Oracle Utilities Live Energy Connect

Configuring Oracle Utilities Live Energy Connect as an OPC UA/ICCP Front End (IFE) to an OEM Application

Release 7.1.0.0.0

F43262-01

May 2021

Oracle Utilities Live Energy Connect Configuring Oracle Utilities Live Energy Connect as an OPC UA/ICCP Front End (IFE) to an OEM Application, Release 7.1.0.0.0

F43262-01

Contents

Preface

This guide is intended for those who would like to create an ICCP Front End (IFE) to an OPC UA Client-Server configuration using the Oracle Utilities Live Energy Connect product. If you need an IFE, you might want to familiarize yourself with ICCP before reading other parts of the guide.

> **Note**: The Oracle Utilities Live Energy Connect product was formerly known as the LiveData Utilities RTI Server Platform.

This preface is designed to give some background information for those who have had little exposure to the Inter-Control Center Communications Protocol (ICCP). ICCP is an international standard which is known as International Electrotechnical Commission 60870-6 (IEC)/Telecontrol Application Service Element 2 (TASE.2). ICCP provides data exchange over wide area networks (WANs) between utility control centers, utilities, power pools, regional control centers, and non-utility generators.

ICCP exchanges data by using three different mechanisms: transfer sets/information reports, writes, and event notification. The blocks of data are categorized into eight distinct types of data. These eight distinct blocks are described in detail in ICCP Service Modeling.

LEC Server supports Blocks 1, 2, 4, 5, and 7.

Block 1 and 2

Provide for the transfer of measurement and state information from an ICCP server to an ICCP client. Block 1 allows for the hosting of variables on the server, and Block 2 is for monitoring changes to these variables. Blocks 1 and 2 are exchanged by the creation of a transfer set composed of a collection of variables. The information in transfer sets is transferred to the client by way of an information report.

Block 4

Provides for the transfer of a buffer of binary bytes from an ICCP server to an ICCP client. Therefore, you can use Block 4 to pass any kind of data, numeric, text, or binary bytes; however, in this application, Block 4 carries mostly informational text messages. Like Blocks 1 and 2, Block 4 uses the transfer set mechanism. In this IFE, customers use Block 4 to encapsulate messages defined by a different protocol OPC UA.

Block 5

Provides for the writing of control information from an ICCP client to an ICCP server. Block 5 uses the write mechanism. It allows a remote ICCP client[1] to control events at devices that are connected to the ICCP server, such as switching an OEM's OPC UA server device on or off. Block 5 also supports setting, clearing, and getting tags. An OEM's OPC UA client can set or

[1] Remote ICCP client refers to a SCADA device that is not part of an OEM's client-server application, nor does it exist within the LEC Server instance.

clear a tag in the local OPC UA server[2] by writing to the tag variable. Similarly, an OEM's OPC UA client can monitor a tag in the local OPC UA server by reading the tag variable.

Block 7

Reports completion status for a Block 5 device control. Block 7 does this by using the mechanism of Event Notification.

[2] Local OPC UA server refers to an OPC UA server within the LEC Server instance, also known as an IFE when it is configured as an OPC UA/ICCP Front End.

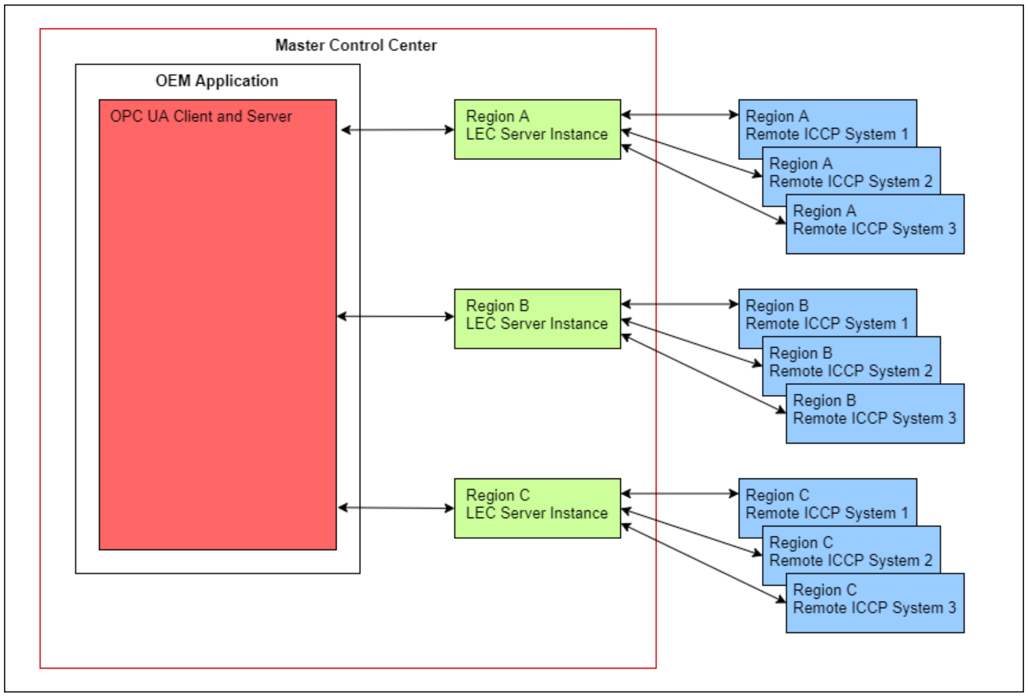
System Overview

Oracle Utilities engineers can configure LEC Server as an ICCP front end (IFE) to OEM applications, such as Distributed Energy Resource Management Systems (DERMS), Energy Management Systems (EMS), for example, ABB's Ability Network Manager EMS, Outage Management Systems (OMS), and the Industrial Internet of Things (IIOT). Throughout this guide, the OEM application is often referred to as "the application" or "an application."

When Oracle Utilities LEC Server is configured as an IFE, LEC Server connects the application to remote ICCP SCADA systems; this allows the application's OPC UA client to read data from and write data to remote ICCP points. LEC Server converts ICCP data and controls to be compatible with OPC UA; conversely, LEC Server converts OPC UA data and controls to be compatible with ICCP. OPC UA clients will be able to write controls to remote ICCP systems that act upon devices in the field, find out whether the control directives were carried out successfully, receive telemetry data from the remote systems, and subscribe to data that alerts the OPC UA client of changes in the field. For each of the OEM's OPC UA client-server instances, there are one or more LEC Server instances that provide the OPC UA instances with bidirectional communication to remote ICCP devices in different regions.

The OEM application hosts the OPC UA clients and servers. Figure 1 provides an overview of how one OPC UA client and server instance within the SCADA system connects to many LEC Server instances, and how each of these instances connects to many remote ICCP devices; this illustrates how each LEC Server instance serves as a bridge between the OPC UA client-server instance and the remote ICCP devices in each region.

Figure 1: OEM Application, LEC Server Instances, and Remote ICCP Systems in Three Different Regions



Both the OEM's OPC UA client-server application and the LEC Server instances have redundant servers at the Main Control Center so that if one instance fails, the system can fail over to the redundant application on another machine.

System Limits and Other Considerations

LEC OPC UA Limits

The following are limitations of local OPC UA clients[3], servers[4], and subscriptions[5] within an LEC Server instance.

- Local OPC UA client supports up to 1024 DataValue/DeviceControl variables that connect to the same application's OPC UA server point. Each of these variables can be seen as an alias for an OPC UA server point.
- The default maximum number of points per subscription allowed by the local OPC UA agent is 10,000. However, this number is configurable within the OpcUaAgent template in LEC Configuration Manager or by setting the number in a batch file. In LEC Configuration Manager, the **Points per subscription** property allows you to adjust this number.
- The default maximum number of subscriptions per connection is 100. However, this number is configurable within the OpcUaAgent template in LEC Configuration Manager or by batch file. In LEC Configuration Manager, the **Maximum number of subscriptions** property allows you to adjust this number.
- The default maximum number of sessions per connection is 10. However, this number is configurable within the OpcUaAgent template in LEC Configuration Manager or by batch file. In LEC Configuration Manager, the **Maximum total subscriptions** property allows you to adjust this number.

LEC Server Limits

While there are not many actual LEC Server limits, Oracle Utilities engineers have identified the following:

- LEC Server has a limit of 16,383 virtual devices, VMDs[6] and VCCs[7], per LEC Server instance.
- LEC VCCs have a limit of 128 transfer sets per VCC.

---

[3] Local OPC UA client refers to an OPC UA client within the LEC Server instance, also known as an IFE when it is configured as an OPC UA/ICCP Front End.

[4] Local OPC UA server refers to an OPC UA server within the LEC Server instance.

[5] Local OPC UA subscription refers to the subscription an OEM's OPC UA client has to points within the local OPC UA server for Block 1 and Block 4.

[6] A **VMD** is a container of nodes. VMD stands for virtual manufacturing device. Each VMD is associated with a specific type of communications protocol or interface. Most VMD types are intended to handle communications in and out of LEC Server using a particular communications protocol or interface, such as ICCP, OPC UA, Modbus, database access, or others. Thus, a VMD can map a device from the outside world to Oracle Utilities LEC's internal variable model, which allows LEC Server to capture, transform, and route data to other devices, systems, or applications in a form that the other device, system, or application can understand. Each instance of a VMD has a network address.

[7] A **VCC** is a Virtual Control Center. It is the type of virtual device that communicates to other devices using the ICCP protocol.

Subscriptions and Associations

When an LEC Server instance is configured as an IFE, it connects an OEM's OPC UA clients to remote ICCP systems so OPC UA clients can subscribe to data from and exercise control over remote ICCP peers' devices. The way in which LEC IFE connects OPC UA clients to remote ICCP systems follows:

1. An application's OPC UA client subscribes to a set of local OPC UA server points. Some of these points are Block 1 points and others are Block 4 points.
2. The local OPC UA server connects to the local ICCP client.
3. For Block 1, the local ICCP client makes associations with remote ICCP SCADA systems, subscribing to remote ICCP server points with the ICCP Transfer Set mechanism. Once an association exists, the OPC UA client can also exercise control over the remote ICCP server by utilizing the ICCP Block 5 write mechanism.
4. For Block 4, the local OPC UA server connects to the local ICCP server. Through that path, the OPC UA client can cause the local ICCP server to send a block 4 message to the remote SCADA ICCP client.

Similarly, the LEC Server instance connects remote ICCP clients to the OEM application's OPC UA servers so that remote ICCP systems can subscribe to Block 1 and Block 4 data from the OEM application's OPC UA servers.

The remote ICCP clients can also exercise control over the OEM application's OPC UA servers using the Block 5 write mechanism. The way in which each LEC IFE connects remote ICCP systems to OPC UA clients follows:

1. Each remote ICCP client can subscribe to a set of local ICCP Block 1 server points using the ICCP Transfer Set mechanism.
2. The local ICCP server[8] connects to the local OPC UA client.
3. The local OPC UA client in turn connects to the OEM application's OPC UA server. Once an association exists, the remote ICCP client can exercise control over the OEM's OPC UA server by utilizing the ICCP Block 5 write mechanism.

LEC Server supports many protocols, but in this configuration it enables OPC UA clients and servers to have connectivity and control over only ICCP devices. In the future, Oracle Utilities expects to use LEC Server to connect to MultiSpeak and IEC 61850 devices.

For information on Block 1 and Block 4 subscriptions, refer to the subsection Blocks 1 and/or 2 Bottom Dataflow and the section Block 4 Subscription. For more information on Block 5, refer to the subsection Block 5 Top Dataflow.

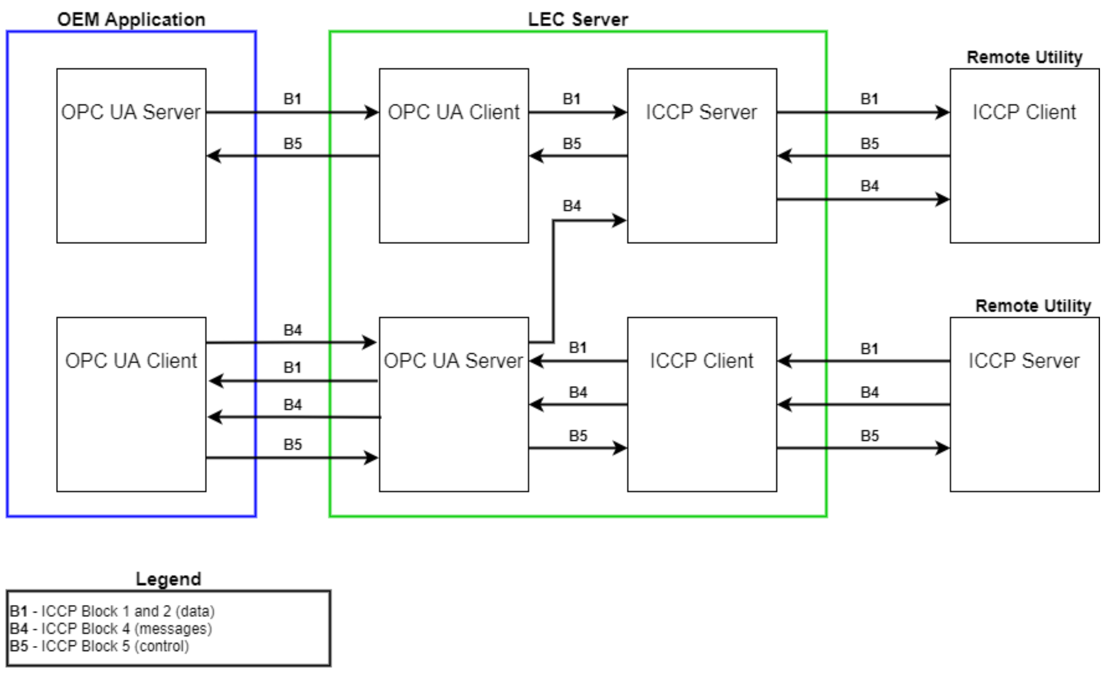
OPC Clients and Servers, LEC Server Instances, and Remote ICCP Peers

LEC Server supports a subset of the ICCP services that are required for compliance with the international ICCP Standard.

[8] Local ICCP server refers to an ICCP server within the LEC Server instance.

Figure 2 illustrates how an OEM application's OPC UA server and OPC UA client connect to remote ICCP devices, using LEC Server to facilitate the OPC UA communication to ICCP devices.

Figure 2: The OEM's OPC UA Client-Server Application, LEC Server Instance, and Remote ICCP Devices



The top dataflow in Figure 2 shows an application's OPC UA server communicating with a remote ICCP client using an LEC Server instance as the ICCP Front End (IFE). This communication transmits both Blocks 1 and 2 and Block 5 data. Blocks 1 and 2 transfer measurement and state information from a local ICCP server to a remote ICCP client. Block 5 provides for the transfer of control information from the remote ICCP client to the local ICCP server.

The lower dataflow in Figure 2 shows an application's OPC UA client communicating with a remote ICCP server using LEC Server as the IFE. This communication allows the OPC UA client to receive Block 1 and/or 2 data as well as Block 4 text messages from the remote ICCP server. It also shows how the OPC UA client can transmits Block 5 control information to the ICCP server.

The IFE implements the transfer of Block 4 messages from the OEM's OPC UA client to the remote ICCP client by passing the Block 4 messages to the local OPC UA server and then to the local ICCP server in the upper dataflow diagram.

Blocks 1 and/or 2 Top Dataflow

1. As the top dataflow in Figure 2 shows, an application's OPC UA server can pass Block 1 (and Block 2) data to a local OPC UA client in an LEC Server instance.
2. LEC Server converts the OPC UA data to ICCP data, and then passes this converted data to an ICCP server in the LEC Server instance.
3. The ICCP server transfers this data to a remote ICCP client device.

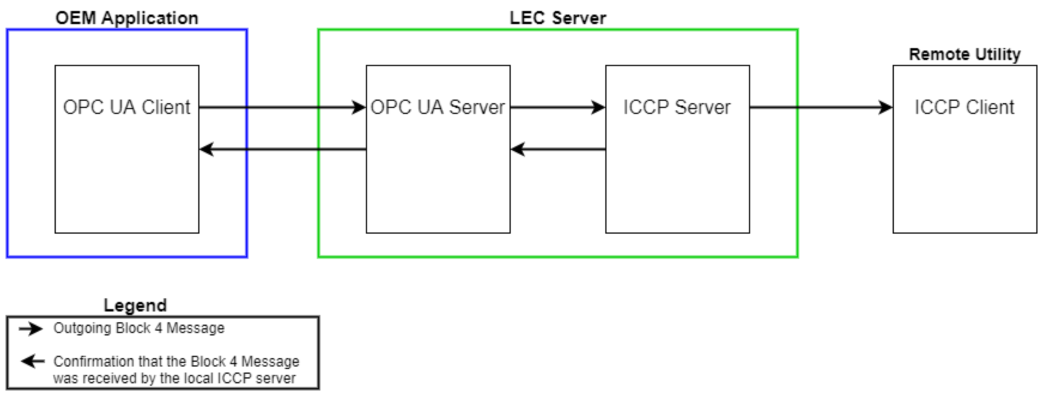
Blocks 1 and/or 2 Bottom Dataflow

1. An application's OPC UA client subscribes to data points on the OPC UA server in the LEC Server instance.
2. LEC Server forwards these requests to corresponding points configured on the ICCP client in the LEC Server instance.
3. The ICCP client requests the creation of a transfer set from the ICCP server, specifying the list of points.
4. Upon receiving the request, the remote ICCP server can push Block 1 and 2 data to the local ICCP client when certain criteria are met. For example, LEC Server can specify that the remote ICCP server push data cyclically after a given period of time or when the data changes or both. LEC Server converts this data into OPC UA data, then passes it to the OPC UA server in the LEC Server.
5. The OPC UA server pushes this data to the external OPC UA client.

Block 4 (ConfirmedMessageFromOpcUa and ConfirmedMessageToIccp)

1. As Figure 2a shows, an application's OPC UA client can pass Block 4 messages to a remote ICCP client. By using a ConfirmedMessageFromOpcUa node in LEC's OPC UA server, the originating OPC UA client can receive confirmation that the local ICCP server received the message.

Figure 2a: OPC UA Client sends a ConfirmedMessageFromOpcUa message to a Remote ICCP Client



2. LEC Server converts the OPC UA data to ICCP data, and then passes this converted data to an ICCP server in the upper dataflow.
3. When the local OPC UA server uses a ConfirmedMessageFromOpcUa node, then the local ICCP server returns confirmation that it received the message to the originating device.
4. This ICCP server also sends the message to the remote SCADA's ICCP client if it is able to do so.

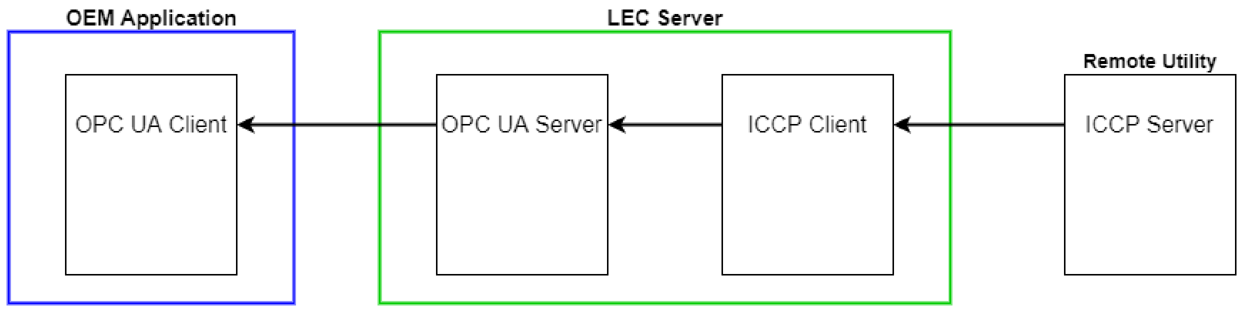
To see the table used to configure Block 4 confirmed messages from OPC UA, see the section called of this document called #ConfirmedBlk4ToIccp. This section describes the nodes that are required to implement the transfer of Block 4 messages from the application's OPC UA server to the remote ICCP client.

