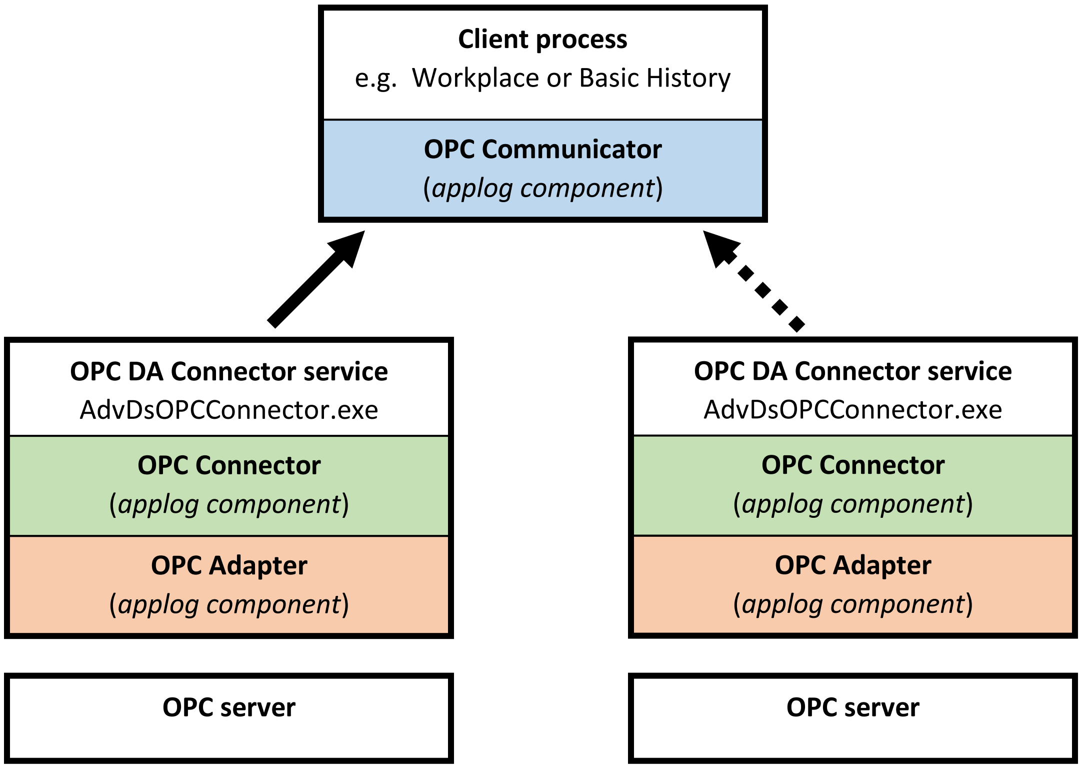
In principle, there are two locations available where logging can be made, one of them is split into two parts:

1. The client process
    - OPC Communicator (facing the OPC Connector)

2. The OPC DA Connector service provider
    - OPC Connector (facing the OPC Communicator in the client)
    - OPC Adapter (facing the source OPC DA server)

In redundant systems, there is often two OPC DA Connector service providers. The choice of server is either governed by *affinity* or random (in lack of affinity).

Some select services (e.g. Basic History and SoftAlarm) can use *Parallel Redundancy* where subscriptions are run in parallel towards both OPC DA Connector service providers. This allow ultra-fast failover, in fact there is no failover to talk about. The subscription is already running in the secondary, the client just need to switch feed.



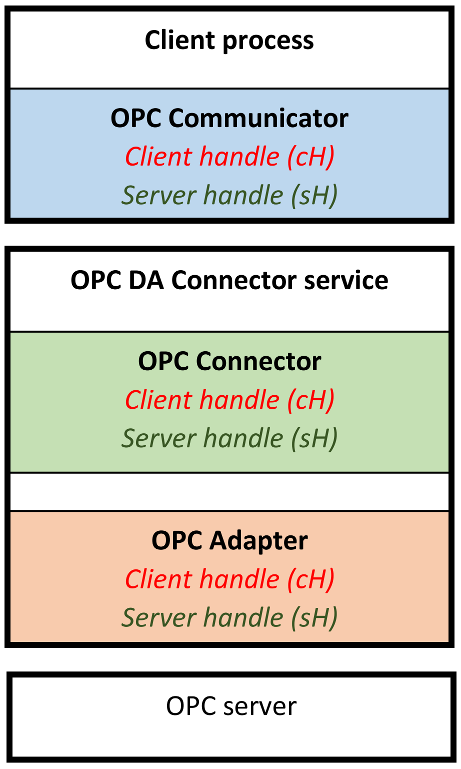
The second key to success is proper identification of the item identities, so called "handles" in the three (3) layers of logging that is possible between the server and client.

Each layer has its own representation of item handle. A handle also has a direction:

- "client handle" (facing towards the client)
- "server handle" (facing towards the server).

Notice that the complete chain involves "four" (4) parties:

1. The client
2. The OPC DA Connector (facing multiple clients)
3. The OPC DA Adapter (facing a single OPC server)
4. The server



1. A sample of logging in the OPC Communicator of Basic History (AdvHtHistorySrv.exe) may look like this:

```
2018-01-23 12:44:45:254#1065, ABBIT51, AdvHtHistorySrv, PID: 29536, Thread: 37848
OPC Communicator, Common, level = 2, Tag =

NewValues_2_0: OpcHandler=0x457b4e8

GroupSH:0 Client:CH(701), OPCHdlr:SH(68096), ItemID:'{7B6DA285-FFB0-4886-814E-
27F87062F6C1}{4C8CCD88-CCBA-4E6D-A642-15A4B7B010CC}:value.IOValue',
NewValue:value(VT_R4) = 101.727, Quality:0xc0, TimeStamp:01/23/18 12:44:44 864ms,
Result:0x0, OPCHdlr:CH(68096), Connector:SH(-1)
```

Comments

The log contains the Object and Aspect ID + property names (the IDs can be found out from object or aspect *Details…* and translated back to name by using the *Find Tool*). Along with the IDs we also see the data value, timestamp and data quality. Then we have the mission critical "handles" used by the OPC Communicator:

- `Client:CH(701)`
- `OPCHdlr:SH(68096)`
- `OPCHdlr:CH(68096)`
- `Connector:SH(-1)`

2. A sample of logging in the OPC Connector may look like this:

```
2018-01-23 12:44:45:242#1002, ABBIT51, AdvDsOPCConnector, PID: 6848, Thread: 8548
AdvDsOPCConnector, Basic, level = 2, Tag =

CConnector::OnDataChange - OPCHdlr:CH(68096), Connector:SH(60672), quality(c0),
1/23/2018 12:44:44 PM.864, value(VT_R4) = 101.727, Connector:CH(60672),
Adapter:SH(237)
```

As we can see, the OPC Connector does not expose any object or aspect IDs, only the data value, timestamp and quality + the "handles"

Handles used by the OPC Connector:

- `OPCHdlr:CH(68096)`
- `Connector:SH(60672)`
- `Connector:CH(60672)`
- `Adapter:SH(237)`

3. A sample of logging in the OPC Adapter may look like this:

```
2018-01-23 12:44:45:232#964, ABBIT51, AdvDsOPCConnector, PID: 6848, Thread: 40048
AdvDsOPCAdapter, Basic, level = 2, Tag =

Connector:CH(60672), Adapter:SH(237), quality(c0), 1/23/2018 12:44:44 PM.864,
value(VT_R4) = 101.727, Adapter:CH(237), OPCSrv:SH(224)
```

Handles used by the OPC Adapter:

- `Connector:CH(60672)`
- `Adapter:SH(237)`
- `Adapter:CH(237)`
- `OPCSrv:SH(224)`

Notice: The OPC item value (101.727) and quality (0xc0 = OPC good) is exactly the same in all levels of logging!

