Amtiri

REFERENCE MANUAL

OPC UA

Generated on November 25, 2026

OPC UA connects an instance to PLCs, gateways and SCADA products in both directions. The client module reads and writes nodes on an OPC UA server. The server module publishes the instance's own variables to OPC UA clients. Both modules use the same certificate, kept in the same folders.

Module	What it does	Typical use
`OPCUA-Client` (OPC UA Client)	Reads nodes into variables, by subscription or on a schedule, and writes variables to nodes when they change	Tags on a PLC with a built-in OPC UA server, or on an OPC UA gateway
`OPCUA-Server` (OPC UA Server)	Publishes chosen variables as nodes, and accepts the writes its configuration permits	Giving a SCADA, HMI or reporting tool the instance's variables

Enable, **Name** and the **Status and messages** tab work the same way in every module. They are described in Settings every module shares.

Certificates

With any security policy other than `None`, both ends of an OPC UA connection prove who they are with a certificate. The instance creates its own the first time either module opens, and keeps it with the certificates it has been shown:

```
instance/pki/opcua/own/certificate.p12   this instance's key pair and certificate
instance/pki/opcua/trusted/              certificates this instance accepts
instance/pki/opcua/issuers/              certificate authorities, for certificates signed by one
instance/pki/opcua/rejected/             certificates that were refused
```

- **One identity per instance.** The client and the server module present the same certificate: self-signed, RSA 2048, valid for 20 years, with the application URI `urn:amtiri:dataorchester:<install id>`.
- **No trust on first use.** A certificate that is neither in `trusted/` nor signed by an authority in `issuers/` is refused, and filed in `rejected/`. The client's error message shows its thumbprint (SHA-1, as OPC UA tools show it).
- **Trusting is moving a file.** Move the certificate from `rejected/` to `trusted/`. Nothing needs restarting: the folders are watched, and a client that failed to open tries again on its own (see Settings every module shares).
- **What is checked.** The certificate's validity dates and its key usage. The host name and the application URI inside it are not checked, because an industrial device's certificate rarely names the address it is reached at.

> **Important**
>
> The first connection to a secured server is expected to fail once, and so is the first secured connection of a new client to the server module. Trusting a certificate is a deliberate act on each side: move the far end's certificate into `trusted/` here, and trust this instance's certificate on the far end, the way that product's documentation says.

OPC UA Client

The `OPCUA-Client` module opens one session to one OPC UA server and keeps it. It has two lists, as every client module does: **collectors** bring a node's value into a variable, and **forwarders** write a variable's value to a node whenever the variable changes.

Settings

Key	Setting	Type	Default	Meaning
`endpointUrl`	Endpoint URL	string	`opc.tcp://localhost:4840`	The server's address, with scheme and port, such as `opc.tcp://10.20.1.15:4840`. Required: without it the module does not open
`securityPolicy`	Security policy	string	`None`	`None`, `Basic256Sha256`, `Aes128_Sha256_RsaOaep` or `Aes256_Sha256_RsaPss`. The policy's full URI is accepted too. A name the engine does not know is read as `None`
`securityMode`	Security mode	string	`None`	`None`, `Sign` or `SignAndEncrypt`. A value the engine does not know is read as `None`
`userName`	User	string	(empty)	The user the session opens with. Empty opens it anonymously
`password`	Password	string	(empty)	The user's password, or `env:NAME` to read it from the environment variable `NAME` each time the module opens
`publishingIntervalMs`	Publishing (ms)	int, ms	1000	How often the server may send the changes of the subscribed nodes
`collectors`	Collector	list of rows	empty	The nodes read into variables
`forewarders`	Forwarder	list of rows	empty	The variables written to nodes. The key is spelled `forewarders`

Collectors and forwarders

A collector row uses every key below. A forwarder row uses `field`, `nodeId` and `dataType`.