Block 4 Subscription

1. An application's OPC UA client subscribes to data points on the local OPC UA server.
2. LEC Server forwards these subscription requests to corresponding points configured on the ICCP client in the LEC Server instance.
3. The ICCP client requests the creation of a transfer set from the ICCP server. Unlike Block 1 and Block 2 transfer sets, a Block 4 transfer set contains no point list; instead, the Block 4 transfer set grants permission to the remote ICCP server to send Block 4 messages.
4. Upon receiving the transfer set, the remote ICCP server can push Block 4 messages to the local ICCP client whenever there is a new message. LEC Server converts this data into OPC UA data, then passes it to the OPC UA server in the LEC Server.

The ICCP Block 4 message is formatted as an InformationBuffer message. The InformationBuffer object provides a unique identifier (InfoReference) and a local identifier (LocalReference). The MessageId identifies the particular instance of a message. Each of these identifiers is a signed 32-bit integer. The Size attribute is the length in octets of the actual data being transferred.

Figure 2b: ICCP Server Pushes Data from Data Points that the OPC UA client has Subscribed to



LEC Server converts the InformationBuffer message into a data type that the local OPC UA server can understand. This data type consists of four-elements, a comma-separated string-type variable, where the elements are the InfoReference, the LocalReference, the MessageId, and the message buffer.

5. The local OPC UA server makes this data available to the external OPC UA client.

For details on how Block 4 data is configured for subscription, see the sections of this document called #Blk4_buflen, #Blk4MessageRouter, and #Blk4FromIccp.

Block 5 Top Dataflow

The remote ICCP client can also initiate Block 5 controls to the application's OPC server as shown in Figure 2.

Block 5 Bottom Dataflow

1. An application's OPC UA client can initiate Block 5 controls to the local OPC UA server.

2. LEC Server converts the OPC UA data to Block 5 ICCP data and passes it to the local ICCP client.
3. The ICCP client then transfers this control data to the remote ICCP server device.

> For more information on Block 5, see the sections of this document called OPC UA Control Over ICCP Devices and the entry for OPC UA Client (OEM Applications) in the Glossary of System Components.

Configuration Setup, Startup, and Remote Batch Load

You first need to install Oracle Utilities LEC Configuration Manager and Server on each of the LEC machines, physical and virtual.

> For more information, refer to *LEC Platform Installation Guide*. Prior to purchasing Oracle Utilities LEC Platform, which provides both LEC Configuration Manager and Server, first read the section of this document called Provisioning a New System.

Generating and Staging Batch and Command Files

Prior to activating LEC IFEs, the OEM must generate batch files using the templates that Oracle Utilities and the OEM develop jointly. These batch files will define the data and control points designed to hold the Block 1, 2, 4, and 5 data that the IFE configuration can transfer between the OEM's application and the remote ICCP systems and, in the opposite direction between the remote ICCP systems and the OEM's application; for examples, see the sections of this document called Header Batch File *and* VCC, VMD, Variables, and Node Batch File Definitions.

In addition, the OEM needs to generate command files to load the batch files. Oracle Utilities and the OEM also develop templates that the OEM can use to create these command files. After generating the batch files and creating the command files, the OEM must stage the batch files and command files on each LEC Server machine.

Staging the IFE Configuration on a Development Machine

Staging LEC IFEs requires that the OEM perform the following steps:

1. Use LEC Configuration Manager to import the LEC IFE configuration .db file.
2. Use LEC Configuration Manager to load the header batch file into the IFE configuration.
3. Save the LEC IFE configuration .db and give it a descriptive name.

Deploying the IFE Configuration

1. Rename the LEC IFE configuration .db to cfg.db, which is the default name of the configuration file used by LEC Server.
2. Push or copy **cfg.db** to the **C:\ProgramData\LiveEnergyConnect** directory of each IFE machine.
3. Remotely start LEC Server as a service on each of the IFE machines.
4. Now you are ready to load the additional batch files onto the remote machines where you or another engineer has completed the following:

- Installed LEC Configuration Manager.
- Made the LEC IFE configuration the default configuration database by naming it cfg.db.
- Started LEC Server as a service.

Loading Batch Files

Loading batch files remotely from the OEM's application depends on six management variables defined in the OPC UA server within the LEC IFE application. These variables are located within the System OPC UA namespace of the OPC UA server and in the LEC System branch.

- **DBLoadRequest**: This input variable sets the path to a command file that will load batch files. Writing to this variable triggers the execution of the command file.
- **DBLoadState**: This output variable indicates the state of the load. 0 is OK; 1 is failed, and -1 is busy.
- **DBLoadRequestedCommandFile**: This output variable indicates the name of the command file to which DBLoadState refers.
- **DBLoadCurrentlyLoaded**: This output variable provides the name of the last command file that was loaded; this variable is set just before DBLoadState changes to 0. If the load fails, DBLoadState is set to 1, indicating failure. In addition, DBLoadErrorString and DBLoadCommandLine are returned to provide more information about the failure.
- **DBLoadErrorString**: This output variable returns a string with the load error message if there is one.
- **DBLoadCommandLine**: This output variable returns the line on which the error occurred in the command file.

The OPC UA client within the OEM application sets the path of the command file that triggers the loading of one or more batch files in each LEC IFE configuration. This in turn will cause each LEC IFE to return the success or failure of the batch load in DBLoadState and DBLoadCurrentlyLoaded. If there is a failure, the OPC UA server will also return DBLoadRequestedCommandFile, DBLoadCurrentlyLoaded, and DBLoadErrorString.

For an example of a batch file that defines these output variables, see the section of this document called Header Batch File Definitions.

Setting Security within a Batch File

Security for OPC UA is set within the batch file that defines the OPC UA client agent. In this file, you can turn on security by setting **Security Options** for the client. To turn on Security Options, set the Security Options parameter to **BEST**. To turn off Security Options, set this parameter to **NONE**. By default, OPC UA security is set.

> Let your Oracle Utilities engineers know whether you intend to use OPC UA security so that they can prepare your batch files accordingly.

Provisioning a New System

Starting with a Windows Server and install LEC Server software with a base IFE configuration, the OEM application will be able to push the batch files of specific VMDs, VCCs, association control variables, and device points to each system. For each system, the OEM application will also be able to specify IP addresses for all VMDs, VCCs, and utilities, and then push these addresses to their respective systems.

Oracle LEC Server Requirements

The hardware requirements depend on the specific application. Most applications can run on an entry-level Microsoft Windows Server. The following are the minimum requirements for a computer hosting an LEC Server instance:

- Multi-core processor (2.4 GHz or better)
- 8 GB RAM
- 500 GB Hard Disk
- Any currently supported Microsoft Windows Server OS

Updating an Existing System

When the OEM application moves to a new version of LEC Server, the new version will include new LEC macros, and someone will need to update the macros in the OEM application's base configuration.

Heartbeat Monitor

The OEM application's OPC UA client writes to a heartbeat variable called HeartBeatFrom<*App*> in the local OPC UA server in the LEC Server instance. **<App>** refers to the OEM application's name or abbreviation. Each LEC Server instance monitors this heartbeat variable, and if it is not updated within the configurable timeout, the LEC Server instance will go offline.

> **Note:** You can disable the monitor in the batch file. See the description of the <active> property in the #HeartBeatFromApp section for details.

The OPC UA server in each LEC Server instance has a heartbeat variable called HeartbeatMon that is updated every 10 seconds. The application's OPC client can monitor this heartbeat variable for purposes of managing failover.

OPC UA Control Over ICCP Devices

You can use either **Select Before Operate (SBO)** or **Direct Operate (DO)** to issue commands and set or clear points on an ICCP peer device.

Select Before Operate (SBO) and Direct Operate (DO)

SBO is a two-stage, select-before-operate process with confirmation following the select. Using this process, a client has to select the control object. After selection, the selecting client is the only one allowed to perform control actions. An ICCP Check Back is required for all **Select**

Before Operate controls. A **Check Back** is identified using an arbitrary number, agreed upon by the local ICCP client and the remote ICCP server; the remote ICCP server returns to the local ICCP client as a result of a **Select** operation. However, if the **Select** operation is successful, the application's OPC UA client is not notified; the OPC UA client is only notified if there is an error returned from the **Select** operation.

DO is a single-stage direct operate process with no select-before-operate stage. The direct control models provide a simple means to start actions on the LEC Server instance. If multiple clients are trying to perform conflicting control actions, the LEC Server instance will finish one of the operations successfully but will fail and return an error message for any additional operations. If conflict prevention is required, the **Select Before Operate** process should be used.

Commands and Setpoints

ICCP distinguishes between device operation (commands) and setting and clearing numeric values (setpoints) using the following objects:

- **COMMAND** – ICCP allows 16-bit, integer-based control actions for each control point such as Open (0) and Close (1) or floating-point values. The following table shows some of the integer-based values and their meanings for different devices.

Table 1: Examples of Control Actions

0	1	Device
Trip	Close	Switch
Open	Close	Switch
Off	On	Switch
Lower	Raise	Transformer

Note: Boolean values are represented by the integer values 0 and 1.

- **SETPOINT** – ICCP's setpoint object can hold floating-point numbers or integers.

LEC Flags for SBO and DO Controls

LEC Server has a Flags property for the ControlToIccp and ControlFromIccp templates that allows you to specify the type of control to use for a particular point. Any point defined by one of these templates is a control point. You can specify more than one flag to define a control point by using a bar (I), for example: *DISCRETE|SBO*.

- **REAL** – The control point is a floating-point number. If a control point is a floating-point number, it is a setpoint.
- **DISCRETE** – The control point is an integer. If a control point is an integer, it is a setpoint.
- **SBO** – The control point supports **Select Before Operate** (SBO). If SBO is not specified, then the control point does not support SBO and uses **Direct Operate** (DO), instead.

- **TAGABLE** – The control point supports tagging. If TAGABLE is not specified, then the point does not support tagging.

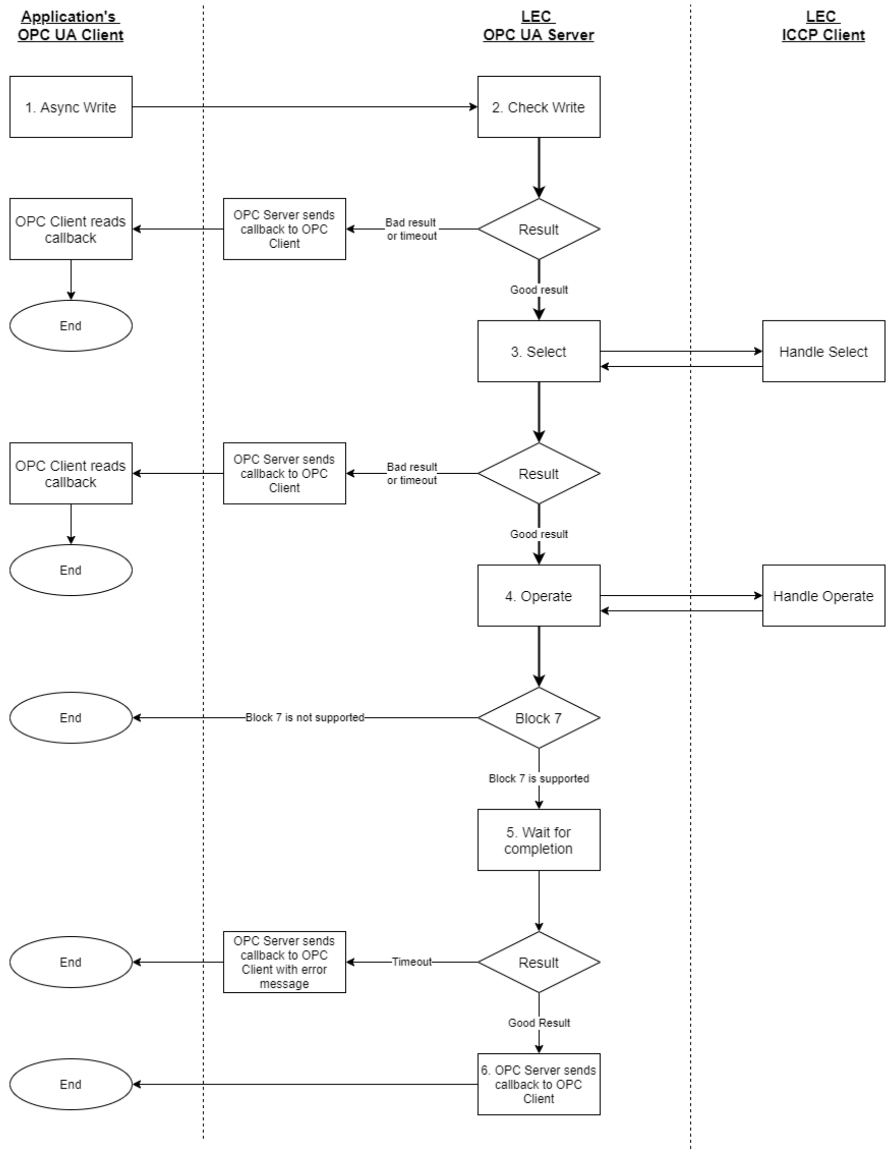
> **Note**: The default type of a control point is a command point. In order to indicate that a point is a command, do not specify either REAL or DISCRETE.

Examples of points that are defined by these flags are described in #Blk5FromIccp and #Blk5ToIccp. For more information on tagging, see the Tagging section.

Issue a Command or Set the Value of a Setpoint Using SBO

The SBO protocol is shown in Figure 3. LEC Server has a green overlay, and the OEM application's OPC UA client has a blue overlay.

Figure 3: Select Before Operate Workflow



The following are steps required to issue a command or set a setpoint using the Select Before Operate protocol with and without Block 7 support.

1. The application's OPC UA client's write request variable to the local OPC UA server. The OPC UA client variable has the following structure: Value, which is a command or a setpoint. The possible value for a command is an integer, specifying one of two possible actions to be taken on a device in the field such as 0 for Open and 1 for Close, or 0 for Off and 1 for On. The possible values for a setpoint are either a floating-point number or an integer.
2. The local OPC UA server checks the write and sends a write callback if the Write times out or returns an error. If there is no error, then the Local OPC UA server goes to step 3 in the sequence without sending a callback.

   If an error occurs, such as an access right violation, error message or number, or something else, the local OPC UA server stops the sequence and sends a callback message with the result back to an application's OPC client. This message is returned in the Status field of the OPC client write response variable.
3. The local OPC UA server sends a Select to the remote ICCP device.
   a. If there is no error, then the local OPC UA server goes to Step 4 without sending a callback.
   b. If there is an error, the local OPC UA server stops the sequence and sends a callback message with result back to the application's OPC client. The result is returned in the Status field of the OPC UA client variable.
4. The local OPC UA server sends an **Operate (Block 5)** to ICCP.
   a. If there is no support for Block 7, the OPC server does not wait for the operation to complete. Instead, local OPC UA server sends the Block 5 response back to the OPC client, which indicates whether or not the ICCP Block 5 Operate was received by the remote ICCP server, but there is no indication whether the Operate succeeded or failed. The result is returned in the Status field of the OPC UA client variable.

      After the Operate completes, whether it succeeds or fails, the **Select** is implicitly cleared.
   b. If there is support for Block 7, then the local OPC UA server goes to Step 5 before sending the Block 5 response.
5. The local OPC UA server waits for the **Block 7** event to complete or the local OPC UA server receives a timeout message. If the Block 7 event completes within the configured time period, the local OPC UA server sends a callback with a translated ICCP result (indicating success) to the application's OPC client.
   a. If Block 7 does not complete within the timeout period or if there is an error, local OPC UA server sends a Write response with the translated ICCP result to the application's OPC client.

   The result whether successful or unsuccessful is returned in the Status field of the OPC UA variable.

For a list of possible messages that ICCP returns and how LEC Server translates these messages into OPC UA status codes, see *Mapping between Returned ICCP Status Names and OPC UA Status Names for Block 5 Data*.

Figure 4 shows a simple illustration of SBO when an OPC UA client issues a Block 5 command without Block 7 support.

Figure 4: Block 5 Command Write to a Remote ICCP Server

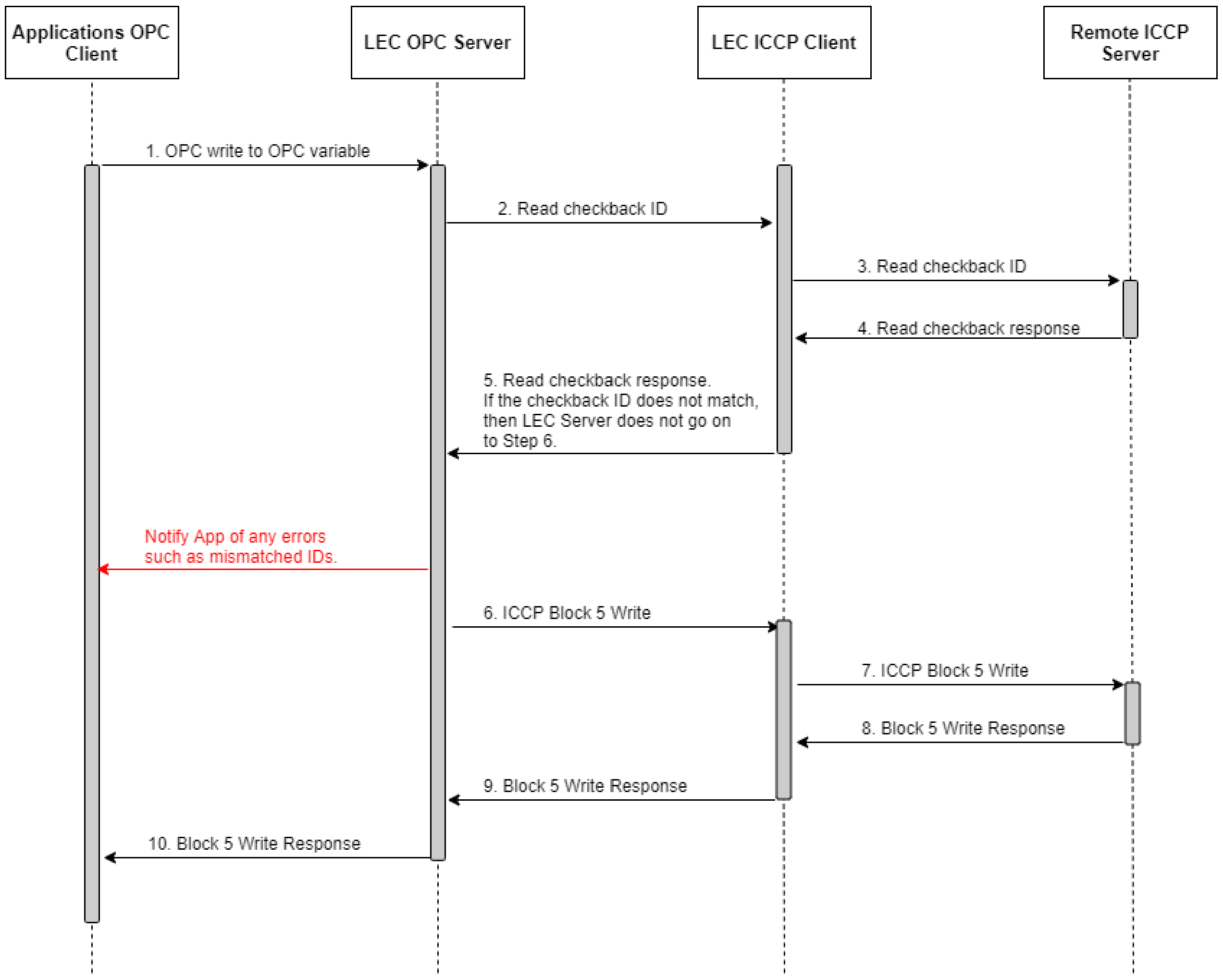
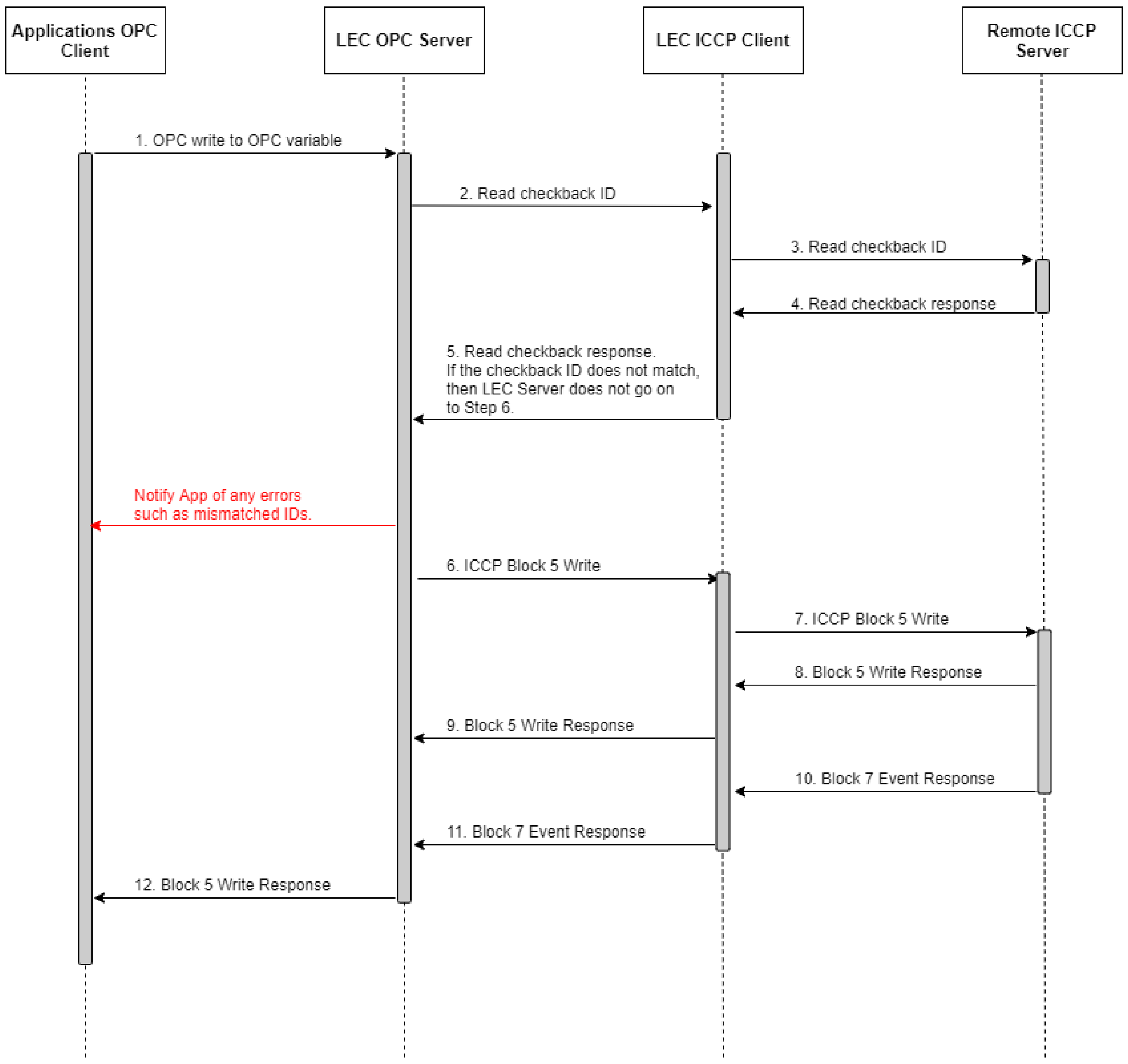

