CygNet OPC UA Server User's Guide

Release Date: July 2, 2021

This document provides important information to help you get started, prepare your system, install, upgrade, and configure the **CygNet OPC UA Server**.

The CygNet OPC UA Server 1.0 operates with CygNet v9.5 and CygNet Bridge v4.4.121.

This document provides the *same* content as is available in the main CygNet Help. Refer to the following sections: CygNet OPC UA Server and CygNet Bridge API.

Contents

CHAPTER 1: About the CygNet OPC UA Server

CygNet provides an OPC Unified Architecture (UA) Server, an application which exposes real-time and archived CygNet data to an OPC UA Client application using the OPC UA (OPC Unified Architecture) specification, version 1.04. OPC UA is a secure, open, reliable mechanism for transferring information between servers and clients.

The CygNet OPC UA Server adheres to the OPC UA specification, which is a service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework.

The CygNet OPC UA Server 1.0 operates with CygNet v9.5 and CygNet Bridge v4.4.121.

This chapter provides an overview of CygNet's OPC UA Server, licensing information, an architectural diagram, the typical workflow required to install and operate the OPC UA server, and terms and concepts relevant to CygNet and OPC UA.

In this chapter:

CygNet OPC UA Server Overview

The CygNet OPC UA Server is a Windows service application installed on a host computer in a SCADA data center, which exposes CygNet data to OPC UA client applications. The server communicates with a CygNet system to request and receive encrypted data via HTTPS using via our flexible intermediary REST API application, CygNet Bridge API.

The diagram on the next page demonstrates this architecture.

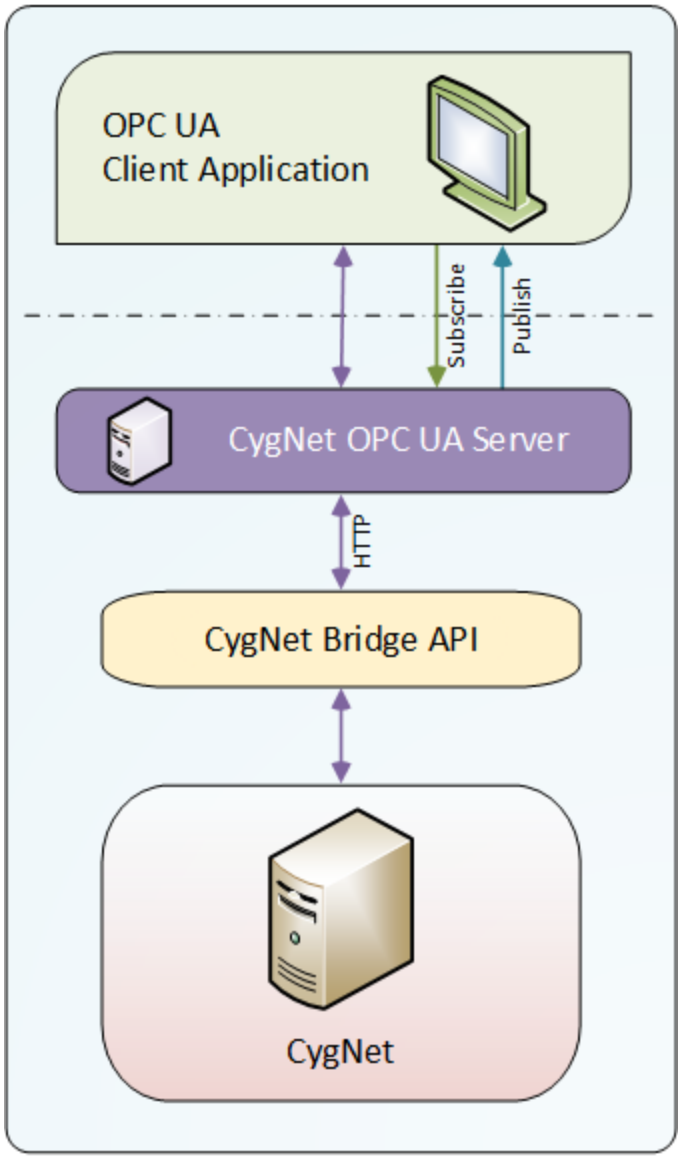
The CygNet OPC UA Server supports the following communication methods for exchanging CygNet data over a secure network connection:

- Client/Server — a request-response model where an OPC UA client makes requests for data and the OPC UA server returns with a response
- Pub/Sub — a publish-subscribe model where an OPC UA client establishes a subscription to data and then any changes will be published by the OPC UA server

The CygNet OPC UA server accesses CygNet data via CygNet Bridge and uses the Bridge API to authenticate access and provide the following types of CygNet data:

- Real-time values from the Current Value Services
- Historical values from the Value History Service
- Facility record and attributes from the Facility Service
- Point records and attributes from the Point Service
- Current alarms from the Common Alarm Service

CygNet OPC UA Server Architecture



CygNet OPC UA Server Architecture

CygNet OPC UA Server Licensing

The CygNet OPC UA Server is a separately licensed component from existing CygNet SCADA software components. A valid license file is required containing the appropriate OPC UA license component. The OPC UA Server will perform a license check that the OPC UA license component is active in the license file.

A CygNet Bridge license is not required, as the server will check for the necessary OPC UA license. If you have only the OPC UA license, you must install CygNet Bridge API, but you will not be able to use it directly. If you intend to access and use Bridge API separately for other purposes, you will need to get a separate CygNet Bridge API license. See **Licensing CygNet Bridge API** in the **Bridge API** section of the CygNet Help for more information about license types for that product.

For more information about obtaining and licensing the CygNet OPC UA and CygNet Bridge products, contact your Account Manager

Typical CygNet OPC UA Server Workflow

The following steps describe a typical workflow scenario for setting up, configuring, and running the CygNet OPC UA Server.

1. Acquire and install appropriate certificates from a Certificate Authority.
2. Install **CygNet Bridge** and **CygNet Bridge API** using the CygNet Bridge setup file: **CygNet Bridge Setup.exe**. See **Installing and Updating CygNet Bridge** and **Accessing and Updating CygNet Bridge API** for information about installing **CygNet Bridge** and accessing **CygNet Bridge API**.
3. Configure the necessary security settings required for **CygNet Bridge API** authentication in the CygNet Access Control Service (ACS). See **CygNet Bridge API (BRDGAPI) Security** for more information this security application.
4. Start the CygNet Bridge website using IIS Manager.
5. Install the CygNet OPC UA Server using the CygNet OPC UA Server installer file: **CygNetOpcUa.Setup.msi**. See Installing the CygNet OPC UA Server for more information.
6. Configure the server configuration file: **CygNetOpcUaServer.Config.xml**. See Configuring the CygNet OPC UA Server for more information about the settings in this file.
7. *Optional but highly recommended*. Encrypt the CygNet Bridge password. See CygNet OPC UA Server Password Encryption for more information about how to do this.
8. Run the model builder using the PowerShell script: **GenerateCygNetModel.ps1** to generate the **CygNetModel.json** file, which is required to create the OPC UA Address Space. See Maintaining the CygNet OPC UA Address Space for more information about this process.
9. Restart the CygNet OPC UA Server service (manually via the Windows Task Manager or via script). Once the CygNet OPC UA Server is running the address space is generated. See Start the CygNet OPC UA Server for more information.
10. *Optional*. Schedule the model builder script to run regularly to pick up changes to the CygNet facility and point configurations. See Maintaining the CygNet OPC UA Address Space for more information about this process.
11. Install and configure your preferred OPC UA client (e.g., Matrikon OPC UA Explorer, dOPC, others) to connect to the CygNet OPC UA Server using an authentication method of your choosing.
12. Program the OPC UA client to request or monitor CygNet data.

CygNet OPC UA Server Concepts

The following terms, concepts, and definitions relate to the CygNet OPC UA Server.

Term	Description
Address Space	The address space is the core of an OPC UA server and is a representative model of the data the server exposes. The collection of objects and related information that an OPC UA server makes available to an OPC UA client is referred to as its address space. The address space defines the internal node and folder hierarchical structure, the access-rights for each node, how the data for each node will be provided, and display names for each node. An OPC UA server exchanges data in a standardized way with OPC UA clients via an address space model. See Maintaining the CygNet OPC UA Address Space for more information about generating the CygNet address space, and keeping the address space current with CygNet data. Once generated the address space can be viewed and navigated in the OPC UA client.
Alarms	The OPC UA client subscribes to CygNetAlarmType events in order to access alarms on CygNet points. The client can subscribe to events on the OPC UA "Server" object to receive notification of alarms on all points in the CygNet system (the "Server" object is a node which exists on all OPC UA servers). Alternatively, the OPC UA client can subscribe to any level of the address space to receive alarm notifications for all points located within the hierarchy under that node. For example, a client could subscribe to a Facility node to receive alarms for all points on that Facility. Or they could subscribe to an individual RealtimeRecord in the system to receive notification of alarms on that point. Alarm acknowledgment is the only action supported by the OPC UA Server. The CygNetAlarmType is one of the CygNet event types defined in the OPC UA server. The RealtimeRecordType is one of the CygNet data types defined in the OPC UA server. Both are described below. See CygNetAlarmType for more information about how alarms are handled by the OPC UA server.
CvsSet	A custom node specified in the address space that contains the CygNet Current Value Services. Found under Objects in the address space hierarchy. For each CygNet Service type you can drill down to the FacilityTypeSet to each specific facility to see facility attributes, point set, point configuration, and real time record, etc.

Term	Description
DataTypes	CygNet OPC UA Server has defined the following extended (custom) data types to represent the CygNet CVS, facility, point, and real-time records: • CvsType — contains the CygNet Current Value Services • FacilityType — contains CygNet facility attributes • PointConfigurationType — contains CygNet point configuration record properties • RealtimeRecordType — contains CygNet real-time record (current value) properties CvsType, FacilityType, PointConfigurationType, and RealtimeRecordType are custom DataTypes found under the Types > ObjectTypes > BaseObjectType in the address space hierarchy. See Extended Data Types and Event Types for more information about each data type properties.
Display name	The name to be displayed for each node in the address space, determined by the **<DisplayNameProperties>** element in the configuration file. There are three types of DisplayNames: **Cvs**, **Facility**, and **Point**. See <DisplayNameProperties> in Configuring the CygNet OPC UA Server for more information.
EventTypes	CygNet OPC UA Server has defined the following extended (custom) event type to represent the CygNet alarm records: • CygNetAlarmType CygNetAlarmType is a custom EventType found under the Types > EventTypes > BaseEventType > ConditionType > AcknowledgeableConditionType > AlarmConditionType in the address space hierarchy. See Extended Data Types and Event Types for more information about the CygNetAlarmType properties.
Local Discovery Server	The Local Discovery Server (LDS) is a UA Discovery Server that maintains a list of all OPC UA servers and gateways available on the host where it is running to make its endpoint information available to OPC UA clients with access. The LDS is a service that runs in the background. The CygNet OPC UA Server will periodically connect to an LDS and register itself as being available. **Note:** CygNet Software does not provide a Local Discovery Server for installation, although you can acquire one from the OPC Foundation.
Model Builder	The model builder is a mechanism for generating the address space for the CygNet system, which is run when you first set up the CygNet OPC UA Server, and when you make changes to CygNet that you want exposed in OPC UA (such as adding or removing points or facilities, changes to metadata, etc.). Specifies the XML namespace used by the server. See Maintaining the CygNet OPC UA Address Space for more information.
"Modified" historical values	The OPC UA specification supports the retrieval of history data entries with the same timestamp, each with distinct values. See Timestamp ordinal below for information about how CygNet handles this type of data.

