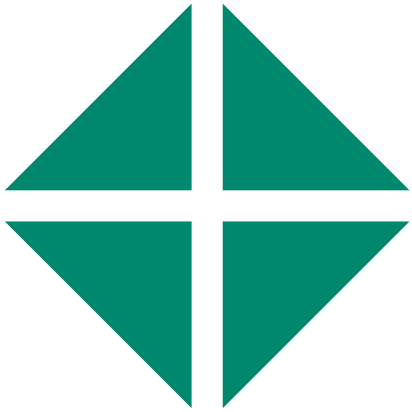
cytiva

UNICORN 7.6

OPC Manual



cytiva.com

Table of Contents

1 Introduction

This chapter contains important user information, compatibility information and a description of the intended use of UNICORN™ OPC server.

OPC server

UNICORN OPC server provides a standardized integration interface to support integration between UNICORN and other software systems such as Laboratory Information Systems (LIMS) and Manufacturing Execution Systems (MES or SCADA). OPC enables open connectivity via open standards created in collaboration with leading automation manufacturers worldwide, including Microsoft®.

OPC provides inter-operability between system components by creating and maintaining open standard specifications. The benefit of following standard specifications in system implementations is greater independence for hardware and software components. This leads to higher flexibility, better quality and overall lower maintenance costs for the system solution. The first standard developed was the Data Access specification, which therefore has the broadest client support.

UNICORN OPC server supports the following three areas:

- UNICORN OPC Data Access gives access to all process data (i.e., real-time values, valve status, process step information and commands). UNICORN Alarm & Events server informs an OPC client application that a system parameter has exceeded an upper or lower limit value. The UNICORN Alarm & Events server also provides information about the process (***RunLog***).
- UNICORN Historical Data Access allows any OPC client application to access the entire batch result generated by UNICORN.
- UNICORN OPC Security controls client access to the UNICORN OPC Data Access, Alarms & Events and Historical Data Access to protect sensitive information and to guard against unauthorized modification of process parameters. This is an important security feature.

UNICORN server supported OPC standards and versions

Product 1 supports:	
Data Access (DA)	1.0 custom interface 2.05A custom interface 3.0 custom interface
Alarms & Events (A&E)	1.1 custom interface

Product 1 supports:	
Historical Data Access (HDA)	1.1 custom interface 1.2 custom interface
Security	1.0 custom interface

To ensure high compatibility, UNICORN OPC has been compliance tested. See the OPC homepage:

www.opcfoundation.org

This document requires basic knowledge of UNICORN, as well as OPC clients. Refer to the UNICORN manuals and *Application Notes* for detailed information about configuring OPC for specific OPC clients.

2 UNICORN OPC server general settings

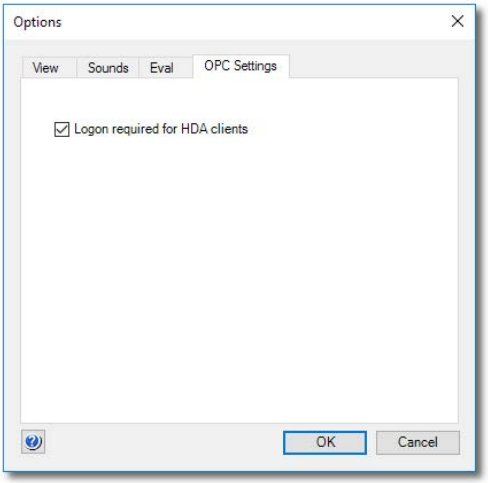
This chapter provides the required system settings for the OPC server.

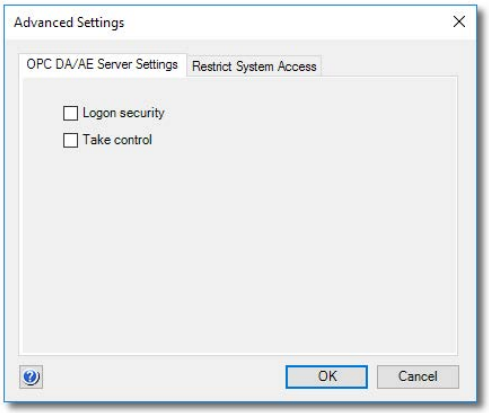
In this chapter

2.1 System start up

UNICORN Administration

When the server is executed, it first checks the basic configuration. This basic configuration is adjustable from ***UNICORN Administration***'s ***Options*** dialog box for HDA server and ***Advanced Settings*** dialog in ***System Properties*** for AE and DA servers (***Tools*** →***System Properties*** →***Edit*** →***Advanced Settings***).





Note: *The OPC connection is not allowed when the system is in* ***Initializing state****.*

2.2 Logon security

UNICORN OPC server supports OPC private security. When enabled, the client must log in before accessing any items.

All UNICORN users can log into OPC by giving username with/without the access group to which user belongs to. For example "`username@accessgroupname`" or "`username`". If the access group is not given during login, then the first available access group for that user will be taken. User access rights are depending on the access group chosen. It is not possible to change password or administer users through OPC. User management is handled through UNICORN ***Administration*** module.

Logon security enables the OPC security interface. Since not all clients support this interface, the default installation setting is ***OFF***. When the interface is enabled, the server requires all clients to login before they can access data. This applies to all interfaces, that is, DA, AE or HDA.

OPC interface dependencies

Security v 1.0		
Data Access v 1.0, 2.05A, 3.0	Alarms & Events v 1.1	Historical Data Access v 1.2

If the ***Logon security*** option is ***OFF***, the server disables the security interface. Instead, it uses "OPC" user when communicating with UNICORN.

2.3 System connection (Take Control)

The ***Take Control*** flag is used when OPC is connecting to the system. If the flag is set to ***ON***, the ***AssignState*** is set to ***Connected in control***. Otherwise, it is set to the default value ***Connected in view***. However, if another user already has control over the system, OPC is assigned ***Connected in view*** state.

3 UNICORN OPC remote

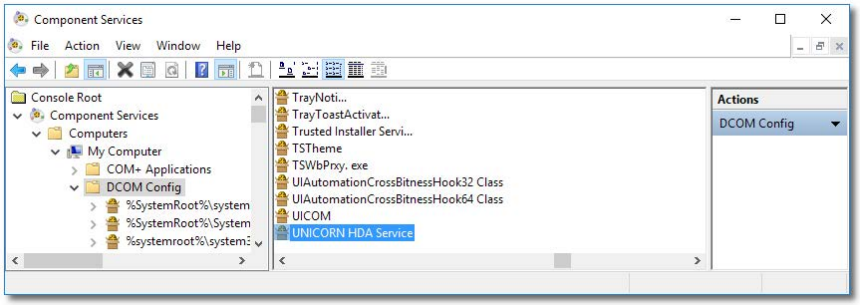
This chapter contains information about configuring remote access, recommended settings and the related error codes.

In this chapter

3.1 Security settings

UNICORN OPC supports remote access via DCOM. Some DCOM security settings might have to be altered to get it to work. The configuration can be changed through ***Component Services***.

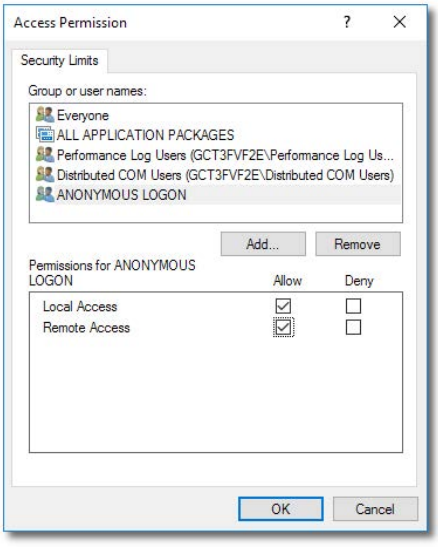
Component Services



The recommended security settings for ***UNICORN HDA Service*** and ***UNICORN Instrument Server*** are shown below. For detailed instructions, see *Troubleshoot communication issues, on page 25*.

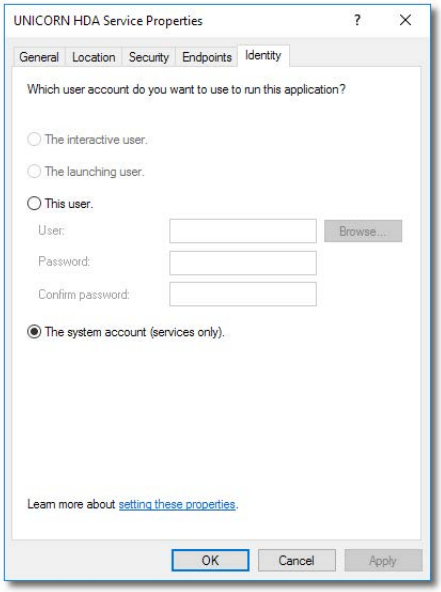
- Add ***Access Permission*** to the users ***ANONYMOUS LOGON***, ***Everyone***, ***INTERACTIVE***, ***NETWORK*** and ***SYSTEM***.

UNICORN *Access Permission* window



- Set ***Which user do you want to use to run this application?*** in the ***Identity***-tab to ***The system account (services only)***.

UNICORN HDA Server Properties window



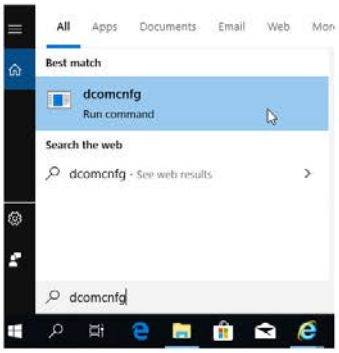
Open DCOM communications port

If you are using a firewall, make sure that port 135 is open.

DCOM encryption settings for OPC

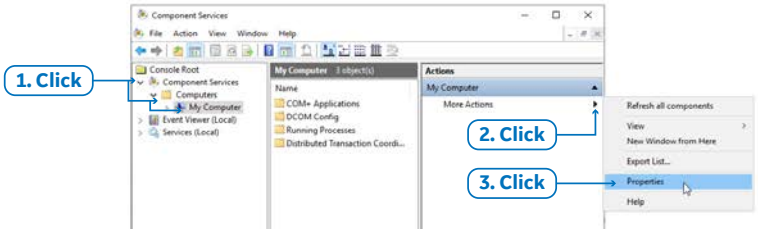
Server-side settings

Step	Action
1	Click the Windows search icon, type in `dcomcnfg` and then click ***dcomcnfg***.

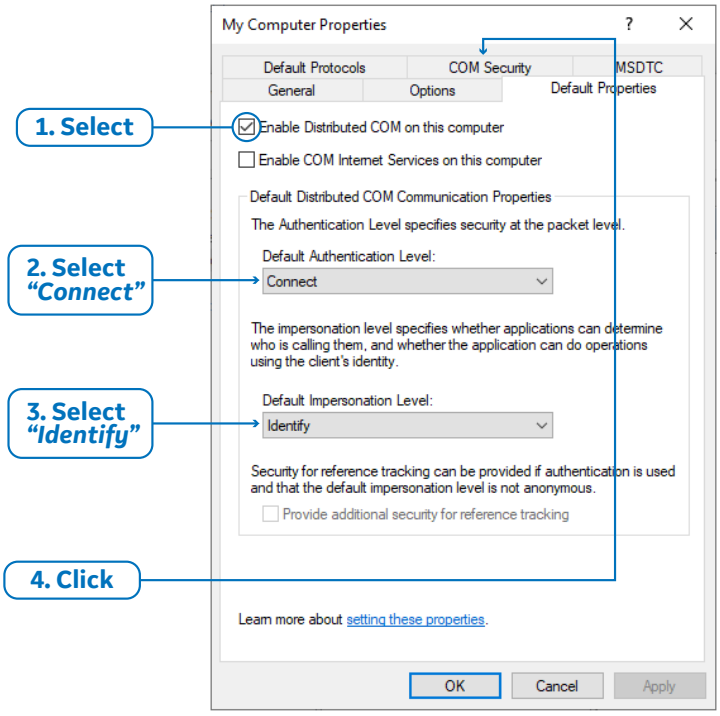


IMPORTANT
If you are logged in as Administrator, you will be prompt for confirmation. Click ***Yes*** to proceed.

2	Follow the instructions in the following illustration:

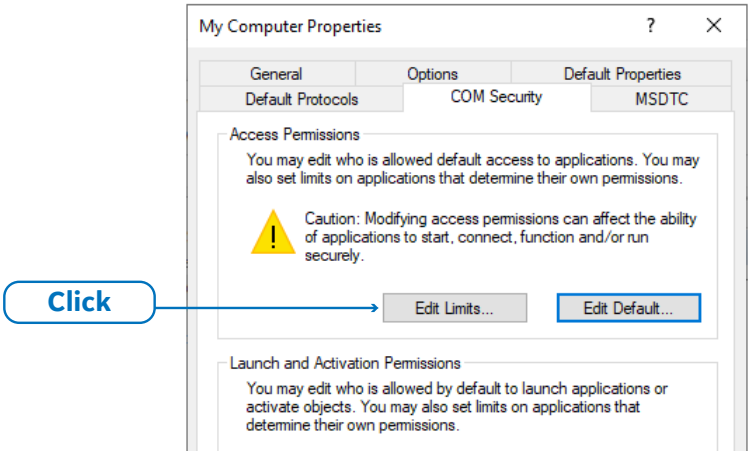


Step	Action
3	**a.** For **non-encrypted** DCOM communication, follow the instructions in the following illustration:

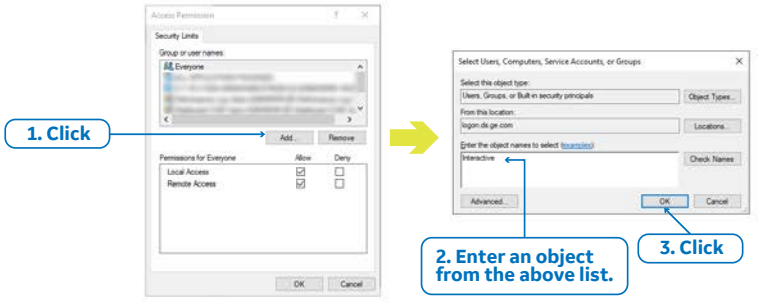


b. For **encrypted** DCOM communication, only change from ***Connect*** to ***Packet Privacy*** under ***Default Authentication level***. See the previous image for reference.