Applications OPC Client
LEC OPC Server
LEC ICCP Client
Remote ICCP Server
1. OPC write to OPC variable
2. Read checkback ID
3. Read checkback ID
4. Read checkback response
5. Read checkback response. If the checkback ID does not match, then LEC Server does not go on to Step 6.
Notify App of any errors such as mismatched IDs.
6. ICCP Block 5 Write
7. ICCP Block 5 Write
8. Block 5 Write Response
9. Block 5 Write Response
10. Block 5 Write Response

Figure 5 shows a simple illustration of SBO with Block 7 support.

Figure 5: SBO Block 5 Write to a Remote ICCP Server with Block 7 Support


Applications OPC Client
LEC OPC Server
LEC ICCP Client
Remote ICCP Server
1. OPC write to OPC variable
2. Read checkback ID
3. Read checkback ID
4. Read checkback response
5. Read checkback response. If the checkback ID does not match, then LEC Server does not go on to Step 6.
Notify App of any errors such as mismatched IDs.
6. ICCP Block 5 Write
7. ICCP Block 5 Write
8. Block 5 Write Response
9. Block 5 Write Response
10. Block 7 Event Response
11. Block 7 Event Response
12. Block 5 Write Response

Issue a Command or Set the Value of a Setpoint using DO

The following list describes the steps required to issue a command or set a point using the Direct Operate protocol with and without Block 7 support.

1. The application's OPC client writes a variable to the local OPC UA server.

The OPC UA client write request variable has this structure:

- Value, which is a command or a setpoint.
    - The possible value for a command is an integer, specifying one of two possible actions to be taken on a device in the field such as 0 for Open and 1 for Close, or 0 for Off and 1 for On.

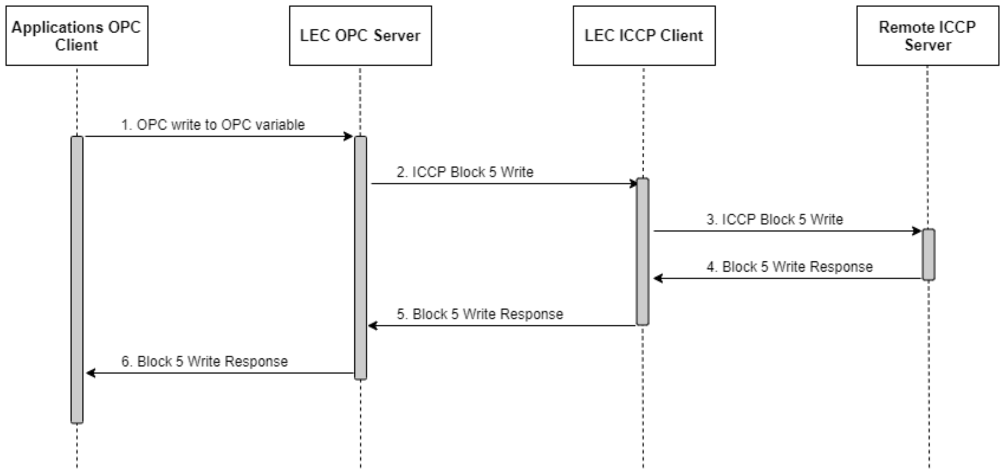
   - The possible values for a setpoint are either a floating-point number or an integer.

2. The local OPC UA server checks the write and sends a write callback.
   a. If there is no error, then the local OPC UA server goes to Step 3 in the sequence without sending a callback.
   b. If an error occurs, such as an access right violation, error message or number, or something else, the local OPC UA server stops the sequence and sends a callback message with the result back to the OPC client.
3. The local OPC UA server sends an **Operate (Block 5)** to the local ICCP client.
4. The local ICCP client then sends an **Operate (Block 5)** to the remote ICCP server.

If Block 7 is not supported, the OPC server does not wait for the operation to complete. Instead, the OPC server waits only for the remote ICCP server to respond to the Operate request. Then the local OPC UA server sends the Block 5 response back to the OPC client in the **Status** field of the OPC client variable. The returned value indicates whether or not the remote ICCP server received the Block 5 Operate request; it does not report whether the Operate succeeded or failed.

Figure 6 displays the DO Block 5 Write Operate and response sequence.

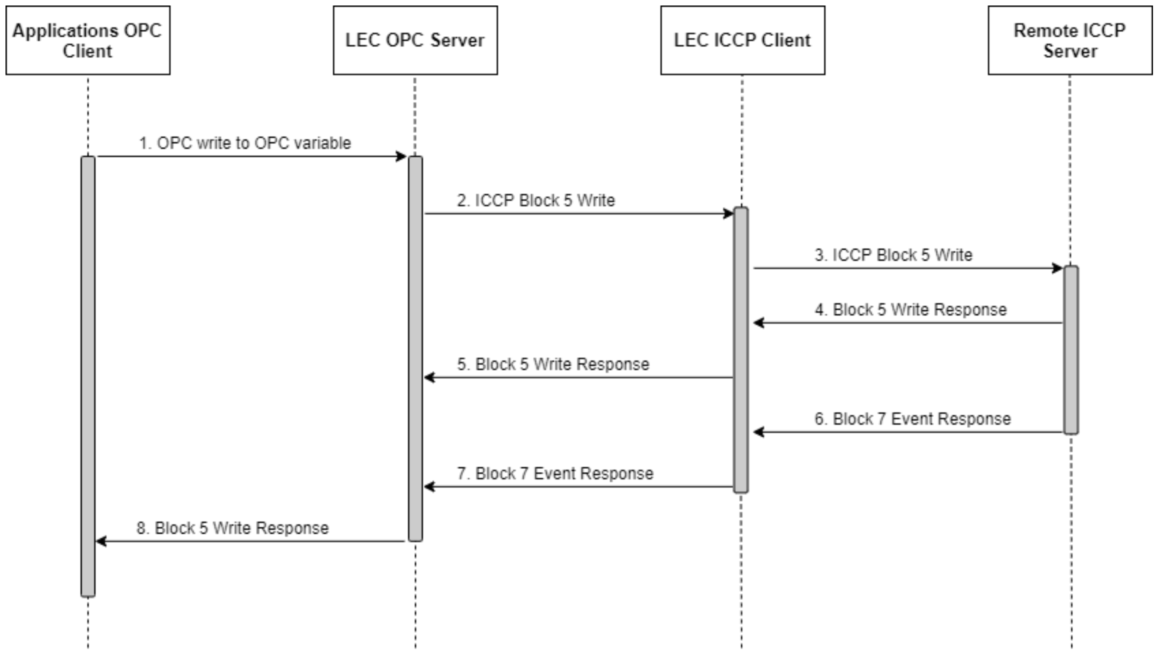
Figure 6: DO Block 5 Write to a Remote ICCP Server



If there is support for Block 7, the local OPC UA server waits for the Block 7 event to complete or issues a timeout message. If the Block 7 event completes within the time period, the local OPC UA server sends a callback with the result (indicating success) to the application’s OPC client.

If Block 7 does not complete or if there is an error, the local OPC UA server returns a Write response to the OPC client in the Status field of the OPC UA client variable as shown in Figure 7.

Figure 7: DO Block 5 Write to a Remote ICCP Server with Block 7 Support


Applications OPC Client
LEC OPC Server
LEC ICCP Client
Remote ICCP Server
1. OPC write to OPC variable
2. ICCP Block 5 Write
3. ICCP Block 5 Write
4. Block 5 Write Response
5. Block 5 Write Response
6. Block 7 Event Response
7. Block 7 Event Response
8. Block 5 Write Response

For a list of possible messages that ICCP returns and how LEC Server translates these messages into OPC UA status codes, see *Mapping between Returned ICCP Status Names and OPC UA Status Names for Block 5 Data.*

Tagging

(Tagging is not implemented yet, but it will become available in a future release.)

Tagging is the mechanism through which OPC UA clients can protect utilities technicians and equipment (usually during maintenance) by hanging a "red tag" (logically, not physically) on a device to prevent the device from executing subsequent operations. OPC UA clients can set, clear, and monitor tags in remote ICCP servers. Setting, clearing, and monitoring are all Block 5 operations.

Setting and Clearing Tags

(Tagging is not implemented yet, but it will become available in a future release.)

The application's OPC client can set or clear a tag in the local OPC UA server by writing to the tag variable. A tag kind of 1 or 2 sets a tag, and a tag kind of 0 clears the tag. The local OPC UA server then passes the data in the tag item to a tag in the specified remote ICCP server. The tag state format for write (set) and monitor/read (get) is composed of three comma-separated fields: **tag kind**, **armed**, and **comment**.

The tag is a structure within an OPC UA variable that consists of the following fields:

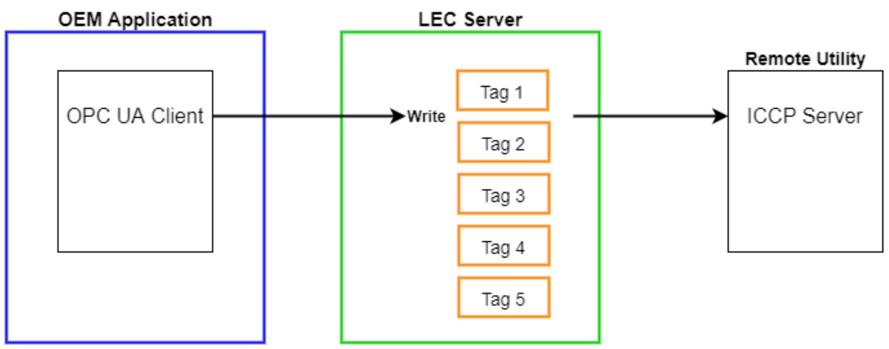
- **Tag** - a ByteString. The tag state format for write (set) and monitor/read (get) is composed of three comma-separated fields: tag kind, armed, and comment as shown in the following table.

Tag kind	NO-TAG=0	OPEN-AND-CLOSE-INHIBIT=1	CLOSE-ONLY-INHIBIT=2
Armed	IDLE=0	ARMED=1	
Comment	Optional unconstrained comment string that can contain commas		

- **Status** - usually initialized to 0.
- **Timestamp of the local OPC UA server** - usually initialized to 0.
- **Timestamp of the receiving ICCP server** - usually initialized to 0 *(not currently used).*

The local OPC UA server then passes the data in the tagged items to corresponding tags in the specified remote ICCP server as shown in Figure 8.

Figure 8: OPC UA Client Writes to a Set of Tags Once the Connection is Made with the ICCP Server



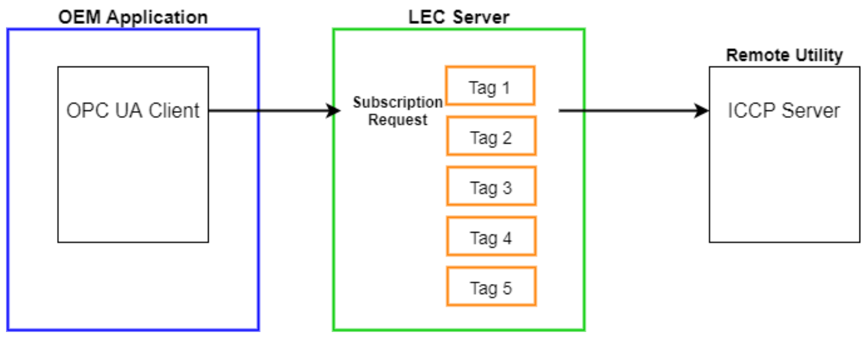
> **Note:** Prior to working with tags, you need to understand it is always the remote ICCP server's view of the tag that is the actual tag state. Even if the OPC client writes to a tag, and it is sent to the ICCP server, the ICCP server can ignore the set tag and consequently will return its view of the tag state in the next get request.

Subscribing to a Tag

(Tagging is not implemented yet, but it will become available in a future release.)

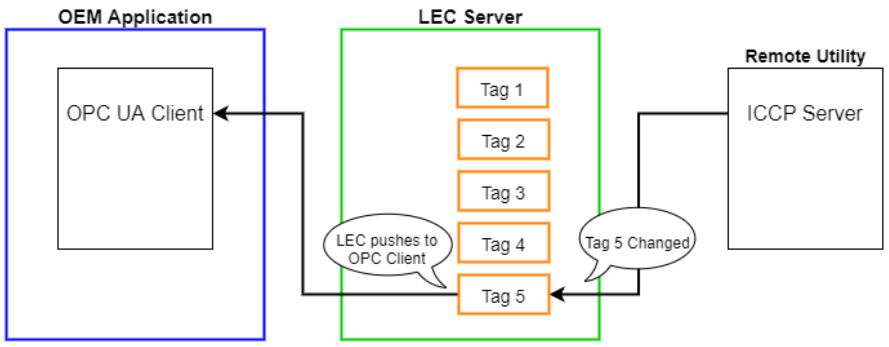
Subscription is the means by which an application's OPC client learns the status of a tag in a remote ICCP server. An OPC client asks to subscribe to a set of tags in the local OPC UA server as shown in Figure 9.

Figure 9: OPC UA Client Writes to a Set of Tags Once the Connection is Made with the ICCP Server



1. After the LEC Server instance has connected to the remote ICCP server, the OPC UA client can issue subscription requests to a set of Tag items in the LEC Server instance. The LEC Server instance will in turn subscribe to these points in the remote ICCP server.
2. From this time forward, LEC Server will cyclically read these tag values, and push any changed tags to the OPC UA client. The cycle time is configurable. The tag values are read cyclically because the ICCP server can change the tags at the remote site at any time for any reason. Figure 10 shows this dataflow.

Figure 10: LEC Server Pushes Changed ICCP Tag Values Back to the OPC UA Client



3. After reading the tagged value, the OPC UA client can be programmed to prevent or allow the issuing of a control directive (Block 5) to the remote ICCP server.

Redundancy

Oracle supports redundancy by running one LEC Server instance and one or more standby LEC Server instances. These instances all run on different physical or virtual machines. All redundant LEC Server instances have identical ICCP link configurations. However, only one LEC Server instance can be online at one time. When an LEC Server instance is online, in Active mode, it is the only LEC Server with active ICCP associations and connections to the application's OPC client connections.

Each LEC Server instance has a Master Control variable that controls whether it is operating online (Active mode) or on standby (Passive mode).

Master Control and Active/Passive Mode

The LEC Server instance, configured as an IFE, hosts a Master Control variable that toggles the instance between Active and Passive modes. A remote OPC UA client can write to this Master Control variable. In this scenario, the remote OPC UA client is one of the application's OPC UA clients. Writing 1 to a Master Control variable sets the LEC Server instance to Active mode, while writing 0 to a Master Control variable sets it to Passive mode. A remote OPC UA client can also read the Master Control variable to query the instance's active/passive status. It is useful for the OPC UA client to read the Master Control variable because, even though the OPC UA client knows what Master Control values it has sent, the Master Control variable may be changed by the Heartbeat mechanism or by a cluster controller.

The Master Control variable is available to read from and write to through the LEC Server instance's OPC UA server using the following points:

Read Point	Write Point
`MasterControlOutFor`<agent> Where **<agent>** is the name of the actual OPC UA server agent node. For example: `MasterControlOutForAgentS`	`MasterControlInFor`<agent> Where **<agent>** is the name of the actual OPC UA server agent node. For example: `MasterControlInForAgentS`

Note: You can configure the OPC UA namespace and branch for each of these points.

Active Mode: In Active mode, LEC Server will allow the enabling of ICCP associations and OPC UA client outbound connections. However, you must enable each ICCP association explicitly with the VCC's Association Control variable. Similarly, you must enable each OPC UA client outbound connection by explicitly setting the VMD's Association Control variable.

Passive Mode: In Passive mode, LEC Server will disable all ICCP associations and OPC UA client outbound connections.

Association Control Variables

Each ICCP VCC has an Association Control variable to enable or disable its associations. Writing 1 to the Association Control variable enables the VCC's associations, while writing 0 to the Association Control variable disables the VCC's associations. The LEC Server instance must be in Active mode for associations to be enabled; writing 1 to the Association Control variable has no effect when the instance is in Passive mode.

The name of the connected remote peer VCC is also read via OPC UA server variable.