It may be easier to interpret all of the “handles” if we put them in their context

The client (Basic History, AdvHtHistorySrv.exe) adds an item which the OPC Communicator in the inside identifies as:

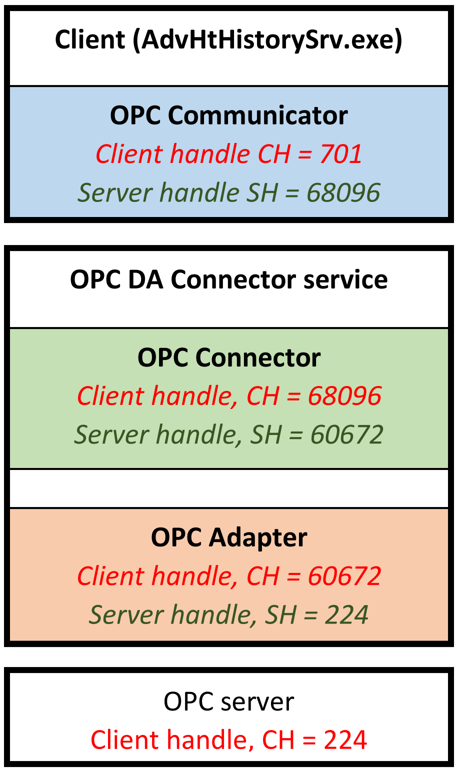
Object ID	`{7B6DA285-FFB0-4886-814E-27F87062F6C1}`
Aspect ID	`{4C8CCD88-CCBA-4E6D-A642-15A4B7B010CC}`
Property name	`value.IOValue`
Client Handle	701
Server Handle	68096 *(this handle is returned by the OPC DA Connector)*

Affinity (or randomization) send the subscription to the OPC DA Connector service in some computer (with Basic History, often the same node since Basic History often run in Connectivity Server nodes). The Connector has the following identifications:

Client Handle	68096
Server Handle	60672 *(this handle is returned by the OPC DA Adapter)*

The OPC Adapter gets the subscription from the OPC DA Connector. The OPC DA Adapter keep the item with the following identifications:

Client Handle	60672
Server Handle	224 (this handle is returned by the true source OPC server)



So, what the true client knows as data item #701 is known by the true server as data item #224. In the layers between, two more identities are used for the very same item!

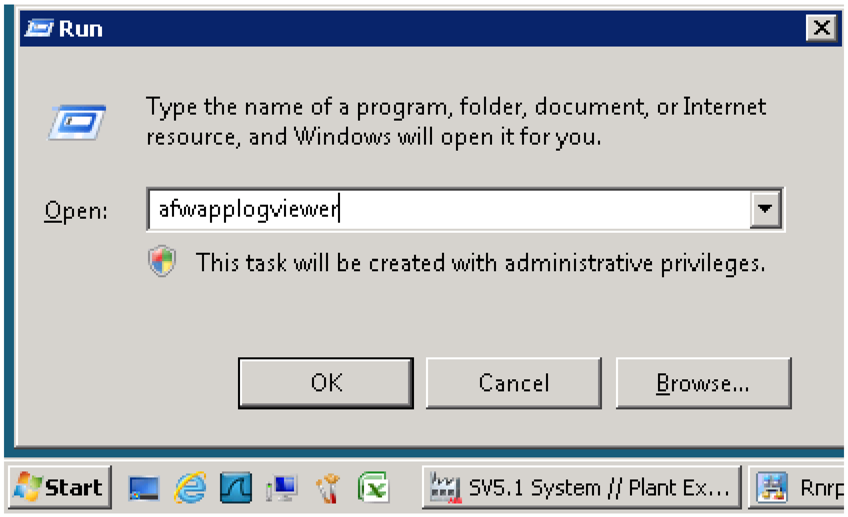
Understanding this chain of data communication and identification is crucial to be able to interpret the logs.

The logging can become quite substantial (an OPC server or client may handle several tens of thousands of items), and filtering is limited.

Only the OPC DA Adapter has a filter that can be applied. This filter takes an OPC group or OPC client as argument. It is indeed possible to limit the OPC Adapter logging to only one client, e.g. Basic History – but the client and server handles require you to log also in the OPC DA Connector. Without the Connector logging, you cannot follow the chain from start to end.

Let's begin with starting the AfwApplogViewer.exe tool, hereafter called "applog". The tool is decentralized, i.e. you can operate it from any node within an 800xA system; logging can be applied to any component in any node of the system. Ergo: it is possible to perform remote logging in a server from a local engineering client. The tool require the user to be member of the *IndustrialITAdmin* user group in the Windows workgroup or domain.

Start→Run… `afwapplogviewer`



Run

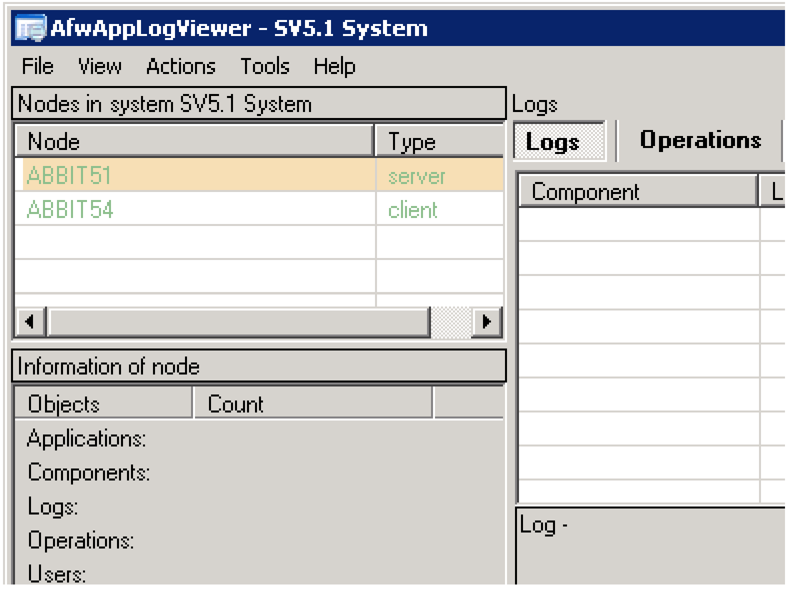
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: afwapplogviewer

This task will be created with administrative privileges.