Step	Action
4	Click ***Edit Limits***.

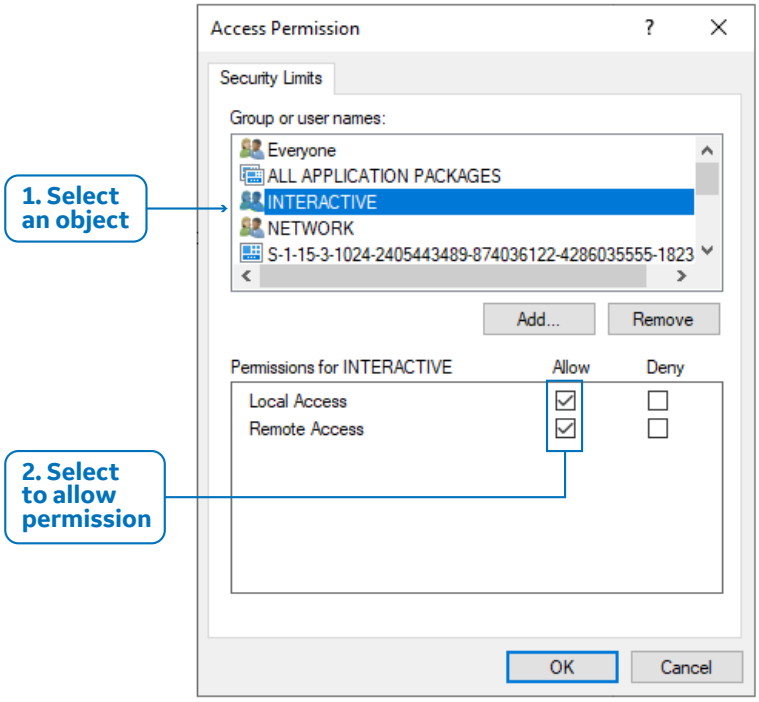


5	Add the following users (objects) as described in the following illustration: • Everyone (by default it is available) • Interactive • Network • System • Anonymous Login

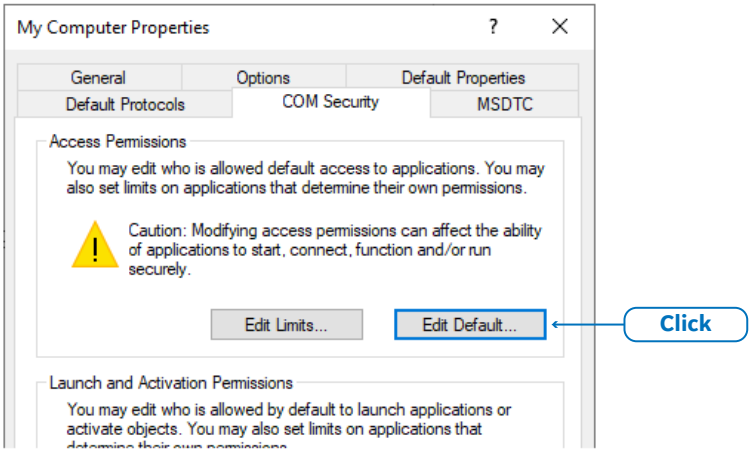


6	Repeat step 5 for all the users in the list.

Step	Action
7	Provide permissions to the users as illustrated below:

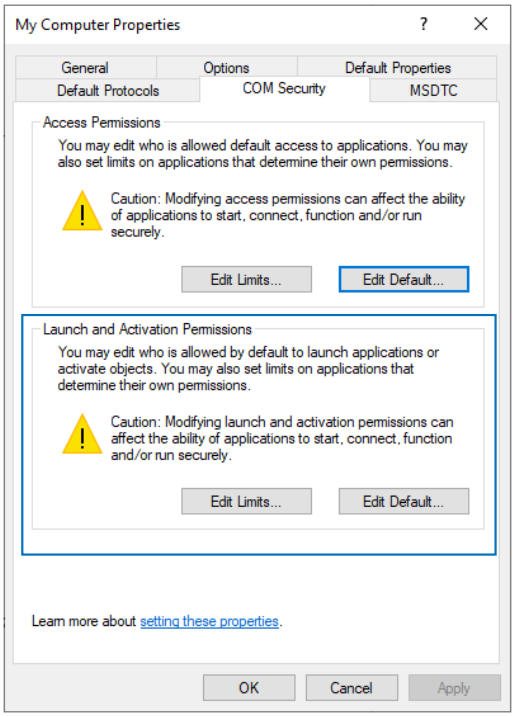


8	Repeat step 7 for all the added (see step 5) users.
9	Click ***Edit*** in the ***My Computer Properties*** window.

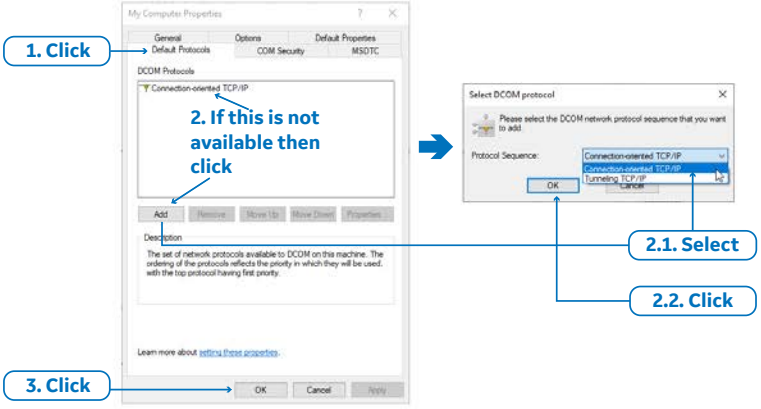


10	Repeat step 5 - step 7.

Step	Action
11	Repeat step 5 to step 10 for ***Launch and Activation Permissions***.

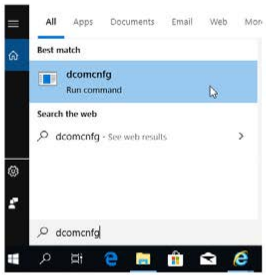


12	Follow the instructions in the following illustration:



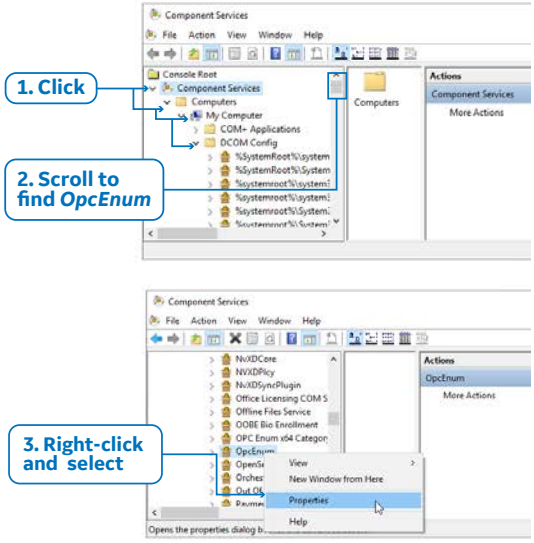
OPC Enum

1. Click the Windows search icon, type in `dcomcnfg` and then click ***dcomcnfg***.

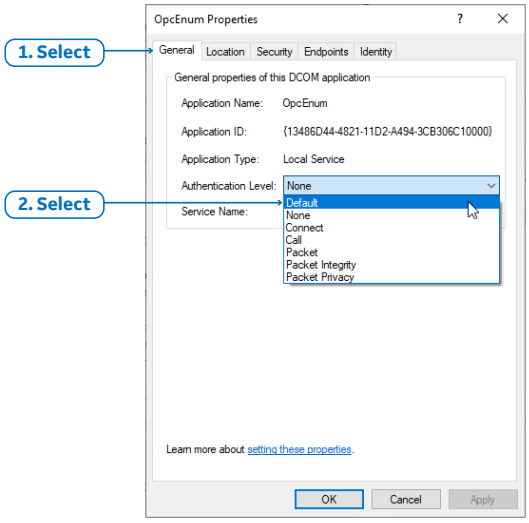


IMPORTANT
If you are logged in as Administrator, you will be prompt for confirmation. Click ***Yes*** to proceed.

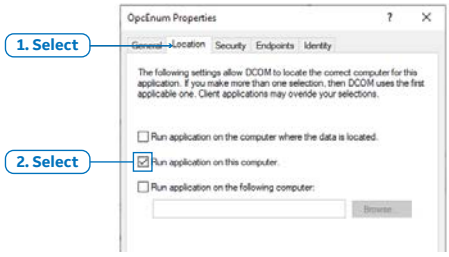
2. Follow the instructions in the following illustrations:



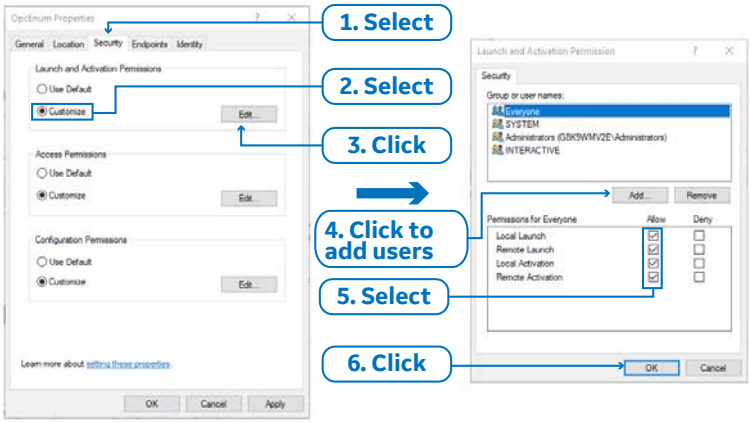
3. In the ***General*** tab, set the ***Authentication level*** to ***Default***.



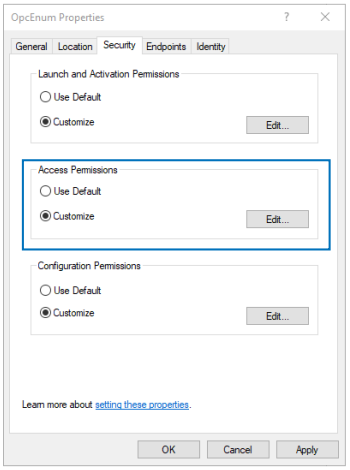
4. Under the ***Location*** tab select ***Run application on this computer***.

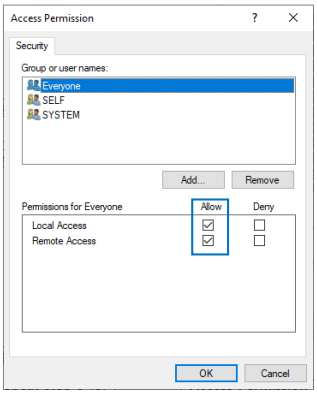


5. Allow permission for ***Everyone, Interactive, Network, Anonymous***, and ***System*** following the instructions in the illustration:

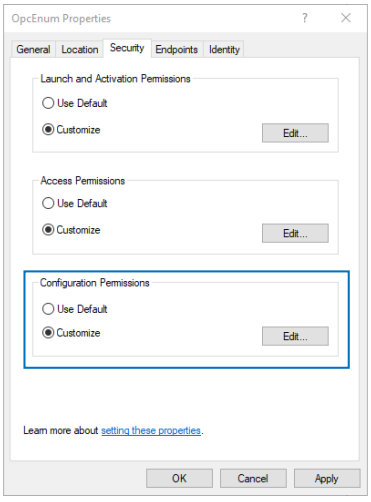


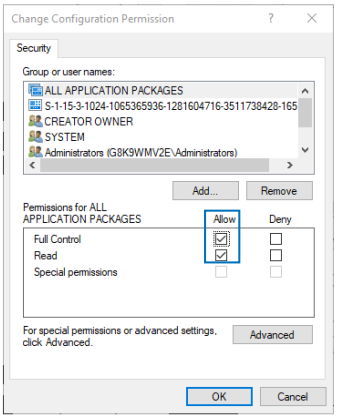
6. Repeat step 5 for ***Access Permission***.



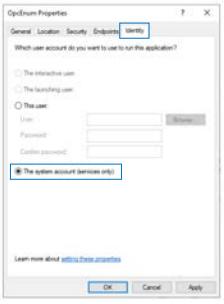


7. Repeat step 6 for ***Configuration Permissions***.

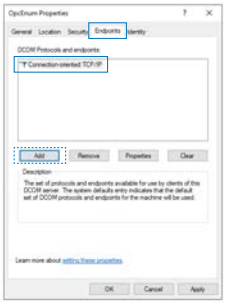




8. In the ***Identity*** tab, select ***The system account***.



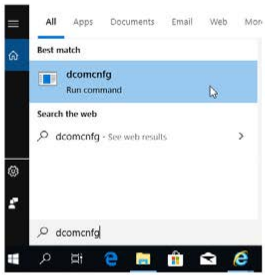
9. In the ***Endpoints*** tab, check if the ***DCOM Protocol and endpoint session*** has ***Connection-oriented TCP/IP***. If no, click ***Add*** and add ***Connection-oriented TCP/IP***.