Association Control variables are available to read from and write to through the LEC Server instance's OPC UA server. The inbound Association Control variables for reading and writing are as follows:

Inbound Association Variables	
Inbound Association: Read Point	<VCC>/`AssocInCtrlStatus`

	Where **<VCC>** is a placeholder for the actual character string used in naming the VCC containing this point. For example: `AppTestVcc/AssocInCtrlStatus`
Inbound Association: Write Point	<VCC>/`AssocInCtrlControl` For example: `AppTestVcc/AssocInCtrlControl`
Inbound Connected Peer: Read Point	<VCC>/`InboundPeerName` For example: `AppTestVcc/InboundPeerName`
Outbound Association Variables	
Outbound Association: Read Point	<VCC>/`AssocOutCtrlStatus` Where **<VCC>** is a placeholder for the actual character string used in naming the VCC containing this point . For example: `AppTestVcc/AssocOutCtrlStatus`
Outbound Association: Write Point	<VCC>/`AssocOutCtrlControl` For example: `AppTestVcc/AssocOutCtrlControl`
Outbound Connected Peer: Read Point	The outbound Association Control variables for reading and writing are as follows: <VCC>/`OutboundPeerName` For example: `AppTestVcc/OutboundPeerName`

Note: You can configure the OPC UA namespace and branch for each of these points.

Alarm State

If an enabled VCC association is not established within the configured timeout period, the association enters an Alarm state. The Alarm state is signaled by setting the Association Control variable to 3. The LEC Server instance clears the Alarm state when an association is made, or when the association is disabled by setting the Association Control variable to 2 (associated), or 0 (disabled), respectively.

You can configure the timeout as the Stale time for the Association Control nodes. See Setting the Stale Time Property for more information.

When an LEC Server instance sets a VCC to the Alarm state, the instance also sets quality flags that reflect this state in all of the VCC's ICCP points that support quality flags. The LEC Server instance can set either the VALID HIGH or VALID LOW quality flag bits, or both, in the Association flags property of these ICCP points. This behavior is configurable through a batch file. Specifically, the `VALID_LOW` flag sets the `Validity_lo` quality bit (bit #3), and the `VALID_HIGH` flag sets the `Validity_hi` quality bit (bit #2). For more information on quality bits, see Mapping Quality Bit Values from ICCP to OPC UA Status Names and States for Block 1 Data.

Setting the Stale Time Property and Association Flags

You can configure the time period before the LEC Server instance signals an alarm by setting the Stale time property for the Association Control node in the header file. A batch file sets the Stale time property for an Association Control point or points as well as the Association flags VALID_HIGH and/or VALID_LOW. You can find an example of a batch file called "AppTestAHeader.csv", which sets VCCs, points, and properties, including the Stale time property in the directory C:\ProgramData\LiveEnergyConnect\Config\ or D:\ProgramData\LiveEnergyConnect\Config\. This batch file is described in its entirety in the section of this document called Header Batch File Definitions.

```
#ConfigControlIn,          <agent>,    <branch>,   <NS_Suffix>,    <type>, <tag>

DBLoadRequest,             AgentS,     LECSystem,  SYSTEM,        <v-s:-128>, DBLoadRequest
```

Prototype Node Name or Property	Node Label/Output Variable or Property Value	Defines
#ConfigControlIn	DBLoadRequest	The label of the input variable that sets the path to a command file which will load batch files. Writing to this variable triggers the execution of the command file.
<agent>	AgentS	The label of the OPC UA server agent that handles communication for the DBLoadRequest node.
<branch>	LECSystem	The OPC UA branch where DBLoadRequest is located. Leaving blank indicates no branch.
<NS_Suffix>	SYSTEM	Specifies the OPC UA namespace suffix for DBLoadRequest. Leaving blank indicates no NS suffix.
<type>	<v-s:-128>	LEC data type of the DBLoadRequest node.
<tag>	DBLoadRequest	The name of the variable that is to be passed to the DBLoader script for the DBLoadRequest point.

#ConfigControlOut

This table defines the output variables that the OPC UA server uses to return error message data to the OPC UA client. For more information, see the section of this document called Loading Batch Files.

```
#ConfigControlOut,         <agent>,    <branch>,   <NS_Suffix>,    <type>,    <tag>

DBLoadCommandLine, AgentS, LECSystem, SYSTEM, <v-s:-128>, DBLoadCommandLine

DBLoadErrorString, AgentS, LECSystem, SYSTEM, <v-s:-128>, DBLoadErrorString

DBLoadState, AgentS, LECSystem, SYSTEM, <int:32>, DBLoadState

DBLoadRequestedCommandFile, AgentS, LECSystem, SYSTEM, <v-s:-128>,
DBLoadRequestedCommandFile
```

DBLoadCurrentlyLoaded, AgentS, LECSystem, SYSTEM, <v-s:-128>, DBLoadCurrentlyLoaded

Prototype Node Name or Property	Node Label/Output Variable or Property Value	Defines
#ConfigControlOut	DBLoadCommandLine	The label of the output variable that returns the line on which the error occurred in the command file if an error occurred. **DBLoadErrorString** is the label of the output variable that returns a string with the load error message if there is one. **DBLoadState** is the label of the output variable that indicates the state of the load. 0 is OK; 1 is failed, and -1 is busy. **DBLoadRequestedCommandFile** is the label of the output variable that contains the name of the command file to which DBLoadState refers. **DBLoadCurrentlyLoaded** is the label of the output variable containing the name of the last command file that loaded successfully; this variable is set just before DBLoadState changes to 0. If the load fails, DBLoadState is set to 1.
<agent>	AgentS	The label of the OPC UA server agent that handles communication for this point and for DBLoadErrorString, DBLoadState, DBLoadRequestedCommandFile, and DBLoadCurrentlyLoaded.
<branch>	LECSystem	The OPC UA branch where this point and DBLoadErrorString, DBLoadState, DBLoadRequestedCommandFile, and DBLoadCurrentlyLoaded are located. Leaving blank indicates no branch.
<NS_Suffix>	SYSTEM	Specifies the OPC UA namespace suffix for this point as well as DBLoadErrorString, DBLoadState, DBLoadRequestedCommandFile, and DBLoadCurrentlyLoaded. Leaving blank indicates no NS suffix.
<type>	<v-s:-128>	Specifies the OPC UA data type for this point and for DBLoadErrorString, DBLoadRequestedCommandFile, and DBLoadCurrentlyLoaded. **Note**: The data type for DBLoadState is <int:32>.
<tag>	DBLoadCommandLine	The name of the variable that is to be passed to the DBLoader script for the DBLoadCommandLine point. In addition, the DBLoadErrorString, DBLoadState, and DBLoadCurrentlyLoaded points pass the DBLoadErrorString DBLoadState, and DBLoadCurrentlyLoaded tags to the DBLoader script, respectively.

		Note: These particular values are set in points on the ConfigLoader VMD, which is internal to the workings of the IFE.

To set the Stale time property and the Association flags VALID_HIGH and/or VALID_LOW:

1. Stop LEC Server and unload the batch files.
2. In the batch file, search for the table header that begins with the prototype node `#AssocInControl`. This node is based on the `VccAssocControl` template. Prototype nodes begin with a pound sign (#). The table header identifies all the prototype node properties that will be set by the following line in the batch file. In addition to setting properties in the `#AssocInControl` points (that are generated when the batch file is loaded), this table sets properties in a connecting node in the `OpcUaServer` VMD that has an Agent label, Branch name, and Namespace. All prototype properties are identified by brackets (<>).

Example 1

```
#AssocInControl, <agent>, <branch>, <NS_Suffix>, <stale_time>, <timeout_valid_bit>
```

The next line, shown in Example 2, identifies the actual values used for the Stale time and Association flags properties in the AssocInCtrl node. In addition, the values for the Agent label, Branch name, and Namespace are set. These values must be in the exact same order as the table header shown in Example 1.

Example 2

```
AssocInControl, AgentS, LECSystem, SYSTEM, 30000, |VALID_LOW|VALID_HIGH
```

The table below shows the Point label value for the node that is set by the table defined in Example 2.; each value is delimited by a comma.

Point label	Agent	Branch	Namespace	Stale time	Timeout valid bit
AssocInCtrl	AgentS	LECSystem	SYSTEM	30000	`VALID_LOW` and `VALID_HIGH`

AgentS is the agent label, which maps to <agent> in the table header. There is no Agent label property for AssocInCtrl, but AssocInCtrl connects to a node in the OpcUaServer VMD that requires an Agent label.

In this example, the following schematic shows the connection between AssocInControl and the OPC UA server node that requires the Agent, Branch, and Namespace properties.

#AssocInControl $\longrightarrow$ AppTestA1A/AssocInCtrl
(which is in a VCC) (which is in the OPC server)

AppTestA1A/AssocInControl allows an external OPC client that is connected to the OPC server to enable or disable an ICCP associate and learn the status of an ICCP association.

LECSystem is the OPC branch name that the connecting node AppTestA1A/AssocInCtrl in the OpcUaServer VMD requires. This value maps to <branch> in the table header.

SYSTEM is the OPC suffix that is to be appended to the OPC namespace Universal Resource Identifier (URI) that identifies the agent node. This suffix in used by AppTestA1A/AssocInCtrl in the OpcUaServer VMD.

Stale time specifies the period of time that is allowed to elapse while there is no association. Stale time is specified in milliseconds. After the period of time has elapsed, the association will time out, and input points from this association are assigned a quality code based on the configured timeout validity bit settings: `VALID_LOW` and `VALID_HIGH`.

VALID_LOW If the VCC is set to the Alarm state, the `VALID_LOW` flag sets the `Validity_lo` quality bit (bit #3).

VALID_HIGH If the stale time period expires, the `VALID_HIGH` flag sets the Validity_hi quality bit (bit #2). For more information on quality bits, see the section of this document called Mapping Quality Bit Values from ICCP to OPC UA Status Names and States for Block 1 Data.

Note that these flags are prefixed by a vertical bar (|) as shown in Example 2. The vertical bar (|) before `VALID_LOW` appends it to other flags that are defined in the `#AssocInControl` prototype node in LEC Configuration Manager. The vertical bar (|) before VALID_HIGH appends it to `VALID_LOW`. If you look at the Association flags for AssocInCtrl in the Properties panel of LEC Configuration Manager, you will notice that `VALID_LOW` comes after `RELAY_FLAG|MASTERRESETFLAG`.

The `RELAY_FLAG` flag is set to 1 in this example. Setting this flag directs LEC Server to treat incoming Information Reports and Read Event Notifications as though they were MMS Writes to a local variable by the same name as is in the report. If this flag is not set, the ICCP Block 4 message will not be reliable.

The `MASTERRESETFLAG` flag is set to 1 in the Association Control nodes in this example configuration. To enable an association, both the Master Control variable and the VCC association control variable need to need to be written 1. When Master Control is written 0, all association control variables will revert to their initialized state of 0. See Determining Which ICCP Points LEC Server Sets in the Alarm State for more information.

1. Specify your own values, if necessary, for the Stale time property in the batch file used for setting points and VCC values at your site, such as AppTestA1.csv. The order of properties might differ in your batch file.
2. Setting this property for the AssocInCtrl node will also set properties in other connecting nodes/points in the OpcUaServer VMD, so the VCC and the OPC server can communicate. Repeat this process for AssocOutCtrl.
3. Reload the header file and the batch file you just edited.
4. Restart LEC Server.

Determining Which ICCP Points LEC Server Sets in the Alarm State

If the timeout set by the Stale time property expires, LEC Server sets the Alarm state to `VALID_HIGH`, `VALID_LOW`, or both in the Association flags as discussed in *Setting the Stale Time Property.*

Whenever a particular VCC association enters the Alarm state, LEC Server will set the ICCP quality flags fields in any of the VCC's ICCP points that contain quality flags, such as those that have either the Data_RealQ data type or the Data_StateQ data types.

Secure OPC and ICCP

This section explains how to secure OPC and ICCP VMDs/VCCs and nodes.

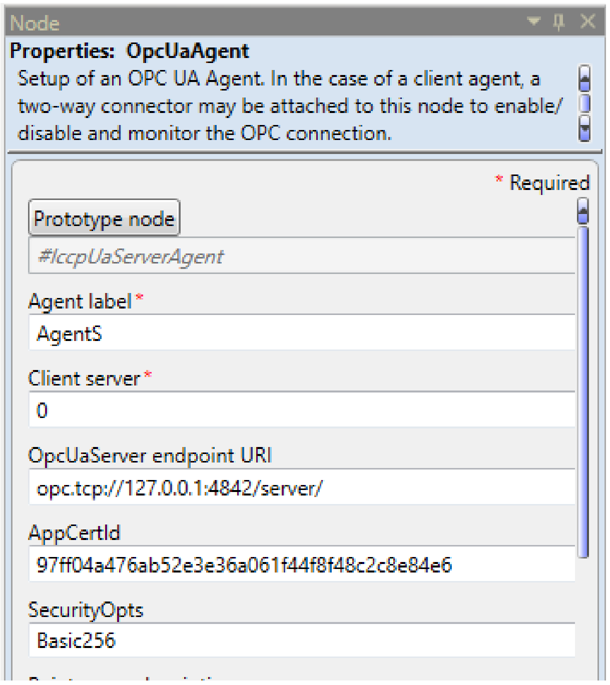
Secure OPC

Secure OPC UA is encrypted at the transport layer. The setup of the encrypted connection requires the OPC UA server and the client to authenticate their peers' certificates.

Certificates are stored in certificate stores. A certificate store is a place where certificates can be stored on a file system. All Windows systems provide a registry-based store called the Windows Certificate Store.

From LEC Configuration Manager, the OPC server and client VMDs have an OPC UA Agent node based on the OpcUaAgent template as shown in Figure 11.

Figure 11: OPC UA Server Fields for Secure OPC UA



Properties: OpcUaAgent

Setup of an OPC UA Agent. In the case of a client agent, a two-way connector may be attached to this node to enable/disable and monitor the OPC connection.

* Required

#IccpUaServerAgent

Agent label*
AgentS

Client server*
0

OpcUaServer endpoint URI
opc.tcp://127.0.0.1:4842/server/

AppCertId
97ff04a476ab52e3e36a061f44f8f48c2c8e84e6

SecurityOpts
Basic256

You can use the following three properties to create a secure OPC UA connection:

- **AppCertId (optional)**: Specifies the application certificate used by this OPC UA client or server. Specify the ID (thumbprint) of a certificate that was previously created and added to the appropriate Windows Certificate Store.
- **Default Behavior**: Required if SecurityOpts is specified as one of the secure options, otherwise the LEC Server's OPC UA client/server will fail to start.
- **SecurityOpts**: Specifies the security modes to use:

- **NONE** means that no encryption is available (server) or requested (client).
- **Basic256** means the Basic256 mode is used by the server. Invalid for client.
- **Basic128Rsa15** means the Basic128Rsa15 mode is used by the server. Invalid for client.
- **Basic256Sha256** means the Basic1256Sha256 mode is used by the server. Invalid for client.
- **ALL** means allow any of the above modes as a server. Invalid for client.
- **BEST** is a shortcut for Basic256Rsa256 as a server, or to use the best available secure option as a client.

Setting Secure OPC Properties in an LEC IFE Instance

The security properties are set in an LEC IFE instance at the time the header batch file is loaded (see the example AppTestAHeader.csv header file). This file is described in its entirety in Header Batch File Definitions.

1. To ensure that the security properties in the nodes AgentS and AgentC are set correctly, you need to edit the header batch file located in C:\ProgramData\LiveEnergyConnect\Config.

Search for the table header in the batch file that begins with the prototype node #IccpUaServerAgent. Prototype nodes begin with the pound sign (#). This line identifies all the prototype node properties that will be set by the batch file. These properties are identified by brackets (<>).

Example 1

```
#IccpUaServerAgent,<ServerIP>,<AppCertId>,<SecurityOpts>,<PointsPerSub>,<ServerName>,<PublishingInterval>,<CodeMapFile>,<MaxSubs>,<SubsPerSession>,<MaxSessions>
```

The next line identifies the actual values used for these fields in the agent node for the local OPC UA server. These values must have the exact same sequence as shown in Example 1.

Example 2

```
AgentS,opc.tcp://127.0.0.1:4842/server/,,NONE,,,,,50,10,5
```

The table below shows the Agent label and the OpcUaServer IP address followed by each security-related property that is defined in the second example line from the batch file as shown in Example 2.

Agent label	OpcUaServer IP	AppCertId	SecurityOpts
AgentS	opc.tcp://127.0.0.1:4842/server/	Not set.	NONE

AgentS is the agent label, which maps to the prototype #IccpUaServerAgent node in the preceding line.

opc.tcp://127.0.0.1:4842/server/ is the OPC UA Server IP address, which maps to <ServerIP>

No specification, as shown under the AppCertId field, indicates the default, which means that the application certificate ID will be generated by LEC Server.

NONE means that no encryption is available (server) or requested (client).

2. Specify your own values for the AppCertId, SecurityOpts properties in the same order as is in the line shown in Example 1. The order of these properties might differ in your header file.
3. Repeat the same process for the OPC UA client agent. Search for the line beginning with the prototype node #IccpUaClientAgent, specifying your own values for the AppCertId and SecurityOpts properties in the following line, similar to Example 2 but with appropriate values for the client.
4. Stop LEC Server and reload the header file.
5. Restart LEC Server.

Disabling Secure OPC in an LEC Server Instance

To stop using Secure OPC, you will need to clear the security properties that are set in an LEC IFE instance at the time the header batch file is loaded.

1. To ensure that the security properties in the nodes based on the OpcUaAgent template, called AgentS and AgentC, are cleared, you need to edit the header batch file, named AppTestAHeader.csv.

Search for the table header in the batch file that begins with the prototype node #IccpUaServerAgent. Prototype nodes begin with the pound sign (#). This line identifies all the prototype node properties that will be set or cleared by the batch file. These properties are identified by brackets (<>). Note that these properties can be listed in any order; however, the following line shown in Example 2 must use the same sequence.

Example 1

```
#IccpUaServerAgent,<ServerIP>,<AppCertId>,<SecurityOpts>,<PointsPerSub>,<ServerNam
e>,<PublishingInterval>,<CodeMapFile>,<MaxSubs>,<SubsPerSession>,<MaxSessions>
```

The next line identifies the actual values used for these fields in the agent node for the local OPC UA server. These values must have the exact same sequence as <AppCertId> and <SecurityOpts> shown in Example 1.

Example 2

```
AgentS,opc.tcp://127.0.0.1:4842/server/,,,,,,,50,10,5
```

The table below shows the Agent label and the OpcUaServer IP address followed by each security-related property that is defined in the second example line from the batch file.

Agent label	OpcUaServer IP	AppCertId	SecurityOpts
AgentS,	opc.tcp://127.0.0.1:4842/server/		

AgentS is the agent label, which maps to the prototype #IccpUaServerAgent node in the preceding line.

opc.tcp://127.0.0.1:4842/server/ is the OPC UA Server IP address, which maps to <ServerIP>

The empty space, nothing specified, usually indicates the default, which means the application certificate ID will be generated by LEC Server. However, in this instance, it actually clears the value for the **AppCertId** property.

The empty space under the **SecurityOpts** property clears the value for this property, too.

2. Clear the values for the **AppCertId**, **AuthorizedCerts**, **SecurityOpts** properties in the same order as is in the line shown in Example 1. Note that the order of the prototype node properties in your batch file might differ from the order shown in Example 1.
3. Repeat the same process for the OPC UA client agent. Search for the line beginning with the prototype node #IccpUaClientAgent, clearing the values for the **AppCertId** and **SecurityOpts** properties in the correct order.
4. Stop LEC Server and reload the header file.
5. Restart LEC Server.

Windows Certificate Stores

LEC Server's OPC UA server and client access certificates stored in the Windows Certificate Store.

Note: The Oracle Utilities Live Energy Connect (LEC Server) software was formerly known as LiveData RTI Server. For this reason, the following certificate sub stores still reference **LiveData**.

The following sub-stores are created and utilized:

- **LocalMachine\LiveData OPCUA Server App** – where LEC's OPC UA server certificates are stored.
- **LocalMachine\LiveData OPCUA Server Trusted** – where LEC's OPC UA server looks to find public certificates of connecting clients. If a copy of the client's presenting certificate is stored here, the client is considered trusted and the connection may be accepted. CA certificates/chains also be stored here. This allows any presented client certificate signed by the CA to be considered trusted. Certificate Revocation Lists (CRLs) may also be imported there to disallow certain client certificates signed by CAs stored here.
- **LocalMachine\LiveData OPCUA Server Issuer** – where LEC's OPC UA server looks for CA certificates/chains to validate trusted client certificates if said certificates are not self-signed.