Click **OK** in the two following dialog boxes (accepting defaults)

After some short time, an application shall be presented having a list of nodes in the upper left corner. The list with nodes may take a while to populate, especially in systems having more nodes in the Node Administration Structure than in reality (e.g. a test system with only a limited number of the computers available).



AfwAppLogViewer - SV5.1 System

Nodes in system SV5.1 System

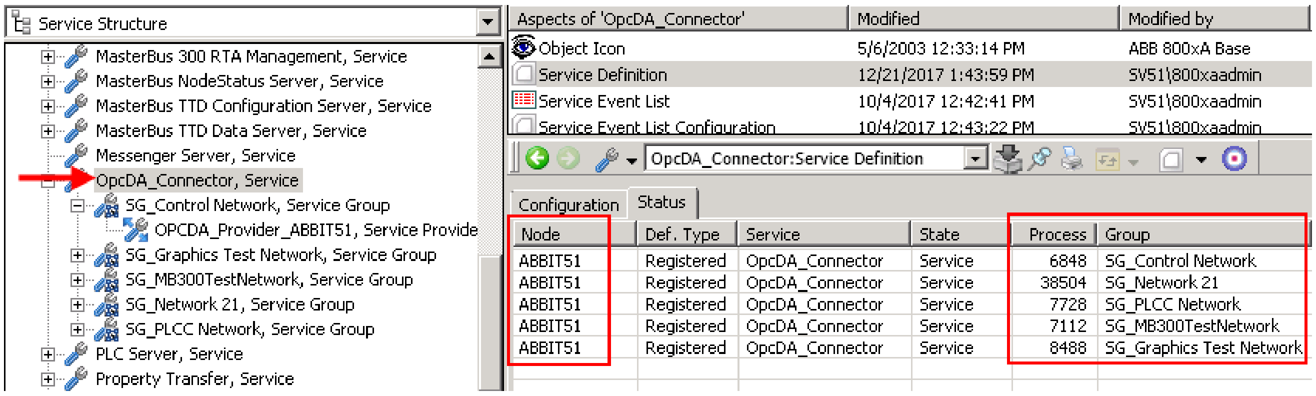
Node	Type
ABBIT51	server
ABBIT54	client

Information of node

Objects	Count
Applications:	
Components:	
Logs:	
Operations:	
Users:	

Lets begin with something simple, to list clients connecting to an OPC server. Before performing this, you must identify the node with the source OPC server and the PID of its associated OPC DA Connector.

The Service Definition aspect of the OPC DA Connector service in the [Service Structure] is an excellent place to make this decision from:

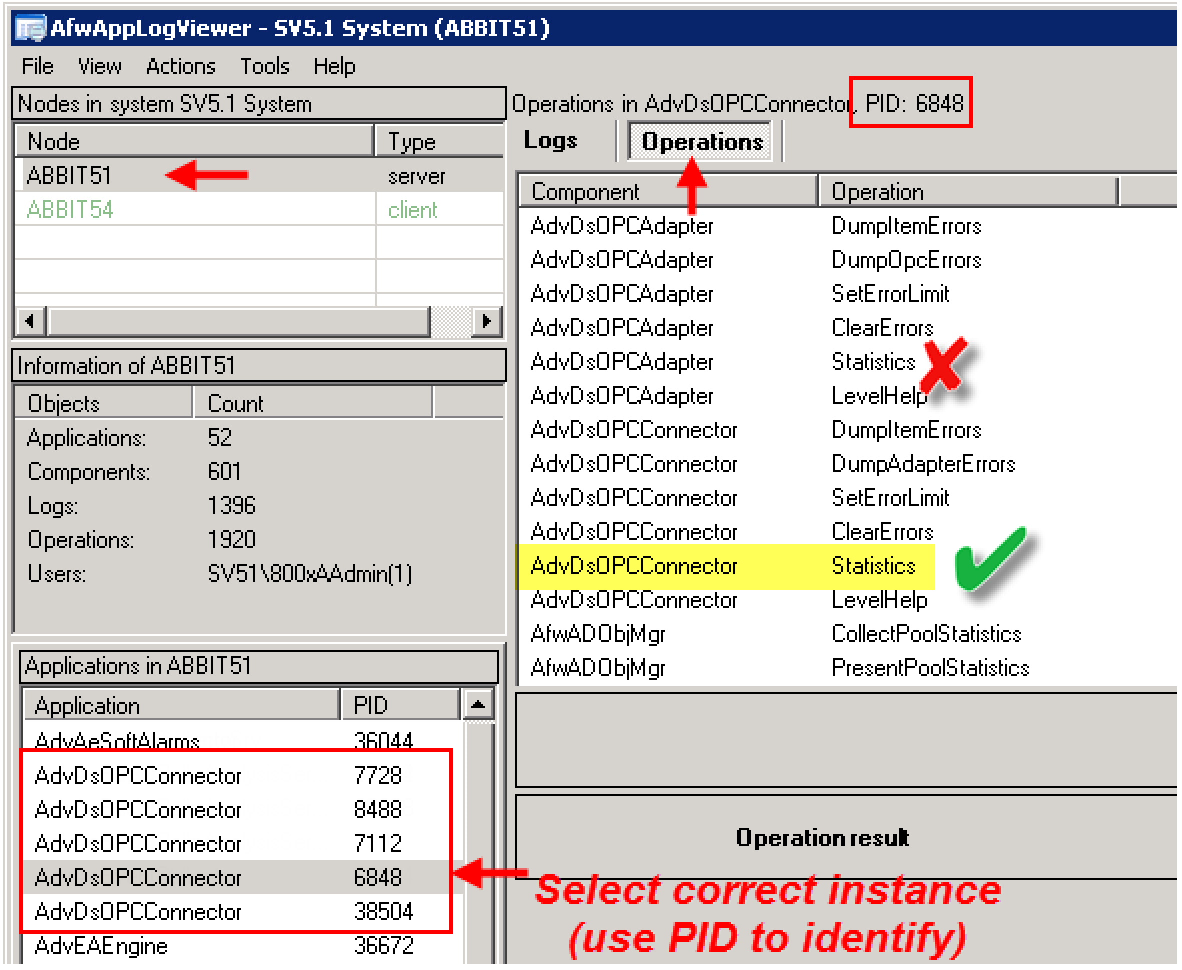

Service Structure
MasterBus 300 RTA Management, Service
MasterBus NodeStatus Server, Service
MasterBus TTD Configuration Server, Service
MasterBus TTD Data Server, Service
Messenger Server, Service
OpcDA_Connector, Service
SG_Control Network, Service Group
OPCDA_Provider_ABBIT51, Service Provide
SG_Graphics Test Network, Service Group
SG_MB300TestNetwork, Service Group
SG_Network 21, Service Group
SG_PLCC Network, Service Group
PLC Server, Service
Property Transfer, Service
OpcDA_Connector:Service Definition
Configuration
Status