10. Click ***Apply*** and then ***OK*** to save the settings and close the ***OpcEnum Properties*** window.

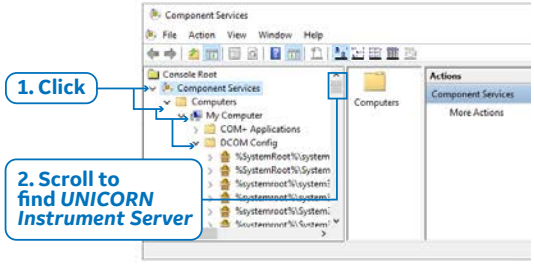
UNICORN Instrument Server

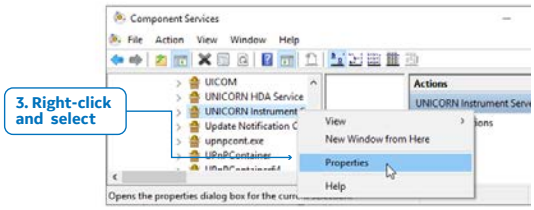
1. Click the Windows search icon, type in `dcomcnfg` and then click ***dcomcnfg***.



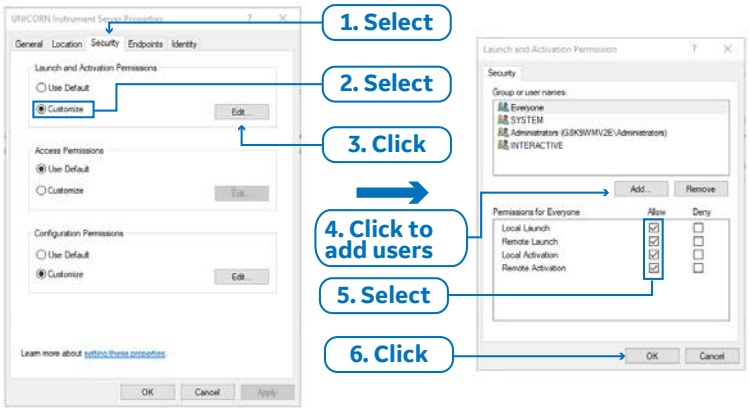
> **IMPORTANT**
> If you are logged in as Administrator, you will be prompt for confirmation. Click ***Yes*** to proceed.

2. Follow the instructions in the following illustrations:





3. In the ***General*** tab, set the ***Authentication level*** to ***Connect***.
4. In the ***Security*** tab select:
   - ***Customize*** for ***Launch and Activation Permissions***,
   - ***Use Default*** for ***Access Permissions***,
   - ***Customize*** for ***Access Permissions***.
5. Follow the instructions in the illustration:



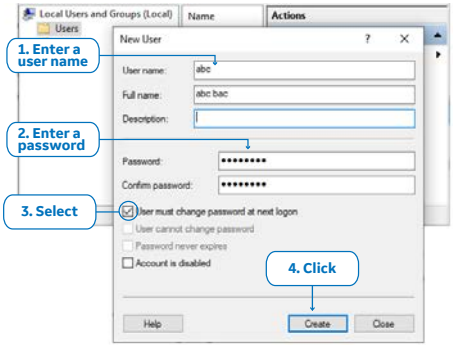
6. In the ***Identity*** tab, make sure that either ***The interactive user*** or ***The system account*** is selected.
7. Click ***Apply*** and then ***OK*** to save the settings and close the ***UNICORN Instrument Server Properties*** window.

Create new user

1. In the Windows search button, type in `lusrmgr.msc` and then click ***lusrmgr.msc***.

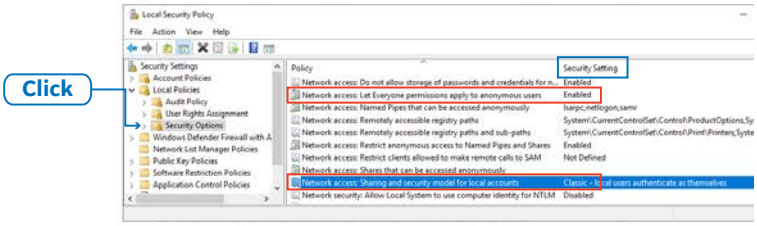
> **IMPORTANT**
> If you are logged in as Administrator, you will be prompt for confirmation. Click ***Yes*** to proceed.

2. Right-click ***Users*** and then select ***New User***.
3. Follow the instructions in the following illustration:



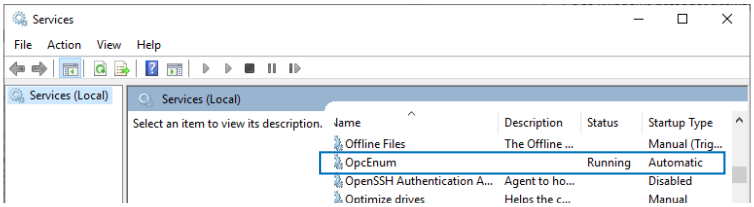
Set security option

1. In the Windows search button, type in `secpol.msc` and then click ***secpol.msc***.
2. Make sure that the ***Security Setting*** for the items highlighted in following illustration are ***Enabled*** and ***Classic-local users authenticate as themselves***:



Check OpcEnum status

1. In the Windows search button, type in `services` and then click ***Services***.
2. Make sure that ***OpcEnum*** is ***Running*** with ***Automatic*** as ***Startup Type***.



Firewall settings

1. In the Windows search button, type in `firewall.cpl` and then click ***firewall.cpl***.
2. Click ***Turn Windows Defender Firewall on or off*** in the left menu and then select ***Turn ON Windows defender firewall*** for both public and private network settings.
3. Add firewall ports as Inbound.

1. Click ***Advanced*** on the left menu.
2. Click ***Inbound Rules*** on the left menu.
3. Enable ***File and Printer Sharing*** (***Right-click Enable***).

   Note: *Client and server machines must be able to ping each other.*
4. Close this window.

4. In the ***Windows Defender Firewall***, click ***Allow an app or feature.....*** on the left menu.
5. Make sure that the following are added in the ***Allowed app*** list:

   - OpcEnum
   - DCOM
   - DCOM Port
   - TCP Ports for UNICORN

   If error occurs, see *Event ID-10016*.

Client-side settings

1. Do the steps 1 to 6 as described in *Server-side settings, on page 12*.
2. Do the steps as described in *OPC Enum, on page 17*.
3. Do the steps as described in *Firewall settings, on page 23*.

3.2 Configuration

Use domain users

It is easier to setup a secure OPC communication between a remote OPC client and the OPC server if the computers in the network are connected to a domain server and the domain users are logged onto each computer.

If the computers are not connected to a domain, add them to a workgroup and create users with the same credentials on all computers in the network.

Get DCOM call/activation log

Detailed information on error caused by incorrect DCOM configuration is logged in the Windows event log. Follow the instruction below to retrieve the information.

In the registry below the key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Ole\` add the following ***DWORD*** values:

- `ActivationFailureLoggingLevel` - ***DWORD*** value with value `1`.
- `CallFailureLoggingLevel` - ***DWORD*** value with value `1`.

Troubleshoot communication issues

Communication failure between an OPC client and an OPC server is often caused by DCOM security issues. There is no DCOM configuration that works on all systems since it depends on the operating system used, network settings, computer configuration and the required security level.

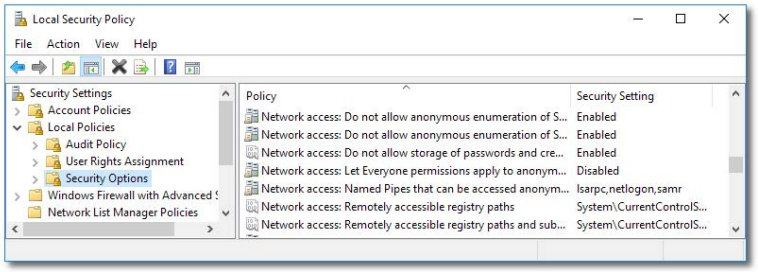
Note: *In case of communication issues, it is recommended to first remove all DCOM security settings to get the communication working and then gradually increase the security level.*

Follow the instructions below to troubleshoot communication issues.

1. Turn off Windows firewall
2. Change local security policy
3. Change default (system wide) DCOM settings
4. Change server specific DCOM configuration

Change local security policy

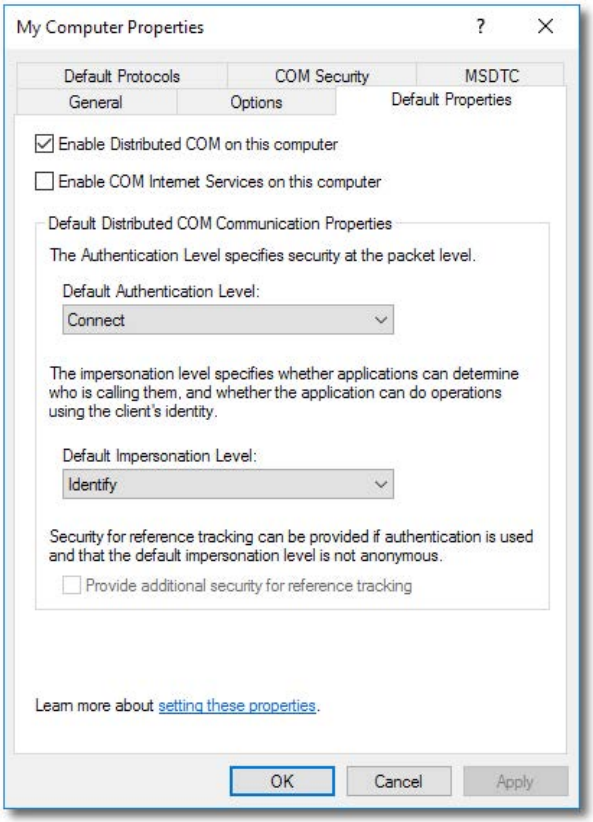
Step	Action
1	Type "`Local Security Policy`" in the Windows Start menu search field, click the ***Local Security Policy*** item that is displayed as the search result.

Step	Action
2	Expand ***Security Settings*** →***Local Policies*** →***Security Options*** 
3	Set ***Network access →Let Everyone permissions apply to anonymous users*** to ***Enabled***.
4	Set ***Network access →Sharing and security model for local accounts*** to ***Classic – local users authenticate as themselves***.

Change default DCOM settings system wide

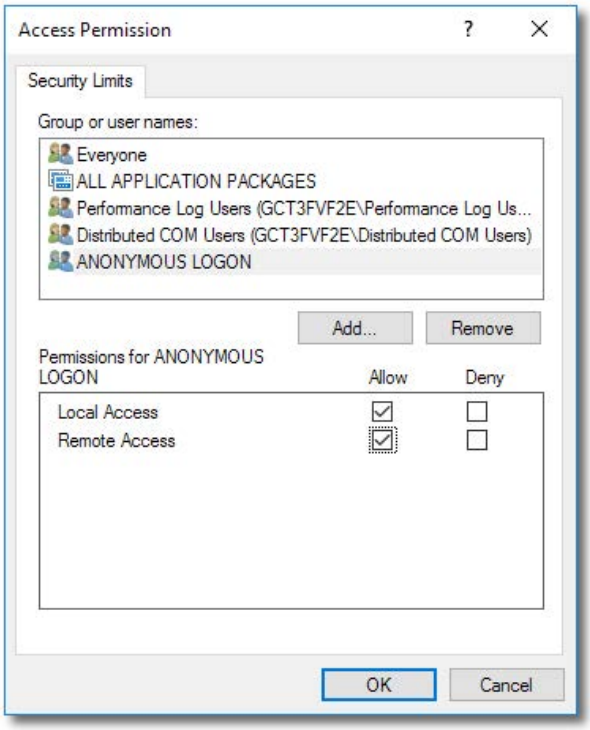
Step	Action
1	Type "`Component Services`" in the Windows start menu search field, click the ***Component Services*** item that is displayed as the search result.

Step	Action
2	Expand ***Component Services*** →***Computers***. Right-click ***My Computer***, then select ***Properties***. In ***Default Properties*** tab set ***Default Authentication Level*** to ***Connect***.



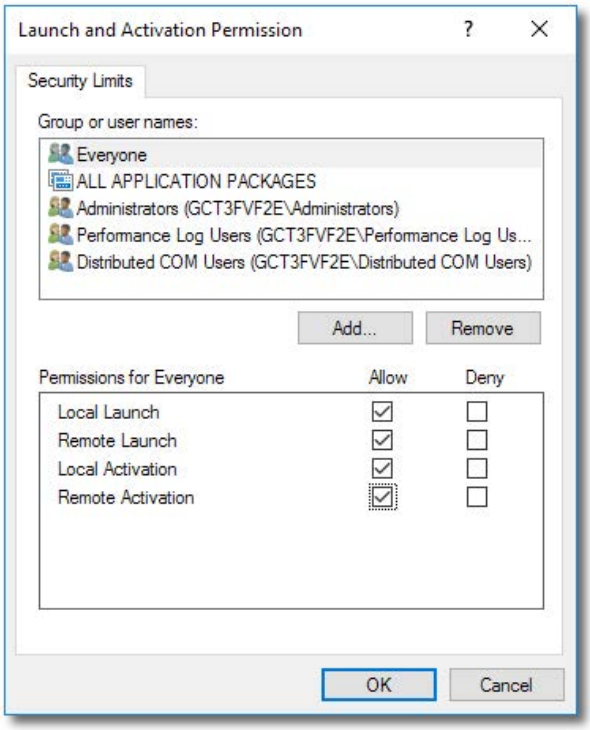
3	In ***COM Security*** tab, ***Access Permissions*** pane, click ***Edit Limits***.

Step	Action
4	Add ***Interactive***, ***Network*** and ***System*** users. Set the ***Everyone***, ***Interactive***, ***Network*** and ***System*** users to have both ***Local Access*** and ***Remote Access***. Click ***OK***.