- **LocalMachine\LiveData OPCUA Server Rejected** – where LEC's OPC UA server stores certificates presented by connecting clients that are not trusted. Rejected certificates may be copied or moved from here to the trusted store.
- **LocalMachine\LiveData OPCUA Client App** – where LEC's OPC UA client certificates are stored.
- **LocalMachine\LiveData OPCUA Client Trusted** – where LEC's OPC UA client looks to find public certificates presented by servers it is connecting to. If a copy of the server's presenting certificate is stored here, the server is considered trusted and the connection may proceed. CA certificates/chains also be stored here. This allows any presented server certificate signed by the CA to be considered trusted. Certificate Revocation Lists (CRLs) may also be imported there to disallow certain client certificates signed by CAs stored here.
- **LocalMachine\LiveData OPCUA Client Issuer** – where LEC's OPC UA client looks for CA certificates/chains to validate trusted server certificates if said certificates are not self-signed.
- **LocalMachine\LiveData OPCUA Client Rejected** – where LEC's OPC UA client stores certificates presented by connected servers that are not trusted. Rejected certificates may be copied or moved from here to the trusted store.

Certificate Details and Requirements

The LEC Server's Secure OPC UA client and server utilize X.509 V3 certificates for encryption and peer authentication. Below are specific certificate requirements:

- A minimum ***key length*** of 2048 bits is required, with a maximum of 4096 bits allowed.
- A ***Common Name*** (CN) is required, though its content is not important.
- ***Key Usage*** required usages: ***Digital Signature, Non-Repudiation, Key Encipherment, Data Encipherment, Certificate Signing***
- Certain X.509 V3 extensions are required:
  - ***Subject Alternative Name*** - must be provided with the following Names:
    - ***URI*** - (at least one) of the form URL=urn:<host>:<peer application name>
    - At least one of the following:
      - ***IP Address*** – one or more specifying and IP address to be associated with the endpoint, as seen by the receiving client or server.
      - ***DNS Name*** – one of more specifying the hostname to be associated with the endpoint, as seen by the receiving client or server.
  - ***Basic Constraints*** – must be provided with self-signed certificates to indicate:
    - ***Subject Type*** = End Entity
    - ***Path Length Constraint*** = 0

- ***Enhanced Key Usage – "Server Authentication"*** and ***"Client Authentication"*** required.

LEC Server ships with and installs OPC UA client and server certificates that is suitable for testing locally on a single machine, with OPC UA client and server endpoints specified with the localhost IP address 127.0.0.1. Also included is *MakeLegalUaCert.bat* that can be used and modified to create suitable certificates for network connections.

Secure ICCP

Secure ICCP provides security for the transport and application layers with certificates.

You need to set up Secure ICCP when you configure your LEC Server. Refer the Oracle Utilities Live Energy Connect Installation Guide and the Oracle Utilities Live Energy Connect Certificate Deployment Procedure for Using Secure ICCP document for instructions on configuring LEC Server for Secure ICCP.

Server Health

Every LEC Server instance exposes Server Health using a Service Level variable that OPC can access remotely. An LEC health monitor script not only monitors VMDs for their health, but the Health monitoring script has been customized to create an overall health score, also known as a service level, for the LEC Server instance. Since the Service Level variable is maintained by the Health Script, it can be exposed using OPC UA or any protocols that are supported by your environment. The Service Level variable is a byte. Health refers to all critical LEC Server functionality, including the ability of the LEC Server instance to communicate via SCADA protocols as well as functions such as:

- LEC WatchDogs
- Data flows: thread checker watchdog
- Link operational status
- Script execution
- Database access
- Logging

The Service Level variable is a node of the OPC UA server. The following table shows what each value or range of values means[9].

Subrange	Name	Description
0-0	Maintenance	The failed LEC Server instance is in the maintenance subrange. Therefore, new clients will not be allowed to connect, and currently connected clients will disconnect.
1-1	No Data	LEC Server is not operational. Therefore, an application's OPC client will not be able to exchange any information with it. LEC Server most likely has no data other than Service Level and diagnostic information available.

[9] In the future, LEC Server will also support the Standard OPC UA service level ns=0;n=2267.

2-199	Degraded	LEC Server is partially operational but is experiencing problems such that portions of the address space are out of service or unavailable. A possible cause for the Degraded service level would be if three of ten ICCP devices connected to LEC Server were unavailable.
200-255	Healthy	LEC Server is fully operational. Therefore, an application's OPC UA client can obtain all information from this server. This subrange allows LEC Server to provide information that clients can use to load balance. LEC Server could use this Service Level subrange to reflect LEC Server's CPU load; this would enable LEC Server to conclude that data would be delivered as expected. LEC Servers in the Healthy Service Level subrange are able to deliver information in a timely manner.

Mapping ICCP to OPC UA and Vice Versa

This section describes how Oracle Utilities LEC Server exposes ICCP monitoring and control information using an OPC UA interface.

ICCP Service Modeling

ICCP communication consists of the following data exchange patterns:

- Writes: The ICCP client writes to the ICCP server.
- Solicited Read: The server replies to a client request.
- Unsolicited Periodic Reporting: The server periodically sends data to the client.
- Unsolicited Exception Reporting: The server sends data that has changed within a period of time.
- Unsolicited Event Reporting: The server sends data based on client-defined criteria.

ICCP conformance blocks define groups for what and how to exchange data.

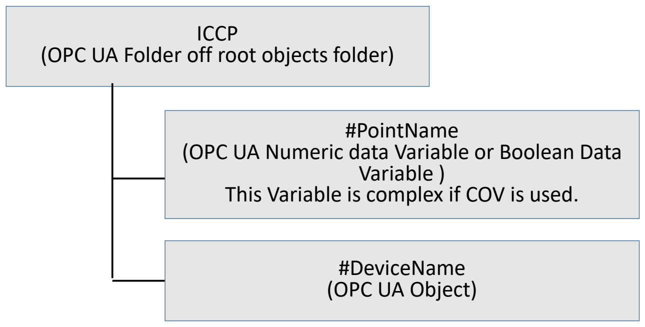
Table 2: ICCP Conformance Blocks

ICCP Block	Description
Block 1	Periodic System Data: Status points, analog points, quality flags, timestamp, change of value counter. ICCP transfers Block 1 data using Solicited Read as well as Unsolicited Periodic Reporting.
Block 2	Extended Data Set Condition Monitoring: Status points, analog points, quality flags, time stamp, and change-of-value (COV) counter. ICCP exchanges Block 2 data using Solicited Read as well as Unsolicited Exception Reporting.
Block 3	Not supported
Block 4	Information Messages: ICCP exchanges simple text and binary files using Unsolicited Exception Reporting.
Block 5	Device Control: Device control requests include on/off, trip/close, raise/lower as well as setting and clearing digital setpoints. Block 5 includes mechanisms for interlocked

	controls and select-before-operate methods. ICCP exchanges Block 5 data using Writes as well as Unsolicited Periodic and Exception Reporting.
Block 6	Not supported
Block 7	Event Reporting: ICCP provides Reporting completion status for a Block 5 device control.
Block 8	Additional User Objects: ICCP provides scheduling, accounting, outage, and plant information.
Block 9	Not supported

Default ICCP Mapping Exposed by a Local OPC UA Server

For OPC UA, an ICCP folder is exposed under the root Objects folder on the LEC Server machine.


ICCP
(OPC UA Folder off root objects folder)
#PointName
(OPC UA Numeric data Variable or Boolean Data Variable)
This Variable is complex if COV is used.
#DeviceName
(OPC UA Object)

While ICCP is the default name for the root of ICCP data, this OPC UA folder name is the name of the configuration database file.

Although only point values and change-of-value (COV) variables are shown above, you can configure variables for individual ICCP Quality tags by modifying the default batch file.

ICCP/OPC UA Mapping for Data from Each ICCP Conformance Block

This section shows how an LEC Server instance, acting as an LEC IFE, maps ICCP to OPC UA types for each ICCP conformance block. Blocks 1 and 2 include the PointTypes, which are REAL, STATE, and DISCRETE, as well as the Quality bits, Timestamp, and COV data. Block 4 includes ICCP data that is defined by the Message attribute, and Blocks 5 and 7 are defined by the ICCP Control and Report attributes, respectively.

Table 3 shows the OPC UA mappings for the data that is supported in each ICCP conformance block.

Table 3: ICCP to OPC UA Mapping by ICCP Conformance Block

ICCP Block	ICCP Attribute	ICCP Attribute Values	Default OPC UA Type Mapping
Blocks 1 & 2	PointType (REAL,STATE,DISCRETE)		

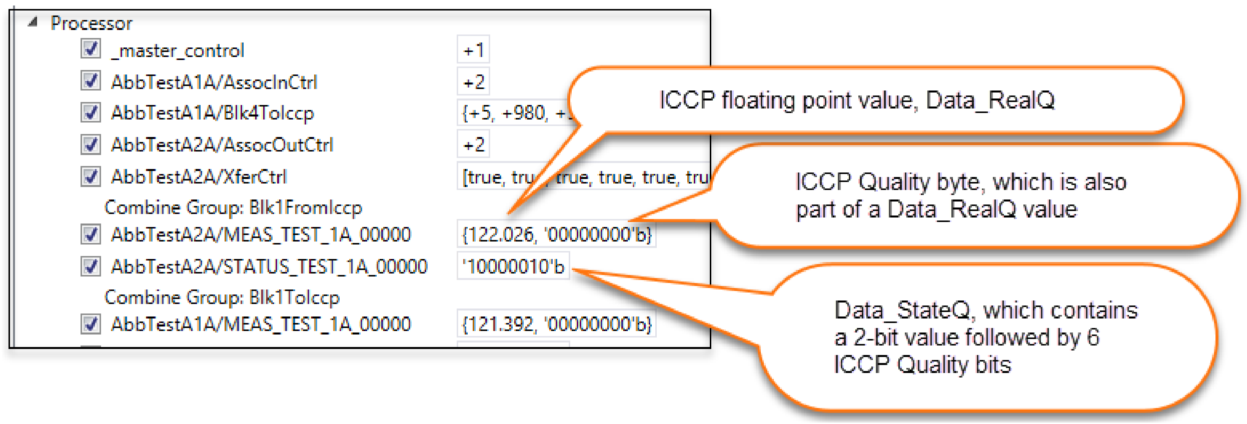
Blocks 1 & 2	PointType= REAL	PointRealValue	Float is defined as a 32-bit floating-point number with an eight-bit exponent.
Blocks 1 & 2	PointState= STATE	PointStateValue	Int32 is defined as a signed 32-bit integer.
Blocks 1 & 2	PointState=DISCRETE	PointDiscreteValue	Int32
Blocks 1 & 2	Quality	See the section called Quality Bits in LEC Configuration Manager for an explanation of the representation of quality bits.	See Table 4.
Blocks 1 & 2	Quality.Validity	0 = Valid, 1 = Held, 2 = Suspect, 3 = Invalid	
Blocks 1 & 2	Quality.CurrentSource	0 = Telemetered, 1 = Calculated, 2 = Entered, 3 = Estimated	
Blocks 1 & 2	Quality.NormalValue	0 = Normal, 1 = Abnormal	
Blocks 1 & 2	Quality.TimeStampQuality	0 = Valid, 1 = Invalid	
Blocks 1 & 2	Quality.State	0 = Between, 1 = Off, 2 =On, 3 = Invalid	
Blocks 1 & 2	TimeStamp	32-bit integer representing Universal Time Coordinated (UTC) in seconds since midnight January 1, 1970 00:00:00 in Greenwich, England and optionally a 16-bit integer for the number of milliseconds elapsed within the current second	ICCP time if available, if not, LEC Server's host machine time Regardless of the source of the timestamp, LEC Server converts it into an OPC UA timestamp.
Blocks 1 & 2	COV	16-bit unsigned integer that holds the change-of-value counter.	COV is exposed as a two-item OPC UA array, where the first item is the usual

			point information, and the second item is a Uint16 COV. Uint16 is an unsigned 16-bit integer.
Block 4 (not used as part of ICCP control)	Message	An octet string of 8 bits for binary data or characters.	Four-element, comma-separated string-type variable, where the elements are respectively the InfoReference, the LocalReference, the MessageId, and the message buffer.
Block 5	Control	See the OPC UA Control over ICCP Devices For information on tagging, see the Setting and Clearing Tags section.	
Block 7	Report	See the OPC UA Control over ICCP Devices section.	
Block 8	Not supported	Not supported	Not supported

Quality Bits in LEC Configuration Manager

The easiest way to see the ICCP quality bits returned for ICCP Block 1 and 2 data is to use the Node Monitor in LEC Configuration Manager. Figure 12 shows an example of the information that you can read from Processor nodes within the Node Monitor.

Figure 12: Node Monitor exposes Quality Bits of ICCP Types Data_RealQ and Data_StateQ



```
◢ Processor
    ☑ _master_control                      +1
    ☑ AbbTestA1A/AssocInCtrl               +2
    ☑ AbbTestA1A/Blk4ToIccp                {+5, +980, +
    ☑ AbbTestA2A/AssocOutCtrl              +2
    ☑ AbbTestA2A/XferCtrl                  [true, tru  true, true, true, tru
   Combine Group: Blk1FromIccp
    ☑ AbbTestA2A/MEAS_TEST_1A_00000        {122.026, '00000000'b}
    ☑ AbbTestA2A/STATUS_TEST_1A_00000      '10000010'b
   Combine Group: Blk1ToIccp
    ☑ AbbTestA1A/MEAS_TEST_1A_00000        {121.392, '00000000'b}
```

ICCP floating point value, Data_RealQ
ICCP Quality byte, which is also part of a Data_RealQ value
Data_StateQ, which contains a 2-bit value followed by 6 ICCP Quality bits

In Figure 12, a Data_RealQ value has the floating point value 122.026 and a Quality byte of '00000000.'

00	**00**	**00**	**0**	**0**
Good	Valid	Telemetered	Normal Value	Normal Timestamp

In Figure 12, a Data_StateQ value has a Quality byte of '10000010.'

10	**00**	**00**	**1**	**0**
On	Valid	Telemetered	Abnormal Value	Normal Timestamp

State is shown in the first pair of bits if the bit string represents a State value.

Bits	00	01	10	11
Meaning	Between	Off	On	Invalid

Validity is shown in the second pair of bits.

Bits	00	01	10	11
Meaning	Good or Valid	Held	Suspect	Bad or Invalid

Current Source is shown in the third pair of bits.

Bits	00	01	10	11
Meaning	Telemetered	Calculated	Entered	Estimated

Normal or Abnormal Value is shown in the next single bit.

Bits	0	1
Meaning	Normal	Abnormal

Valid or Invalid Timestamp is shown in the last single bit.

Bits	0	1
Meaning	Valid	Invalid

Mapping Quality Bit Values from ICCP to OPC UA Status Names and States for Block 1 Data

LEC Server matches incoming ICCP Quality bit values to the outgoing OPC UA status name and OPC UA state value if the incoming value is a State PointType. These Quality flags are returned with the ICCP Real, Discrete, and State PointTypes. Table 5 shows the outgoing OPC UA status names and state values that map to the ICCP Quality flags, State, Validity, Normal Value, and TimeStampQuality.

Table 4: Mapping ICCP Quality Bit Values (Flags) to OPC UA Statuses and States

ICCP State	ICCP Validity	ICCP CurrentSource	ICCP Normal Value	ICCP TimeStamp Quality	OPC UA Status Name	OPC UA State Value
No Problems						
0, 1, 2, 3	0	0	0	0 or 1	Good	Do not set state
Normal Value Problem						
0, 1, 2, 3	0	0	1	0 or 1	UncertainEngineeringUnitsExceeded	Do not set state
Current Source Problems						
0, 1, 2, 3	0	1	0 or 1	0 or 1	GoodEdited	Do not set state
0, 1, 2, 3	0	2	0 or 1	0 or 1	GoodLocalOverride	Do not set state
0, 1, 2, 3	0	3	0 or 1	0 or 1	GoodEdited	Do not set state
State Problems						
0	0	0, 1, 2, or 3	0 or 1	0 or 1	Uncertain	Do not set state
3	0	0, 1, 2, or 3	0 or 1	0 or 1	UncertainEngineeringUnitsExceeded	Do not set state
Validity Problems						
0, 1, 2, 3	1	0, 1, 2, or 3	0 or 1	0 or 1	UncertainLastUsableValue	Do not set state
0, 1, 2, 3	2	0, 1, 2, or 3	0 or 1	0 or 1	Uncertain	Do not set state
0, 1, 2, 3	3	0, 1, 2, or 3	0 or 1	0 or 1	BadUnexpectedError	Do not set state
Transfer State Values						
0	0	0, 1, 2, or 3	0 or 1	0 or 1	Do not set status	0
1	1	0, 1, 2, or 3	0 or 1	0 or 1	Do not set status	1
2	2	0, 1, 2, or 3	0 or 1	0 or 1	Do not set status	2

3	3	0, 1, 2, or 3	0 or 1	0 or 1	Do not set status	3

Mapping OPC UA Status Names and States to ICCP Quality Bit Values for Block 1 Data

LEC Server matches an incoming OPC status name and transfer state value to the outgoing ICCP Quality bit values and the State value if the incoming value is the State PointType. Table 5 shows the outgoing ICCP quality bit values, State, Validity, Normal Value, and TimeStampQuality that map to the incoming OPC status names and transfer state values.

Table 5: Converting OPC UA Statuses and States to ICCP Quality Bit Values

OPC UA Status Name	OPC UA State Value	State	Validity	Current Source	Normal Value	TimeStamp Quality	ICCP Description
No Problems							
Good	0, 1, 2, or 3	Don’t set	0	0	0	0	ICCP_Good Quality
Current Source Problem							
GoodEdited	0, 1, 2, or 3	Don’t set	0	2	0	0	ICCP_Entered
GoodLocalOverride	0, 1, 2, or 3	Don’t set	0	2	0	0	ICCP_Entered
Normal Value Problem							
UncertainEngineering	0, 1, 2, or 3	Don’t set	0	0	1	0	ICCP_AbnormalValue
Validity Problem							
UncertainLastUsableValue	0, 1, 2, or 3	Don’t set	1	0	0	0	ICCP_Held
Uncertain	0, 1, 2, or 3	Don’t set	2	0	0	0	ICCP_Suspect
Bad	0, 1, 2, or 3	Don’t set	3	0	0	0	ICCP_NotValid
BadUnexpectedError	0, 1, 2, or 3	Don’t set	3	0	0	0	ICCP_NotValid
BadWaitingForInitial	0, 1, 2, or 3	Don’t set	Don’t set	Don’t set	Don’t set	Don’t set	Do not pass data
Transfer State Values							
Any incoming status	0	0	Don’t set	Don’t set	Don’t set	Don’t set	

Any incoming status	1	1	Don’t set	Don’t set	Don’t set	Don’t set	
Any incoming status	2	2	Don’t set	Don’t set	Don’t set	Don’t set	
Any incoming status	3	3	Don’t set	Don’t set	Don’t set	Don’t set	
Default Values							
DEFAULT_GOOD	0, 1, 2, or 3	Don’t set	0	0	0	0	ICCP_Good Quality
DEFAULT_UNCERTAIN	0, 1, 2, or 3	Don’t set	2	0	0	0	ICCP_Suspect
DEFAULT_BAD	0, 1, 2, or 3	Don’t set	3	0	0	0	ICCP_NotValid

Mapping of ICCP Measurement Data Types to OPC UA Data Types for Block 1 Data

Table 6 shows how LEC Server maps ICCP (Block 1) measurement data types to equivalent OPC UA data types.