Term	Description
Nodes	A node is the basic unit of data in the address space, which provides a standard way for an OPC UA server to represent objects to an OPC UA client. A node is a piece of information (for example, a temperature value) and consists of attributes, the actual data value, and one or more references to other nodes, each in its own address space.
NodeId	The NodeId uniquely identifies every node in the entire OPC UA address space. For the CygNet OPC UA Server the NodeId is derived from the CygNet tag to ensure uniqueness, allowing the client to make a request to the server to read the Facility node directly. See NodeId Formats for more information about how NodeIds are formatted.
Point state mapping	**Mapping CygNet Point States to OPC UA Status Codes** The CygNet OPC UA Server can retrieve (via Bridge API) the point status fields and resolved point states for real-time and historical values. The OPC UA server maps CygNet point states to OPC quality and sub-status values in the same way status mappings are handled by the legacy CygNet OPC DA and CygNet OPC HDA servers. The **CvsMetadata** provides an external status attribute "ext_status" on the **PointStateDefinition** element for mapping a point state to a numeric OPC quality and sub-status value. This mapping is used by our legacy OPC DA and OPC HDA servers to communicate the OPC quality and sub-status of a current or historical value. In order to maintain compatibility between our legacy OPC DA and HDA applications and the OPC UA server application, values assigned to the "ext_status" attribute of a PointStateDefinition node in the CvsMetadata file are translated to the corresponding OPC UA status codes and passed to your OPC UA clients. To be clear, the value assigned to the "ext_status" attribute must still comply with the legacy OPC DA specification for quality and sub-status, but that value will be translated to the corresponding OPC UA status code. See Mapping CygNet Point States to OPC Quality and Sub-Status for more information about the CygNet point state mappings to the OPC quality and sub-status values. **Note:** The OPC foundation has made legacy OPC quality and sub-status concepts analogous for OPC DA, OPC HDA, and OPC UA. OPC UA quality mapping is described in OPC 10000-8: OPC UA Specification Part 8: Data Access, Annex A.3.2.3, Table A.33.
Profiles	A profile indicates what specific features of the OPC UA specification are supported. A profile comprises a collection of facets, which describe the specific capabilities supported by the OPC UA server.

Term	Description
Timestamp ordinal	CygNet supports history data entries with the same timestamp each with distinct values with ordinals. CygNet assigns each unique timestamp entry an ordinal to allow differentiation between the multiple entries. OPC UA historical retrieval supports a concept of "modified" values, representing entries with the same timestamp with distinct values. This closely resembles the CygNet concept of "timestamp ordinals". This release of the CygNet OPC UA Server does not support the retrieval of "modified" historical values. This is because the OPC UA Server talks to CygNet Bridge and the Bridge interface does not provide historical entry timestamp ordinals in the data it returns. Since the Bridge API does not allow timestamp ordinals to be specified when requesting historical values nor return historical value timestamp ordinals, the underlying value iteration logic in the Bridge API has been modified to skip to the last ordinalized value entry to avoid a possible endless loop of historical value requests and responses.
Transport channel	The transport channel is the secure communication medium used to exchange serialized OPC UA messages between a server and a client over a network. OPC UA supports the sharing of data via many different transport layers using IP-based protocols, such as SOAP/HTTP, HTTPS, and UA TCP.
Types	A node in the address space hierarchy that defines all the available OPC UA types, including ObjectTypes, Variabletypes, DataTypes, ReferenceTypes, EventTypes, and InterfaceTypes. See DataTypes and EventTypes above.

CHAPTER 2: CygNet OPC UA Server Security

This chapter describes important information about OPC UA and security: certificates, user and application authentication, transport security, CygNet Bridge API access, general security considerations, and password encryption.

In this chapter:

CygNet OPC UA Server Security

Security is integral to OPC UA. OPC UA supports a sophisticated security model that ensures the authentication of users, authentication of client and servers, and the integrity of communication channels and shared data. The OPC UA standard requires that all network endpoints communicate over a secure connection. The CygNet OPC UA Server provides security via data encryption, user authentication, and application authentication, ensuring secure transmission of CygNet data to the OPC UA client.

See the following subsections for more information:

- OPC UA Certificates
- OPC UA Client User Authentication
- OPC UA Server Application Authentication
- Transport Channel Security
- CygNet Bridge API Access
- CygNet Security Considerations
- CygNet OPC UA Server Password Encryption

OPC UA Certificates

Security in OPC UA is handled using certificates. Each OPC UA server instance and OPC UA client instance must provide a trusted certificate to identify itself in order to establish a secure connection. OPC UA uses the X.509 certificate standard, which defines a standard public key format, by signing and encrypting the OPC UA messages to validate integrity and providing application identification to ensure trustworthiness.

The default CygNet OPC UA Server certificates store is **C:\ProgramData\Weatherford\CygNetOpcUaServer\Certificates**.

OPC UA Client User Authentication

An OPC UA client must provide appropriate user credentials (username and password) to establish an OPC UA session. These user credentials must resolve to a valid user identity to authenticate against Windows Active Directory on the CygNet Bridge API server. User credentials are passed through to CygNet Bridge to authenticate against the Client Login API. There will be one Bridge session per user identity in CygNet OPC UA Server. ACS permissions are enforced per-user in CygNet according to the end-user's Windows userid.

Trusted Certificates

If the OPC UA client fails to authenticate with the OPC UA server, you might see a message in the OPC UA log indicating that a client certificate is untrusted. The client certificate will be rejected if it isn't in the trusted folder, so it should be added to the trusted folder before attempting to connect.

The **Certificates** folder locations can be overridden in the **CygNetOpcUaServer.Config.xml** file. See Security Configuration for more information about this option.

OPC UA user authentication is described in OPC 10000-2: OPC UA Specification Part 2: Security Model, Section 4.9.

OPC UA Server Application Authentication

Application authentication is achieved via trusted certificates. An OPC UA client provides a trusted certificate and the OPC UA server has that certificate in its trusted certificate store (e.g., **C:\ProgramData\Weatherford\CygNetOpcUaServer\Certificates\Trusted**). There is no user identity associated with the client's session, so all access to CygNet Bridge (and CygNet) will be using the OPC UA server identity configured (and encrypted) in the OPC UA configuration file. See Configuring the CygNet OPCUA Server for more information about configuring the OPC UA server credentials and CygNet OPC UA Server Password Encryption below for more information about password encryption.

OPC UA application authentication is described in OPC 10000-2: UA Specification Part 2: Security Model, Section 4.10.

Transport Channel Security

Both server and clients authenticate against each other before communicating data over transport layers, where messages are encrypted and signed end-to-end, ensuring highly secure communication channels.

CygNet Bridge API Access

Since the CygNet OPC UA Server communicates with CygNet Bridge API, the BRDGAPI/ACCESS security event must be configured in the CygNet ACS. The OPC UA server requires that all user identities (both client identities and the server identity) have the BRDGAPI/ACCESS event configured.

Note that the OPC UA client identity is only required in the CygNet ACS when using User Authentication. If using Application Authentication only, the server identity is required, and all OPC UA clients will have access to the same data that the server identity is allowed.

See **CygNet Bridge API (BRDGAPI) Security** for more information about the BRDGAPI/ACCESS security event.

> **Note:**
> CygNet Bridge API supports a two-factor authentication (2FA) option for an additional layer of security for user authentication. However, this release of the CygNet OPC UA Server *cannot* be used with a user with 2FA enabled in CygNet Bridge.

CygNet Security Considerations

Note the following items regarding CygNet ACS permissions for the OPC UA Server identity for alarms, subscriptions, and browsing nodes.

Alarms

- To control which alarms are exposed in the OPC UA Server, the CygNet ACS permissions for the OPC UA Server identity are used. This is the identity specified in the CygNetBridgeUsername configuration file element.
- All OPC UA clients will have access to the alarms exposed by the OPC UA Server identity's ACS permissions.

Real-time value subscriptions

- The CygNet ACS permissions of the client's identity are checked at the time the subscription is established. Denied permissions will return the error **BadNotFound**.

- If the OPC UA client session is established using an X509 certificate for authentication, then the OPC UA Server identity's ACS permissions are used for determining access to the CygNet current values.
- If the OPC UA client session is established using user credentials, then the client user identity's ACS permissions are used for determining access to the CygNet current values.
- Changes to a client user's ACS permissions will only affect new subscriptions. Existing subscriptions will not be affected, and will continue to publish new real-time values for that subscription.
- The OPC UA Server will need to be restarted in order to force the new ACS permissions to take effect, for existing OPC UA client session subscriptions.

Browsing nodes in the CygNet address space

- When the CygNet address space is built using the CygNet OPC UA Server model builder script (GenerateCygNetModel.ps1), all CygNet facilities and points that the configured OPC UA Server identity has permission to read will be included in the address space.
- Consequently, when browsing the address space using a client application such as the Matrikon OPC UA Explorer, all facilities and points included in the address space will be viewable regardless of the permissions of the logged-in user session.
- An attempt to subscribe to a real-time value for a point for which the current user has no permissions to read will return the error **BadNotFound**.
- Reading facility and point property values through the OPC UA Server's Read service will enforce the ACS permissions for that client's user identity.

CygNet OPC UA Server Password Encryption

Best practice recommends that the OPC UA server password be saved in an encrypted format in the CygNet OPC UA Server configuration file.

This is accomplished by starting the server with an "-encrypt" parameter, which will start the server in a special mode used only for password encryption and nothing else.

> **Note:**
>
> When the server is started with the "-encrypt" parameter, it will stop immediately after performing the encryption routine. The server will need to be started without the "-encrypt" parameter in order to run normally.