5	In ***COM Security*** tab, ***Launch and Activation Permissions*** pane, click ***Edit Limits***.

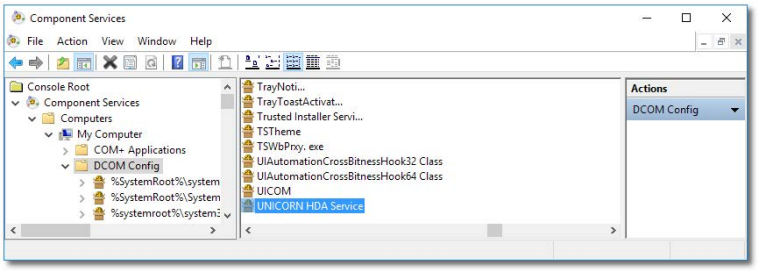
Step	Action
6	Add ***Anonymous Logon***, ***Interactive***, ***Network***, ***System*** and ***Everyone***. Set them to have ***Local Launch***, ***Remote Launch***, ***Local Activation*** and ***Remote Activation*** permissions.



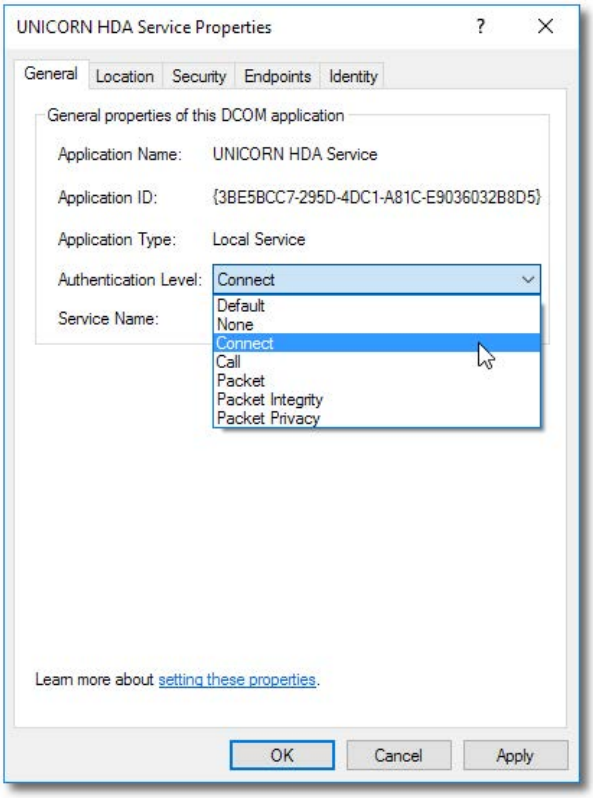
7	Click ***OK*** to close the ***Launch Permission*** window.

Change server-specific DCOM configuration

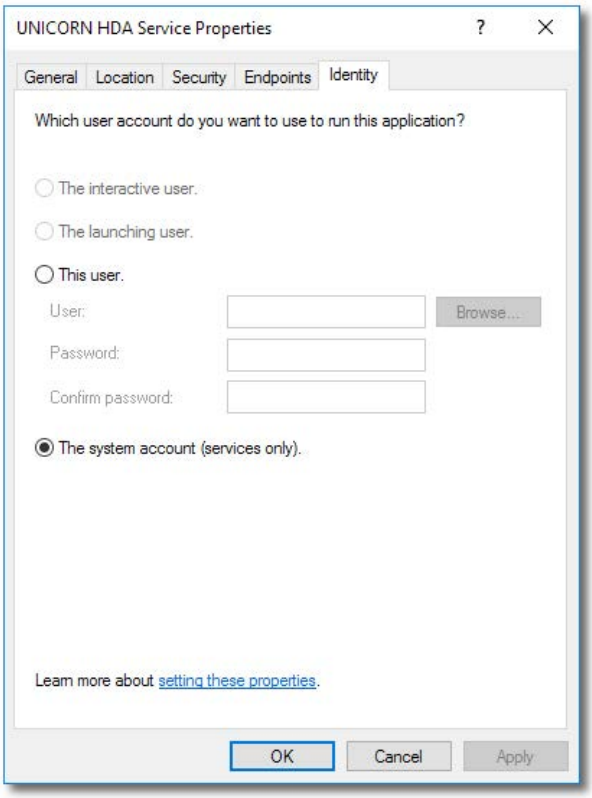
Step	Action
1	Expand ***Component Services*** →***Computers*** →***My Computer*** →***DCOM Config***. Answer ***Yes*** on any question asked when expanding ***DCOM Config***.
2	Locate ***UNICORN HDA Service, UNICORN Instrument Server*** and ***OpcEnum*** in the list.
3	Right-click ***UNICORN HDA Service***, select ***Properties***.



Step	Action
4	In the ***General*** tab, set ***Authentication level*** to ***Connect***.

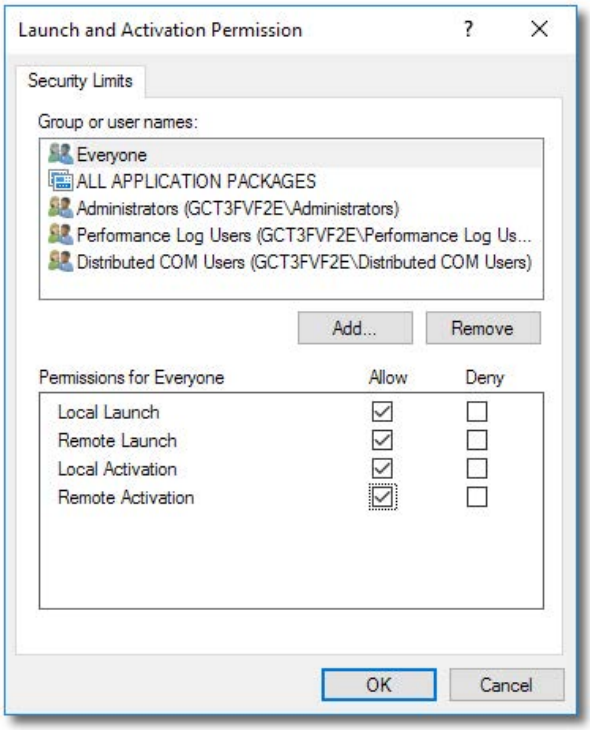


Step	Action
5	In the ***Identity*** tab, set ***Authentication level*** to ***The system account (services only)*** (or to ***The interactive user***, if not a service)



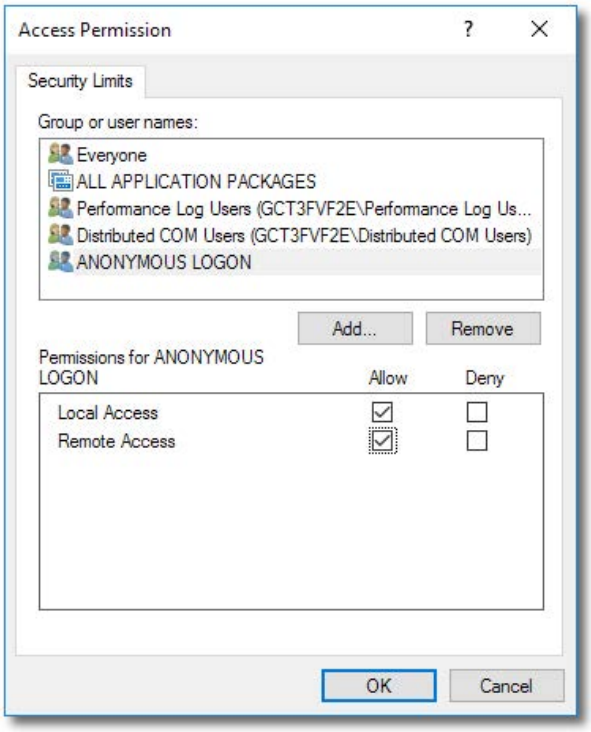
6	Click the ***Security*** tab, then in the ***Launch and Activation Permissions*** pane click ***Customize***, then click ***Edit***.

Step	Action
7	Set the ***INTERACTIVE***, ***NETWORK***, ***Everyone*** and ***SYSTEM*** users ***Permissions*** to ***Allow Local Launch***, ***Remote Launch***, ***Local Activation*** and ***Remote Activation***.



8	Click ***OK*** to close the ***Launch and Activation Permissions window***.
9	In the ***Access Permissions*** pane click ***Customize***, then click ***Edit***.

Step	Action
10	Set the ***INTERACTIVE***, ***NETWORK***, ***Everyone*** and ***SYSTEM*** users ***Permissions*** to ***Allow Local Access*** and ***Remote Access***.



11	Click ***OK*** to close the ***Access Permission*** window
12	Click OK to close the ***UNICORN HDA Service Properties*** window.
13	Redo step 3 to 12 for ***UNICORN Instrument Server*** and ***OpcEnum***.

3.3 UNICORN OPC custom error codes

The server defines several custom error codes that extend the OPC error codes. They are returned when reading or writing items. The error codes can be avoided by checking the ***OPC_PROP_RIGHTS*** property (for OPC Data Access) of an item. UNICORN OPC defines ***UNICORN_OPC_SECURITYACCESS*** and ***UNICORN_OPC_CONTROLACCESS*** to indicate if the item is accessible or not.

Custom error code	is returned if...
UNICORN_OPC_E_ACCESSDENIED (0xC0048002)	...the client tries to read/write to an item to which, the user for some reason currently does not have access. If security is enabled, the error code is returned if no user is logged in or if access to the user set-up of UNICORN is denied.
UNICORN_OPC_E_NOTINCONTROL (0xC0048003)	...the client tries to write to an item and the OPC server is not in control of the system. To gain control, write ***Control (1)*** to the ***AssignState*** item in the ***STATE*** branch. If another UNICORN user is in control of the system, taking control will fail.
ServerVendorEFailedToTakeControl (0xC0048004)	...the client failed to get control of the system.
ServerVendorEQuestionsNotAnswered (0xC0048006)	...a mandatory question is not answered prior to starting a method run.

3.4 Connection Identifiers

An OPC client can connect to the below connection identifiers, respectively, for DA, AE, and HDA servers:

OPC interface	ProgID	CLSID
DA 2.05A	UNICORN_DAServer20.System1.1	{03635E5D-8B72-4A00-99B9-692AC7451F83}
DA 3.0	UNICORN_DAServer30.System1.1	{1A509DE1-6B6C-47AC-B629-B5894F9D80B6}
AE 1.1	UNICORN_AEServer11.System1.1	{0FEB68A8-BC41-4D98-AB5C-8374A2F2C1F3}
HDA 1.2	UNICORN_HDAServer12.1	{E6E8E03C-A36E-4F6F-9BFB-F279D12A0188}

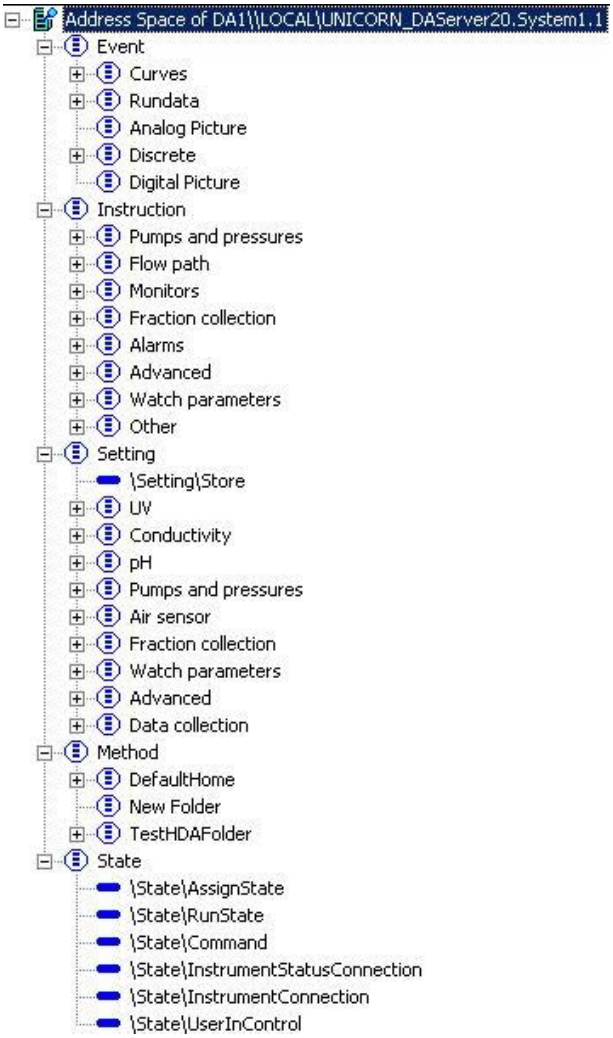
Note: *Other connection identifiers exist but only intended for internal use.*

4 UNICORN OPC Data Access address space

This chapter provides information about UNICORN OPC Data Access address space, method execution, settings, run data, picture data etc.

The OPC Data Access (DA) address space is a real-time representation of process control data for connected instruments.

Data Access address space view



Tip: *The address space depends on the instrument configuration. Different items will appear depending on the instrument configuration used.*

In this chapter

4.1 Run data and Picture data

UNICORN run data are available via the `EVENT\Rundata` branch and, can be analog or digital depending on data type. Picture data are available via `EVENT\ANALOGPICTURE` and `EVENT\DIGITALPICTURE`.

Example location (<=> ***OPC total Item ID***) in server configuration tree:

```
Event\Rundata\Accumulated time
Event\Rundata\System flow
Event\Rundata\BufferPrep
Event\Analog Picture\UV1
Event\Digital Picture\Frac Comp
```

Properties for analog data

Properties for analog data are:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_R4*** or ***VT_I4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
100	EU unit	***VT_BSTR***
102	High EU	***VT_R8***
103	Low EU	***VT_R8***
5001	Decimals	***VT_R8***

Properties for binary data

Properties for binary data are:

PID	Description	Data type
1	Canonical data type	***VT_I2***

PID	Description	Data type
2	Current value	***VT_I4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_EMPTY***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***

4.2 Trend data

UNICORN trend data are available via the `EVENT\Curves`. These data have the highest resolution. ***ANALOGTREND*** data are always an array of values.