Key	Column	Type	Default	Meaning
`field`	Variable	variable code	(none)	Collector: the variable the node's value is written to. Forwarder: the variable whose changes are written to the node
`nodeId`	NodeId	string	(empty)	The node, in OPC UA notation: `ns=3;s="DB_Tank"."Level"`, `ns=2;i=1001`
`dataType`	Type	string	(empty)	The node's OPC UA type, from the table below. Optional, but see what it decides
`mode`	Mode	string	`poll`	`subscription`: the server sends the value whenever it changes. Any other value, `poll` in the editor: the module reads the node on `cron`
`cron`	Schedule / sampling	Quartz cron	`0 * * * * ? *`	A polled collector's schedule. Required for a polled collector
`samplingIntervalMs`	Schedule / sampling	int, ms	1000	How often the server samples a subscribed node

The **Schedule / sampling** column follows the mode: a cron for a polled collector, a sampling interval for a subscribed one.

Types

`dataType` decides two things. It tells the editor the variable type of the row, so the diagram shows the module's ports and the pencil beside **Variable** can create a variable of that type before the server has ever been reached. And it decides the OPC UA type a forwarder sends: OPC UA writes are typed, so a node declared `Int16` refuses a write carrying an `Int64`, however small the number in it.

`dataType`	Variable type	Notes
`Boolean`	boolean	
`SByte`, `Byte`, `Int16`, `UInt16`, `Int32`, `UInt32`, `Int64`, `UInt64`	int	Unsigned values arrive as whole numbers. A `UInt64` above 9223372036854775807 arrives as a negative number, the value less 18446744073709551616: 18446744073709551615 arrives as -1
`Float`, `Double`	float	
`String`	string	
`LocalizedText`	string	The text, without its locale
`DateTime`	string	ISO 8601 in UTC, to the second: `2026-09-29T14:05:00Z`
`Boolean[]`, `Int32[]`, `Double[]`, `String[]`	list	An array arrives as a list

- **Reading**, the server has already said what the value is: it is converted as the table says, and stored with the variable's own type, as every value a module writes is. A value that type refuses leaves the variable as it was, and the module warns `Variable {variable} refused the value: {reason}`. Before engine 6.18.8 the refusal was only logged.
- **Writing**, the value is converted to `dataType` before it is sent. Without a `dataType`, and for `DateTime` and the array types, it is sent as the variable holds it.
- **Ranges are not checked.** Keep a forwarded variable within the node's range: 70000 sent as `Int16` arrives as 4464.

Behaviour

- **Opening.** Every polled collector's schedule is checked first, then the endpoint URL. A problem stops the module opening, with a message naming it.
- **The endpoint.** The module connects only to an endpoint the server offers with exactly the configured policy and mode. It never falls back to another one, so a server that offers `None` first cannot quietly downgrade a module configured for `SignAndEncrypt`. It keeps the host and port written in **Endpoint URL**, even when the server's endpoints name another address, as a server behind NAT or in a container often does.
- **Subscribed collectors** share one subscription, with the publishing interval of the module and the sampling interval of each row. The subscription is created again every time the session is re-activated.
- **Polled collectors** are read once when the module opens, then on their own schedules. Each node is read on its own, never from a cache on the server.
- **Forwarders** write their variable's current value once when the module opens, then every change.
- **A bad status is not a lost link.** A node that answers with a bad status leaves its variable as it was, and the module warns. A variable is never cleared to say that its reading failed.
- **Reconnection** is the OPC UA library's own: a dropped session is re-activated, and the module reports what happened. A module that could not open at all is tried again by the engine.

Messages

A problem is written into the module's message variables, and cleared once the module succeeds again. "then `ERROR` " follows the thresholds in Settings every module shares: a lost connection becomes an error after `errorAfterSeconds`, and a failure repeated `errorAfterFailures` times in a row becomes one. All texts are in English; the full list is in Module messages.

`{server}` is the **Endpoint URL**. In `{node} refused the write: {reason}`, `{reason}` is the status code the server answered the write with, written out with its name, such as `Bad_NotWritable`, and its value.