When running the encryption routine, the server will look for the **<CygNetBridgePassword>** element in the **CygNetOpcUaServer.Config.xml** file and will replace it with a **<CygNetBridgePasswordEncrypted>** element that contains a 64-bit encoded string as its value. The server will also generate a second element called **<CygNetEncryptionKeyPath>**, which points to the location of a generated key file that is used by the server anytime the encrypted password needs to be decrypted (i.e., whenever the server needs to supply the unencrypted password to CygNet Bridge). This key file can be moved to a secure location, such as to a flash drive, and the **<CygNetEncryptionKeyPath>** in the configuration file must be updated accordingly.

See the <CygNetBridgePassword> element in the **CygNetOpcUaServer.Config.xml** file for more information.

To encrypt the CygNet Bridge password

The CygNet OPC UA Server supports a command-line interface to start the server and encrypt the CygNet Bridge password:

1. Open a Command Prompt window.
2. Navigate to the directory where the server executable is installed: **C:\Program Files\Weather-ford\CygNetOpcUa\CygNetOpcUaServer**
3. Type the following command:

```
CygNetOpcUaServer -encrypt
```

4. The CygNet OPC UA Server will stop after the encryption routine is completed.

CHAPTER 3: Installing CygNet OPC UA Server

This chapter describes system requirements, how to prepare your machine and CygNet software for CygNet Bridge, how to install and update CygNet Bridge, log file location, how to install and update the CygNet OPC UA server, how to start the server, and a list of required files. Also included is information about server memory usage estimates.

In this chapter:

Installing the CygNet OPC UA Server

The CygNet OPC UA Server accesses CygNet services via an instance of CygNet Bridge API, in order to access data via HTTP. The following topic describes the OPC UA server system requirements, links to information about installing CygNet Bridge and CygNet Bridge API, the OPC UA log files, and how to install the OPC UA Server, and a list of files required by the server.

OPC UA Server System Requirements

CygNet's OPC UA Server is installed as a Windows service on a computer in your CygNet environment. The following CygNet components must also be installed in the same environment:

1. CygNet SCADA Services
2. CygNet Bridge API
3. CygNet OPC UA Server

We recommend that the CygNet OPC UA Server be installed on a different computer from the machine where the CygNet SCADA system is running. CygNet Bridge and CygNet OPC UA Server may or may not be on the same server. It is recommended that these components be installed on separate servers to optimize performance.

There is more flexibility as to where the CygNet OPC UA server is installed. Since it communicates through CygNet Bridge, the server can be located on a computer that isn't necessarily in the CygNet environment, since it does not require CygNet messaging or RUDP communication, as the server only needs to communicate over an HTTPS network connection. However close network proximity between the CygNet OPC UA Server, CygNet Bridge, and the CygNet SCADA system is recommended, to lower network latency and maximize performance.

Install CygNet Bridge API

Before you install the CygNet OPC UA Server you will need to install and configure CygNet Bridge API. See the following topics for more information:

Information about Installing CygNet Bridge

- **CygNet Bridge** (an Overview)
- **Preparing Your System for CygNet Bridge** (includes enabling .NET Framework, installing IIS, and activating HTTPS)
- **Installing and Updating CygNet Bridge** (includes installing, starting, and updating CygNet Bridge)
- **Troubleshooting CygNet Bridge**

Information about Configuring CygNet Bridge API

- **CygNet Bridge API** (an Overview)
- **Preparing Your System for CygNet Bridge API** (includes licensing, ACS security settings, and optional two factor authentication)
- **Accessing and Updating CygNet Bridge API** (includes accessing and updating CygNet Bridge)
- **Troubleshooting CygNet Bridge API**

CygNet OPC UA Server Log Files

The CygNet OPC UA Server log files are found in the following default storage location:

- **C:\ProgramData\Weatherford\CygNetOpcUaServer\Logs**

Consult the log file to view any issues or errors related to the OPC UA service itself, installation, or its operational functions.

See the <LoggingConfiguration> element in the **CygNetOpcUaServer.Config.xml** file for more information about configuring logging for the OPC UA server.

Install the CygNet OPC UA Server

Accomplish all prerequisite system preparation tasks before attempting to install CygNet OPC UA Server. Preparation tasks include the following:

- Acquire and install CygNet OPC UA license. Note that you do not need a CygNet Bridge API license to use the CygNet OPC UA Server. See CygNet OPC UA Server Licensing.
- Start the CygNet Bridge API website.
- Configure the BRDGAPI/ACCESS security event in the CygNet ACS. This will be required for the OPC UA Server default user (configured via <CygNetBridgeUsername> in the **CygNetOpcUaServer.Config.xml** file), as well as for any users who will be connecting to the CygNet OPC UA Server using User Authentication. See CygNet Bridge API Access.

After you have prepared your server(s) and system to meet all requirements, install and configure CygNet OPC UA Server as follows.

To Install CygNet OPC UA Server

1. Install the CygNet OPC UA Server:
   a. Obtain the product source files from the **CygNet Software Download Website** on the Weatherford software support portal (login required).
   b. Copy the CygNet OPC UA Server installer file (**CygNetOpcUa.Setup.msi**) to the staging location on the computer where you plan to install the service.
   c. Use the setup program to install the OPC UA service on the computer:
      i. Start the CygNet OPC UA installer, and click **Next**.
      ii. Read (follow the link) and accept the terms in the End-User License Agreement (EULA), and click **Next**.
      iii. Select the setup type that best suites your needs: **Typical**, **Custom**, or **Complete**.
      iv. On the **Ready to install** page, click **Install** to install the CygNet OPC UA service.
      v. On the **Completed ...** page, click **Finish**.
2. Configure the CygNet OPC UA Server. Open the installed **CygNetOpcUaServer.Config.xml** file and configure it to your specifications. Save and close the file.

> **Note:**
> The **CygNetOpcUaServer.Config.xml** file is located in **C:\ProgramData\Weatherford\CygNetOpcUaServer**.
> See Configuring the CygNet the OPC UA Server for more information about the settings in this file.

3. Encrypt the CygNet Bridge password. To encrypt the password, start the server using the "-encrypt" parameter. The server will stop after the password is encrypted. See CygNet OPC UA Server Password Encryption for more information about this step.
4. Run the model builder script, **GenerateCygNetModel.ps1**, to generate the **CygNetModel.json** file, which is required to create the OPC UA internal node hierarchy (the address space). CygNet Bridge must be running. See Maintaining the CygNet OPC UA Address Space for more information about this process.
5. Restart the the CygNet OPC UA Server. After editing the **CygNetOpcUaServer.Config.xml** file and building the **CygNetModel.json** file, you will need to restart the CygNet OPC UA Server, as described next.

Start CygNet OPC UA Server

Any time you finish installing or updating OPC UA Server, complete the following steps to initialize and launch the server.

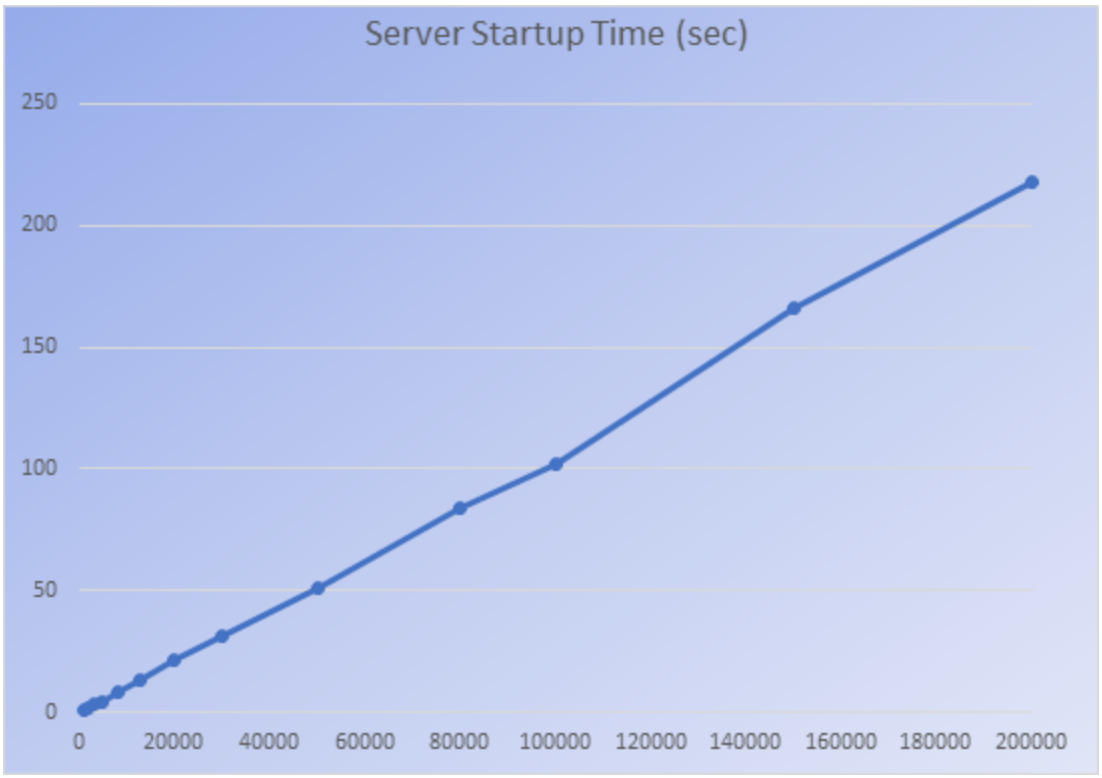
To Start CygNet OPC UA Server

1. Open Windows Task Manager (run **TaskMgr.exe**).
2. Click on the **Services** tab.
3. Find **CygNetOpcUaServer** in the list; the description is **Weatherford CygNet OPC UA Server**.
4. Right-click and select **Start**.

> **Important Note:**
> You will need to restart the service each time you encrypt the password, edit the **CygNetOpcUaServer.Config.xml** file, or rebuild the address space.

CygNet OPC UA Server Startup Time

When the CygNet OPC UA Server starts, it first builds its internal node hierarchy from the contents of the **CygNetModel.json** file. The time required for this startup process to complete will depend upon the number of CygNet points represented in the **CygNetModel.json** file. The following chart estimates the number of seconds that the CygNet OPC UA Server will take to build its internal node hierarchy based upon the number of CygNet points included in the CygNet model. Your results will vary depending on attributes of your host machine and network environment such as CPU, memory, and network latency. See CygNet OPC UA Server Memory Usage for more information about how the server uses system memory.