BINARYTREND data

BINARYTREND contains textual trending data:

Data	Description
CurrentMethod	Name of the current running method.
CurrentResult	Name of the current result to which the method run is saved.
CurrentScouting	Scouting number.
CurrentBlock	Name of the block currently executing.
RunLog	Messages for the current run.
Fractions	Current fraction mark.
Inject ions	Current injection mark.

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`Event\Curves\UV1`

`Event\Discrete\Injections`

Properties for analog data

Properties for analog data are:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_ARRAY | VT_R4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
100	EU unit	***VT_BSTR***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***

PID	Description	Data type
5001	Decimals	***VT_R8***

Properties for binary data

Properties for binary data are:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_BSTR***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***

4.3 Manual instructions

UNICORN manual instructions from the ***Manual Instructions*** box in ***System Control*** are available in the ***INSTRUCTION*** branch. To execute an instruction, first enter the parameter values and then enter 0 or 1 to execute the item, or 2 to ignore the instruction. When 2 is entered, the instruction reaches UNICORN layer but UNICORN ignores the instruction.

If the instruction in UNICORN uses text, it is possible to get that text from the EU type.

During method runs, manual instruction parameters will be updated with current values, that is, flow changes are reflected in the flow parameter.

In this section

4.3.1 Start methods

It is recommended not to start runs by sending manual commands from OPC.

Note: *It is recommended not to start manual runs via OPC. Start methods instead and then send manual commands only if needed during the method run.*

4.3.2 Execute parameter

Execute parameter is related to instruction information of the instrument configuration. The parameter name is constructed by adding "`_Execute`" to the actual instruction name. Set the scan rate to the fastest update rate of the server (i.e., 500 ms).

The canonical data type is ***VT_I2***. The instruction description is read from instrument configuration.

Instruction execute objects are only ***OPC_WRITABLE***. Enter `0` or `1` to execute the item, or `2` to ignore the instruction. When `2` is entered, the instruction reaches UNICORN layer but UNICORN ignores the instruction.

Note: *The instructions can only be executed if the module is in control of UNICORN system.*

Instruction execution leads to run state ***MANUAL (5)***, unless already in ***RUN*** state. It is possible to change run state to ***END (4)*** by using the state control object.

Quality is good on successful execution of the instruction, which means that the corresponding manual command is successfully submitted.

Implemented properties

Implemented execute parameter properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_EMPTY***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
101	Item description	***VT_BSTR***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***

Example location (***<=> OPC total Item ID***) in server configuration tree:

```
\Instruction\Pumps and pressures\System flow\System
flow_Execute
```

4.3.3 Analog parameter

Unit, parameter description, minimum and maximum values are read from instrument configuration. The canonical data type is either ***VT_I4*** or ***VT_R4***. If instrument configuration position names are defined, the EU type becomes ***OPC_ENUMERATED***, otherwise ***OPC_ANALOG***. Set the scan rate to the fastest update rate of the server (i.e., 500 ms).

Analog parameter objects are both ***OPC_READABLE*** and ***OPC_WRITABLE***, but only from/to ***CACHE***. When executing an instruction (described above), the ***CACHE*** value is included in the manual command structure.

Note: *The instructions can only be executed if the module is in control of UNICORN system.*

Implemented properties

Implemented analog parameter properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_R4*** or ***VT_I4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
100	EU unit	***VT_BSTR***
101	Item description	***VT_BSTR***
102	High EU	***VT_R8***
103	Low EU	***VT_R8***
5001	Decimals	***VT_R8***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

```
\Instruction\Pumps and pressures\System flow\Flow rate
```

4.3.4 Digital parameter

The canonical data type is always ***VT_I2***. Parameter description, open and close labels are read from instrument configuration. If instrument configuration position names are defined, the EU type becomes ***OPC_ENUMERATED***, otherwise ***OPC_NOENUM***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

Digital parameter objects are both ***OPC_READABLE*** and ***OPC_WRITABLE***, but only from/to ***CACHE***. When executing an instruction (described above), the ***CACHE*** value is included in the manual command structure.

Note: *The instructions can only be executed if the module is in control of UNICORN system.*

Implemented properties

Implemented digital parameter properties are: as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_I2***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
100	EU unit	***VT_BSTR***
101	Item description	***VT_BSTR***
106	Maximum value	***VT_R8***
107	Minimum value	***VT_R8***
5001	Decimals	***VT_R8***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

```
\Instruction\Flow path\Inlet A\Position
```

4.3.5 String parameter

The canonical data type is always ***VT_BSTR***. The EU type is always ***OPC_NOENUM***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

String parameter objects are both ***OPC_READABLE*** and ***OPC_WRITABLE***, but only from/to ***CACHE***. When executing an instruction (described above), the ***CACHE*** value is included in the manual command structure. The actual parameter is converted to an integer value before the instruction is executed.

Note: *The instructions can only be executed if the module is in control of UNICORN system.*

Implemented properties

Implemented string parameter properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_BSTR***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_EMPTY***
101	Item description	***VT_BSTR***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\Instruction\Other\Set mark\Mark text`

4.3.6 Virtual output instructions feature

This feature may exist depending on the UNICORN instrument configuration. If the UNICORN instrument configuration contains virtual output instructions then they will be visible below `INSTRUCTION\VirtualOutput`.

The following differs when comparing to other instructions:

- Virtual output instructions can be executed by an OPC client even though the client is not in control of the system.
- Virtual output instructions are not visible in the UNICORN run log.

4.4 Method execution

A UNICORN method can be run via the ***METHOD*** branch.

The ***METHOD*** branch is dynamic. Whenever a new method is created or an existing method is modified, the ***METHOD*** branch has to be browsed again to see the changes and has to reconfigure the items to use. If a method is deleted, it is removed from the address space. Depending on the general server settings, different users are shown in the ***METHOD*** branch.

Each method file contains several items.

In this section

4.4.1 Scout Run Index

Scout Run Index is used to include any scout runs. When a client writes `0`, all scout runs will be included in the method run unless the excluded flag is set for any particular run. Entering any particular scout run number will include only that run in the method run. The canonical data type is ***VT_I2*** and the EU type is always ***OPC_NOENUM***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

The object is always ***OPC_READABLE*** and ***OPC_WRITEABLE***. The object is writable only when in control of the system. Executing the method stores the text in the result

Implemented properties

Implemented scout run index properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_I2***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***

Location (<=> ***OPC total Item ID***) in server configuration tree:

`\Method\DefaultHome\OPCSystem1(Manual)\TestMethod1\Scout Run Index`

4.4.2 Run Method

Available methods are read from the share directory of the current user, OPC user, or all users, depending on setup. The canonical data type is ***VT_I2*** and the EU type is always ***OPC_ENUMERATED***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

Defined object values are as follows:

Value	Description
0	Execute
1	Execute
2	NoExecute ***Note:*** *This is to ignore the instruction.*

The object is ***OPC_READABLE*** and ***OPC_WRITEABLE***.

When a client writes ***END (3)*** to the state command, the current run state is checked to see if the result to be saved or not. If a method is running, the run state is ***RUN (0)***, ***PAUSE (1)*** or ***HOLD (2)*** and the result is saved. Other run states imply that it is a manual run being executed and the result is saved in the `[SystemName](Manual)` folder.

If a method is started when another method is already running, then the method will be placed the queue.

Quality is good when a method command is successfully issued. Writing the method run command stores the start notes, batch number, variables and questions before the method executes. The method will not start if a result name is not given.

Implemented properties

Implemented properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_EMPTY***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***

PID	Description	Data type
8	Value texts	***VT_ARRAY→VT_BSTR***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***

Location (<=> ***OPC total Item ID***) in server configuration tree:

`\Method\DefaultHome\OPCSystem1(Manual)\TestMethod1\Run Method`

4.4.3 Result Name

The canonical data type is ***VT_BSTR*** and the EU type is always ***OPC_NOENUM***. Set the scan rate to the fastest update rate of the server (i.e., 500 ms). Result name objects are both ***OPC_READABLE*** and ***OPC_WRITABLE***, but only readable from ***CACHE***. If a result name is set up in OPC, then UNICORN will not generate a result name. If not, then UNICORN will generate a unique result name when the method starts.

Note: *The methods can only be run if the module is in control of UNICORN system.*

Implemented properties

Implemented result name properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_BSTR***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\Method\DefaultHome\OPCSystem1(Manual)\TestMethod1\Result Name`

4.4.4 Batch ID

Batch ID is used to set the batch number before running a method. The canonical data type is ***VT_BSTR*** and the EU type is always ***OPC_NOENUM***.

Set the scan rate to the fastest update rate of the server (which means 500 ms).

Batch ID objects are both ***OPC_READABLE*** and ***OPC_WRITABLE***, but only readable from ***CACHE***. Clients write the batch number to the object. If a batch ID is set up in OPC, then UNICORN will not generate a batch ID. If a batch ID is not setup in OPC, then UNICORN will generate a unique batch ID when the method starts.

Note: *The methods can only be run if the module is in control of UNICORN system.*

Implemented properties

Implemented batch ID properties are: as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_BSTR***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\Method\DefaultHome\OPCSystem1(Manual)\TestMethod1\Batch ID`

4.4.5 StartNotes

StartNotes is used to read and write the start notes. The canonical data type is ***VT_BSTR*** and the EU type is always ***OPC_NOENUM***.

Set the scan rate to the fastest update rate of the server (which means 500 ms).

StartNotes objects are ***OPC_READABLE*** and ***OPC_WRITEABLE***. The object is writable only when in control of the system. Executing the method stores the text in the result.

Implemented properties

Implemented start notes properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_BSTR***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\Method\DefaultHome\OPCSystem1(Manual)\TestMethod1\StartNotes`

4.4.6 MethodNotes

MethodNotes is used to read the method notes. The canonical data type is ***VT_BSTR*** and the EU type is always ***OPC_NOENUM***.

Set the scan rate to the fastest update rate of the server (which means 500 ms).

MethodNotes objects are ***OPC_READABLE***.

Implemented properties

Implemented method notes properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_BSTR***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\Method\DefaultHome\OPCSystem1(Manual)\TestMethod1\Method Notes`

4.4.7 PreCompile

The canonical data type is always ***VT_BSTR*** (text string value). The EU type is always ***OPC_NOENUM***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

This forces UNICORN OPC to compile the method and check if it is runnable on this system. If it is, the result will be an xml string containing "`Succeed`" as the status. If not, the string will contain an error text.

Implemented properties

Implemented method compile properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	current value	***VT_BSTR***
3	Quality	***VT_I2***
4	Timestamp	***ST_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

```
\Method\DefaultHome\OPCSystem1(Manual)
\TestMethod1\PreCompile Method
```

4.4.8 Questions

The ***QUESTIONS*** branch is used to read and write the start protocol questions. One object is created for each question in a start protocol. Objects are added to the ***QUESTIONS*** branch for every method. The canonical data type is either ***VT_BSTR*** or ***VT_UI4***, depending on the question type.

Table 4.1: Question types and data types

Data type	Question type
NoReply	***VT_BSTR***, ***OPC_NOENUM***
Entry field	***VT_BSTR***, ***OPC_NOENUM***
Multiple choice	***VT_UI4***, ***OPC_ENUMERATED*** as ***VT_ARRAY | VT_BSTR***
Integer	***VT_BSTR***, ***OPC_NOENUM***
Float	***VT_BSTR***, ***OPC_NOENUM***

Integer and ***Float*** can have minimum and maximum ranges. Writing out of range is not allowed. Empty strings are valid unless the mandatory flag is set. Authorized questions are allowed without supplying any signature. Also, it is possible to run methods without answering all the questions unless the mandatory flag is set.

Set the scan rate to the fastest update rate of the server (which means 500 ms).

Question objects are ***OPC_READABLE*** and ***OPC_WRITEABLE***. The object is writable only when in control of the system. Executing the method stores the answers in the result.

Implemented properties

Implemented properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_BSTR*** or ***VT_UI4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***

PID	Description	Data type
102	Maximum value	***VT_R8*** (if ***Float***, ***Integer*** or ***Multiple***)
103	Minimum value	***VT_R8*** (if ***Float***, ***Integer*** or ***Multiple***)

Example location (<=> ***OPC total Item ID***) in server configuration tree:

```
\Method\DefaultHome\OPCSystem1(Manual)
\TestMethod1\Question(s)\Question1{Mandatory}
```

4.4.9 Variables

The ***VARIABLES*** branch is used to read and write the method variables. One object is created for each method variable. Objects are added to the ***VARIABLES*** branch for every method if the ***StartProtocol*** for the method supports the variable value change. The variables on the following are not supported:

- Scouting
- Instruction breakpoint
- Frac-950 Fractionation instruction parameters
- Frac-950 Peak Fractionation parameters

The canonical data type depends on the following variable types:

Table 4.2: Data types and variable types

Data type	Variable type
Float	***VT_R4***, ***OPC_NOENUM***
Integer	***VT_I4***, ***OPC_NOENUM***
String	***VT_BSTR***, ***OPC_NOENUMR***
Multiple choice	***VT_I4***, ***OPC_ENUMERATED*** as ***VT_ARRAY | VT_BSTR***

Integer and ***Float*** have minimum and maximum ranges. Writing out of range is not allowed.

Set the scan rate to the fastest update rate of the server (which means 500 ms).

Variables objects are ***OPC_READABLE*** and ***OPC_WRITEABLE***. The object is writable only when in control of the system. Executing the method uses the variable settings during method run.

Implemented properties

Implemented start protocol variables general properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_R4***, ***VT_I4***, ***BSTR*** or ***VT_UI4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***

Properties of start protocol variables specific variables

Float, ***Integer*** or ***Multiple choice*** also implements the following properties:

PID	Description	Data type
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
100	Unit	***VT_BSTR***
101	Description	***VT_BSTR***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***
5000	Default value	***VT_R4***, ***VT_I4***, ***VT_BSTR*** or ***VT_UI4***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

```
\Method\DefaultHome\OPCSystem1(Manual)\TestMethod1\Method
Variable(s)\Inlet B
```

4.5 State

The ***STATE*** branch contains important state information for the system.