Aspects of 'OpcDA_Connector'	Modified	Modified by
Object Icon	5/6/2003 12:33:14 PM	ABB 800xA Base
Service Definition	12/21/2017 1:43:59 PM	SV51\800xaadmin
Service Event List	10/4/2017 12:42:41 PM	SV51\800xaadmin
Service Event List Configuration	10/4/2017 12:43:22 PM	SV51\800xaadmin

Node	Def. Type	Service	State	Process	Group
ABBIT51	Registered	OpcDA_Connector	Service	6848	SG_Control Network
ABBIT51	Registered	OpcDA_Connector	Service	38504	SG_Network 21
ABBIT51	Registered	OpcDA_Connector	Service	7728	SG_PLCC Network
ABBIT51	Registered	OpcDA_Connector	Service	7112	SG_MB300TestNetwork
ABBIT51	Registered	OpcDA_Connector	Service	8488	SG_Graphics Test Network

In this case I am using AC 800M Connect of PID 6848 running in the node ABBIT51

To list OPC clients connecting to the AC 800M OPC server in ABBIT51 via System 800xA OPC framework, select the ABBIT51 node in the upper left list in applog, then select the button **Operations**. Locate the *AdvDsOPCConnector Statistics* operation, then double click that row (or mark it, and click the **Invoke** button on the right side)


AfwAppLogViewer - SV5.1 System (ABBIT51)
File
View
Actions
Tools
Help
Nodes in system SV5.1 System
Operations in AdvDsOPCConnector, PID: 6848
Logs
Operations
Operation result
Select correct instance
(use PID to identify)

Node	Type
ABBIT51	server
ABBIT54	client

Information of ABBIT51

Objects	Count
Applications:	52
Components:	601
Logs:	1396
Operations:	1920
Users:	SV51\800xAAdmin(1)

Applications in ABBIT51

Application	PID
AdvAeSoftAlarms	36044
AdvDsOPCConnector	7728
AdvDsOPCConnector	8488
AdvDsOPCConnector	7112
AdvDsOPCConnector	6848
AdvDsOPCConnector	38504
AdvEAEngine	36672

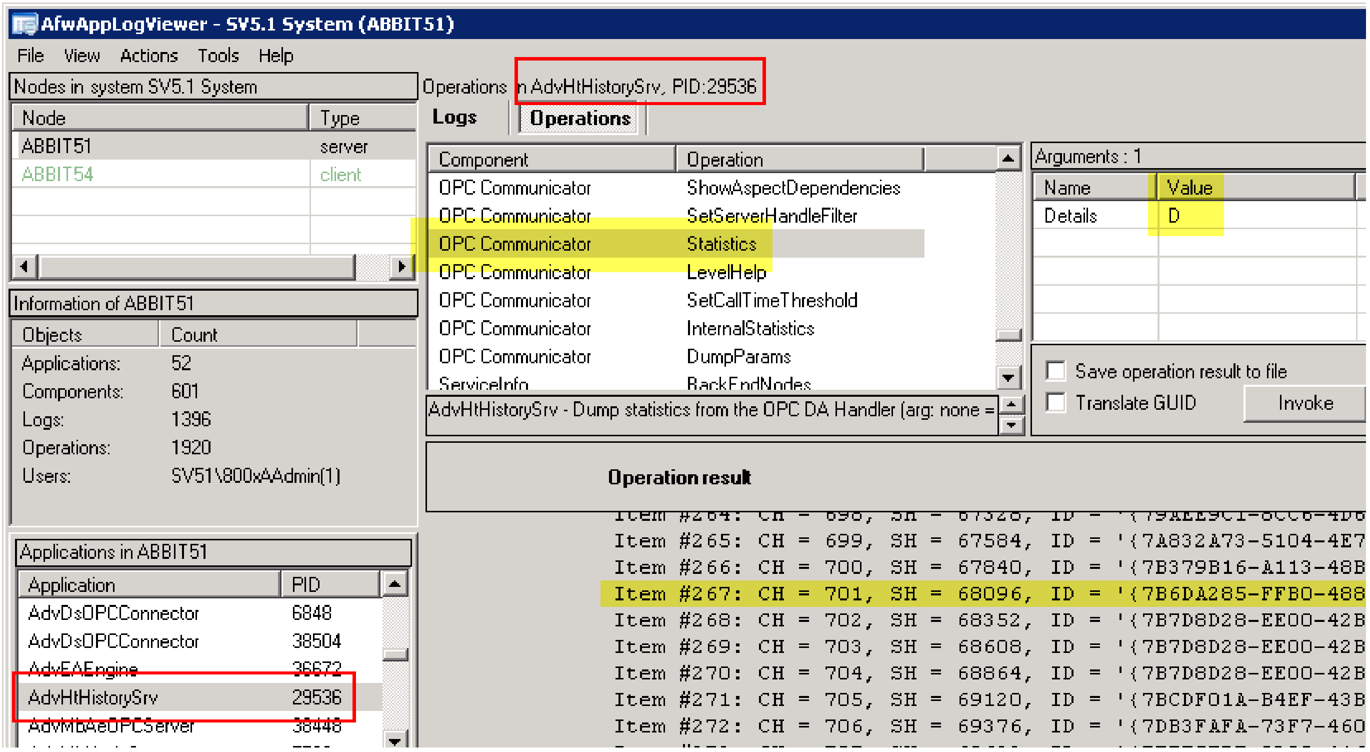
Component	Operation
AdvDsOPCAdapter	DumpItemErrors
AdvDsOPCAdapter	DumpOpcErrors
AdvDsOPCAdapter	SetErrorLimit
AdvDsOPCAdapter	ClearErrors
AdvDsOPCAdapter	Statistics
AdvDsOPCAdapter	LevelHelp
AdvDsOPCConnector	DumpItemErrors
AdvDsOPCConnector	DumpAdapterErrors
AdvDsOPCConnector	SetErrorLimit
AdvDsOPCConnector	ClearErrors
AdvDsOPCConnector	Statistics
AdvDsOPCConnector	LevelHelp
AfwADObjMgr	CollectPoolStatistics
AfwADObjMgr	PresentPoolStatistics