CygNet OPC UA Server Startup Time
(seconds by number of points)

Update CygNet OPC UA Server

The CygNet OPC UA Server can be updated any time a new version is released. Update the CygNet OPC UA Server files as follows.

1. **Stop** CygNet OPC UA Server in the Task Manager.
2. Obtain the product source files from the **CygNet Software Download Website** on the Weatherford software support portal (login required).
3. Copy the CygNet OPC UA Server installer file (**CygNetOpcUa.Setup.msi**) to the staging location on the computer where you previously installed the service.
4. Open the CygNet OPC UA Server installer file, and follow the prompts to install the software.
5. **Start** CygNet OPC UA Server in the Task Manager, as described above.

Required Files

The following table lists some of the important files you will need to run or configure when using the CygNet OPC UA Server and CygNet Bridge API.

File	Folder	Purpose
CygNetModel.json	The location for this file is determined by the <CygNetModelFilePath> element in the **CygNetOpcUaServer.Config.xml** file. The default path is **C:\ProgramData\Weatherford\CygNetOpcUaServer**	A file generated by the model builder script (**GenerateCygNetModel.ps1**) that encapsulates the address space. The CygNet OPC UA Server uses the **CygNetModel.json** file on startup to populate its node hierarchy. The **CygNetModel.json** is created when the model builder script is first run. Each subsequent time the script is run the existing **CygNetModel.json** is backed up, and either archived (on success) or restored (on failure). Backups are stored in **C:\ProgramData\Weatherford\CygNetOpcUaServer\CygNetModelBackups**. The **CygNetModel.json** file should never be modified manually or deleted.
CygNetOpcUa.Setup.msi	Available from the **CygNet Software Download Website** on the Weatherford software support portal (login required)	The CygNet OPC UA installer file.
CygNetOpcUaServer.Config.xml	**C:\ProgramData\Weatherford\CygNetOpcUaServer**	The CygNet OPC UA Server configuration file. The server must be restarted each time you modify this file for changes to take effect. See Configuring the CygNet OPC UA Server for more information.

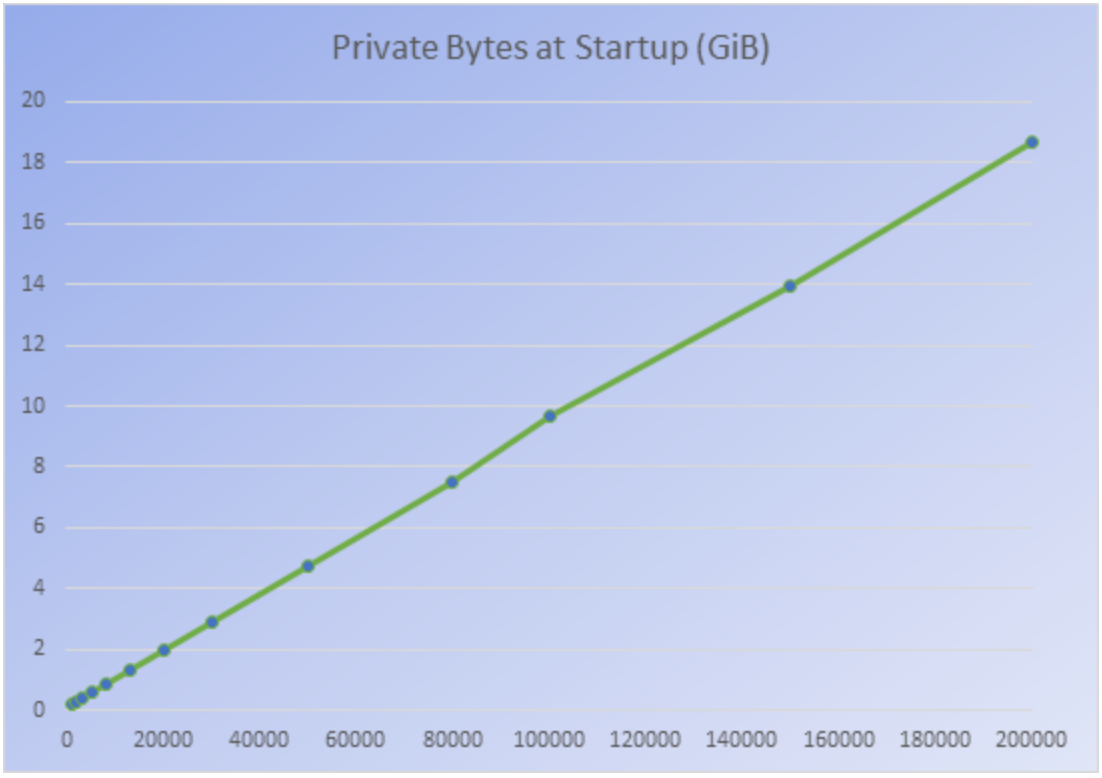
File	Folder	Purpose
CygNetOpcUaServer.exe	**C:\Program Files\Weatherford\CygNetOpcUa\CygNetOpcUaServer**	The CygNet OPC UA Server executable. There is no need to interact with this file, unless you plan to start the server or encrypt the password CygNetBridgePassword manually. See CygNet OPCUA Server CLI for more information.
CygNetOpcUaServer.log	**C:\ProgramData\Weatherford\CygNetOpcUaServer\Logs**	The log file(s) generated by the CygNet OPC UA Server. See Configuring the CygNet OPC UA Server for more information about configuring logging.
GenerateCygNetModel.ps1	**C:\ProgramData\Weatherford\CygNetOpcUaServer**	The model builder script that generates the address space. Schedule this file to execute regularly, to pick up changes in your CygNet system. See Maintaining the CygNet OPC UA Address Space for more information.

CygNet OPC UA Server Memory Usage

The following graphs estimate the amount of system memory the CygNet OPC UA Server will allocate when loading the address space on startup and loading active current alarms.

Memory Requirements on Startup

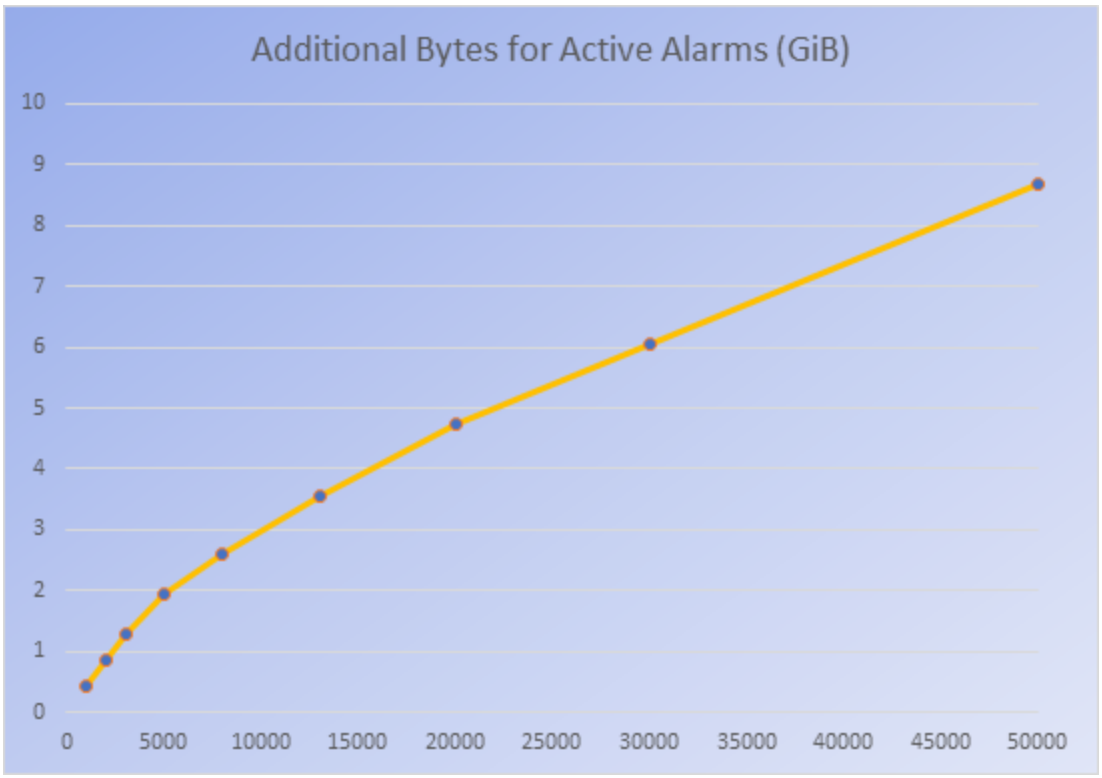
The CygNet OPC UA Server builds its internal node hierarchy on startup and stores these structures in memory. The more CygNet points that are included in this in-memory node hierarchy, the more system memory is required. The following chart estimates the amount of memory the CygNet OPC UA Server will allocate based upon the number of CygNet points included in the node hierarchy.



Private Bytes at Startup
(GiB by number of points)

Memory Requirements for Active Alarms

In addition to the system memory required to load the Address Space on startup, additional system memory will be allocated depending upon how many active CygNet current alarms exist at any given time. The following chart estimates the amount of additional memory the CygNet OPC UA Server will require based upon the number of active CygNet current alarms.



Additional Bytes for Active Alarms
(GiB by number of alarms)

CHAPTER 4: Configuring the CygNet OPC UA Server

This chapter describes how to configure the CygNet OPC UA Server application configuration file, CygNetOpcUaServer.Config.xml, how to build and maintain the OPC UA addess space, describes the extended data types and event types defined by the server, information about the NodeId formats, how to run the server from a command line, information about mapping point state to quality and sub-status values, and a few troubleshooting tips.

In this chapter:

Configuring the CygNet OPC UA Server

After installing the CygNet OPC UA Server application, a configuration file, **CygNetOpcUaServer.Config.xml**, is included in the installation package, and must be configured to your system's specifications. The configuration file must be located in **C:\ProgramData\Weatherford\CygNetOpcUaServer**.

The CygNet OPC UA Server will only pick up any changes to this file on startup, so the server must be restarted anytime a change is made.

See the following subsections for more information:

- Default Configuration — describes elements in the default CygNetOpcUaServer.Config.xml after installation
- Server Configuration — describes elements that can be added to manage subscriptions or monitored items
- Security Configuration — describes elements that can be added to override default certificate locations.

Default Configuration

The following table describes the default elements found in the default CygNetOpcUaServer.Config.xml file.