In this section

4.5.1 AssignState

AssignState enables the client to change assign mode to UNICORN system. Default assign mode is ***View***, a mode where the client can monitor ***EVENT*** and ***STATE*** branch items. By changing mode to ***Control***, it is also possible to execute instructions, run methods and save system settings. The canonical data type is ***VT_I4*** and the EU type is always ***OPC_ENUMERATED***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

Object values

Defined assign state values are as follows:

Value	State
0	***View***
1	***Control***

Quality is good if a correct ***AssignState*** is written to the object, otherwise bad.

Implemented properties

Implemented assign state properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_I4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\STATE\AssignState`

4.5.2 RunState

Object values

The canonical data type is ***VT_I2*** and the EU type is always ***VT_I4***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

Defined object values (meaning the state) are as follows:

- ***Ready***
- ***Method Run***
- ***System Pause***
- ***Hold***
- ***Pause Hold***
- ***Hold Pause***
- ***Manual Pause***
- ***Wash***
- ***Alarms and errors***
- ***Initializing system***
- ***Resetting***
- ***Starting method run***
- ***Starting manual run***

Quality is good on successful request of run state data or on valid update from online data, otherwise bad.

Implemented properties

Implemented run state properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_I4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_EMPTY***
102	Maximum value	***VT_R8***

PID	Description	Data type
103	Minimum value	***VT_R8***

Example location (<=> ***OPC total Item ID***) in server configuration tree:
`\STATE\RunState`

4.5.3 Command

Object values

The canonical data type is ***VT_I2*** and the EU type is always ***OPC_ENUMERATED***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

Defined object values are as follows:

Value	State
0	***Pause***
1	***Hold***
2	***Continue***
3	***End***
4	***Next breakpoint***

Method command objects are ***OPC_WRITABLE***. Writing a valid value causes the utility object to submit a manual command.

Note: *Methods can only be run if the module is in control of UNICORN system.*

When a client writes ***END (3)***, the current run state is checked to see if the result should be saved or not. If a method is running, the run state is ***RUN (1)***, ***PAUSE (4)*** or ***HOLD (5)*** and the result is saved. Other run state values imply a manual instruction being executed and the result is created in the `[SystemName] (Manual)` folder.

Implemented properties

Implemented general command properties are:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_I2***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
102	Maximum value	***VT_R8***

PID	Description	Data type
103	Minimum value	***VT_R8***

Example location (<=> ***OPC total Item ID***) in server configuration tree:
`\STATE\Command`

4.5.4 InstrumentConnection

Object values

InstrumentConnection handles the instrument connection state. The canonical data type is ***VT_I4***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

Defined object values are as follows:

Value	State
0	***Ready***
1	***Scanning***
2	***Unknown***

InstrumentConnection objects are ***OPC_READABLE***.

Note: *The instrument connection must be valid (connected) for the other items to work as expected.*

Implemented properties

Implemented instrument connection properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_I4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\STATE\InstrumentConnection`

4.5.5 InstrumentStatusConnection

Object values

InstrumentStatusConnection handles the instrument status state. The canonical data type is ***VT_I2***. Set the scan rate to the fastest update rate of the server (which means 500 ms).

Defined object values are as follows:

Value	Connection
0	Yes
1	Partially
2	No

InstrumentStatus objects are ***OPC_READABLE.***.

Note: *The instrument status must be valid (ready) if the other items are to work as expected.*

Implemented properties

Implemented instrument status properties are as follows:

PID	Description	Data type
1	Canonical data type	***VT_I2***
2	Current value	***VT_I4***
3	Quality	***VT_I2***
4	Timestamp	***VT_DATE***
5	Access rights	***VT_I4***
6	Server scan rate	***VT_R4***
7	EU type	***VT_I4***
8	Value texts	***VT_ARRAY→VT_BSTR***
102	Maximum value	***VT_R8***
103	Minimum value	***VT_R8***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\STATE\InstrumentStatusConnection`

4.5.6 UserInControl

Object values

UserInControl handles the instrument control state. The canonical data type is ***VT_I2***. Defined object value is the user name of the current user.

UserInControl objects are ***OPC_READABLE***.

Implemented properties

Implemented instrument status properties are as follows:

PID	Description	Data type
1	Data Type	***VT_I2***
2	Value	***VT_BSTR***
3	Quality	***VT_I2***
4	Time stamp	***VT_DATE***
5	Access right	***VT_I4***
6	Scan rate	***VT_R4***

Example location (<=> ***OPC total Item ID***) in server configuration tree:

`\STATE\UserInControl`

4.6 Recommendations

It is recommended to write to items synchronously. The server handles both synchronous and asynchronous writes but when making asynchronous writes it is up to the client to wait until the server is ready with previous asynchronous write.

All items whose names ends with _***EXECUTE*** are used as "batch" commands to execute methods and instructions with parameters and system settings. Before writing to the _***EXECUTE*** item, the items, which are included in the "batch" (for example, instruction parameters) have to be written. A recommendation is to add a delay before writing to the _***EXECUTE*** items since when using a mirroring client between two OPC servers it has been seen that item writes might arrive to the UNICORN OPC server in an incorrect order.

Note: *UNICORN system settings are available via the* ***SETTINGS*** *branch, which works like the* ***INSTRUCTION*** *branch with one exception. There is only one execute instruction,* `\Setting\Store`*, to store all system settings at once.*

Note: *System settings can only be saved when the system is in* ***Ready*** *state. If UNICORN changes system settings, it updates the settings parameters in OPC.*

5 UNICORN OPC Alarms & Events address space

This chapter describes UNICORN OPC Alarms & Events address space, various types of alarms, run log, error messages and event categories.

The Alarms & Events address space shows the internal alarm functionality of the instrument configuration. Each item in the address space might trigger an alarm or event.

Each component's failure/loss triggers an alarm and the AE client connected in control/view of the system is notified. If the connection to the instrument is lost, then AE client is notified through the restart error alarm. Once the restart error occurs and when the connection issue has been fixed thereafter, then either the UNICORN Instrument Server computer has to be restarted or the restart error has to be acknowledged in UNICORN ***System Control***.

In order to acknowledge alarms or warnings from the OPC client, the client (UNICORN OPC server) has to be in control of the system. To take control, use the ***Data Access*** server in parallel and take control over the system via the ***AssignState*** item. It is also possible to use the ***TakeControl*** option.

The UNICORN OPC AE server does not support the inactive/acknowledged/enabled state. UNICORN OPC AE server will not be able to notify that the alarm or warning becomes inactive. It only notifies if the signal reaches an alarm or warning state, but it will not notify the opposite when reaching the normal state.

Alarms & Events address space view

Tip: *The address space depends on the instrument configuration. Different items will appear depending on which instrument configuration is used.*

In this chapter

5.1 Alarms and errors

Analog alarm

There are always four conditions for analog alarm objects (sources). Corresponding condition names are as follows:

1. ***LO_ALARM***
2. ***LO_WARNING***
3. ***HI_ALARM***
4. ***HI_WARNING***

Both ***Condition Events*** and ***Tracking Events*** can be generated. ***Condition Events*** are generated when a condition becomes active or inactive and when a condition is acknowledged. ***Tracking Events*** are generated when a condition is disabled but still active or inactive and unacknowledged. The event categories are ***Level*** for ***Condition Events*** and ***Enable/Disable*** for ***Tracking Events***. No attributes are supported for analog alarms.

No attributes are supported for analog alarms.

Example location (<=> ***OPC qualified source name***) in server area space:

```
Analog alarms.Cond2
```

Digital alarm

There is only one condition for each digital alarm object (source). The condition names for the two different types are:

1. ***WARNING***
2. ***ALARM***

Both ***Condition Events*** and ***Tracking Events*** can be generated. ***Condition Events*** are generated when a condition becomes active or inactive and when a condition is acknowledged. ***Tracking Events*** are generated when a condition is disabled but still active or inactive and unacknowledged. Event categories are ***DiscreteW*** or ***DiscreteA*** for ***Condition Events*** and ***Enable/Disable*** for ***Tracking Events***. No attributes are supported for digital alarms.

Example location (<=> ***OPC qualified source name***) in server area space:

```
Digital alarms.FlowWarning
```

Errors

This is the error dialog box in UNICORN ***System Control***. The event category of the simple error event is ***Run Log***. The event category of the condition error event is ***Errors***.

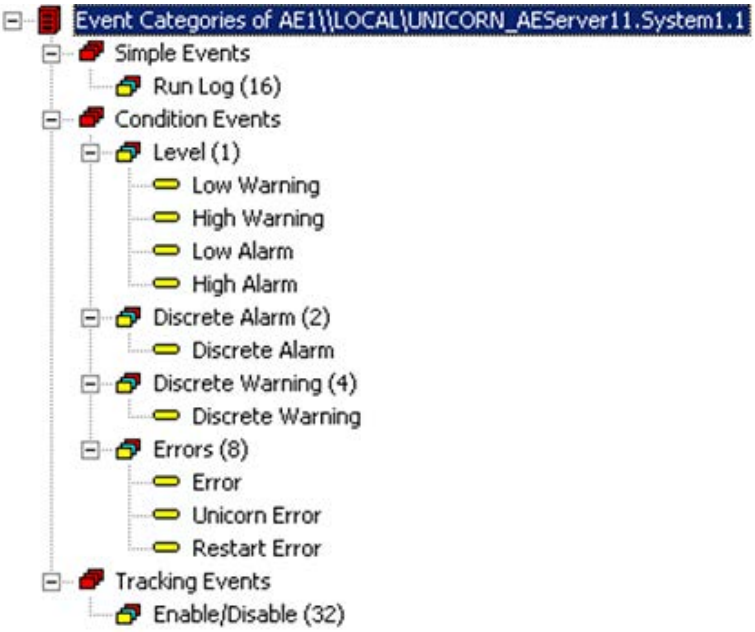
No attributes are defined for error events and no acknowledgements are required for this event type.

Location (<=> ***OPC qualified source name***) in server area space:

```
Other.Errors
```

5.2 Event categories

Alarms & Events address space view



Event categories for UNICORN OPC server

Event categories for UNICORN OPC server are defined as follows:

UNICORN Alarm & Event	Event types	Event categories
Analog Alarm	***OPC_CONDITION_EVENT*** ***OPC_TRACKING_EVENT***	***Level*** (ID=1) ***Enable/Disable*** (ID=32)
Digital Alarm	***OPC_CONDITION_EVENT*** ***OPC_TRACKING_EVENT***	***Discrete Alarm*** (ID=2) or ***Discrete Warning*** (ID=4)
Error	***OPC_SIMPLE_EVENT*** ***OPC_CONDITION_EVENT***	***Run Log*** (ID=16) ***Errors*** (ID=8)

6 UNICORN OPC Historical Data Access address space

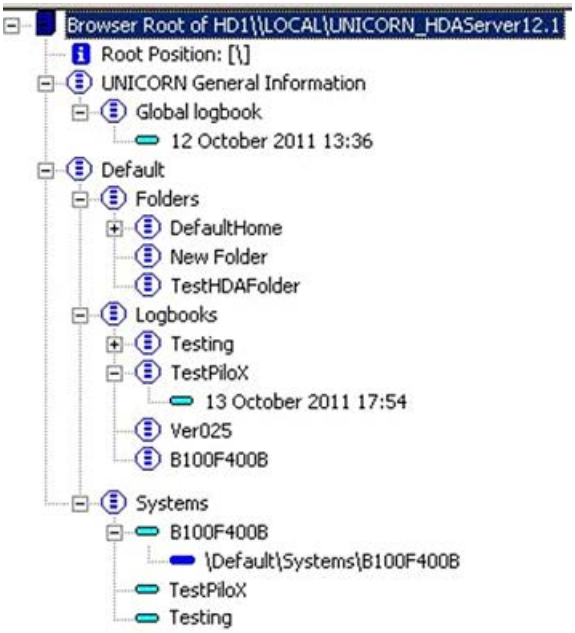
This chapter provides information about UNICORN OPC Historical Data Access (HDA) address space, audit trail, HDA XML format definition etc.

The HDA address space is dynamic. A result file can be browsed whenever a new result is created. If a result is deleted, it is removed from the address space.

Tip: *The address space depends on the result files that have been created based on a specific instrument configuration. Different items will appear in the result file depending on the instrument configuration used when the result file was created.*

In this chapter

HDA address space view

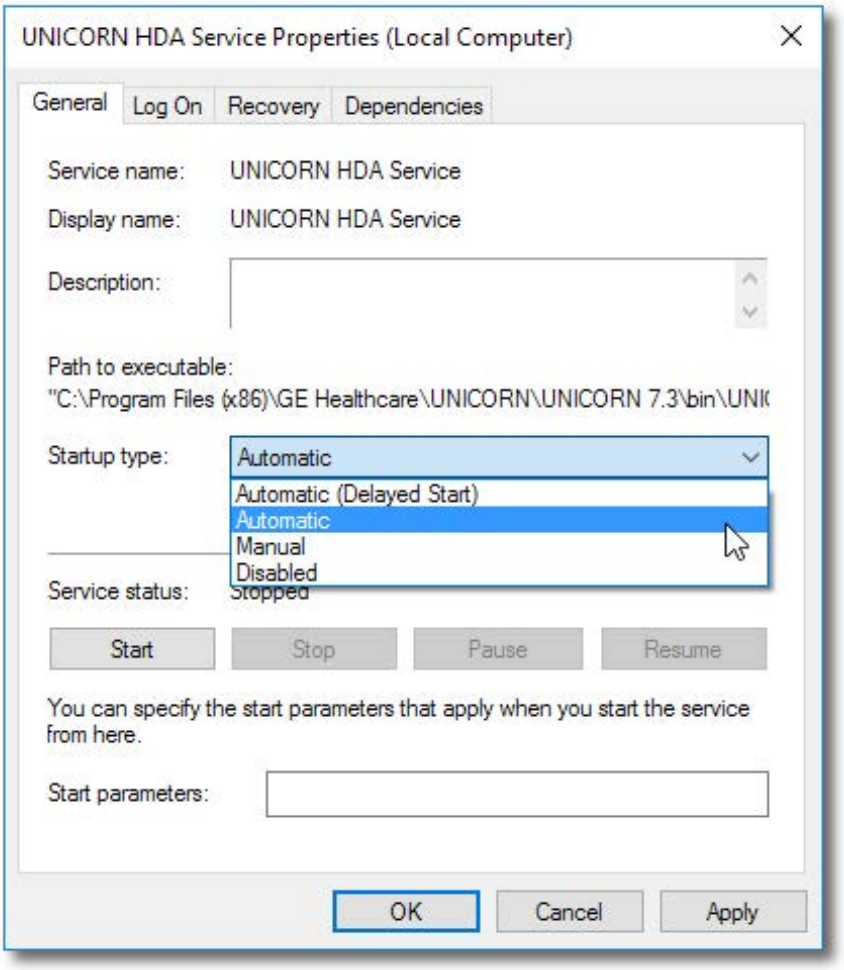


Tip: *The address space depends on the result files that has been created based on a specific instrument configuration. Different items will appear in the result file depending on which instrument configuration that was used when the result file was created.*

Configuration and settings

UNICORN HDA Service supports manual activation by default. The UNICORN HDA Service settings have to be altered to get it to start automatically. The setting can be changed through the Services-utility (`Run: services.msc` to start it).