When	Level	Text
Opening: a polled collector has no schedule	`ERROR`	`No schedule given`
Opening: a schedule does not parse	`ERROR`	`Invalid schedule '{cron}': {reason}`
Opening: no endpoint URL	`ERROR`	`No endpoint URL`
Opening: the server's certificate is not trusted	`ERROR`	`Server certificate not trusted: {thumbprint}`
Opening: the server cannot be reached, or offers no endpoint with this policy and mode	`ERROR`	`Server {server} unreachable: {reason}`
The session is active	`INFO`	`Connected to {server}`
Every subscribed node was accepted	`INFO`	`Subscribed to {n} node(s)`
The server refused some subscribed nodes	`WARNING`	`{n} of {total} node(s) refused by the server`
The subscription could not be created	`WARNING`	`Subscription refused: {reason}`
The session dropped	`WARNING`, then `ERROR`	`Disconnected from {server}`, then `Server {server} unreachable`
A polled read failed	`WARNING`, then `ERROR`	`Reading {node} failed: {reason}`
A node answered with a bad status	`WARNING`	`{node} returned {status}`
The variable could not take the value	`WARNING`	`Variable {variable} refused the value: {reason}`
The server refused a write	`WARNING`	`{node} refused the write: {reason}`
A write could not be sent	`WARNING`	`Writing {node} failed: {reason}`

Example

A tank level taken as the PLC changes it, a pump's running hours checked every five minutes, and a level setpoint written back when an operator moves it, over an encrypted session:

```
Endpoint URL      opc.tcp://10.20.1.15:4840
Security policy   Basic256Sha256
Security mode     SignAndEncrypt
User              orchester
Password          env:OPCUA_PLC1_PASSWORD

Collector   LIT_101        ns=3;s="DB_Tank"."Level"      Double   Subscription   500
Collector   P_101_HOURS    ns=3;s="DB_Pump"."Hours"      UInt32   Poll           0 */5 * * * ?
Forwarder   LIT_101_SP     ns=3;s="DB_Tank"."Setpoint"   Double
```

The first connection fails with `Server certificate not trusted: {thumbprint}`. Once the PLC's certificate is moved into `trusted/`, and the PLC trusts this instance's certificate, the next attempt connects. `LIT_101` then follows the PLC without being asked, `P_101_HOURS` costs one read every five minutes, and writing `LIT_101_SP` from a dashboard or a formula sends it to the PLC as a `Double`.

OPC UA Server

The `OPCUA-Server` module makes the instance an OPC UA server. A SCADA, an HMI or a reporting tool connects to it, browses the published variables, reads or subscribes to them, and writes the ones it is allowed to.

The endpoint URL is:

```
opc.tcp://<host>:<port>/orchester
```

`<host>` is the address the client reaches the machine at, and `<port>` is the module's port. In the endpoint descriptions it publishes, the server names itself by the machine's host name; a client that cannot resolve that name should keep the address it dialled.

Settings

Key	Setting	Type	Default	Meaning
`port`	Port	int	4840	The TCP port. The server listens on every network interface
`securityPolicies`	Security policy	list of strings	`Basic256Sha256`	The policies offered, one endpoint each, chosen with four checkboxes: `None`, `Basic256Sha256`, `Aes128_Sha256_RsaOaep`, `Aes256_Sha256_RsaPss`. At least one must be usable
`allowAnonymous`	Allow anonymous access	boolean	false	Whether a session may open without a user. An anonymous session may read, and may never write
`users`	Users	list of rows	empty	Who may open a session
`variables`	Published variables	list of rows	empty	The variables the server publishes

- **Every policy except `None` is offered with the mode `SignAndEncrypt` only.** `None` is offered with the mode `None`, and only when it is ticked.
- **With anonymous access off and no users, no client can open a session.**

Users

Key	Column	Type	Default	Meaning
`userName`	User	string	(empty)	The name a client signs in with. Compared exactly, upper and lower case included
`password`	Password	string	(empty)	The password, or `env:NAME` to read it from the environment variable `NAME` when the module opens. A user without a password, or whose variable is not set, can never sign in
`writable`	Writable	boolean	false	Whether this user may write the variables that permit it