Table 6: Mapping of ICCP Data Types to OPC UA Data Types for Block 1

ICCP Data Type	Meaning	OPC UA Data Type Mapping
Data_State	Discrete 2-bit value	OPC UA UInt32
Data_StateQ	Discrete 2-bit value + 6 ICCP Quality bits	OPC UA UInt32 with ICCP Quality reflected in the OPC UA Status
Data_StateQTimeTag	Discrete 2-bit value + ICCP Quality bits + Timestamp	OPC UA UInt32 with ICCP Quality reflected in the OPC UA Status
Data_StateExtended	Discrete 2-bit value + ICCP Quality bits + COV	Two element OPC UA array type, where the first element is ICCP Quality, which is reflected in the OPC UA Status, and the second element is the Uint32 COV
Data_StateTimeTagExtended	Discrete 2-bit value + ICCP Quality bits + Extended timestamp	OPC UA UInt32 with ICCP Quality reflected in the OPC UA Status

Data_Discrete	Integer value 32-bit signed	OPC UA Int32
Data_DiscreteQ	Integer value 32-bit signed + ICCP Quality bits	OPC UA Int32 with ICCP Quality reflected in the OPC UA Status
Data_DiscreteQTimeTag	Integer value 32-bit signed + ICCP Quality + Timestamp	OPC UA Int32 with ICCP Quality reflected in the OPC UA Status
Data_DiscreteExtended	Integer value 32-bit signed + ICCP Quality bits + COV	Two element OPC UA array type, where the first element is ICCP Quality, and the second element is the Uint32 COV
Data_DiscreteTimeTagExtended	Integer value 32-bit signed + ICCP Quality bits + Extended timestamp	OPC UA Int32 with ICCP Quality reflected in the OPC UA Status
Data_Real	Float 32	OPC UA Float
Data_RealQ	Float 32 + ICCP Quality bits	OPC UA Float with ICCP Quality reflected in the OPC UA Status
Data_RealQTimeTag	Float 32 + ICCP Quality bits + Timestamp	OPC UA Float with ICCP Quality reflected in the OPC UA Status
Data_RealExtended	Float 32 + ICCP Quality bits + Current Source + COV	Two element OPC UA array type, where the first element is ICCP Quality, and the second element is the Uint32 COV
Data_RealTimeTagExtended	Float 32 + ICCP Quality bits + Current Source + Extended timestamp	OPC UA Float with ICCP Quality reflected in OPC UA Status

Mapping of ICCP Control and Tag Data Types to OPC Data Types for Block 5 Data

Table 7 shows how LEC Server maps ICCP (Block 5) command, setpoint, and tag data types to equivalent OPC UA data types.

Table 7: Mapping of ICCP Data Types to OPC UA Data Types for Block 5

Block 5 Controls and Tags	ICCP Data Type	OPC UA Data Type Mapping
Command	16-bit integer	OPC UA Int32

Setpoint	32-bit integer or Floating point	OPC UA Int32 OPC UA Float
Tag	A data structure containing three strings for tag kind, armed, and comment.	A ByteString containing three comma-separated fields: tag kind, armed, and comment

See the section of this document called Setting and Clearing Tags for more information on the specific tag fields.

Mapping between Returned ICCP Status Names and OPC UA Status Names for Block 5 Data

The mapping between ICCP status names and OPC UA status names allows the application's OPC UA server to see device control results. ICCP controls (Block 5) have no quality bits, but when an ICCP client receives an OPC UA device control, the ICCP client returns an MMS status name and number.

LEC Server maps these MMS status names and numbers to a 32-bit integer (shown in hex) that represents an OPC UA status name. If there is no mapping for an MMS status name, LEC Server gives it the default OPC UA status name BadUnknownResponse. The following table shows the OPC UA number that represents each of the MMS status names and codes returned by ICCP.

Table 8: Mapping of ICCP Data Type to OPC UA Data Type

32-bit Integer returned by LEC Server in hex	OPC UA Status Name for returned MMS Status NameC	MMS Status Name returned by ICCP	MMS Status Code returned by ICCP
0	Good	SUCCESS	-1
0x803F0000	BadObjectDeleted	INVALIDATED	0
0x808B0000	BadDeviceFailure	FAULT	1
0x80040000	BadResourceUnavailable	UNAVAILABLE	2
0x801F0000	BadUserAccessDenied	DENIED	3
0x80340000	BadNodeIdUnknown	UNDEFINED	4
0x80640000	BadSourceNodeIdInvalid	INVALID_ADDRESS	5
0x80110000	BadDataTypeIdUnknown	TYPE_UNSUPPORTED	6
0x80740000	BadTypeMismatch	TYPE_INCONSISTENT	7
0x80620000	BadNodeAttributesInvalid	ATTRIBUTE_INCONSISTENT	8
0x803D0000	BadNotSupported	ACCESS_UNSUPPORTED	9
0x803E0000	BadNotFound	NON_EXISTENT	10

0x80850000	BadRequestTimeout	TIMEOUT	11

Mapping between OPC UA Status Codes and ICCP Access Results for Block 5 Data

The mapping between OPC status codes and ICCP access results allows the ICCP server to see device control results. LEC Server provides a mapping between each OPC UA status name and each MMS status name in the file StandardOpcToIccp.csv. You can change these mappings by editing this file that is in the directory C: or D:\ProgramData\LiveEnergyConnect\Config.

Since there are far fewer MMS status numbers/names than OPC UA status names, more than one OPC UA status name has to be mapped to a single MMS status name. If an OPC UA name is not mapped to an MMS status name, LEC Server will map the OPC UA status name to the MMS status name as shown in the DEFAULT row of the table.

Table 9: Mapping of OPC Status Names to MMS Status Names and Numbers

OPC UA Status Name	MMS Status Name	MMS Number
GoodCompletesAsynchronously	IN_PROCESS	-2
Good	SUCCESS	-1
GoodLocalOverride	SUCCESS	-1
GoodEntryInserted	SUCCESS	-1
GoodEntryReplaced	SUCCESS	-1
GoodNonCriticalTimeout	SUCCESS	-1
BadObjectDeleted	INVALIDATED	0
BadDeviceFailure	FAULT	1
BadSensorFailure	FAULT	1
BadResourceUnavailable	UNAVAILABLE	2
BadNoCommunication	UNAVAILABLE	2
BadInvalidState	UNAVAILABLE	2
BadOutOfService	UNAVAILABLE	2
BadStateNotActive	UNAVAILABLE	2
BadUserAccessDenied	DENIED	3
BadRequestTypeInvalid	DENIED	3
BadMethodInvalid	DENIED	3
BadNodeIdUnknown	UNDEFINED	4
BadSourceNodeIdInvalid	INVALID_ADDRESS	5

BadDataTypeIdUnknown	TYPE_UNSUPPORTED	6
BadNotImplemented	TYPE_UNSUPPORTED	6
BadDataEncodingUnsupported	TYPE_UNSUPPORTED	6
BadTypeMismatch	TYPE_INCONSISTENT	7
BadIndexRangeInvalid	TYPE_INCONSISTENT	7
BadOutOfRange	TYPE_INCONSISTENT	7
BadNodeAttributesInvalid	ATTRIBUTE_INCONSISTENT	8
BadDataEncodingInvalid	ATTRIBUTE_INCONSISTENT	8
BadArgumentsMissing	ATTRIBUTE_INCONSISTENT	8
BadDeadbandFilterInvalid	ATTRIBUTE_INCONSISTENT	8
BadAttributeIdInvalid	ATTRIBUTE_INCONSISTENT	8
BadInvalidArgument	ATTRIBUTE_INCONSISTENT	8
BadNotSupported	ACCESS_UNSUPPORTED	9
BadNotWritable	ACCESS_UNSUPPORTED	9
OPC UA Status Name	MMS Status Name	MMS Number
BadWriteNotSupported	ACCESS_UNSUPPORTED	9
BadNotFound	NON_EXISTENT	10
BadTimeout	TIMEOUT	11
BadRequestTimeout	TIMEOUT	11
DEFAULT	INVALID_ADDRESS	5

Header Batch File Definitions

The header batch file sets up the Master Control, Heartbeat, and ServiceLevel variables, and Input and Output Control points as well as the server and client agents. You must load the header batch file from the machine on which LEC Server is installed after the configuration .db file has been imported and prior to any other batch files. (See the section of this document called *Configuration Setup, Startup, and Remote Batch Load* section for more information on loading batch files.) This batch file is called AppTestAHeader.csv, and by default it is located in C:\ProgramData\LiveEnergyConnect\Config. The following section defines each table and row in the example header batch file.

#IccpUaServerAgent

This table defines the agent in the OPC UA server VMD.

```
#IccpUaServerAgent, <ServerIP>, <PublishingInterval>, <ServerName>, <CodeMapFile>,
<AppCertId>,   <AuthorizedCerts>, <SecurityOpts>, <PointsPerSub>, <MaxSubs>,
<SubsPerSession>, <MaxSessions>,<push_by_status>,<AssocDownStatus>
```

```
AgentS, opc.tcp://127.0.0.1:4842/server/,,,,, *, NONE, , 50, 10, 50,0,BadCommunicationError
```

Table 10: Properties and Values of the OPC UA Server Agent

Node or Property	Value	Defines
#IccpUaServerAgent	AgentS	Name given to the OPC UA server agent node.
<ServerIP>	opc.tcp://127.0.0.1:4842/server/	The URI address of the OPC UA server. The application's OPC UA client would connect to the OPC UA server by specifying this address.
<PublishingInterval>		Minimum time in seconds between the transmissions of subscribed data. By default, the OPC UA server agent will send transmissions immediately.
<ServerName>		Name of the OPC UA server. The default name is LiveData_OPC_UA.
<CodeMapFile>		The path of the ICCP-OPC code mapping file. The default is StandardOpcIccp.csv. See the section called Mapping between Returned ICCP Status Names and OPC UA Status Names for Block 5 Data.
<AppCertId>		Descriptor that specifies the certificate to use from the Windows Certificate Store. No value is given since the certificate used in this example does not come from the Windows Certificate Store.
<SecurityOpts>	NONE	NONE specifies the security mode. There are other security mode options available.
<PointsPerSub>		The maximum number of points per subscription. The default is 10,000.
<MaxSubs>	50	The total number of subscriptions permitted for this agent. The default is 100.
<SubsPerSession>	10	The number of subscriptions that are allowed per session.
<MaxSessions>	50	The maximum number of sessions per connection. The default is 10.
<push_by_status>	0	Enables or disables (default) OPC UA server toggling unused bits in the OPC UA status field to force otherwise unchanged data propagate.

<AssocDownStatus>	BadCommunicationError	Specifies which OPC UA status code to apply to OPC UA server points when the source ICCP association is down

#IccpUaClientAgent

This table defines the agent in the OPC UA client VMD.

```
#IccpUaClientAgent, <ServerIP>, <PublishingInterval>, <ServerName>, <NS_URI>,
<CodeMapFile>, <AppCertId>, <SecurityOpts>, <PointsPerSub>, <MaxSubs>,
<SubsPerSession>, <MaxSessions>
,<StaleTime>,<TimeoutValidBit>,<ConnStatusBranch>,<ConnStatusNS>,<ConnStatusOpcLabe
l>,<ConnStatusServerAgent>,<InactiveTimeout>

AgentC, opc.tcp://127.0.0.1:4840/server/,,Test,,,, NONE, , 50, 10, 50 , 30000,
|VALID_HIGH|VALID_LOW,RTISystem, SYSTEM, OpcClientConnectionStatusForAgentC,
AgentS
```

Node or Property	Value	Defines
#IccpUaClientAgent	AgentC	Name given to the OPC UA client agent node.
<ServerIP>	opc.tcp://127.0.0.1:4840/server/	The URI address of the OPC UA client. The application's OPC UA server would connect to this client by specifying this address.
<PublishingInterval>		Minimum time in seconds between the transmissions of subscribed data. By default, the OPC UA server agent will send transmissions immediately.
<ServerName>	Test	The name of the OPC UA server to which the OPC UA client will connect.
<NS_URI>		URI for all points under this agent.
<CodeMapFile>		The path of the ICCP-OPC UA code mapping file. The default is StandardOpcIccp.csv. See the section called Mapping between Returned ICCP Status Names and OPC UA Status Names for Block 5 Data.
<AppCertId>		Descriptor that specifies the certificate to use from the Windows Certificate Store.
<AuthorizedCerts>	*	* is a wildcard that specifies all certificates except for those in

		the Windows Certificate Store. There are additional options for specifying certificates.
<SecurityOpts>	NONE	NONE specifies the security mode. There are other security mode options available.
<PointsPerSub>		The maximum number of points per subscription. The default is 10,000.
<MaxSubs>	50	The total number of subscriptions permitted for this agent. The default is 100.
<SubsPerSession>	10	The number of subscriptions that are allowed per session.
<MaxSessions>	50	The maximum number of sessions per connection. The default is 10.
<StaleTime>	30000	Number of milliseconds before data is considered stale.
<TimeoutValidBit>	VALID_HIGH|VALID_LOW	Which ICCP quality validity bits to set to indicate data is stale.
<ConnStatusBranch>		OPC UA server branch for connection status variable for this OPC UA client.
<ConnStatusNS>		OPC UA server namespace for connection status variable for this OPC UA client.
<ConnStatusOpcLabel>		Alternative name to get the connection status OPC UA server variable.
<ConnStatusServerAgent>	AgentS	Name of OPC UA server agent where client connection status variable is available.
<InactiveTimeout>		Timeout in milliseconds data inactivity. Change client connection status to 4 when timed out

#Master Control

Each IFE hosts a Master Control variable that toggles the instance between Active and Passive modes. A remote OPC UA client can write to this Master Control variable. If an application's OPC UA client writes 1 to a Master Control variable, the IFE instance is set to Active mode; conversely, if an application's OPC UA client writes 0 to a Master Control variable, the IFE instance is set to Passive mode. The #MasterControl table defines this variable, which is exposed in the MasterControlOutForAgentS and MasterControlInForAgentS nodes.

```
#MasterControl,     <agent>,   <branch>,   <NS_Suffix>

MasterControl,      AgentS,   LECSystem,   SYSTEM
```

Prototype Node or Property	Value	Defines
#MasterControl	MasterControl	The name of the master control variable.
<agent>	AgentS	The label of the OPC UA server agent that handles communication for the nodes that have access to the Master Control variable. These are the MasterControlOutForAgentS and MasterControlInForAgentS nodes.
<branch>	LECSystem	The OPC UA branch where this point is located. Leaving blank indicates no branch.
<NS_Suffix>	SYSTEM	The OPC UA namespace suffix for the MasterControl node. Leaving blank indicates no NS suffix.

#Heartbeat

The OPC UA server in each LEC Server instance has a heartbeat variable that is updated every 10 seconds. The #Heartbeat table defines the Heartbeat variable within the OPC UA server's HeartbeatForAgentS node.

```
#Heartbeat,    <agent>,    <branch>,   <NS_Suffix>

Heartbeat,     AgentS,     LECSystem,  SYSTEM
```

Prototype Node or Property	Value	Defines
#Heartbeat	Heartbeat	The OPC UA variable name within HeartbeatForAgentS.
<agent>	AgentS	The label of the OPC UA server agent that handles communication for the Heartbeat node.
<branch>	LECSystem	The OPC UA branch where this point is located. Leaving blank indicates no branch.
<NS_Suffix>	SYSTEM	The OPC UA namespace suffix for the Heartbeat node. Leaving blank indicates no NS suffix.

#HeartBeatFromApp

Each application's OPC UA server writes to a heartbeat variable in an OPC UA client within LEC Server. The application's OPC UA server updates the value of the variable every 10 seconds. The local OPC UA client monitors this heartbeat value, and if it is not updated in 30 seconds, the LEC Server instance will go offline. The #HeartBeatFromApp table defines the HeartbeatFromApp variable within the local OPC UA client.

```
#HeartBeatFromApp, <active>, <agent>, <branch>, <NS_Suffix>, <type>, <timeout>, <tag>,
<opc_label>

HeartbeatFromApp, 0, AgentC, LECSystem, SYSTEM, <int:32>, 30000, Heartbeat,
HeartbeatFromApp
```

Prototype Node Name or Property	Node Label or Property Value	Defines
#HeartBeatFromApp	HeartbeatFromApp	A node within the OPC UA server that gets data (a heartbeat) from an application's OPC UA client.
<active>	0	0 indicates that the Heartbeat variable is not actively monitored, meaning that LEC Server will not revert to Passive mode even if the Heartbeat variable ceases to increment. 1 would indicate that the Heartbeat variable is actively monitored. If the Heartbeat variable is actively monitored, the LEC Server instance would revert to Passive mode if the Heartbeat variable ceases to increment. **Note**: This is an internal variable required by the script that is referenced in the #HeartBeatFromAppMonitor node.
<agent>	AgentC	The label of the OPC UA client agent that handles communication for the HeartbeatFromApp node.
<branch>	LECSystem	The OPC UA branch where this point is located. Leaving blank indicates no branch.
<NS_Suffix>	SYSTEM	Specifies the OPC UA namespace suffix for the HeartbeatFromApp node. Leaving blank indicates no NS suffix.
<type>	<int:32>	LEC data type of the HeartbeatFromApp node.
<timeout>	30000	Specifies the time in milliseconds that the IFE will wait for the application's heartbeat to increment. **Note**: This particular value is internal to the workings of the IFE.
<tag>	Heartbeat	The name of the variable that is to be passed to the HeartBeatFromAppMonitor script.
<opc_label>	HeartbeatFromApp	The OPC UA variable name that contains the counter number. This variable name overrides the default name. The default name is the same as the node label.

#ConfigControlIn

The #ConfigControlIn table defines the input control point that a remote application's OPC UA client can use to modify an IFE configuration. DBLoadRequest is currently the only input control point. For more information, see Loading of Batch Files.

```
#ConfigControlIn,         <agent>,    <branch>,   <NS_Suffix>,    <type>, <tag>

DBLoadRequest,            AgentS,     LECSystem,  SYSTEM,        <v-s:-128>, DBLoadRequest
```

Prototype Node Name or Property	Node Label or Property Value	Defines
#ConfigControlIn	DBLoadRequest	The label of the input variable that sets the path to a command file which will load batch files. Writing to this variable triggers the execution of the command file.
<agent>	AgentS	The label of the OPC UA server agent that handles communication for the DBLoadRequest variable.
<branch>	LECSystem	The OPC UA branch where the DBLoadRequest variable is located. Leaving blank indicates no branch.
<NS_Suffix>	SYSTEM	Specifies the OPC UA namespace suffix for the DBLoadRequest variable. Leaving it blank indicates no NS suffix.
<type>	<v-s:-128>	LEC data type of the DBLoadRequest variable.
<tag>	DBLoadRequest	The name of the variable that is to be passed to the DBLoader script for the DBLoadRequest variable.

#ConfigControlOut

This table defines the output variables that the OPC UA server uses to return error message data to the OPC UA client. For more information, see the section of this documents called *Loading of Batch Files.*

```
#ConfigControlOut,         <agent>,    <branch>,   <NS_Suffix>,    <type>,    <tag>

DBLoadCommandLine, AgentS, LECSystem, SYSTEM, <v-s:-128>, DBLoadCommandLine

DBLoadErrorString, AgentS, LECSystem, SYSTEM, <v-s:-128>, DBLoadErrorString

DBLoadState,  AgentS, LECSystem, SYSTEM, <int:32>, DBLoadState

DBLoadRequestedCommandFile, AgentS, LECSystem, SYSTEM, <v-s:-128>,
DBLoadRequestedCommandFile

DBLoadCurrentlyLoaded, AgentS, LECSystem, SYSTEM, <v-s:-128>, DBLoadCurrentlyLoaded
```

Prototype Node Name or Property	Node Label/Output Variable or Property Value	Defines
#ConfigControlOut	DBLoadCommandLine	The label of the output variable that returns the line on which the error occurred in the command file if an error occurred. **DBLoadErrorString** is the label of the output variable that returns a string with the load error message if there is one.