Element	Description	Required	Default
<ApplicationConfiguration>	The parent element for the configuration file. Specifies the XML namespaces and schemaLocation used by the OPC UA Server. Nested within this element are extended settings specific to the CygNet OPC UA Server.	yes	xmlns:ua-a="http://opcfoundation.org/UA/2008/02/Types.xsd" xmlns-="h-ttp://opcfoundation.org/UA/SDK/Configuration.xsd" schem-aLocation="./Schema/ApplicationConfiguration.xsd"
<ServerConfiguration>	The parent element for general server configuration information for the application. Nested within this element are other elements that define the server settings for the CygNet OPC UA Server. See ServerConfiguration below.	yes	

Element	Description	Required	Default
<BaseAddresses>	The parent element for the base address URL used by the CygNet OPC UA Server.	no	
<ua:String>	Specifies the base address (URL) for accessing the CygNet OPC UA server.	yes, if <BaseAddresses> is included	opc.tcp://localhost:51210/CygNetOpcUaServer "localhost" restricts client access to a client running on the server host. To allow access from a remote client machine, modify this setting to specify the hostname and/or port, for example, opc.tcp://opcuahost:51211/CygNetOpcUaServer
<Extensions>	The parent element for the extended settings specific to the CygNet OPC UA Server application. Nested within this element are other elements that define CygNet configuration settings, logging settings, and model builder configuration settings for the CygNet OPC UA Server application.	yes	
<ua:XmlElement>	Contains the extended XmlElements specific to the CygNet configuration.	yes	Numerous XmlElement sets are supported.
<CygNetConfiguration>	The parent element for the custom CygNet configuration settings, including the Bridge connection information. Specifies the XML namespace used by the CygNet OPC UA Server.	yes	xmlns="https://weatherford.com/UA/cygnet/"

Element	Description	Required	Default
<CygNetBridgeBaseUrl>	Specifies the URL of the installed CygNet Bridge application.	yes	http://localhost
<CygNetBridgeUsername>	Certain operations of the CygNet OPC UA Server require that the server can log into CygNet Bridge using default credentials. **CygNetBridgeUsername** specifies the "default" username that will be used in those instances. The username specified will be supplied to Bridge when generating CygNet nodes, when a user subscribes to real-time and alarm notifications, and when a user authenticates with a certificate rather than a username and password. See CygNet Bridge API Access for more information.	yes	bridgeuser
<CygNetBridgePassword>	Specifies the unencrypted password used along with **CygNetBridgeUsername** so that the server can log into Bridge using default credentials.	yes	bridgepwd

Element	Description	Required	Default
<CygNetBridgePass-wordEncrypted>	Specifies the encrypted password used along with **CygNetBridgeUsername** so that the server can log into Bridge using default credentials. The CygNet OPC UA Server can use either **CygNetBridgePass-wordEncrypted** or **CygNetBridgePassword**. **CygNetBridgePass-wordEncrypted** will replace **CygNetBridgePassword** once the server is started with the "-encrypt" parameter. See CygNet OPC UA Server Password Encryp-tion for information about encrypt-ing the **CygNetBridgePassword**.	no	
<CygNetEncryptionKeyPath>	Specifies the path to the encryp-tion key. When the encryption routine is executed to encrypt **CygNetBridgePassword**, the process will generate a file that decrypts **CygNetBridgePass-wordEncrypted** internally and on an as-needed basis. See CygNet OPC UA Server Pass-word Encryption for information about encrypting the **CygNetBridgePassword**.	no	

Element	Description	Required	Default
<CygNetModelFilePath>	Specifies the path to the **CygNetModel.json** file, the predefined CygNet node set binary file, which encapsulates the address space definition and is the output from the model builder script. The CygNet OPC UA Server uses the **CygNetModel.json** file on startup to populate its node hierarchy.	no	%CommonApplicationData%\Weatherford\CygNetOpcUaServer\CygNetModel.json **Note:** The %CommonApplicationData% token will be resolved by the server at run time.
<ua:XmlElement>	Contains the extended XmlElements specific to the logging configuration.	no	Numerous XmlElement sets are supported.
<LoggingConfiguration>	The parent element for the custom CygNet logging configuration settings. Specifies the XML namespace used by the OPC UA Server.	no	xmlns="https://weatherford.com/UA/cygnet/"
<FilePath>	The path to the log file(s) generated by the server.	no	%CommonApplicationData%\Weatherford\CygNetOpcUaServer\Logs\CygNetOpcUaServer.log **Note:** The %CommonApplicationData% token will be resolved by the server at run time.
<FileSizeLimitBytes>	The maximum size, in bytes, that the log file can grow to before the server creates a new log file (in addition to the original one, not in place of it).	no	52428800

Element	Description	Required	Default
<RetainedFileCountLimit>	The maximum number of log files that will be written to before the server begins to overwrite the earliest ones.	no	5

Element	Description	Required	Default
<MinimumLevel>	The minimum level of logging detail that will be logged to the log file. The options are: • **Fatal** — To see only messages related to a fatal termination of the server, use **Fatal**. • **Error** — To see everything prior, as well as any non-fatal errors, use **Error**. • **Warning** — To see everything prior, as well as warnings, use **Warning**. • **Information** — To see everything prior, as well as useful but non-critical information about the operation of the server, use **Information**. This is the default setting. • **Debug** — To see everything prior, as well as diagnostic messages to help identify the source of problems, use **Debug**. • **Verbose** — To see every possible message about the server, use **Verbose**.	no	Information

Element	Description	Required	Default
<ua:XmlElement>	Contains the extended XmlElements specific to the model builder configuration.	yes	Numerous XmlElement sets are supported.
<ModelBuilderConfiguration>	The parent element for the custom model builder configuration settings. The model builder is a mechanism for generating the address space for the CygNet system, which is run when you first set up the CygNet OPC UA Server, and when you make changes to CygNet that you want exposed in OPC UA (such as adding or removing points or facilities, changes to metadata, etc.). Specifies the XML namespace used by the server. See Maintaining the CygNet OPC UA Address Space for more information. Specifies the XML namespace used by the OPC UA Server.	yes	xmlns="https://weatherford.com/UA/cygnet/"
<CygNetServices>	Specifies a comma-separated list of CygNet current value services (CVS) in the format [DOMAIN] SITE.SERVICE that will be used to generate the **CygNetModel.json** file when the **GenerateCygNetModel.ps1** script is run.	yes	[00000]CYGNET.UIS,[00000]CYGNET.HSS

Element	Description	Required	Default
<DisplayNameProperties>	The parent element for configuring the display names of each nodes in the address space. Contains three child elements for each configurable node property: **<Cvs>**, **<Facility>**, and **<Point>**. **Note:** The entire **DisplayNameProperties** element and any or all of the three child elements may be left out of the configuration file. If missing, the default values for the child elements will be: • **Cvs** — DomainSiteService • **Facility** — Description • **Point** — Description	no	

Element	Description	Required	Default
<Cvs>	Specifies the display name to be used for the Current Value Service (CVS) node. Possible values for the **<Cvs>** element are: • DomainSiteService • SiteService • Description **Note:** The **<Cvs>** element is not required. If missing, the default value for the element will be *DomainSiteService*. Also see the note about blank attributes below this table.	no	DomainSiteService

Element	Description	Required	Default
<Facility>	Specifies the display name to be used for the Facility node. Possible values for the **<Facility>** element are: • FacilityId • FacilityDescription • Description • FacilityInfoAttribute0 • FacilityInfoAttribute1 • FacilityTextAttribute00 — FacilityTextAttribute39 • FacilityTableAttribute00 — FacilityTableAttribute59 • FacilityTableAttribute00Description — FacilityTableAttribute59Description • FacilityTag **Note:** The **<Facility>** element is not required. If missing, the default value for the element will be *Description*. Also see the note about blank attributes below this table.	no	Description

Element	Description	Required	Default
<Point>	Specifies the display name to be used for the Point node. Possible values for the **<Point>** element are: • Description • Tag • TagLong • TagFull • PointId • PointIdLong • UDC • UDCDescription • LongDescription • SystemDescription • PointDataTypeDescription • AlarmCategoryDescription • Comment • Indexed1 • Indexed2 • Indexed3 • TableDriven1 • TableDriven2 • TableDriven3 • TableDriven1Description • TableDriven2Description • TableDriven3Description • GeneralData1 • GeneralData2 • GeneralData3 • ExternalId **Note:** The **<Point>** element is not required. If missing, the default value for the element will be *Description*. Also see the note about blank attributes below this table.	no	Description

Blank Display Name Attributes

Note that there is another kind of "default" for the **<Cvs>, <Facility>, and <Point>** elements. If the value resolved by the CygNet service is blank for whatever attribute is chosen (for example, if you choose *Description* for **Facility**, but a facility's description is blank), then the following defaults will be used:

- **Cvs** — DomainSiteService
- **Facility** — FacilityId
- **Point** — UDC

This is because none of these attributes above can ever be blank in a CygNet system, so this guarantees that nodes will never have a blank display name.

Server Configuration

The **<ServerConfiguration>** section of the CygNetOpcUaServer.Config.xml defines general server information for the CygNet OPC UA Server application. This section appears in the default CygNetOpcUaServer.Config.xml file, but can be supplemented with the following elements to override default server settings.

When subscribing to large numbers of points, it is recommended to use a single Subscription for multiple MonitoredItems. We recommend between 5,000 and 10,000 MonitoredItems per subscription. If you run into issues with a large number of Subscriptions or MonitoredItems, you can increase one or more of the following parameters by adding these elements to your CygNet OPC UA Server configuration file. See Troubleshooting the CygNet OPC UA Server for more information about Subscriptions and MonitoredItems.

To configure server settings

1. Open the CygNetOpcUaServer.Config.xml in a text or xml editor.
2. Copy the three elements between the **<ServerConfiguration>** parent elements and paste into the **<ServerConfiguration>** section of the CygNetOpcUaServer.Config.xml. The elements are described in the table below.
3. Edit the elements as necessary.
4. Restart the CygNet OPC UA Server.

ServerConfiguration XML

Add the following XML to your CygNetOpcUaServer.Config.xml to set the maximum message queue size, notifications per publish, and subscription count for the server.

```
<ServerConfiguration>
     <MaxMessageQueueSize>10000</MaxMessageQueueSize>
     <MaxNotificationsPerPublish>1000</MaxNotificationsPerPublish>
     <MaxSubscriptionCount>100</MaxSubscriptionCount>
</ServerConfiguration>
```

ServerConfiguration Elements

The following table describes the **<ServerConfiguration>** elements that can be added to the CygNetOpcUaServer.Config.xml file.