The output should be like the below example:

```
2018-01-23 14:01:52:482 ABBIT51     AdvDsOPCConnector PID : 6848  AdvDsOPCConnector
      ThreadId : 8548   Statistics
Execution time: 0 seconds, 33 miliseconds
…
      Total number of connected clients:    5

      Connected clients:
        Id: -1, BackendId: 0x02CF91E0, Node: 'ABBIT51', App:
'AfwWorkplaceApplication', User: 'SV51\800xaadmin', Items: 0 (0 active + 0
inactive)
        Id: 0, BackendId: 0x02CFA188, Node: 'ABBIT51', App: 'AdvAeSoftAlarms',
User: 'sv51\800xaservice', Items: 5 (5 active + 0 inactive)
        Id: 1, BackendId: 0x02CFB530, Node: 'ABBIT51', App: 'AdvAeAlarmManager',
User: 'sv51\800xaservice', Items: 5 (5 active + 0 inactive)
        Id: 2, BackendId: 0x02CFD340, Node: 'ABBIT51', App: 'AfwPropertyTransfer',
User: 'sv51\800xaservice', Items: 4 (1 active + 3 inactive)
        Id: 3, BackendId: 0x02CFE7C0, Node: 'ABBIT51', App: 'AdvHtHistorySrv',
User: 'sv51\800xaservice', Items: 245 (245 active + 0 inactive)
…
```

There are more lines output, but for this investigation, it is enough to identify Basic History in ABBIT51 as client ID 3 (we can also see that it has 245 items subscribed towards this OPC Connector).

Next thing we can do is to ask for the complete set of OPC items subscribed by the Basic History service. Select the node where it is running (here ABBIT51) and the AdvHtHistorySrv.exe application, then locate the operation OPC Communicator Statistics, fill out the argument field with "D" (for Detailed analysis) then press Invoke (or double click the row with the operation name)


AfwAppLogViewer - SV5.1 System (ABBIT51)
File View Actions Tools Help
Nodes in system SV5.1 System
Operations n AdvHtHistorySrv, PID:29536
Logs
Operations
Arguments : 1
Save operation result to file
Translate GUID
Invoke
Operation result

Node	Type
ABBIT51	server
ABBIT54	client

Information of ABBIT51

Objects	Count
Applications:	52
Components:	601
Logs:	1396
Operations:	1920
Users:	SV51\800xAAdmin(1)

Applications in ABBIT51

Application	PID
AdvDsOPCConnector	6848
AdvDsOPCConnector	38504
AdvEAEngine	36672
AdvHtHistorySrv	29536
AdvMbAeOPCServer	38448

Component	Operation
OPC Communicator	ShowAspectDependencies
OPC Communicator	SetServerHandleFilter
OPC Communicator	Statistics
OPC Communicator	LevelHelp
OPC Communicator	SetCallTimeThreshold
OPC Communicator	InternalStatistics
OPC Communicator	DumpParams
ServiceInfo	BackEndNodes

Name	Value
Details	D

AdvHtHistorySrv - Dump statistics from the OPC DA Handler (arg: none =

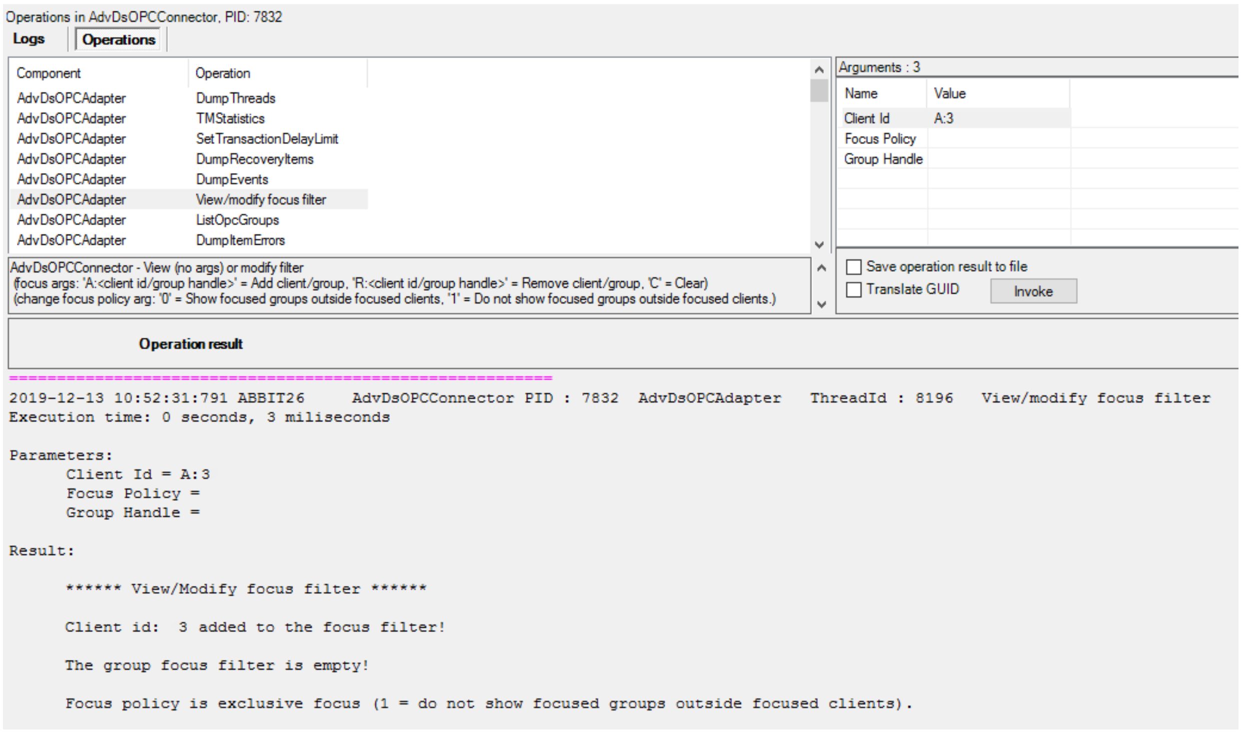
```
Item #264: CH = 698, SH = 67328, ID = '{79AEE9C1-8CC6-4D6
Item #265: CH = 699, SH = 67584, ID = '{7A832A73-5104-4E7
Item #266: CH = 700, SH = 67840, ID = '{7B379B16-A113-48B
Item #267: CH = 701, SH = 68096, ID = '{7B6DA285-FFB0-488
Item #268: CH = 702, SH = 68352, ID = '{7B7D8D28-EE00-42B
Item #269: CH = 703, SH = 68608, ID = '{7B7D8D28-EE00-42B
Item #270: CH = 704, SH = 68864, ID = '{7B7D8D28-EE00-42B
Item #271: CH = 705, SH = 69120, ID = '{7BCDF01A-B4EF-43B
Item #272: CH = 706, SH = 69376, ID = '{7DB3FAFA-73F7-460
```

If the output is too large, it will be directly saved to a text file (the name will be output in the lower right window) or if small be put directly in the window.

Here, all items subscribed will be listed (which can be far larger than the 245 belonging to AC 800M Connect). I have marked out the item described in the earlier pages. Please notice the CH and SH identities which is important to know.