		DBLoadState is the label of the output variable that indicates the state of the load. 0 is OK; 1 is failed, and -1 is busy. **DBLoadRequestedCommandFile** is the label of the output variable that contains the name of the command file to which DBLoadState refers. **DBLoadCurrentlyLoaded** is the label of the output variable containing the name of the last command file that loaded successfully; this variable is set just before DBLoadState changes to 0. If the load fails, DBLoadState is set to 1.
<agent>	AgentS	The label of the OPC UA server agent that handles communication for this point and for the DBLoadErrorString, DBLoadState, DBLoadRequestedCommandFile, and DBLoadCurrentlyLoaded variables.
<branch>	LECSystem	The OPC UA branch where this point and the DBLoadErrorString, DBLoadState, DBLoadRequestedCommandFile, and DBLoadCurrentlyLoaded variables are located. Leaving blank indicates no branch.
<NS_Suffix>	SYSTEM	Specifies the OPC UA namespace suffix for this point as well as for the DBLoadErrorString, DBLoadState, DBLoadRequestedCommandFile, and DBLoadCurrentlyLoaded variables. Leaving blank indicates no NS suffix.
<type>	<v-s:-128>	Specifies the OPC UA data type for this point and for the DBLoadErrorString, DBLoadRequestedCommandFile, and DBLoadCurrentlyLoaded variables. **Note**: The data type for DBLoadState is <int:32>.
<tag>	DBLoadCommandLine	The name of the variable that is to be passed to the DBLoader script for the DBLoadCommandLine point. In addition, the DBLoadErrorString, DBLoadState, and DBLoadCurrentlyLoaded variables pass the DBLoadErrorString, DBLoadState, and DBLoadCurrentlyLoaded tags to the DBLoader script, respectively. **Note:** These particular values are set in points on the ConfigLoader VMD, which is internal to the workings of the IFE.

VCC, VMD, Variables, and Node Batch File Definitions for AppTestA1 CSV

The following section shows an example batch file AppTestA1.csv that describes local and remote VCCs as well as properties, nodes, and transfer sets. AppTestA1.csv contains the tables that define the VCCs, VMDs, association control variables, and device points that allow for the following types of communication:

- An application’s OPC UA server to pass Block 1 and Block 2 data to a remote ICCP client.
- An application’s OPC UA server to pass Block 4 data to a remote ICCP client.
- A remote ICCP client to pass Block 5 controls to an application’s OPC UA server.

#sharemode

The share mode is set to none, meaning that its content is not for use by other batch files. It is set at the beginning of the batch file.

#sharemode

none

#My_VCC

The #My_VCC table defines one or more local VCCs, along with their association configuration parameters.

#My_VCC, <local_dom>, <remote_vcc>, <remote_dom>,<assoc_out>,<assoc_in>,<client>,<server>, <iccp_features>, <security_flags>, <assoc_verify>,<codemap_file>

AppTextA1A, AppLocDom, AppTestA2A, AppRemDom, 0, 1, 0, 1, 110110000000, 0, 1,AbbCodeMapping.csv

Prototype VCC or Property	Value	In the VccCreate template, defines or indicates
#My_VCC	AppTestA1A	The common name of the local ICCP virtual device, also known as a VCC.
<local_dom>	AppLocDom	The local domain name for server-side data.
<remote_vcc>	AppTestA2A	The common name of the remote VCC.
<remote_dom>	AppRemDom	The remote domain name for server-side data.
<assoc_out>	0	A Boolean value (0 or 1) that indicates whether the VCC can make an outbound association or not. 0 prevents this VCC from making an outbound association.
<assoc_in>	1	A Boolean value (0 or 1) that indicates whether the VCC can make an inbound association or not. 1 enables an inbound association.
<client>	0	A Boolean value (0 or 1) that indicates whether this VCC can serve the client role or not. 0 prevents this VCC from serving the client role. **Note**: A VCC can be configured to be both a client and a server.
<server>	1	A Boolean value (0 or 1) that indicates whether this VCC can serve the server role or not.

		1 enables this VCC to play the server role.
<iccp_features>	110110000000	Locally supported features represented with 12 1s or 0s, or a combination of both. The features represent the blocks that are supported by the VCC. These features must be the same as the remote node's features.
<security_flag>	0	A Boolean value (0 or 1) that indicates whether this VCC utilizes secure ICCP or not. 0 indicates that this VCC does not use secure ICCP.
<assoc_verify>	1	A Boolean value (0 or 1). If <assoc_verify> is set to 1, the VCC periodically requests the peer VCC to identify itself. If the request is not acknowledged by the time specified in the timeout parameters, then the ICCP server will abort the association. The ICCP server can then establish a new association.
<codemap_file>	AbbCodeMapping.csv	Text name of alterative codemap file for mapping ICCP quality to/from OPC UA status. Overrides default and any OPC UA server or client setting for this VCC's points.

#remote_vcc

The #remote_vcc table defines a named list of up to six remote ICCP peer VCCs for incoming or outgoing associations. Note that the name is only a local identifier and not part of the exchanged ICCP association information.

#remote_vcc,<remote_vcc_1>,<remote_vcc_2>,<remote_vcc_3>,<remote_vcc_4>,<remote_vcc_5>,<remote_vcc_6>

AppTestA2A

Prototype VCC or Property	Value	Defines or indicates the names of the remote VCCs associated with this instance of LEC Server
#remote_vcc	AppTestA2A	The common name of the remote VCC. This remote VCC is configured to have an association with AppTestA1A.
<remote_vcc_1>	Not set by this batch file.	The common name of another remote VCC. This remote VCC will also be configured to have an association AppTestA1A.
<remote_vcc_2>	Not set by this batch file.	
<remote_vcc_3>	Not set by this batch file.	
<remote_vcc_4>	Not set by this batch file.	
<remote_vcc_5>	Not set by this batch file.	
<remote_vcc_6>	Not set by this batch file.	

#CommonName

The #CommonName table defines the association parameters for each local and remote VCC defined in the #My_VCC and #remote_vcc tables. You can see these settings in the LDIB Editor.

#CommonName,IpAddress,TSEL,SSEL,PSEL,ApTitle,AeQualifier,Secure

AppTestA1A, ,01,   01,   01,   1 3 9999 111,   102, 0

AppTestA2A, ,2,   2,   2,   1 3 9999 111,   102, 0

Prototype VCC or Property	Row 1	Row 2	Defines or indicates addressing and other values in the LDIB
#CommonName	AppTestA1A	AppTestA2A	The common names of two different VCCs
IpAddress	Not set by the batch file.	Not set by the batch file.	The IP addresses of each of the named VCCs
TSEL	01	02	Transport selector
SSEL	01	02	Session selector
PSEL	0001	0002	Presentation selector
ApTitle	1 3 9999 111	1 3 9999 111	The ISO object IDs of the 3 devices
AeQualifier	102	102	Application entity qualifier[10]
Secure	0	0	Enables Secure ICCP if it is set to 1

#AssocInControl

The #AssocInControl table defines the association control variable and its configuration parameters.

#AssocInControl

In the #AssocInControl example, the following schematic shows the connection between the ICCP AssocInControl node and the OPC UA client node that has the specified Agent, Branch, and Namespace properties.

#AssocInControl	$\longrightarrow$	AppTestA1A/AssocInCtrl
(which is in a VCC)		(which is in the OPC UA server)

AppTestA1A/AssocInControl allows an external OPC UA server that is connected to the local OPC UA client to enable or disable an ICCP association. This connection is key to receiving

[10] The original ISO model terminology talked about **entities** in each layer that communicated, using the services of the layer below. There were network entities, transport entities, session entities, presentation entities, and application entities. Application entity (Ae) refers to the top-level layer, known as the application layer. An application entity is the Open System Interconnection (OSI) portion of an Application Process (AP).

Block 5 controls from a remote ICCP point or sending Block 1 and Block 2 data to this remote point.

#AssocInControl, <agent>, <branch>, <NS_Suffix>, <stale_time>, <timeout_valid_bit>

AssocInCtrl, AgentS, LECSystem, SYSTEM, 30000, |VALID_LOW|VALID_HIGH

Prototype Node Name or Property	Value	In the VccAssocControl template or OpcUaAgent template
#AssocInControl	AssocInCtrl	Specifies the name of the node that controls inbound associations.
<agent>	AgentS	Specifies the agent label. There is no Agent label property for AssocInCtrl, but AssocInCtrl connects to a node AppTestA1A/AssocInCtrl in the OPC UA server that requires an Agent label.
<branch>	LECSystem	The OPC UA branch where OPC UA server points are located. There is no Branch property for AssocInCtrl, but AssocInCtrl connects to a node AppTestA1A/AssocInCtrl in the OPC UA server that require an Agent branch. Leaving blank indicates no branch.
<NS_Suffix>	SYSTEM	Specifies the OPC UA namespace suffix for AppTestA1A/AssocInCtrl. Leaving blank indicates no NS suffix.
<stale_time>	30000	Specifies the period of time in milliseconds that is allowed to elapse while there is no association. After the period of time has elapsed, the association will time out, and input points from this association are assigned a quality code based on the configured timeout validity bit settings: VALID_LOW and VALID_HIGH.
<timeout_valid_bit>	|VALID_LOW|VALID_HIGH	When the VCC is set to the Alarm state, specifies whether the VALID_LOW or the VALID_HIGH flag is set or if both are set.

#OutboundPeerName and #InboundPeerName

The #OutboundPeerName and #InboundPeerName table defines the OPC UA server point that provides the text name of the connected outbound or inbound peer VCC.

#OutboundPeerName, <agent>, <branch>, <NS_Suffix>, <opc_label>

OutboundPeer, AgentS, RTISystem, ,

#InboundPeerName, <agent>, <branch>, <NS_Suffix>, <opc_label>

InboundPeer, AgentS, RTISystem, ,

Prototype Node Name or Property	Value	In the VccAssocControl template or OpcUaAgent template
#OutboundPeerName/InboundPeerName	OutboundPeer/InboundPeer	Specifies the name of the node that reports name of outbound/inbound peer
<agent>	AgentS	Specifies the agent label. There is no Agent label property for AssocInCtrl, but AssocInCtrl connects to a node AppTestA1A/AssocInCtrl in the OPC UA server that requires an Agent label.
<branch>	LECSystem	The OPC UA branch where OPC UA server points are located. There is no Branch property for AssocInCtrl, but AssocInCtrl connects to a node AppTestA1A/AssocInCtrl in the OPC UA server that require an Agent branch. Leaving blank indicates no branch.

#Blk1ToIccp

The #Blk1ToIccp table defines the local ICCP server points that receive telemetry and status data from an external OPC UA. This data is transferred to a peer ICCP client.

$$\begin{tikzcd}
\text{External OPC UA server} \arrow[r] & \text{Local OPC UA client} \arrow[r] & \text{Local ICCP server} \arrow[r] & \text{Remote ICCP client}
\end{tikzcd}$$

#Blk1ToIccp, <NS_Suffix>, <branch>, <type>, <agent>, <dom_scope>, <opc_label>

MEAS_TEST_1A_00000, Reals, , Data_RealQTimeTagExtended, AgentC, 0, "DevMeas_1A_00000"

MEAS_TEST_1A_FI1, Reals, , Data_RealQ, AgentC, 0, "DevMeas_1A_FI"

MEAS_TEST_1A_FI2, Reals, , Data_RealQ, AgentC, 0, "DevMeas_1A_FI"

STATUS_TEST_1A_00000, Statuses, , Data_StateQTimeTagExtended, AgentC, 1, "DevStatus_1A_00000"

STATUS_TEST_1A_FI1, Statuses, , Data_StateQ, AgentC, 0, "DevStatus_1A_FI1"

STATUS_TEST_1A_FI2, Statuses, , Data_StateQ, AgentC, 0, "DevStatus_1A_FI2"

Prototype Node Name or Property	Rows 1 through 6	Defines or indicates
#Blk1ToIccp	MEAS_ 1A_00000 MEAS_TEST_1A_FI1 MEAS_TEST_1A_FI2	Specifies local ICCP node names that receive data from the external OPC UA server. The nodes beginning with MEAS

	STATUS_TEST_1A_00000 STATUS_TEST_1A_FI1 STATUS_TEST_1A_FI2	receive telemetry data, and the nodes beginning with STATUS receive status data.
<NS_Suffix>	Reals Reals Reals Statuses Statuses Statuses	Specifies the OPC UA namespace suffix for the external OPC UA server nodes where the telemetry and status data originate. Leaving blank indicates no NS suffix.
<branch>	Not set.	Specifies the OPC UA branch on the external OPC UA server where the telemetry and status data is located. Leaving blank indicates no branch.
<type>	E Data_RealQTimeTagExtended Data_RealQ Data_RealQ Data_StateQTimeTagExtended Data_StateQ Data_StateQ	ICCP data type of the data in the ICCP nodes: MEAS_TEST_1A_00000 MEAS_TEST_1A_FI1 MEAS_TEST_1A_FI2 STATUS_TEST_1A_00000 STATUS_TEST_1A_FI1 STATUS_TEST_1A_FI2
<agent>	AgentC	Label that specifies the client OPC UA agent node that handles the connection to the application's OPC UA server.
<branch>	RTISystem	The OPC UA server branch where the variable is located. Leaving blank indicates no branch.
NS_Suffix>		The OPC UA server namespace suffix for this variable. Leaving blank indicates no namespace suffix.
<opc_label>		Alternative OPC UA server variable name for this point.

#Blk5FromIccp

The #Blk5FromIccp table defines local ICCP Block 5 setpoint nodes that originate on a remote ICCP client; the LEC Server instance writes these setpoint values to nodes on an external OPC UA server.

$$\begin{tikzcd}
\text{Remote ICCP client} \arrow[r] & \text{Local ICCP server} \arrow[r] & \text{Local OPC UA client} \arrow[r] & \text{External OPC UA server}
\end{tikzcd}$$

#Blk5FromIccp, <NS_Suffix>, <branch>, <tagging_branch>, <type>, <flags>, <tagging_point>, <agent>, <ChkBkId>, <dom_scope>, <opc_label>

CONTROL_TEST_1A_00000, Discretes, ASYS, , <int:16>, DISCRETE, , AgentC, 681, 0, "DevCtrl_1A_00000"

CONTROL_TEST_1A_00001, Discretes, ASYS, , <int:16>, DISCRETE|SBO, , AgentC, 681, 0, "DevCtrl_1A_00001"

CONTROL_TEST_ 00002, Discretes, ASYS, , <int:16>, DISCRETE|TAGABLE, , AgentC, 681, 0, "DevCtrl_1A_00002"

CONTROL_TEST_ 00003, Discretes, ASYS, , <int:16>, DISCRETE|TAGABLE|SBO, , AgentC, 681, 0, "DevCtrl_1A_00003"

CONTROL_TEST_FO1, Discretes, ASYS, , <int:16>, DISCRETE| SBO, , AgentC, 681, 0, "DevCtrl_1A_FO1"

CONTROL_TEST_FO2, Discretes, ASYS, , <int:16>, DISCRETE| SBO, , AgentC, 681, 0, "DevCtrl_1A_FO2"

SETPOINT_TEST_1A_00000, Reals, ASYS, , <f-p:32:8>, REAL, , AgentC, 681, 0, "DevSet_1A_00000"

SETPOINT _TEST_1A_00001, Reals ASYS, , <f-p:32:8>, REAL|SBO, , AgentC, 9, 0, "DevSet_1A_00001"

SETPOINT _TEST_1A_00002, Reals, ASYS, , <f-p:32:8>, REAL|TAGABLE, , AgentC, 9, 0, "DevSet_1A_00002"

SETPOINT_TEST_1A_00003, Reals, ASYS, , <f-p:32:8>, REAL|TAGABLE|SBO, , AgentC, 9, 0, "DevSet_1A_00003"

SETPOINT_TEST_1A_FO1, Reals, ASYS, , <f-p:32:8>, REAL, , AgentC, 9, 0, "DevSet_1A_FO1"

SETPOINT_TEST_1A_FO2, Reals, ASYS, , <f-p:32:8>, REAL, , AgentC, 9, 0, "DevSet_1A_FO2"

Prototype Node Name or Property	Rows 1 through 12	Defines or indicates
#Blk5FromIccp	CONTROL_TEST_1A_00000 CONTROL_TEST_1A_00001 CONTROL_TEST_1A_00002 CONTROL_TEST_1A_00003 CONTROL_TEST_1A_ FO1 CONTROL_TEST_1A_ FO2 SETPOINT_TEST_1A_00000 SETPOINT_TEST_1A_00001 SETPOINT_TEST_1A_00002 SETPOINT_TEST_1A_00003 SETPOINT_TEST_1A_ FO1 SETPOINT_TEST_1A_ FO2	Defines names of the local ICCP nodes that transmit control data from the ICCP peer through a two-way connector.
<NS_Suffix>	Discretes Discretes Discretes	Specifies the OPC UA namespace suffix for the external OPC UA server points that

	Discretes Discretes Discretes Reals Reals Reals Reals Reals Reals	receive the ICCP controls. Leaving blank indicates no NS suffix.
<branch>	ASYS	The OPC UA branch where these OPC UA server points are located. Leaving blank indicates no branch. In this example, all of the receiving server points are located in the ASYS branch.
<tagging_branch>	Not set.	OPC UA tagging point branch name; the default is the OPC UA control point branch name.
<type>	<int:16> <int:16> <int:16> <int:16> <int:16> <int:16> <f-p:32:8> <f-p:32:8> <f-p:32:8> <f-p:32:8> <f-p:32:8> <f-p:32:8>	Specifies the data type of the external, receiving OPC UA server nodes. All of the first set of points are <int:16>. All of the next set of points are <f-p:32:8>.
<flags>	DISCRETE DISCRETE|SBO DISCRETE|TAGABLE DISCRETE|TAGABLE|SBO DISCRETE|SBO DISCRETE|SBO REAL REAL|SBO REAL|TAGABLE REAL|TAGABLE|SBO REAL REAL	Specifies the flag or flags of the receiving ICCP control points within LEC Server: REAL – The control point is a floating-point number DISCRETE – The control point is an integer SBO – The control point supports Select Before Operate TAGABLE – The control point supports tagging
<tagging_point>	Not set.	OPC UA control tagging point name; the default is the OPC UA control point name with _TAG appended.
<agent>	AgentC	Label of the local OPC UA client agent node to handle the connection with the external OPC UA server.
<ChkBkId>	681 681 681 681 681 681 9 9 9 9 9 9	Expected Check back ID. The expected Check back ID for the first set of control points is 681. The expected Check back ID for the second set of control points is 9.
<dom_scope>	0	Indicates the scope of the variables in the remote VCC. The scope in that VCC is VMD-specific as indicated by <dom_scope> of 0 for all points. A <dom_scope> of 1 would indicate Domain-

		specific scope, which refers to the domain specified by <remote_dom> in the #My_VCC table.
<opc_label>	"DevCtrl_1A_00000" "DevCtrl_1A_00001" "DevCtrl_1A_00002" "DevCtrl_1A_00003" "DevCtrl_1A_ FO1" "DevCtrl_1A_ FO1" "DevSet_1A_00000" "DevSet_1A_00001" "DevSet_1A_00002" "DevSet_1A_00003" "DevSet_1A_ FO1" "DevSet_1A_ FO1"	OPC UA control point labels. This variable name overrides the default name. The default name is the same as the node label.