Element	Description	Required	Default
<ServerConfiguration>	The parent element for general server configuration information for the application. Nested within this element are other elements that define the server settings for the CygNet OPC UA Server.	yes	
<MaxMessageQueueSize>	The maximum number of messages in the internal queue in the CygNet OPC UA Server. If this limit is reached, the server will drop messages, and they won't be delivered to the client. This will be logged in the log file with a message such as, "WARNING: QUEUE OVERFLOW. Dropping <X> Messages. Increase MaxMessageQueueSize. SubId=<id>, MaxMessageQueueSize=10000"	no	10,000
<MaxNotificationsPerPublish>	The maximum number of notifications in each publish from the CygNet OPC UA Server. Things that can affect this are the total number of notifications being processed by the server, and the Publishing Interval specified by the client when creating the Subscription.	no	1,000
<MaxSubscriptionCount>	The maximum allowed number of subscriptions.	no	100

Security Configuration

The **<SecurityConfiguration>** section of the CygNetOpcUaServer.Config.xml defines certificate information for the CygNet OPC UA Server application. This section does not appear in the default CygNetOpcUaServer.Config.xml file, but can be added to override default certificate locations.

To override the default certificate locations

1. Open the CygNetOpcUaServer.Config.xml in a text or xml editor.
2. Copy the following xml and paste into the CygNetOpcUaServer.Config.xml as a child of **<ApplicationConfiguration>** element (a sibling of **<Extensions>** element). The **<SecurityConfiguration>** elements are described in the table below.
3. Edit the elements as necessary.
4. Restart the CygNet OPC UA Server.

SecurityConfiguration XML

Add the following XML block to your CygNetOpcUaServer.Config.xml to override the certificate locations.

```
<SecurityConfiguration>
     <!-- Where the application instance certificate is stored (MachineDefault) -->
     <ApplicationCertificate>
          <StoreType>X509Store</StoreType>
          <StorePath>CurrentUser\My</StorePath>
          <SubjectName>CN=CygNet OPC UA Server, C=US, S=Arizona, O=OPC Foundation,
          DC=localhost</SubjectName>
     </ApplicationCertificate>

     <!-- Where the issuer certificate is stored (certificate authorities) -->
     <TrustedIssuerCertificates>
          <StoreType>Directory</StoreType>
          <StorePath>%CommonApplicationData%/Weatherford/CygNetOpcUaServer/Certificates/Issuer</StorePath>
     </TrustedIssuerCertificates>


     <!-- Where the trust list is stored (UA Applications) -->
     <TrustedPeerCertificates>
          <StoreType>Directory</StoreType>
          <StorePath>%CommonApplicationData%/Weatherford/CygNetOpcUaServer/Certificates/Trusted</StorePath>
     </TrustedPeerCertificates>

     <!-- The directory used to store invalid certificates for later review by the administrator. -->
     <RejectedCertificateStore>
          <StoreType>Directory</StoreType>
          <StorePath>%CommonApplicationData%/Weatherford/CygNetOpcUaServer/Certificates/Rejected</StorePath>
     </RejectedCertificateStore>
</SecurityConfiguration>
```

SecurityConfiguration Elements

The following table describes the **<SecurityConfiguration>** elements that can be added to the CygNetOpcUaServer.Config.xml file.

Element	Description	Required	Default
<SecurityConfiguration>	The parent element for the security configuration for the application. Nested within this element are other elements that define the certificate settings for the CygNet OPC UA Server.	no	
<ApplicationCertificate>	Contains the application certificate storage details. The **<ApplicationCertficate>** element identifies the CygNet OPC UA Server application itself. A certificate will be generated by the application the first time it is run. Alternatively, you can provide a certificate from a Certificate Authority (CA), and use the **<ApplicationCertficate>** element to specify where it resides.	no	

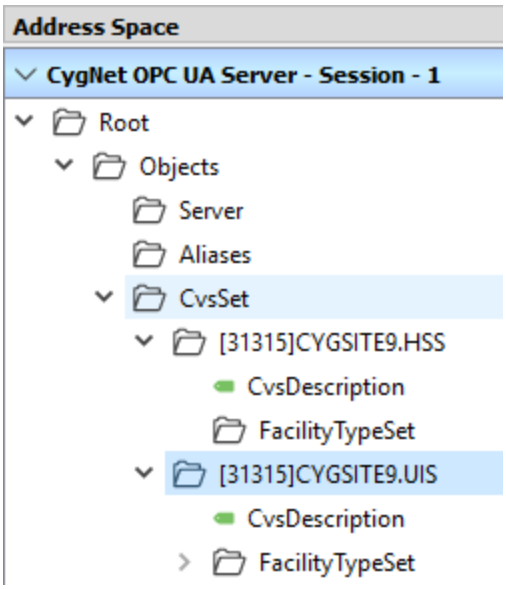
Element	Description	Required	Default
<StoreType>	The type of application certificate.	no	X509Store
<StorePath>	The path to the application certificate for the current user.	no	CurrentUser\My
<SubjectName>	The subject name to use for the application certificate.	no	CN=CygNet OPC UA Server, C=US, S=Arizona, O=OPC Foundation, DC=localhost
<TrustedIssuerCertificates>	Contains a list of issuer certificates that can be trusted by the server.	no	
<StoreType>	The storage type for the trusted issuer certificate.	no	Directory
<StorePath>	The directory where the trusted issuer certificate is stored.	no	%CommonApplicationData%/Weatherford/CygNetOpcUaServer/Certificates/Issuer>
<TrustedPeerCertificates>	Contains a list of peer certificates that can be trusted by the server.	no	
<StoreType>	The storage type for the trusted peer certificate.	no	Directory
<StorePath>	The directory where the trusted peer certificate is stored.	no	%CommonApplicationData%/Weatherford/CygNetOpcUaServer/Certificates/Trusted

Element	Description	Required	Default
<RejectedCertificateStore>	Contains a list of rejected certificates. Invalid certificates will be copied to the specified location and should be reviewed by the system administrator at some later time.	no	
<StoreType>	The storage type for the rejected certificate.	no	Directory
<StorePath>	The directory where the rejected certificates are stored.	no	%CommonApplicationData%/Weatherford/CygNetOpcUaServer/Certificates/Rejected

Maintaining the CygNet OPC UA Address Space

The address space is the core of an OPC UA server and is a representative model of the data the server exposes. The collection of objects and related information that an OPC UA server makes available to an OPC UA client is referred to as its address space. The address space defines the internal node and folder hierarchical structure, the access-rights for each node, how the data for each node will be provided, and display names for each node. An OPC UA server exchanges data in a standardized way with OPC UA clients via an address space model.

The content of the address space is determined by a set of selected information models, which control what nodes are created, the relationship between nodes, and the value of selected attributes of the nodes.



OPC UA Server Address Space

Run the Model Builder Script

The CygNet OPC UA Server needs a CygNet Address Space to define the CygNet data to be shared. This is generated via a special model builder script, which is included in CygNet OPC UA setup program and is installed along with the CygNet OPC UA Server application. This script must be executed to generate the CygNet address space for the CygNet system at the following times:

- when the CygNet OPC UA Server is first set up, and then
- every time changes are made to the CygNet system that must be exposed in OPC UA (such as adding or removing points or facilities, changes to metadata, etc.).

To create the CygNet Address Space

The node creation process works in the following way:

1. Modify the **CygNetOpcUaServer.Config.xml** file with the appropriate <ModelBuilderConfiguration> settings.

   > **Note:**
   > This step only needs to be performed the *first* time you build the address space, and then again only if you want the OPC UA Server to reference a different set of CygNet services, or if you want to change which CVS, Facility, or Point attributes are used to create display names for their nodes.
   > This step does *not* need to be completed each time you want your OPC UA Server to reflect changes to your CygNet services.

2. Ensure the CygNet Bridge website is started using IIS Manager.

3. Run the **GenerateCygNetModel.ps1** script. The model builder script generates a **CygNetModel.json** file, which encapsulates the CygNet Address Space. This file is used by the OPC UA Server and should never be modified manually. The script is installed here: **C:\ProgramData\Weatherford\CygNetOpcUaServer\GenerateCygNetModel.ps1**
   - Right-click the script and select **Run with Powershell** to execute the script.
   - Or, start Windows PowerShell and navigate to the folder where the script file is located and execute the script. Use this option if you want to monitor the build process.
   - Once complete the **CygNetModel.json** file will be written to the same folder.

4. Restart the OPC UA Server.

> **Note:**
> The amount of time it takes to generate the **CygNetModel.json** file, which defines the address space, depends on the size of your CygNet system.

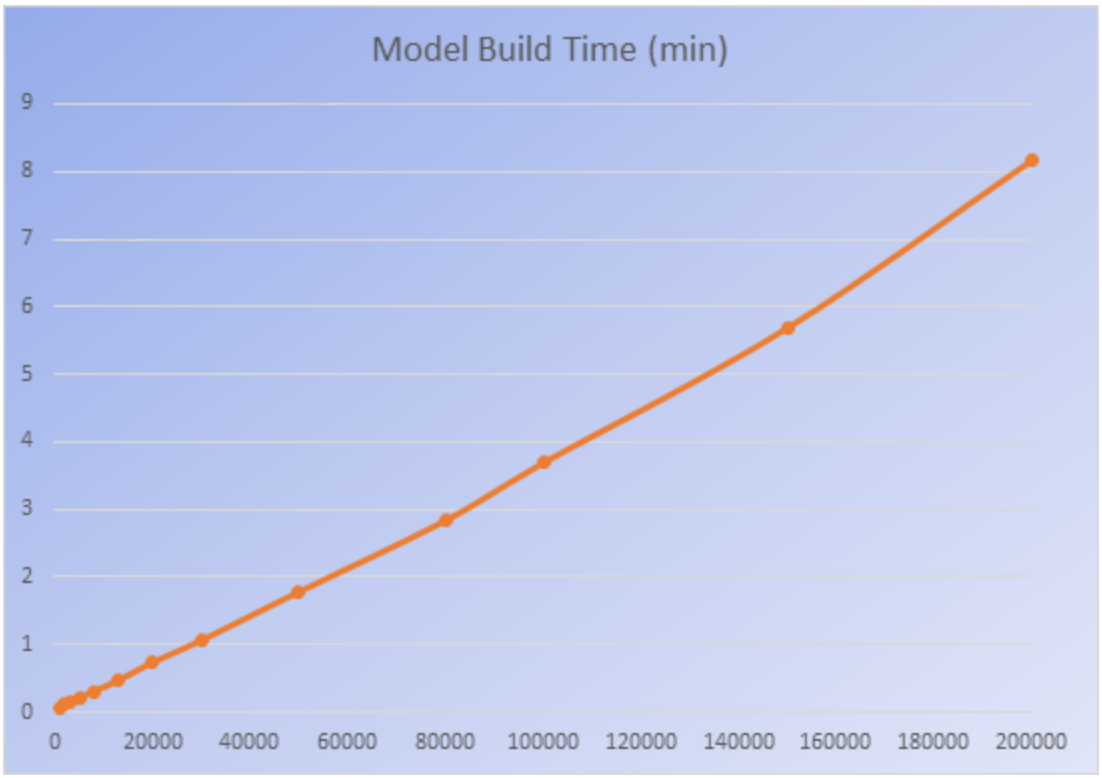
Schedule the Model Builder Script

The **GenerateCygNetModel.ps1** script can either be run manually with Windows PowerShell, or it can be scheduled to run as a task with the Windows Task Scheduler.