To limit the volume of the logging being applied next, we shall set filter in the OPC DA Adapter (which runs as a subcomponent inside the AdvDsOPCConnector.exe) – so we go back to that node and *AdvDsOPCConnector* process with the PID we identified as being in use for the OPC server in question.

Now we locate the *AdvDsOPCAdapter View/modify focus filter* operation, fill out the ID of the application (ID 3 was found in use by AdvHtHistorySrv.exe by running the *AdvDsOPCConnector Statistics* operation a few pages back)


Operations in AdvDsOPCConnector, PID: 7832

Component	Operation
AdvDsOPCAdapter	DumpThreads
AdvDsOPCAdapter	TMStatistics
AdvDsOPCAdapter	SetTransactionDelayLimit
AdvDsOPCAdapter	DumpRecoveryItems
AdvDsOPCAdapter	DumpEvents
AdvDsOPCAdapter	View/modify focus filter
AdvDsOPCAdapter	ListOpcGroups
AdvDsOPCAdapter	DumpItemErrors

Arguments : 3

Name	Value
Client Id	A:3
Focus Policy	
Group Handle	

AdvDsOPCConnector - View (no args) or modify focus filter
(focus args: 'A:<client id/group handle>' = Add client/group, 'R:<client id/group handle>' = Remove client/group, 'C' = Clear)
(change focus policy arg: '0' = Show focused groups outside focused clients, '1' = Do not show focused groups outside focused clients.)

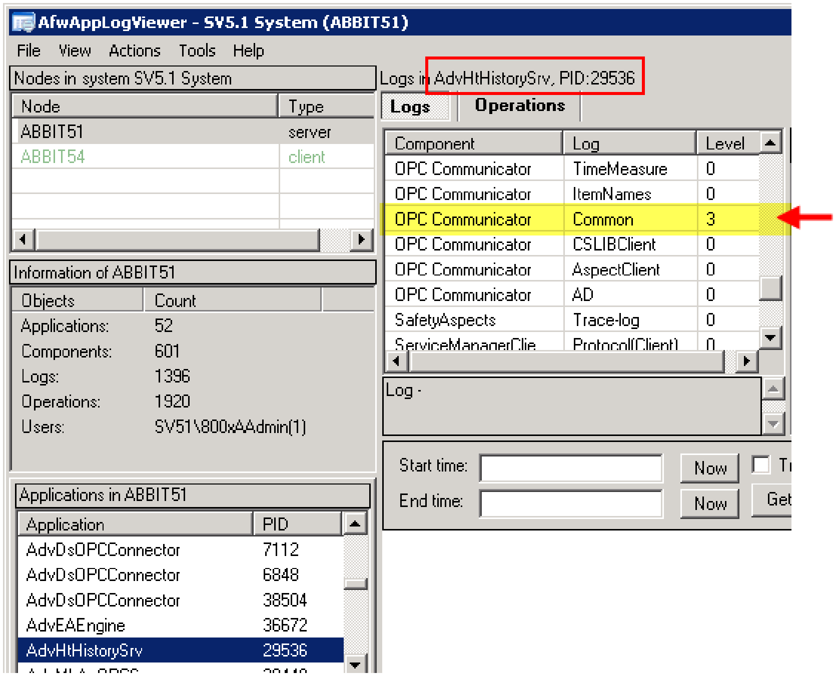
Operation result

```
2019-12-13 10:52:31:791 ABBIT26     AdvDsOPCConnector PID : 7832  AdvDsOPCAdapter   ThreadId : 8196   View/modify focus filter
Execution time: 0 seconds, 3 miliseconds

Parameters:
      Client Id = A:3
      Focus Policy =
      Group Handle =

Result:

      ****** View/Modify focus filter ******

      Client id:  3 added to the focus filter!

      The group focus filter is empty!

      Focus policy is exclusive focus (1 = do not show focused groups outside focused clients).
```

From now on, the OPC DA Adapter logging is limited to this client (Basic History) only.

Now, lets apply logging onto these three locations discussed earlier (OPC Communicator in client, OPC DA Connector and OPC DA Adapter in the server).

First click the Logs button. Then select the AdvHtHistorySrv client, locate the *OPC Communicator Common* log, change log level to three (3).


AfwAppLogViewer - SV5.1 System (ABBIT51)

Nodes in system SV5.1 System

Node	Type
ABBIT51	server
ABBIT54	client

Information of ABBIT51

Objects	Count
Applications:	52
Components:	601
Logs:	1396
Operations:	1920
Users:	SV51\800xAAdmin(1)

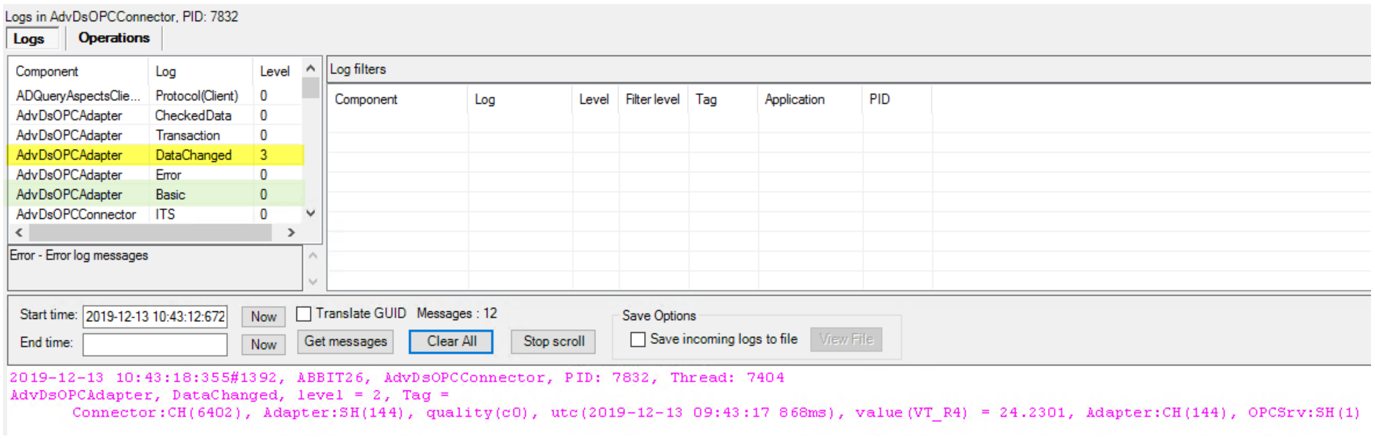
Applications in ABBIT51

Application	PID
AdvDsOPCConnector	7112
AdvDsOPCConnector	6848
AdvDsOPCConnector	38504
AdvEAEngine	36672
AdvHtHistorySrv	29536