To start automatically, change ***Startup Type*** through ***UNICORN HDA Service Properties*** window.



The HDA address space is dynamic. A result file can be browsed whenever a new result is created. If a result is deleted, it is removed from the address space. The UNICORN HDA server only supports one OPC client connection.

6.1 AuditTrail

AuditTrail from UNICORN are available under the HDA server address space. The global logbook is found under the ***UNICORN GENERAL INFORMATION*** branch and the system specific logs are found under the ***Logbooks*** folder in the user (for example ***Default***) folder.

Attributes defined by *AuditTrail*

Each ***AuditTrail*** leaf (item) has the following attributes defined:

Name	Description	Data type
Datatype	Item data type	***VT_UI4***
ItemId	Item ID	***VT_BSTR***
StartLowFileTime	Start time low ***FILETIME***	***VT_UI4***
StartHighFileTime	Start time high ***FILETIME***	***VT_UI4***
EndLowFileTime	End time low ***FILETIME***	***VT_UI4***
EndHighFileTime	End time high ***FILETIME***	***VT_UI4***

Start and end time are equal for a leaf. Time is set to the creation time of the audit trail file, meaning the renew time.

6.2 Result file information

Each result file contains ***Curves***, ***Documentation*** and ***Peak tables***.

Information about the system is available under the ***Systems*** folder in the user folder.

In this section

6.2.1 Curves

All curves in a result file are listed as items directly under the result file branch in the address space.

Attributes supported by *Curves*

Each curve has a number of supported attributes that allow the kind of curve, the item represents to be identified:

Name	Description	Data type
Datatype	Item data type	***VT_UI4***
EngUnits	Unit	***VT_BSTR***
NormalMaximum	Maximum value	***VT_R8***
NormalMinimum	Minimum value	***VT_R8***
Itemid	Item ID	***VT_BSTR***
CurveType	Curve type	***VT_I2***
StartLowFileTime	Start time low ***FILETIME***	***VT_UI4***
StartHighFileTime	Start time high ***FILETIME***	***VT_UI4***
EndLowFileTime	End time low ***FILETIME***	***VT_UI4***
EndHighFileTime	End time high ***FILETIME***	***VT_UI4***
ZeroLowFileTime	Zero tlime low ***FILETIME***	***VT_UI4***
ZeroHighFileTime	Zero time high ***FILETIME***	***VT_UI4***
TimeInt	Sample time interval in sec	***VT_R8***
FirstInjectionLowFileTime	First injection time low ***FILETIME***	***VT_UI4***
FirstInjectionhighFileTime	First injection time high ***FILETIME***	***VT_UI4***
ChromStartLowFileTime	Chrom start time low ***FILETIME***	***VT_UI4***
ChromStartHighFileTime	Chrom start time high ***FILETIME***	***VT_UI4***
UnicornRawData	UNICORN raw curve data	***VT_BSTR***

Types of *Curves*

There are several types of curves (***HDA_CURVE_TYPE***):

Curve type	Data type
HDA_CURVETYPE_GENERIC_RAW	Array of ***VT_R8***
HDA_CURVETYPE_GENERIC_EVALUATED	Array of ***VT_R8***
HDA_CURVETYPE_FRACMARKS_RAW	Array of ***VT_BSTR***
HDA_CURVETYPE_FRACMARKS_EVALUATED	Array of ***VT_BSTR***
HDA_CURVETYPE_INJECTIONMARKS_RAW	Array of ***VT_BSTR***
HDA_CURVETYPE_INJECTIONMARKS_EVALUATED	Array of ***VT_BSTR***
HDA_CURVETYPE_SETMARKS_RAW	Array of ***VT_BSTR***
HDA_CURVETYPE_SETMARKS_EVALUATED	Array of ***VT_BSTR***

Curve types bit mask

In addition to curve type, a bit mask is added to the type:

Curve type bit mask	Description
HDA_CURVETYPE_TIMEDEFINED	Curve data exist as time axis
HDA_CURVETYPE_VOLUMEDEFINED	Curve data exist as volume axis
HDA_CURVETYPE_MARKERDEFINED	Curve data exists as text (both time and volume axes)

The first injection time and chrom start time might not always be defined for a curve. The start time is not the absolute time when the curve was created. Instead, the result file creation time is used as a base. The real absolute time for a curve is read from chrom start time. The difference is usually around 10 to 15 seconds.

Note: *chrom time is only defined on RAW curve types.*

Zero time defines the time of zero. The curve can start at any time before and after zero time. It can even end before zero time. Applying zero time to a curve allows zero to be located on the x-axis.

The result file might contain curves only stored in volume base. These curves cannot be read and will fail ***ReadRaw/ReadProcessed***, since the HDA specification only allows time as x-axis. However, the volume curves are available via the ***HDA_UNICORN_RAW_DATA*** attribute. This attribute returns all curve data (both time and volume) as an XML-formatted string.

The low and high attributes are mapped to the ***dwLowDateTime*** and ***dwHighDateTime*** members of the ***FILETIME*** structure to enable ***FILETIME*** accuracy on the client side.

6.2.2 Documentation

Evaluation in UNICORN has a documentation dialog for each result file. The ***DOCUMENTATION*** branch for each result file in HDA contains the same data as ***Evaluation*** in documentation.

Items in *Documentation*

The following items are available:

Item	Data type
BatchNumber	***String***
BufferPro	***XML string***
Calibration	***XML string***
Columns	***XML string***
Evaluation logbook	***XML string***
Evaluation procedures	***XML string***
FracXY	***XML string***
Run logbook	***XML string***
Method name	***String***
Method creator	***String***
Method creation date	***DateTime***
Method created for system	***String***
Method last modifier	***String***
Method last modification date	***DateTime***
Method signatures	***XML string***
Method notes	***XML string***
Start notes	***XML string***
Run notes	***XML string***
Evaluation notes	***XML string***
Method strategy notes	***String***
Result name	***String***
Result creator	***String***

Item	Data type
Result creation date	***DateTime***
Result run system	***String***
Batch number	***String***
Result signatures	***XML string***
Result strategy used components	***XML string***
Scouting	***XML string***
System Settings	***XML string***
Snapshot	***XML string***
Text Instructions	***XML string***
Variables	***XML string***
Method Instrument Configuration	***XML string***
Result Instrument Configuration	***XML string***
Questions	***XML string***

Attributes in *Documentation*

Each documentation item defines the following attributes:

Attribute	Description	Data type
DataType	Item data type	***VT_UI4***
Itemid	Item ID	***VT_BSTR***
StartLowFileTime	Start time low ***FILETIME***	***VT_UI4***
StartHighFileTime	Start time high ***FILETIME***	***VT_UI4***
EndLowFileTime	End time low ***FILETIME***	***VT_UI4***
EndHighFileTime	End time high ***FILETIME***	***VT_UI4***

Start and end times are equal in the ***DOCUMENTATION*** branch. To get data from ***ReadRaw***, this time stamp must be included in the time range, otherwise no data will be returned.

6.2.3 Peak tables

The ***PEAK TABLES*** branch contains all peak tables stored in the result file. When reading from the item, the peak table is returned as an XML-formatted string.

Items in *Peak Tables*

Each item defines the following attributes:

Attribute	Description	Data type
Datatype	Item data type	***VT_UI4***
Itemid	Item ID	***VT_BSTR***
StartLowFileTime	Start time low ***FILETIME***	***VT_UI4***
StartHighFileTime	Start time high ***FILETIME***	***VT_UI4***
EndLowFileTime	End time low ***FILETIME***	***VT_UI4***
EndHighFileTime	End time high ***FILETIME***	***VT_UI4***

6.2.4 Systems

Information about the system is available under the ***Systems*** folder in the user folder.

Items in *Systems*

Each system leaf defines the following attributes:

Attribute	Description	Data type
Datatype	Item data type	***VT_UI4***
StartLowFileTime	Start time low ***FILETIME***	***VT_UI4***
StartHighFileTime	Start time high ***FILETIME***	***VT_UI4***
EndLowFileTime	End time low ***FILETIME***	***VT_UI4***
EndHighFileTime	End time high ***FILETIME***	***VT_UI4***
ItemId	Item ID	***VT_BSTR***

6.3 UNICORN OPC HDA XML format definition

The XML format is defined for each result file item. This section describes the different XML structures.

In this section

6.3.1 BufferPro, v 1.0

This XML structure is exported by reading the ***BufferPro*** leaf.

Note: *For migrated UNICORN 5.x results, the* ***BufferPro*** *OPC item contains* ***BufferPrep*** *data. Both the S function and the S function share the same storage format.*

BufferPro XML structure

Tag name	Description
BufferPro	Root item
RecipeName	Recipe name
pHRange	pH range, separated with "**-**"
B100	100% buffer B
AcidOrBase	Acid or base name
Salt	Salt name and salt stock concentration
Notes	Recipe name
Buffer	Specifications of the different buffers
Name	Buffer name
pKa1	pK_a1
pKa2	pK_a2
pKa3	pK_a3
dpKa1dt	dpk_a1 delta
dpKa2dt	dpk_a2 delta
dpKa3dt	dpk_a3 delta
AcidicProtons	Number of acidic protons
ChargeDeprotonatedIon	Number of charge deprotonated ions
SaltName	Salt name
ChargeAnion	Number of charged anions
ChargeCation	Number of charged cations

6.3.2 Calibration, v 1.0

This XML structure is exported by reading the ***Calibration*** leaf.

Calibration XML structure

Tag name	Description
CalibrationList	Root item
Calibration	A calibration
Tagname	Name of item that has been calibrated
Date	Date
Username	User name
Point1	Point 1
Point2	Point 2
Constant	Constant
Offset	Offset
Remote	Calibration local or remote. (YES/NO)
P	Proportional control component
I	Integral control component
D	Derivative control component
ReferenceValue1	Reference value 1
ReferenceValue2	Reference value 2
CalibrationConstantDescription1	Calibration constant description 1
CalibrationConstantValue1	Calibration constant value 1
CalibrationConstantDescription2	Calibration constant description 2
CalibrationConstantValue2	Calibration constant value 2
CalibrationConstantDescription3	Calibration constant description 3
CalibrationConstantValue3	Calibration constant value 3

6.3.3 EvaluationLog, v 1.0

This XML structure is exported by reading the ***Evaluation logbook*** leaf.

EvaluationLog XML structure

Tag name	Description
EvaluationLog	Root item.
Item	Item containing evaluation logbook event.

6.3.4 Run logbook, v 1.1

This XML structure is exported by reading the ***Run logbook*** leaf.

Logbook XML structure

Tag name	Description
Logbook	Root item
Item	A logbook item
AccumulatedTime	Accumulated time in seconds
AccumulatedVolume	Accumulated volume, including unit
Event	Logbook event
EventType	Event type
EventSubType	Subtype of event

6.3.5 SignatureList, v 1.0

This XML structure is exported by reading the ***Method Signatures*** and ***Result Signatures*** leaves.

SignatureList XML structure

Tag name	Description
SignatureList	Root item
Signature	A signature
Username	Name of user
Fullname	Full name of user
Position	Position of user
Date	Date, formatted in server local time
Meaning	Signature meaning
Locked	If present, the file is locked against further change

6.3.6 SnapshotList, v 1.0

This XML structure is exported by reading the ***Snapshot*** leaf.

Note: *To get the actual y-axis value, use the time or volume value as a reference into the curves.*

SnapshotList XML structure

Tag name	Description
SnapshotList	Root item
Chromatogram	A signature
Name	Name of chromatogram
Snapshot	Run number
TimeRetention	Time retention
VolumeRetention	Volume retention
Curve	Curve
Name	Curve name
Unit	Curve unit
Time Value	Time stamp of snapshot
Volume Value	Volume stamp of snapshot

6.3.7 UsedComponents, v 1.0

This XML structure is exported by reading the ***Result Strategy Used Components*** leaf.

UsedComponents XML structure

Tag name	Description
UsedComponents	Root item.
Component	Component name used during run.

6.3.8 Notes, v 1.0

This XML structure is exported by reading the ***Method Notes***, ***Start Notes***, ***Run Notes***, and ***Evaluation Notes*** leaves.

Notes XML structure

Tag name	Description
Notes	Root item, the note follows.

6.3.9 ScoutingList, v 1.1

This XML structure is exported by reading the ***Scouting*** leaf.