Published variables

Key	Column	Type	Default	Meaning
`field`	Variable	variable code	(none)	The variable published
`browseName`	Browse name	string	(empty)	The name the node is shown under. Empty uses the variable's code
`dataType`	Type	string	(empty)	The node's OPC UA type: `Boolean`, `Int16`, `UInt16`, `Int32`, `UInt32`, `Int64`, `UInt64`, `Float`, `Double`, `String` or `DateTime`. Empty publishes the node as `BaseDataType`, which takes any value
`writable`	Writable	boolean	false	Whether the variable may be written from outside

The address space

- **One folder per module**, under the Objects folder, named after the module's **Name**.
- **One node per published variable**, addressed as `ns=<index>;s=<code>`, with the code as written in the row. The browse name and the display name are the row's **Browse name**.
- **The namespace index is assigned by the server**, not chosen. On an instance with one server module it is usually 2. The engine log gives it when the module opens, in a line such as `[SCADA_LINK][OPEN] 2 variable(s) published as ns=2;s=<code> on port 4840`, where `SCADA_LINK` is the module's name: read it there rather than assuming it.
- **Values follow the variables.** A node holds its variable's value when the module opens, and every change of the variable is written to the node, so a client that subscribes hears about it.

Who may write

A write is accepted only when the variable is marked **Writable** *and* the session's user is marked **Writable**. Neither implies the other: a variable that must never be driven from outside stays read-only whoever asks, and a read-only user cannot write even the variables that permit it.

The variable	The session	The client gets
Not writable	Any	`Bad_NotWritable`
Writable	A user marked **Writable**	The write is accepted, and the value is written to the variable
Writable	A user not marked **Writable**	`Bad_UserAccessDenied`
Writable	Anonymous	`Bad_UserAccessDenied`
Writable	A user marked **Writable**, with a value the variable's declared type refuses	`Bad_TypeMismatch`
Writable	A user marked **Writable**, when writing the variable fails for another reason	`Bad_InternalError`

An accepted value is stored with the variable's type, as every value a module writes is. A value that type refuses changes neither the variable nor the node, and the module warns `Variable {variable} refused the value: {reason}`. With `orchester.typing.strict=false` in the configuration file, such a value is written anyway. Before engine 6.18.8 a refused value was answered as accepted, and the node kept the client's value while the variable did not; a write that failed for another reason was answered `Bad_OutOfRange`.

Messages

When	Level	Text
Opening: no ticked policy is usable	`ERROR`	`No usable security policy: {policies}`
Opening: the port cannot be bound, or the server cannot start	`ERROR`	`Cannot listen on port {port}: {reason}`
The server is listening	`INFO`	`Listening on port {port}`
A variable's new value could not be written to its node	`WARNING`	`Variable {variable} could not be published: {reason}`
A client wrote a value the variable refused	`WARNING`	`Variable {variable} refused the value: {reason}`

The server module opens before the other modules, and a server that could not open, because another program still holds its port for example, is tried again by the engine.

Example

A tank level a plant SCADA may read, and a setpoint it may write:

```
Port              4840
Security policy   Basic256Sha256, Aes256_Sha256_RsaPss
Allow anonymous   off

Published variables
  LIT_101      Tank101Level      Double   writable: no
  LIT_101_SP   Tank101Setpoint   Double   writable: yes

Users
  scada        env:OPCUA_SCADA_PASSWORD     writable: yes
  reports      env:OPCUA_REPORTS_PASSWORD   writable: no
```

The SCADA connects to `opc.tcp://<host>:4840/orchester` with `SignAndEncrypt`, finds both nodes under the module's folder as `ns=2;s=LIT_101` and `ns=2;s=LIT_101_SP`, and signs in as `scada`. It can write `LIT_101_SP`; writing `LIT_101` answers `Bad_NotWritable`. `reports` can read both and write neither: writing `LIT_101_SP` answers `Bad_UserAccessDenied`, and writing `LIT_101` answers `Bad_NotWritable`.

> **Warning**
>
> Ticking `None` publishes an endpoint that neither signs nor encrypts. Leave it off unless the network between the two ends is closed to everything else.

Next steps

- Module catalogue
- Settings every module shares
- Module messages
- Siemens S7 and EtherNet/IP, for controllers without OPC UA
- Connectivity

This page describes Data Orchester engine 6.18.8.