Logs in AdvHtHistorySrv, PID:29536

Component	Log	Level
OPC Communicator	TimeMeasure	0
OPC Communicator	ItemNames	0
OPC Communicator	Common	3
OPC Communicator	CSLIBClient	0
OPC Communicator	AspectClient	0
OPC Communicator	AD	0
SafetyAspects	Trace-log	0
ServiceManagerClie	Protocol(Client)	0

Then locate the OPC DA Connector process, and the *AdvDsOPCAdapter* component's *DataChanged** log and it to log level three (3):
**) In older versions of 800xA Base, this log does not exist, but the same content will be available in the Basic log.*



Component	Log	Level
ADQueryAspectsClie...	Protocol(Client)	0
AdvDsOPCAdapter	CheckedData	0
AdvDsOPCAdapter	Transaction	0
AdvDsOPCAdapter	DataChanged	3
AdvDsOPCAdapter	Error	0
AdvDsOPCAdapter	Basic	0
AdvDsOPCConnector	ITS	0

Component	Log	Level	Filter level	Tag	Application	PID

Start time: 2019-12-13 10:43:12:672
End time:
Messages : 12

```
2019-12-13 10:43:18:355#1392, ABBIT26, AdvDsOPCConnector, PID: 7832, Thread: 7404
AdvDsOPCAdapter, DataChanged, level = 2, Tag =
      Connector:CH(6402), Adapter:SH(144), quality(c0), utc(2019-12-13 09:43:17 868ms), value(VT_R4) = 24.2301, Adapter:CH(144), OPCSrv:SH(1)
```

Probably, by now, there will be some output sent to the lower right part of the window. If not, click the upper **Now** button to enter a *Minimum Time* filter preventing old data to be logged, then click **Get messages** to refresh the logging settings. Now logging should become visible in the output window.

Let the log run for a minute, then reset the log levels back to zero (0). A service restart, or computer reboot will likewise reset the logging back to default settings. Please note that some logs are enabled by default. That is fully normal.

Now use menu item *File→Save log messages as…* to a text file of your choice.

The rest is decoding the output as listed on one of the early pages of this document; carefully mapping the Client and Server handles so that you can see if data for a certain handle is sent by the true source OPC server or not.

A missing item can be attempted to be "refreshed" by disabling and re-enabling the subscription, e.g. toggle the **Enabled** flag of a Log Configuration. Please note that doing so will most likely generate a new chain of Client and Server handles starting in the OPC Communicator in the client and ending in the OPC DA Adapter. I.e. you must establish the "chain" once more.

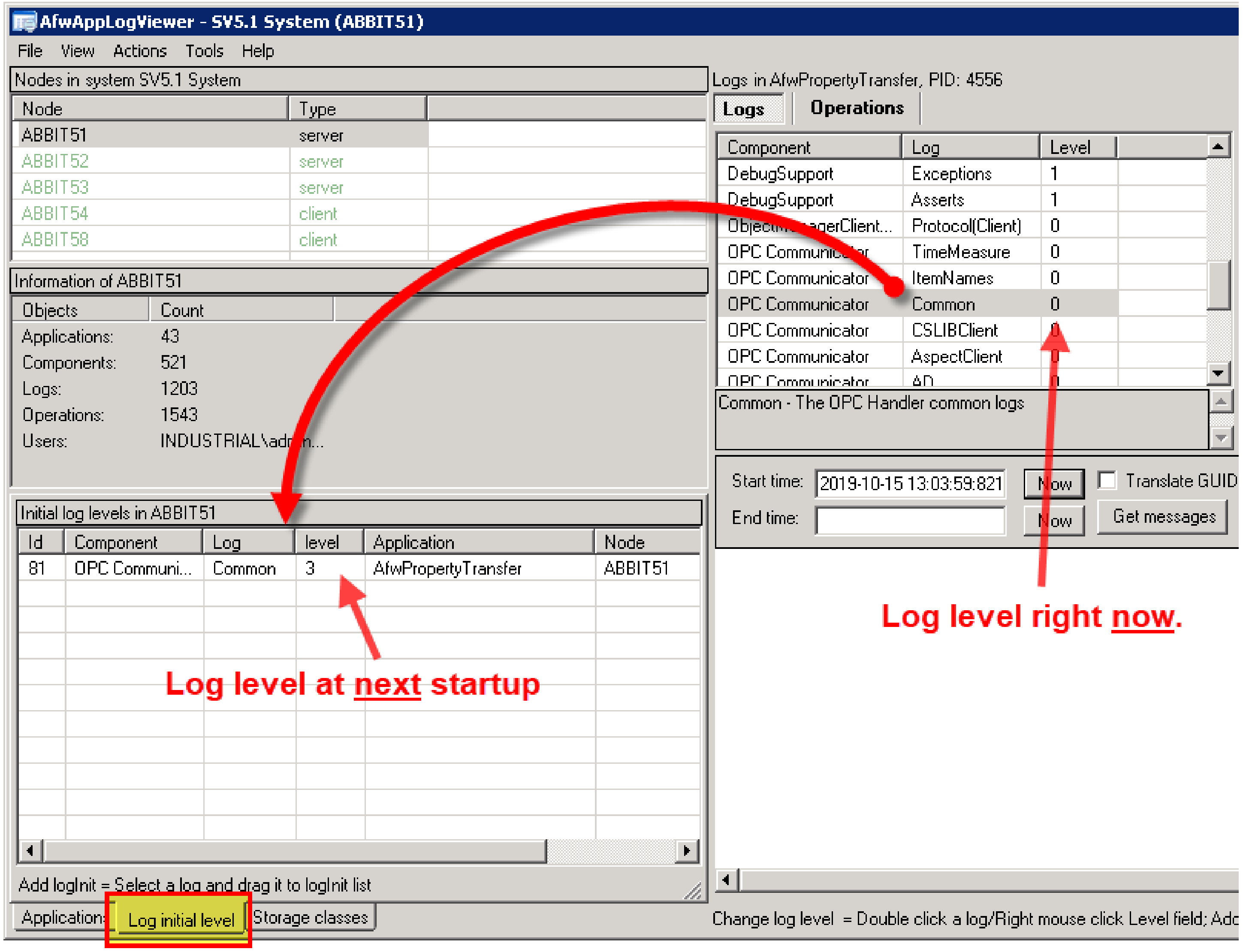
If the "new chain" deliver data, but not the "old chain", it points to the source OPC server having "forgot" or erroneously "cleared" the subscription. In OPC of 800xA, the client does not monitor the feed, it is every OPC server's duty to remember all items added to it. It must not "forget" an item until the client has removed the subscription.

With all the above in memory, you now understand that it is not possible to verify any *other* client's subscription by performing "Subscribe for live data", opening a new faceplate or process graphics aspect. You can only check your own subscriptions, from "now". To see what Basic History subscribed to yesterday or previous week you must use applog.

Happy troubleshooting! ☺
/Stefan

If it is desired to have a log enabled from startup of a process, it is necessary to add the log(s) with *Log Init.* A log init is made by dragging & dropping the logs to the list of log inits on the *Log Initial Level* tab.