Note: *To get the actual y-axis value, use the time or volume value as a reference into the curves.*

ScoutingList XML structure

Tag name	Description
ScoutingList	Root item.
Scouting	A scouting run.
TotalNumberOfScoutings	Total number of scouting items available.
RunScouting	If available, this item indicates the scouting run number of this result file.
Run	Run number.
ThisScoutingWasUsedDuringRun	This leaf is available if this scouting run was used when the method was run. The run number is the same as ***RunScouting***.
Variable	Variables defined in this run.
Block	Block name of this variable.
Name	Name of variable.
Unit	Unit, only included if variable has a unit.
Value	Value of variable.

6.3.10 SettingsList, v 1.0

This XML structure is exported by reading the ***System Settings*** leaf.

SettingsList XML structure

Tag name	Description
SettingsList	Root item.
Group	A group.
GroupName	The name of the group.
Instruction	Instruction names. There is at least one instruction per group.

6.3.11 TextInstructions, v 1.0

This XML structure is exported by reading the ***Text Instructions*** leaf.

TextInstructions XML structure

Tag name	Description
TextMethod	Root item, the complete text method follows. Each row is separated by a carriage return and a new line.

6.3.12 VariableList, v 1.1

This XML structure is exported by reading the ***Variables*** leaf.

Start protocol variables XML structure

Tag name	Description
VariableList	Root item.
Variable	A variable.
Block	Block name of the variable.
Name	The variable name.
VisibleInScouting	Indicates if the variable is visible in scouting.
VisibleInDetails	Indicates if the variable is visible in details only.
Value	Value. Can be a number or a string, depending on the variable.
Unit	Unit of the value. Not included if no unit is present.

6.3.13 QuestionList, v 1.0

This XML structure is exported by reading the ***Questions*** leaf.

Start protocol questions XML structure

Tag name	Description
QuestionList	Root item
Item	A question item
Question	The question
Answer	The answer

6.3.14 PeakTable, v 1.1

This XML structure is exported by reading leaves under the ***PEAK TABLES*** branch.

PeakTable XML structure

Tag name	Description
PeakTable	Root item
Summary	Summary of the peak table
TotalNumberOfDetectedPeaks	Total number of peaks
TotalArea	Total area
Unit	Unit used.
AreaInEvaluatedPeaks	Area in evaluated peaks
RationPeakareaTotalarea	Ratio peak area/total area
TotalPeakDuration	Total peak duration
ColumnHeight	Column height
ColumnV0	Column V_0
ColumnVt	Column V_t
SourceCurve	Source curve name
Baseline	Baseline
Rejection	Rejection
MinHeight	Minimum height
MinWidth	Minimum width
MaxWidth	Maximum width
MinArea	Minimum area
MaxNumberOfPeaks	Maximum number of peaks
Quantitation	Quantitation (molecular size, standard addition, internal standard, external standard, recovery)
Peak	A peak
No	Peak number
Name	Peak name

Tag name	Description
Retention	Retention
Start	Start
End	End
Width	Width
Area	Area
AreaPerTotalArea	Area/total area
AreaPerPeakArea	Area/peak area
Height	Height
WidthAtHalfHeight	Width at half peak height
HeightAtStart	Height at start
HeightAtEnd	Height at end
BaselineAtStart	Baseline at start
BaselineAtMax	Baseline at max
BaselineAtEnd	Baseline at end
ConductivityHeightStart	Conductivity height at start
ConductivityHeightMax	Conductivity height at maximum
ConductivityHeightEnd	Conductivity height at end
FractionTubeAtStart	Fraction tube at start
FractionTubeAtMax	Fraction tube at maximum
FractionTubeAtEnd	Fraction tube at end
PeakStartType	Peak start type (dropline, skim, baseline, skimdrop, unknown)
PeakEndType	Peak end type (dropline, skim, baseline, skimdrop, unknown)
Sigma	Sigma
Resolution	Resolution
CapacityFactor	Capacity factor
Kav	K_{av}
PlateHeight	Plate height

Tag name	Description
PlatesPerMeter	Plates per meter
AsymmetryStart	Asymmetry start
AsymmetryEnd	Asymmetry end
Asymmetry	Asymmetry
AverageConductivity	Average conductivity
ExtinctionCoefficient	Extinction coefficient
ExtCoeffConcentration	Concentration calculated using extinction coefficient.
ExtCoeffAmount	Amount calculated using extinction coefficient.
Concentration	Concentration
Amount	Amount
MolecularSize	Molecular size
Type	Type

6.3.15 Unicorn raw, v 1.0

This XML structure is exported by reading the attribute ***HDA_UNICORN_RAW_DATA*** for a ***Curve*** leaf.

UnicornRawData XML structure

Tag name	Description
UnicornRawData	Root item.
Name	Name of curve.
Type	Curve type. Does not contain the bit mask, which the ***HDA_CURVE_TYPE*** does, only the actual type.
Unit	Curve unit (y-axis).
TimeUnit	Unit of time curve (x-axis).
VolumeUnit	Unit of volume curve (x-axis).
CurveMin	Minimum value of y-axis.
CurveMax	Maximum value of y-axis.
ZeroAdjust	Present if zero adjust is enabled.
ZeroAdjustTime	Zero adjust time.
ZeroAdjustVolume	Zero adjust volume.
StartTimeRetention	Retention start time.
InjectionTimeRetention	Injection start time.
NumberOfTimeDataPoints	Number of curve time data points.
CTD	Curve Time Data. This is repeated for all time data points. The values are assigned as attributes to the tag (x and y attributes).
StartVolumeRetention	Retention start volume.
InjectionVolumeRetention	Injection start volume.
NumberOfVolumeDataPoints	Number of curve volume data points.
ColumnVolumeDefined	Present if column volume is defined.
ColumnVolume	Column volume value for curve.
ColumnVolumeUnit	Unit of ***ColumnVolume*** curve (X-axis).

Tag name	Description
CVD	Curve Volume Data. This is repeated for all volume data points. The values are assigned as attributes to the tag (***x***, ***y*** and, if available, ***cv***).
CMD	Curve Markers Data. This is repeated for all marker (text) data points. The values are assigned as attributes to the tag (***t*** = time, ***v*** = volume, ***t1*** = text1 and, if available, ***t2*** = text2).

6.3.16 Columns, v 1.1

This XML structure is exported by reading the ***Columns*** leaf.

Columns XML structure

Tag name	Description
Columns	Root item.
Column	A column.
Name	Name of column.
Height	Height of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
Diameter	Diameter of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
Volume	Volume of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
VolumeUnit	Height of column. Attribute mandatory defines if item is mandatory.
Technique	Technique used by column. Attribute mandatory defines if item is mandatory.
Vt	Total volume of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
Vo	Empty volume of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
MaxPressure	Maximum pressure of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
DefaultFlowrate	Default flow rate of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
MaxFlowrate	Maximum flow rate of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.

Tag name	Description
TypPeakWidthAtBase	Typical peak width at base. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
pHLongMax	Long-term maximum pH. Attribute mandatory defines if item is mandatory.
pHLongMin	Long-term minimum pH. Attribute mandatory defines if item is mandatory.
pHShortMax	Short-term maximum pH. Attribute mandatory defines if item is mandatory.
pHShortMin	Short-term minimum pH. Attribute mandatory defines if item is mandatory.
AverageParticleDiameter	Average particle diameter. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
Code	Code of column. Attribute mandatory defines if item is mandatory.
TypicalLoadingRange	Typical loading range. Attribute mandatory defines if item is mandatory.
MolWeightRange	Molecular weight range. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.

6.3.17 EvaluationProcedures, v1.0

This XML structure is exported by reading the ***Evaluation procedure*** leaf.

EvaluationProcedures XML structure

Tagname	Description
EvaluationProcedures	Root item.
EvaluationProcedure	An evaluation procedure.
Name	Name of evaluation procedure.
WasRun	Indicates if the evaluation procedure was run during method run.

6.3.18 FracXY, v 1.0

This XML structure is exported by reading the ***FracXY*** leaf.

FracXY XML structure

Tagname	Description
FracXY	Root item
FractionOrder	Fraction order (Serpentine row, Row-by-row, Serpentine column or Column-bycolumn)
Name	Name of rack
Group	Rack group
Name	Name of rack group
LastTube	Last tube of group

6.3.19 AuditTrail, v 1.1

This XML structure is exported by reading a ***Global Logbook*** or ***System Logbook*** leaf.

AuditTrail XML structure

Tag name	Description
AuditTrail	Root item.
Type	Type of audit trail: ***Global***, ***System*** or ***Backup***. ***Backup*** is used when accessing a direct audit trail file address.
CheckSumError	If this item is available, the file has been modified outside UNICORN or is corrupt.
Audit	An audit item.
Time	Time of audit.
TimeZone	Time zone.
Message	Audit message.
Details	Log entry details.

6.3.20 Method/Result Instrument Configuration, v 1.0

This XML structure is exported by reading a ***Method Instrument Configuration*** or ***Result Instrument Configuration*** leaf.

Method/Result Instrument Configuration XML structure

Tag name	Description
InstrumentConfiguration	Root item
Name	Name of instrument configuration
Version	Instrument configuration version
Help text	Instrument configuration help text
StrategyName	The instrument configuration name
StrategyNotes	The instrument configuration notes
PhaseConfiguration	Phase configuration name and version
TemplateMethodConfiguration	Protocol configuration name and version
PerformanceTestConfiguration	Performance test configuration name and version
FlowSchemeConfiguration	Flow scheme configuration name and version
VIDConfiguration	VID configuration name and version
FirmwareConfiguration	Firmware configuration name and version
CompatibleUNICORNClientVersion	UNICORN version
CompatibleUNICORNInstrumentServerVersion	UNICORN version

6.3.21 System

This XML structure is exported by reading a ***System*** leaf.

System XML structure

Tag name	Description
SystemName	Name of system.
SystemType	Type of system.
InstrumentConfiguration	Instrument configuration name and version.
SystemIsNonActive/SystemIsActive	Active system.
ControlledByUNICORNInstrumentServer	Instrument server computer name. ***Note:*** *Displayed only if the system is Active.*
InstrumentSerialNo	Instrument serial number.
ConnectionBasedOnSerial/ ConnectionBasedOnIP	Type of connection based on serial number or instrument IP address.
InstrumentIPAddress	Instrument IP address.

7 Reference Information

In this chapter

7.1 DA server supported OPC interfaces

OPC Server

Interface	1.0	2.05A	3.0	Optional interface
IUnknown	Required	Required	Required	
IOPCServer	Required	Required	Required	
IOPCCommmon	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	Not implemented
IOPCBrowseAddressSpaceBrowse	Optional	Optional	N/A	Not implemented
IOPCItemIO	N/A	N/A	Required	Implemented

OPCGroup

Interface	1.0	2.05A	3.0	Optional interface
IUnknown	Required	Required	Required	
IOPCItemMgt	Required	Required	Required	
IOPCGrouptStateMgt	Required	Required	Required	
IOPCGrouptStateMgt2	N/A	N/A	Required	
IOPCPublicGroupStateMgt	Optional	Optional	N/A	Not implemented
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	Implemented
IConnectionPointContainer	N/A	Required	Required	

Interface	1.0	2.05A	3.0	Optional interface
IOPCAsyncIO	Required	Optional	N/A	
IDataObject	Required	Optional	N/A	

7.2 AE server supported OPC interfaces

OPCEventServer

Interface	AE v1.1	Optional interface
IOPCEventServer	Required	
IOPCEventServer2	Optional	Not implemented
IOPCEventAreaBrowser	Optional	Implemented

OPCEventSubscription Object

Interface	AE v1.1	Optional interface
IUnknown	Required	
IOPCEventSubscriptionMgt	Required	
IConnectionPointContainer	Required	
IOPCEventSubscriptionMgt2	Optional	Not implemented

OPC Common Requirements

Interface	AE v1.1	Optional interface
IOPCCommon	Required	

7.3 HDA server supported OPC interfaces

Custom interface

Interface	HDA v1.20	Optional interface
IOPC_HDAServer	Required	
IOPCHDA_Browser	Required	
IOPCHDA_SyncRead	Required	
IOPCCommon	Required	
IOPCHDA_SyncUpdate	Optional	Not implemented
IOPCHDA_SyncAnnotations	Optional	Not implemented
IOPCHDA_AsyncRead	Optional	Not implemented
IOPCHDA_AsyncUpdate	Optional	Not implemented
IOPCHDA_AsyncAnnotations	Optional	Not implemented
IOPCHDA_Playback	Optional	Not implemented
IConnectionPointContainer	Required if any ***Async*** interface is supported	Not implemented

OPC security requirements

Interface	HDA v1.2	Optional interface
IOPCSecurityPrivate	Required	

7.4 Differences between UNICORN OPC 5.x and UNICORN OPC 7.0

OPC server

Feature	OPC 5.x	OPC 7.0
Number of OPC servers supported per UNICORN Instrument Server computer[1]	4	1
OPC servers (DA & AE)	External	part of OPC service inside the UNICORN Instrument Server
HDA memory cache	Yes	No
HDA file cache	Yes	No

Data access

Feature	OPC 5.x	OPC 7.0
Support for DA 3.0 interface	No	Yes
Ability to handle asynchronous commands from client	Yes	Yes[2]
Method queueing	No	Yes
Zero deadband feature	Yes	No
Quarternary gradient variables	No	Yes
Perform scouting runs from OPC	No	Yes

Historical data access

Feature	OPC 5.x	OPC 7.0
Support for multiple HDA clients	Yes	No
Support browsing through all systems	No	Yes
Display instrument configuration (strategy) details.	No	Yes

1 ÄKTAprocess™ and ÄKTApilot™ only
2 Better handling since the commands are also queued up.

Alarms & Events

Feature	OPC 5.x	OPC 7.0
RunLog event	Condition event	Simple event
Support for events like ***Accumulated time*** and ***Accumulated volume***	Yes	No
Restart error event	No	Yes

Security

Feature	OPC 5.x	OPC 7.0
Automatic creation of the OPC user	No	Yes
Support for access group based login for UNICORN users	No	Yes

cytiva

cytiva.com/unicorn

For local office contact information, visit cytiva.com/contact

29503109 AA V:1 09/2020