The frequency with which the script is run is dependent on the requirements of your enterprise, but it is important to keep the CygNet OPC UA Server in sync with any changes made to the CygNet system (e.g., new and modified points and facilities, updated metadata, etc.), so best practice recommends that the script be run whenever a change is made that would need to be reflected in the OPC UA server.

Time to Rebuild the CygNet Model

Generating a new CygNet model will take more time as the number of points in your system increases. The following chart estimates the time in minutes that the CygNet model will take to rebuild based upon the number of CygNet points included in the model. Your results will vary depending on attributes of your host machine and network environment such as CPU, memory, and network latency.



CygNet Model Build Time
(minutes by number of points)

Extended Data Types and Event Types

The following extended data types and event types are defined by the CygNet OPC UA Server and found in the address space:

- DataTypes
  - CvsType
  - FacilityType
  - PointConfigurationType
  - RealtimeRecordType
- EventTypes
  - CygNetAlarmType

See NodeId Formats for more information about how NodeIds are formatted in the CygNet OPC UA server's address space.

The supported extended types and their properties are listed in the following tables:

CvsType

The following CVS description property is defined in the CygNet OPC UA address space.

Data Type	Description	Type
CvsDescription	The name of the Current Value Service(s) (CVS).	string

FacilityType

The following facility attributes are defined in the CygNet OPC UA address space.

Data Type	Description	Type
FacilityCategory	The category type of facility, e.g., COMMDEV, DDSFAC, EXPDEV, GENFAC, REMDEV, SERVICE, etc. Generated by the Facility Service. Defined in the SYSFCCAT table.	string
FacilityDescription	The description of the facility.	string
FacilityId	The facility identifier of the point.	string
FacilityInfoAttribute0	The facility text attribute (Facility Info 0). The default name is "Facility Location".	string
FacilityInfoAttribute1	The facility text attribute (Facility Info 1). The default name is "Facility Contact".	string
FacilityIsActive	Indicates whether the facility record is active.	boolean
FacilitySecurityApplication	The facility security application.	string
FacilityService	The name of the facility's current value service.	string
FacilitySite	The name of the facility's site.	string

Data Type	Description	Type
FacilitySiteService	The name of the facility's combined site and service, in the format SiteService.	string
FacilityTableAttribute00 - FacilityTableAttribute59	The facility table attributes (Facility Table 00 through Facility Table 59).	string
FacilityTableAttribute00Description - FacilityTableAttribute59Description	The facility table description attributes (Facility Table 00 Desc through Facility Table 59 Desc).	string
FacilityTag	The tag of the facility in the format, Site.Service::FacilityId.	string
FacilityTextAttribute00 - FacilityTextAttribute39	The facility text attributes (Facility Text 00 through Facility Text 39).	string
FacilityType	The facility type, e.g., CDP, METER, PUMP, TANK, WELL, etc. Defined in the SYSFCTYP table.	string
FacilityYesNoAttribute00 - FacilityYesNoAttribute19	The facility Yes/No attributes (Facility Yes/No 0 through Facility Yes/No 19).	boolean

PointConfigurationType

The following point configuration record properties are defined in the CygNet OPC UA address space.

Data Type	Description	Type
AlarmCategory	The alarm category for the point. Defined in the PNTALCAT table.	string
AlarmCategoryDescription	The description of the alarm category for the point. Defined in the PNTALCAT table.	string
AlternateUnits	The alternate units set for the point.	string
AnyUserFlag	An internal flag that indicates that at least one User Flag in the point record has been set.	string
AnyVerifiedFlag	An internal flag that indicates that at least one Verified Flag in the point record has been set.	string
Comment	The comment set for the point.	string
ExternalId	The external identifier used for an external point.	string
Facility	The point's facility.	string
GeneralData1 - GeneralData3	The general data field(s) (General 1, General 2, or General 3) set for the point.	string
Indexed1 - Indexed3	The indexed field(s) (Indexed 1, Indexed 2, or Indexed 3) set for the point.	string
LongDescription	The long free-form description of the point.	string
PointDataType	The data type of the point, e.g., Analog Input, Digital Output, Enumeration Input, String Output.	string
PointDataTypeDescription	The description of the point data type.	string
PointDescription	The description of the point.	string

Data Type	Description	Type
PointId	The identifier of the point.	string
PointIdLong	The long point identifier for the point.	string
PointScheme	The point scheme used for the current CygNet system, which governs system-wide alarm priorities, alarm categories, alarm colors, point states, etc.	string
PointSchemeDescription	The description of the point scheme, e.g., CygNet Standard, CygNet enhanced, etc.	string
QuestionableFlag	The questionable flag set for the point, indicating the data is questionable.	string
QuestionableTimestamp	The timestamp of the last questionable flag change.	string
Service	The name of the CygNet service.	string
Site	The name of the CygNet site.	string
SiteService	The name of the CygNet site and services in the format SiteService.	string
System	The name of the CygNet system.	string
SystemDescription	The description of the CygNet system.	string
TableDriven1 - TableDriven3	The table driven field(s) (Table Driven 1, Table Driven 2, or Table Driven 3) set for the point. Defined in the CMT1, CMT2, and CMT3 tables.	string
TableDriven1Description - TableDriven3Description	The table driven field descriptions (Table Driven 1, Table Driven 2, or Table Driven 3) set for the point. Defined in the CMT1, CMT2, and CMT3 tables.	string
Tag	The tag string identifier for the point in the format Site.Service.PointID.	string
TagFull	The full tag sting identifier for the point in the format Site.Service.PointID:LongID.	string
TagLong	The long tag sting identifier for the point in the format Site.Service:LongID.	string
UDC	The Uniform Data Code (UDC) of the point.	string
UDCDescription	The description that accompanies the actual UDC.	string
Units	The primary units set for the point.	string
UserFlag1 - UserFlag8	The generic user flag(s) (User Flag 1 through User Flag 8) set for the point.	string
VerifiedFlag1- VerifiedFlag4	The verified flag(s) (Verified Flag 1 through Verified Flag 4) set for the point.	string
VerifiedTimestamp	The timestamp of the last verified flag change.	string

RealtimeRecordType

The following real-time record (current value) properties are defined in the CygNet OPC UA address space.

Data Type	Description	Type
AlarmConditionDescription	The description of the alarm condition for the point.	string
AlarmPriorityCategoryDescription	The description of the alarm priority category for the real-time record for the point, e.g. Low, Medium, High, Critical, etc.	string
AlternateValue	The alternate value for the point in the real-time record for the point.	string
PointStateDescription	The description of the current state of the point.	string
Status	The point's status (the status bits in the real-time record).	string
Timestamp	The date and/or time at which the real-time record for the point changed.	DateTime
UserStatus	The point's user status (the user status bits in the real-time record).	string
Value	The current value of the real-time record for the point.	string

CygNetAlarmType

The following alarm record properties are defined in the CygNet OPC UA address space. The **CygNetAlarmType** extends the OPC UA **AlarmConditionType**, and introduces the following alarm record properties.

OPC UA AlarmConditionType is described in OPC 10000-9: OPC UA Specification Part 9: Alarms & Conditions, Section 5.8.2.

These properties map directly to the same-named properties in the AlarmRecordDTO that is returned from CygNet Bridge.

Data Type	Description	Type
AlarmCategory	The alarm category for the point.	string
AlarmCondition	Alarm condition is the same as point state, in that it is the highest precedented state for a point record as defined in the current point scheme, except that in the point state definition for each point state, the alarm condition attribute must be set to "true" for the alarm condition to be considered.	string
AlarmPriority	The alarm priority range for the point.	unsigned integer
AlarmPriorityCategory	The category of the alarm priority range for the point.	unsigned integer
AlarmPriorityCategoryDescription	The description of the category of the alarm priority range for the point.	string
AlarmRecordVersion	The version of the current alarm record.	string

Data Type	Description	Type
HighestAlarmPriority	The highest alarm priority range present.	unsigned integer
HighestAlarmPrioritySinceAck	The highest alarm priority range present since acknowledgment.	unsigned integer
IsHidden	Indicates whether or not the alarm is hidden.	boolean
IsSet	Indicates whether or not the alarm is set.	boolean
PointState	The highest precedented state for a point record as defined in the current point scheme.	string
PointValue	The point value that triggered the alarm.	string

CygNetAlarm

When an alarm exists in the CygNet CAS, an instance of a **CygNetAlarm** will be added to the OPC UA address space as a child of the RealtimeRecord for the point that is in the alarm state. Any changes in the alarm state will cause the instance to be updated, and an event containing the updated CygNetAlarm instance will be published to any subscribed OPC UA clients.

Some of the other inherited properties of the **CygNetAlarm** property are populated in the following way:

CygNetAlarm Property	Value
SourceNode	The NodeId of the corresponding RealTimeRecord.
SourceName	The PointTag of the corresponding CygNet point.
Time	The ReportedTimestamp of the AlarmRecordDTO from Bridge.
ReceiveTime	The Timestamp that the OPC UA server received the new alarm state from Bridge.
Retain	True indicates that the alarm is currently in the CygNet CAS. False if it has been cleared.
Message	A message string containing the PointTag and AlarmCondition string.
AckedState	A boolean flag indicating if the alarm has been acknowledged.
ActiveState	A boolean flag indicating if the point is currently in alarm (not in the normal, black state in CAS).
Severity	The OPC UA severity of the alarm (1 – 1000), which is calculated as CygNet AlarmPriority * 10.
SuppressedOrShelved	A boolean flag will be set to true if the CygNetAlarm is in the suppressed state.

When an alarm in CygNet is cleared, and therefore removed from the Current Alarm Service (CAS), the OPC UA server will publish a new instance of the CygNetAlarm to subscribed clients. The "Retain" flag will be false on this CygNetAlarm instance, and the client should ignore the other properties indicating the state of the alarm in this instance, since it no longer exists in the CAS, and the state is therefore unknown.