IMPORTANT NOTE: Log Inits must be manually cleared, or they will continue to be in effect.



Nodes in system SV5.1 System

Node	Type
ABBIT51	server
ABBIT52	server
ABBIT53	server
ABBIT54	client
ABBIT58	client

Information of ABBIT51

Objects	Count
Applications:	43
Components:	521
Logs:	1203
Operations:	1543
Users:	INDUSTRIAL\admin...

Initial log levels in ABBIT51

Id	Component	Log	level	Application	Node
81	OPC Communi...	Common	3	AfwPropertyTransfer	ABBIT51

Add logInit = Select a log and drag it to logInit list

Logs in AfwPropertyTransfer, PID: 4556

Component	Log	Level
DebugSupport	Exceptions	1
DebugSupport	Asserts	1
ObjectManagerClient...	Protocol(Client)	0
OPC Communicator	TimeMeasure	0
OPC Communicator	ItemNames	0
OPC Communicator	Common	0
OPC Communicator	CSLIBClient	0
OPC Communicator	AspectClient	0
OPC Communicator	AD	0

Common - The OPC Handler common logs

Start time: 2019-10-15 13:03:59:821

End time:

Change log level = Double click a log/Right mouse click Level field; Add

Applog operation: **OPC DA Communicator Statistics (with argument "D" for full detail) on the client process**

```
OPC Group statistics:
=====================
Group #1: Name: 'Group1', Update rate: 1000, Number of items: 4, Pending write: 0, Pending read: 0
    Item #1: CH = 1, SH = 0, ID = 'Root/Control Network/Supportline_24x365/Applications/Application_1/Diagrams/Diagram2:temp_eow'
    Item #2: CH = 2, SH = 256, ID = 'Root/Control Network/Supportline_24x365/Applications/Application_1/Diagrams/Diagram2:temp_serverrum'
    Item #3: CH = 3, SH = 512, ID = 'Root/Control Network/Supportline_24x365/Applications/Application_1/Diagrams/Diagram2:temp_syd'
    Item #4: CH = 4, SH = 768, ID = 'Root/Control Network/Supportline_24x365/Applications/Application_1/Diagrams/Diagram2:temp_tavlan'
```

Applog log: ***OPC DA Connector/OPC Adapter Data Changed log*** **- an update with 4 items is received from the source OPC server**

```
2019-12-13 16:13:17:650#40302, ABBIT26, AdvDsOPCConnector, PID: 7832, Thread: 5332
AdvDsOPCAdapter, DataChanged, level = 2, Tag =
       Connector:CH(6402), Adapter:SH(144), quality(c0), utc(2019-12-13 15:13:17 313ms), value(VT_R4) = 24.4083, Adapter:CH(144), OPCSrv:SH(1)


2019-12-13 16:13:17:651#40303, ABBIT26, AdvDsOPCConnector, PID: 7832, Thread: 5332
AdvDsOPCAdapter, DataChanged, level = 2, Tag =
       Connector:CH(6918), Adapter:SH(175), quality(c0), utc(2019-12-13 15:13:17 313ms), value(VT_R4) = 18.9566, Adapter:CH(175), OPCSrv:SH(2)


2019-12-13 16:13:17:651#40304, ABBIT26, AdvDsOPCConnector, PID: 7832, Thread: 5332
AdvDsOPCAdapter, DataChanged, level = 2, Tag =
       Connector:CH(6662), Adapter:SH(176), quality(c0), utc(2019-12-13 15:13:17 313ms), value(VT_R4) = 22.6334, Adapter:CH(176), OPCSrv:SH(3)


2019-12-13 16:13:17:651#40305, ABBIT26, AdvDsOPCConnector, PID: 7832, Thread: 5332
AdvDsOPCAdapter, DataChanged, level = 2, Tag =
       Connector:CH(3597), Adapter:SH(177), quality(c0), utc(2019-12-13 15:13:17 313ms), value(VT_R4) = 25.1212, Adapter:CH(177), OPCSrv:SH(4)
```

Applog log: **OPC Communicator Common log - the update is received in the client**

```
2019-12-13 16:13:17:655#4592, ABBIT28, AdvDSOPCClient, PID: 5224, Thread: 5612
OPC Communicator, Common, level = 2, Tag =
       NewValues_2_0: OpcHandler=0x60acbd0, DataSource=Connector
       GroupSH:0 Client:CH(1), OPCHdlr:SH(0), ItemID:'Root/Control Network/Supportline_24x365/Applications/Application_1/Diagrams/Diagram2:temp_eow',
       (+) NewValue:value(VT_R4) = 24.4083, Quality:0xc0, Time(UTC):2019-12-13 15:13:17 313ms, Result:0x0, OPCHdlr:CH(0), Connector:SH(6402)

2019-12-13 16:13:17:655#4593, ABBIT28, AdvDSOPCClient, PID: 5224, Thread: 5612
OPC Communicator, Common, level = 2, Tag =
       NewValues_2_0: OpcHandler=0x60acbd0, DataSource=Connector
       GroupSH:0 Client:CH(2), OPCHdlr:SH(256), ItemID:'Root/Control Network/Supportline_24x365/Applications/Application_1/Diagrams/Diagram2:temp_serverrum',
       (+) NewValue:value(VT_R4) = 18.9566, Quality:0xc0, Time(UTC):2019-12-13 15:13:17 313ms, Result:0x0, OPCHdlr:CH(256), Connector:SH(6918)

2019-12-13 16:13:17:655#4594, ABBIT28, AdvDSOPCClient, PID: 5224, Thread: 5612
OPC Communicator, Common, level = 2, Tag =
       NewValues_2_0: OpcHandler=0x60acbd0, DataSource=Connector
       GroupSH:0 Client:CH(3), OPCHdlr:SH(512), ItemID:'Root/Control Network/Supportline_24x365/Applications/Application_1/Diagrams/Diagram2:temp_syd',
       (+) NewValue:value(VT_R4) = 22.6334, Quality:0xc0, Time(UTC):2019-12-13 15:13:17 313ms, Result:0x0, OPCHdlr:CH(512), Connector:SH(6662)

2019-12-13 16:13:17:655#4595, ABBIT28, AdvDSOPCClient, PID: 5224, Thread: 5612
OPC Communicator, Common, level = 2, Tag =
       NewValues_2_0: OpcHandler=0x60acbd0, DataSource=Connector
       GroupSH:0 Client:CH(4), OPCHdlr:SH(768), ItemID:'Root/Control Network/Supportline_24x365/Applications/Application_1/Diagrams/Diagram2:temp_tavlan',
       (+) NewValue:value(VT_R4) = 25.1212, Quality:0xc0, Time(UTC):2019-12-13 15:13:17 313ms, Result:0x0, OPCHdlr:CH(768), Connector:SH(3597)
```