Current Alarm State

If an OPC UA client wants to get the current state of all of the alarms in the CygNet CAS, they can use the **ConditionRefresh** method.

The OPC UA ConditionRefresh Methods is described in OPC 10000-9: OPC UA Specification Part 9: Alarms & Conditions, Section 5.5.7.

When this method is called, the server will publish an instance of **RefreshStartEventType** first, and an instance of **RefreshEndEventType** last, and in between will publish a CygNetAlarm instance for each alarm that currently is in the CygNet CAS.

Typically, a new client that has just connected might use this ConditionRefresh method to get the state of all of the current alarms. After that, they can just get notifications of changes.

CygNet.Opc.Ua.AlarmSample

A sample .NET Framework WPF Windows application, **CygNet.Opc.Ua.AlarmSample**, is available for download on GitHub and demonstrates subscribing to the CygNet OPC UA Server to receive CygNetAlarm notifications. The CygNet.Opc.Ua.AlarmSample application includes a solution file that was created using MS Visual Studio 2019 Version 16.8.3, targeting .NET Framework 4.8. When you run the application a list of the alarms in your CygNet system is displayed and kept current as changes occur.

See https://github.com/cygnet-software/CygNet.Opc.Ua.AlarmSample. The **CygNet.Opc.Ua.AlarmSample** is also included in the product source files from the **CygNet Software Download Website** on the Weatherford software support portal.

CygNet OPC UA Server NodeId Formats

This topic describes the formats of the various NodeIds that are part of the CygNet address space in the OPC UA server.

The structure of the OPC UA NodeId is described in OPC 10000-3: OPC UA Specification Part 3: Address Space Model, Section 8.2.1.

The NodeId is a standard built-in DataType composed of three elements that identify a Node within a Server. It is defined as having the following three elements:

- **namespaceIndex** — nodes in the CygNet OPC UA Server's address space have a **namespaceIndex** of *2*
- **identifierType** — nodes in the CygNet OPC UA Server's address space have an **identifierType** value of *STRING_1* (String value)
- **identifier** — the **Identifier** element's format varies according to the type of node in the address space as defined in the table below:

Identifier by DataType

DataType	NodeID Format	Example
CvsType	[{Domain}]{Site}.{Service}	[5410]CYGNET.UIS
FacilityType	{FacilityTag}	CYGNET.UIS::BRADY_WL
PointConfigurationType	{Site.Service:LongId}_PointConfiguration	CYGNET.UIS:BRADY_WL_PRCASXIN_PointConfiguration
RealtimeRecordType	{Site.Service:LongId}_RealtimeRecord	CYGNET.UIS:BRADY_WL_PRCASXIN_RealtimeRecord

CygNet OPC UA Server Command Line Interface

The CygNet OPC UA Server can be started in console mode from the command line. You can also encrypt the CygNet Bridge password via the command line.

To start the CygNet OPC UA Server

1. Open a Command Prompt window.
2. Navigate to the directory where the server executable is installed: **C:\Program Files\Weather-ford\CygNetOpcUa\CygNetOpcUaServer**
3. Type the following command:

```
CygNetOpcUaServer -console
```

To encrypt the CygNetBridgePassword

1. Open a Command Prompt window.
2. Navigate to the directory where the server executable is installed: **C:\Program Files\Weather-ford\CygNetOpcUa\CygNetOpcUaServer**
3. Type the following command:

```
CygNetOpcUaServer -encrypt
```

4. The CygNet OPC UA Server will stop after the encryption routine is completed.

See CygNet OPC UA Server Password Encryption for more information.

Mapping CygNet Point States to OPC Quality and Sub-Status

The CygNet OPC Servers provide a sophisticated method for mapping "point state" to OPC quality and sub-status values. The CygNet CvsMetadata XML file provides default mappings from the defined "point states" and "point state instances" for Point Scheme 0 to the OPC quality and sub-status values as follows.

CygNet Scheme 0 Point State	Quality and Sub-status Bytes
NORMAL	Hex 0xC0, decimal 192, OPC Quality = "Good"
UNRELIABLE	Hex 0x44, decimal 68, OPC Quality = "Uncertain", Sub-status = "Last Usable Value"
UNINITIALIZED	Hex 0x08, decimal 8, OPC Quality = "Bad", Sub-status = "Not Connected"

Note: When data is retrieved from the CygNet OPC HDA server the OPC quality property is also returned, appended with a sub-status of OPCHDA_RAW (if timestamp is exact) or OPCHDA_INTERPOLATED (if timestamp is interpolated).

See Mapping CygNet Point States to OPC UA Status Codes for more information about how this is handled in the CygNet OPC UA Server.

CVS Metadata Mapping

Point Scheme definitions can be modified to customize the mapping of the user-defined "point states" and "point state instances" to the desired OPC quality and sub-status values. The CygNet OPC Servers load the CVS Metadata XML and resolve the appropriate OPC quality and sub-status values based on the highest precedence "point state."

See the following **CvsMetadata** elements in the **Points** section of the CygNet Help for more information:

- **PointStateDefinition** attribute: **ext_status**
- **PointStateInstanceDefinition** attribute: **ovrext_status**

The CygNet OPC Servers will also set the OPC quality and sub-status values, regardless of resolved point state, based on the following conditions.

OPC Server	Quality and Sub-status Byte
Unknown tag	Hex 0x01, decimal 1, OPC Quality = "Bad", Sub-status = "Configuration error"
Unknown Point Scheme	Hex 0x01, decimal 1, OPC Quality = "Bad", Sub-status = "Configuration error"
Failed to connect to CygNet	Hex 0x18, decimal 24, OPC Quality = "Bad", Sub-status = "Comm. Failure"
CygNet message response error	Hex 0x18, decimal 24, OPC Quality = "Bad", Sub-status = "Comm. Failure"

If the CygNet CVS Metadata XML file fails to load when the CygNet OPC Server starts the following will occur:

- A warning is added to the log file and displayed on the status bar of the **CygNet OPC Server** dialog box.
- The CygNet OPC Server will continue to try to load the CVS Metadata until the metadata no longer fails to load.
- The CygNet OPC Server will revert back to the old OPC server behavior, described below:

OPC Server	Quality and Sub-status Byte
Success	Hex 0xC0, decimal 192, OPC Quality = "Good"
Unknown tag	Hex 0x01, decimal 1, OPC Quality = "Bad", Sub-status = "Configuration error"
Failed to connect to CygNet	Hex 0x18, decimal 24, OPC Quality = "Bad", Sub-status = "Comm. Failure"
CygNet message response error	Hex 0x18, decimal 24, OPC Quality = "Bad", Sub-status = "Comm. Failure"

On Success the OPC servers will return "Good" instead of mapping the point state to an OPC quality and sub-status value.

Troubleshooting the CygNet OPC UA Server

The following tips might be helpful in solving issues that may arise when using the CygNet OPC UA Server.

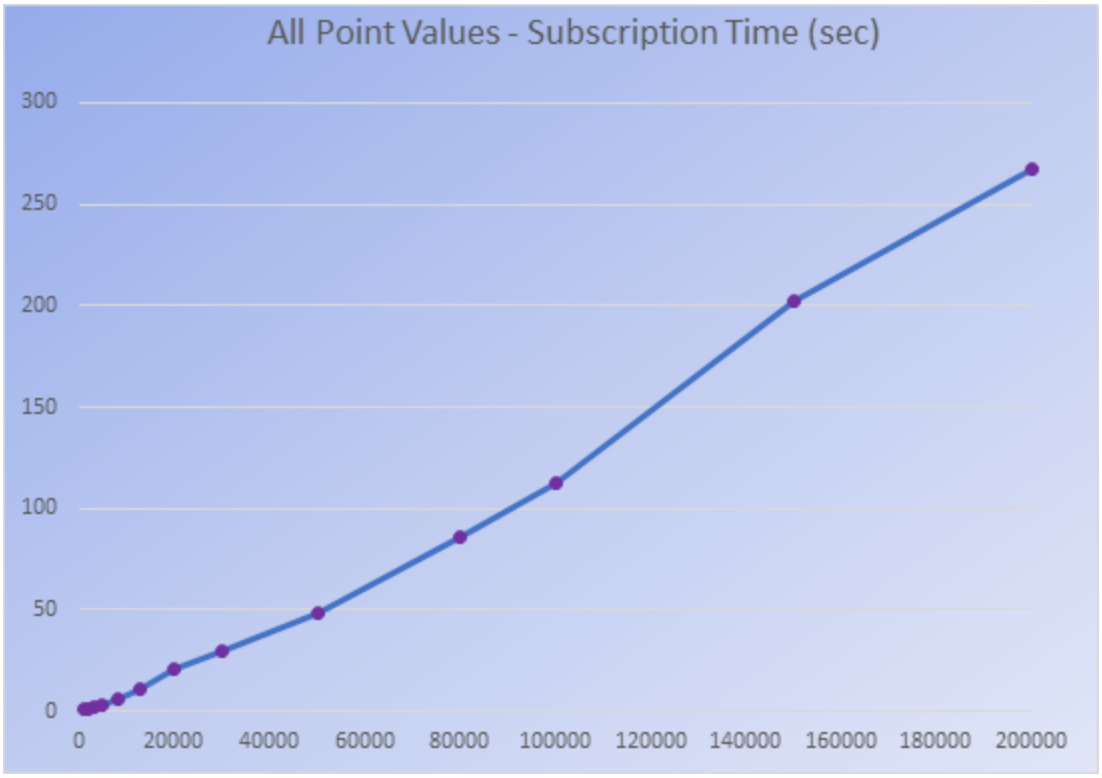
Subscription Considerations

When subscribing to large numbers of points, it is recommended to use a single Subscription for multiple MonitoredItems. We recommend between 5,000 and 10,000 MonitoredItems per subscription. If you run into issues with a large number of Subscriptions or MonitoredItems, you can increase one or more of the following parameters by adding these elements to your CygNet OPC UA Server configuration file. See Server Configuration for more information about how to do this.

- <MaxMessageQueueSize>
- <MaxNotificationsPerPublish>
- <MaxSubscriptionCount>

Subscription Process Time

The more CygNet points that you subscribe to, the longer the full subscription process will take to complete. Below is a chart that estimates the time, in seconds, that the full subscription process will take based upon the number of CygNet points being subscribed to. The measured times include the retrieval and publishing of each points current value. Your results will vary depending on attributes of your host machine and network environment such as CPU, memory and network latency.



Subscription Time (seconds by point values)