#ConfirmedBlk4ToIccp

The #ConfirmedBlk4ToIccp table defines local ICCP Block 4 messaging points to hold data that originate from an external OPC UA client; the local ICCP server sends each message to the remote ICCP client points. The underlying ConfirmedMessageToIccp node receives the message from the local OPC UA server node, which is a ConfirmedMessageFromOpcUa node, through a two-way connector. The ConfirmedMessageToIccp node returns a confirmation to the originating OPC UA client through the ConfirmedMessageFromOpcUa node. This confirmation indicates only that the message got to a local ICCP server and is likely to get to the remote ICCP point, but the remote ICCP client sends no confirmation indicating whether or not it received the message.

$$\begin{tikzcd}
\text{External OPC UA client} \arrow[r] & \text{Local OPC UA server} \arrow[r] & \text{Local ICCP server} \arrow[r] & \text{Remote ICCP client}
\end{tikzcd}$$

```
#ConfirmedBlk4ToIccp,   <NS_Suffix>, <branch>,  <InfoRef>,  <LocalRef>,   <agent>,
<dom_scope>, <opc_label>

MSG_TEST_1A_1_2,   Messages,          ,       1,          2,       AgentS,    0,
DevMess_1A_1_2

MSG_TEST_1A_X_4,   Messages,          ,        ,          4,       AgentS,    0,
DevMess_1A_X_4

MSG_TEST_1A_5_X,   Messages,          ,       5,           ,       AgentS,    0,
DevMess_1A_5_X

MSG_TEST_1A_X_X,   Messages,          ,        ,           ,       AgentS,    0,
DevMess_1A_X_X
```

Prototype Node Name or Property	Rows 1 to 3	Defines
#Blk4ToIccp	MSG_TEST_1A_1_2 MSG_TEST_1A_X_4 MSG_TEST_1A_5_X	The local ICCP server point names for nodes that receive Block 4 message

	MSG_TEST_1A_X_X	data for specific InfoRef/LocalRef pairs from an OPC UA server.
<NS_Suffix>	Messages	Specifies the OPC UA namespace suffix for these points. Leaving blank indicates no NS suffix.
<branch>	Not set.	The OPC UA branch where this point is located. Leaving blank indicates no branch.
<InfoRef>	1 Not set. 5 Not set.	ICCP InfoReference number for this message. The default is to accept any number. MSG_TEST_1A_1_2 sets the number to 1 MSG_TEST_1A_X_4 does not set this number. MSG_TEST_1A_5_X sets the number to 5. MSG_TEST_1A_X_X does not set this number. The InfoReference number in combination with the LocalReference number are used to identify the context of a message, and internal to Oracle Utilities LEC Server to route messages to and from LEC variables.
<LocalRef>	2 4 Not set. Not set.	ICCP LocalReference number for this message. The default is to accept any number. MSG_TEST_1A_1_2 sets the number to 2 MSG_TEST_1A_X_4 sets the number to 4. MSG_TEST_1A_5_X does not set this number. MSG_TEST_1A_X_X does not set this number.
<agent>	AgentS	Label of the OPC UA server agent node to handle the connection to the external OPC UA client.
<dom_scope>	0	Indicates the scope of the variables in the local VCC. The scope in this VCC is VMD-specific as denoted by <dom_scope> of 0. A <dom_scope> of 1 would indicate Domain-specific scope, which refers to the domain specified by <local_dom> in the #My_VCC table.
<opc_label>	DevMess_1A_1_2 DevMess_1A_X_4 DevMess_1A_5_X	Specifies the OPC UA variable name. These OPC UA variables are in the external OPC UA client where the Block

	DevMess_1A_X_X	4 messages originate. They also identify where the confirmed messages are sent from the local ICCP client. These variable names override the default names. The default name is the same as the node label.

VCC, VMD, Variables, and Node Batch File Definitions for AppTestA2 CSV

The following section shows an example batch file AppTestA2.csv that describes local and remote VCCs as well as properties, nodes, and transfer sets. AppTestA2.csv contains the tables that define the VCCs, VMDs, association control variables, and device points that allow for the following types of communication:

- A remote ICCP server to pass Block 1 and Block 2 data to an application's OPC UA client.
- A remote ICCP server to pass Block 4 data to an application's OPC UA client.
- An application's OPC UA client to pass Block 5 controls to a remote server.

#sharemode

The share mode is set to none, meaning that its content is not for use by other batch files. It is set at the beginning of the batch file.

#sharemode

none

#ts_num

The #ts_num table defines one or more ICCP transfer sets and their configuration parameters.

#ts_num,<do_interval>,<interval_period>,<do_integrity>,<integrity_period>,<do_change>,<buffer_time>,<do_rbe>,<do_initial_read>,<critical>, <branch>, <NS_Suffix>, <agent>

1,1,10,1,60,1,2,1,1,0

Property	Value	In DsTransferSet template
#ts_num	1	Assigns a transfer set number that is unique to this VCC.
<do_interval>	1	A Boolean value (0 or 1) that indicates whether or not interval-based transmissions are enabled. 1 enables interval-based transmissions.
<interval_period>	10	Specifies the interval in seconds between transmissions if interval-based transmissions are enabled.
<do_integrity>	1	A Boolean value (0 or 1) that indicates whether or not integrity-based transmissions are enabled. 1 enables integrity-based transmissions.

<integrity_period>	60	Specifies the integrity period in seconds between transmissions if integrity-based transmissions are enabled.
<do_change>	1	A Boolean value (0 or 1) that indicates whether or not change-based transmissions are enabled. If <do_change> is set to 1, a change in any point in the transfer set will trigger the sending of a report. In this example, <do_change> is set to 1. See the description of <do_rbe> for more information.
<buffer_time>	2	Indicates the buffer time in seconds. A non-zero buffer time with change-based transmissions enabled (<do_change> is set to 1) causes a delay of the specified number of seconds after the first change before sending a report. This delay allows subsequent changes to be included in the same report. In this example, <buffer_time> is set to 2 seconds.
<do_rbe>	1	A Boolean value (0 or 1) that indicates whether report by exception is enabled. When a report is sent because an object changed <do_change> or because a specified time period <do_interval> elapsed: Only points that changed since the last report are sent if <do_rbe> is set to 1. The whole transfer set is sent if <do_rbe> is set to 0. In this example, <do_rbe> is set to 1.
<do_initial_read>	1	A Boolean value (0 or 1) that indicates whether an initial read of data values is required. 1 directs LEC Server to perform an initial read.
<critical>	0	If <critical> is set to 1, acknowledgement is required. If not set to 1, no acknowledgement is required.
<branch>		OPC UA server branch for this transfer set's control and status variables. Leaving blank indicates no branch.
<NS_Suffix>		OPC UA server namespace suffix for this transfer set's control and status variables. Leaving blank indicates no namespace suffix.
<agent>	AgentS	Name of the OPC UA server agent where this transfer set's control and status variables are available.

#Blk4_buflen

The #Blk4_buflen table defines a set of local ICCP Block 4 data buffers of various sizes that are created to hold Block 4 data received from the remote ICCP server. The data flow diagram shown in *#Blk4MessageRouter* in the next section outlines the path the buffer data takes from the remote ICCP server to the OEM's OPC UA client.

```
#Blk4_buflen,     <BufLen>,   <dom_scope>
Info_Buff_16,      16,         0
Info_Buff_64,      64,         0
Info_Buff_256,     256,        0
Info_Buff_1024,    1024,       0
```

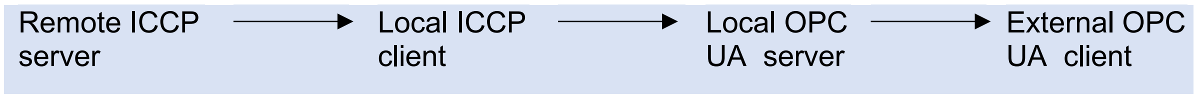
Info_Buff_4096,   4096,   0

Prototype Node Name or Property	Rows 1 through 5	Defines
#Blk4_buflen	Info_Buff_16 Info_Buff_64 Info_Buff_256 Info_Buff_1024 Info_Buff_4096	The node name
<BufLen>	16 64 256 1024 4096	The buffer length in bytes.
<dom_scope>	0	Indicates the scope of the variables in the local VCC. The scope in this VCC is VMD-specific as denoted by <dom_scope> of 0. A <dom_scope> of 1 would indicate Domain-specific scope, which refers to the domain specified by <local_dom> in the #My_VCC table.

#Blk4MessageRouter

The #Blk4MessageRouter table defines the fields in the messageRouterToOpcUa template that are used in the OPC UA server. The messageRouterToOpcUa mechanism provides ICCP Block 4 message routing based on the InfoRef/LocalRef pairs so that a message from ICCP will be associated with the local OPC UA variable that the remote OPC UA client has subscribed to.

The data point in Blk4FromIccpRouter originates from a remote ICCP server and follows the path that is described below.



> **Note**: The messageRouterToOpcUa template takes merged ICCP message buffers from the local ICCP client and associates each buffer with the corresponding OPC UA variable in the local OPC UA server. If the OEM’s OPC UA client subscribes to these variables, it can receive updates through an update connector.

! Message router to route block 4 messages from external ICCP server to specific OPC UA points subscribed to by external UPC UA client via LEC IFE.

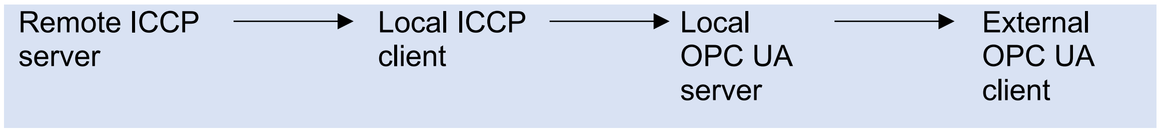
#Blk4MessageRouter, <agent>

Blk4FromIccpRouter, AgentS

Prototype Node Name or Property	Rows 1 through 5	Defines
#Blk4MessageRouter	Blk4FromIccpRouter	The node name
<agent>	AgentS	Specifies whether the OPC UA VMD is a client or a server. In this case, the VMD is an OPC UA server.

#Blk4FromIccp

The #Blk4FromIccp table defines local ICCP Block 4 messaging nodes in the local OPC UA server that hold Block 4 messages originating from a remote ICCP server; the LEC Server configuration makes this data available to an external OPC UA client that has subscribed to these points by specifying InfoRef-LocalRef pairs.



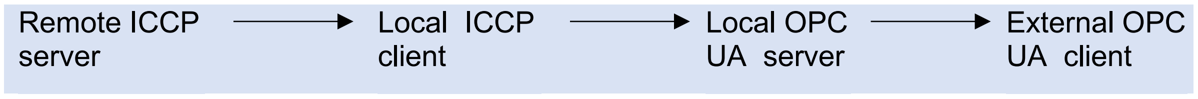
```
#Blk4FromIccp,    <NS_Suffix>, <branch>,      <InfoRef>,     <LocalRef>,    <agent>,
<opc_label>

MSG_TEST_00A_1_2,    Messages,          ,         1,           2,        AgentS,
DevMESS_00A_1_2

MSG_TEST_00A_X_4,    Messages,          ,          ,           4,        AgentS,
DevMESS_00A_X_4

MSG_TEST_00A_5_X,    Messages,          ,         5,            ,        AgentS,
DevMESS_00A_5_X

MSG_TEST_00A_X_X,    Messages,          ,          ,            ,        AgentS,
DevMESS_00A_X_X
```

Prototype Node Name or Property	Rows 1 to 4	Defines
#Blk4FromIccp	MSG_TEST_00A_1_2 MSG_TEST_00A_X_4 MSG_TEST_00A_5_X MSG_TEST_00A_X_X	The local ICCP client message names for nodes that receive ICCP Block 4 message data in the local OPC UA server from a remote ICCP server. These batch nodes are generated from the messageToOpcUa template.
<NS_Suffix>	Messages	Specifies the OPC UA namespace suffix for local OPC UA server points that are available to receive this Block 4 messages from the local ICCP client. Leaving blank indicates no NS suffix.
<branch>	Not set	The OPC UA branch where these points are located. Leaving blank indicates no branch.
<InfoRef>	1	ICCP InfoReference number for this message.

	blank 5 blank	MSG_TEST_00A_1_2 sets the number to 1 MSG_TEST_00A_X_4 does not set the InfoReference number. MSG_TEST_00A_5_X sets the number to 5. MSG_TEST_00A_X_X does not set the InfoReference number. This number is a 32-bit signed integer.
<LocalRef>	2 4 blank blank	ICCP LocalReference number for this message. The default is to accept any number. MSG_TEST_00A_1_2 sets the number to 2 MSG_TEST_00A_X_4 sets the number to 4. MSG_TEST_00A_5_X does not set the LocalReference number. MSG_TEST_00A_X_X does not set the LocalReference number. This number is a 32-bit signed integer.
<agent>	AgentS	Label of the OPC UA server agent node that is to handle the connection to the external OPC UA client.
<dom_scope>	0	Indicates the scope of the variables in the remote VCC. The scope in that VCC is VMD-specific as indicated by <dom_scope> of 0. A <dom_scope> of 1 would indicate Domain-specific scope, which refers to the domain specified by <remote_dom> in the #My_VCC table.
<opc_label>	DevMESS_00A_1_2 DevMESS_00A_X_4 DevMESS_00A_5_X DevMESS_00A_X_X	Specifies the OPC UA variable name in the external OPC UA client. The default name is the same as the node label.

#Blk1FromIccp

The #Blk1FromIccp table defines local ICCP Block 1 points. The data points in these nodes originate from a remote ICCP server and are transferred to an external OPC UA client through the OPC UA mechanism of subscription.



```
#Blk1FromIccp, <NS_Suffix>, <branch>, <type>, <agent>, <dom_scope>, <ts_num>,
<opc_label>, <filter>, <slope>, <offset>, <s0>, <s1>, <s2>, <s3>, <data_timeout>,
<codemap_file>,,,,,,,,,

MEAS_TEST_1A_00000, Reals, , Data_RealQTimeTagExtended, AgentS, 0, 1,
"DevMeas_1A_00000",,,,,,,,,

MEAS_TEST_1A_FI1, Reals, , Data_RealQ, AgentS, 0, 1, "DevMeas_1A_FI1",,,,,,,,,

MEAS_TEST_1A_FI2, Reals, , Data_RealQ, AgentS, 0, 1, "DevMeas_1A_ FI2",,,,,,,,,
```

STATUS _1A_00000, Statuses, , Data_StateQTimeTagExtended, AgentS, 0, 1, "DevStatus_1A_00000",,,,,,,,

STATUS_TEST_1A_FI1, Statuses, , Data_StateQ, AgentS, 0, 1, "DevStatus_1A_ FI1",,,,,,,,

STATUS_TEST_1A_FI2, Statuses, , Data_StateQ, AgentS, 0, 1, "DevStatus_1A_ FI2",,,,,,,,

Prototype Node Name or Property	Rows 1 and 2	Defines or indicates
#Blk1FromIccp	MEAS_TEST_1A_00000 MEAS_TEST_1A_FI1 MEAS_TEST_1A_FI2 STATUS_TEST_1A_00000 STATUS_TEST_1A_FI1 STATUS_TEST_1A_FI2	Specifies local ICCP node names which will contain data that the configuration transmits to the local OPC UA server. The nodes beginning with MEAS will contain telemetry data, and the nodes beginning with STATUS will contain status data.
<NS_Suffix>	Reals Reals Reals Statuses Statuses Statuses	Specifies the OPC UA namespace suffix for the OPC UA server nodes that receive this data from LEC's ICCP client nodes. Leaving blank indicates no NS suffix.
<branch>	Not set.	The OPC UA branch where these receiving points are located. Leaving it blank indicates no branch.
<type>	Data_RealQTimeTagExtended Data_RealQ Data_RealQ Data_StateQTimeTagExtended Data_StateQ Data_StateQ	LEC data type of the data in the local ICCP client nodes MEAS_TEST_1A_00000, MEAS_TEST_1A_FI1, MEAS_TEST_1A_FI2, STATUS_TEST_1A_00000, STATUS_TEST_1A_FI1, and STATUS_TEST_1A_FI2.
<agent>	AgentS	Specifies a label for the local OPC UA server agent node that handles the connection to the external OPC UA client.
<dom_scope>	0	Indicates the scope of the variables in the remote VCC. The scope is VMD-specific as indicated by <dom_scope> of 0. A <dom_scope> of 1 would indicate Domain-specific scope, which refers to the domain specified by <remote_dom> in the #My_VCC table.
<ts_num>	1	Specifies the transfer set number that is unique to this VCC.
<opc_label>	"DevMeas_1A_00000" "DevMeas_1A_FI1" "DevMeas_1A_FI2" "DevStatus_1A_00000" "DevStatus_1A_FI1" "DevStatus_1A_FI2"	Specifies the OPC UA data point names. The default OPC UA point name is the node label. These points reside on the local OPC UA server. The nodes beginning with DevMeas will receive telemetry data, and the nodes beginning with DevStatus will receive status data.

<filter>		Name, either STATE or SCALE, of the filter to use for this point. Leaving blank indicates no filter.
<slope>		If SCALE filter is used, the slope (multiplier) to apply to this point. Leaving blank indicates no slope.
<offset>		If SCALE filter is used, the offset (summand) to add to this point. Leaving blank indicates no offset.
<s0>		When STATE filter is used, value to replace this point's value with when actual value is 0. Leaving blank indicates no replacement.
<s1>		Same as above but for actual value 1.
<s2>		Same as above but for actual value 2.
<s3>		Same as above but for actual value 3.

#Blk5ToIccp

The #Blk5ToIccp table defines local ICCP Block 5 setpoint nodes. The data points that are written to these nodes originate from an external OPC UA client; the LEC Server instance transfers the values in these setpoint nodes to nodes on a peer ICCP server using the ICCP Write mechanism. #Blk5ToIccp is based on a ControlFromOpcUa template within the local OPC UA server.

$$\begin{tikzcd}
\text{External OPC UA client} \arrow[r] & \text{Local OPC UA server} \arrow[r] & \text{Local ICCP client} \arrow[r] & \text{Remote ICCP server}
\end{tikzcd}$$

#Blk5FToIccp, <NS_Suffix>, <branch>, <tagging_branch>, <type>, <flags>, <tagging_point>, <agent>, <ChkBkId>, <dom_scope>, <timeout>, <tag_pollclass>,<opc_label>

CONTROL_TEST_1A_00000, Discretes, ASYS,,<int:16>, DISCRETE,, AgentS, 681, 0, 6000, ,"DevCtrl_1A_00000"

CONTROL_TEST_1A_00001, Discretes, ASYS, , <int:16>, DISCRETE|SBO, , AgentS, 681, 0, 6000, ,"DevCtrl_1A_00001"

CONTROL_TEST_1A_00002, Discretes, ASYS, , <int:16>, DISCRETE|TAGABLE, , AgentS, 681, 0, 6000, ,"DevCtrl_1A_00002"

CONTROL_TEST_1A_00003, Discretes, ASYS, , <int:16>, DISCRETE|TAGABLE|SBO, , AgentS, 681, 0, 6000, ,"DevCtrl_1A_00003"

CONTROL_TEST_1A_FO1, Discretes, ASYS, ,<int:16>, DISCRETE|SBO, , AgentS, 681, 0, 6000, ,"DevCtrl_1A_ FO1"

CONTROL_TEST_1A_FO2, Discretes, ASYS, ,<int:16>, DISCRETE|SBO, , AgentS, 681, 0, 6000, ,"DevCtrl_1A_ FO2"

SETPOINT_TEST_1A_00000, Reals, ASYS, , <f-p:32:8>, REAL,, AgentS, 9, 0, 6000, , "DevCtrl_1A_00000"

SETPOINT_TEST_1A_00001, Reals, ASYS, , <f-p:32:8>, REAL|SBO, , AgentS, 9, 0, 6000, ,"DevCtrl_1A_00001"

SETPOINT_TEST_1A_00002, Reals, ASYS, , <f-p:32:8>, REAL|TAGABLE, , AgentS, 9, 0, 6000, ,"DevCtrl_1A_00002"

SETPOINT_TEST_1A_00003, Reals, ASYS, , <f-p:32:8>, REAL|TAGABLE|SBO, , AgentS, 9, 0, 6000, ,"DevCtrl_1A_00003"

SETPOINT_TEST_1A_FO1, Reals, ASYS, , <f-p:32:8>, REAL, , AgentS, 9, 0, 6000,,"DevCtrl_1A_ FO1"

SETPOINT_TEST_1A_FO2, Reals, ASYS, , <f-p:32:8>, REAL, , AgentS, 9, 0, 6000,,"DevCtrl_1A_ FO2"

Prototype Node Name or Property	Rows 1 through 12	Defines or indicates
#Blk5ToIccp	CONTROL_TEST_1A_00000 CONTROL_TEST_1A_00001 CONTROL_TEST_1A_00002 CONTROL_TEST_1A_00003 CONTROL_TEST_1A_ FO1 CONTROL_TEST_1A_ FO2 SETPOINT_TEST_1A_00000 SETPOINT_TEST_1A_00001 SETPOINT_TEST_1A_00002 SETPOINT_TEST_1A_00003 SETPOINT_TEST_1A_ FO1 SETPOINT_TEST_1A_ FO2	Defines internal names for passing setpoint data in the local OPC UA server.
<NS_Suffix>	Discretes Discretes Discretes Discretes Discretes Discretes Reals Reals Reals Reals Reals Reals	Specifies the OPC UA namespace suffix for the local OPC UA server points that transmit the ICCP data. Leaving blank indicates no NS suffix.
<branch>	ASYS	The OPC UA branch where these OPC UA server points are located. Leaving it blank indicates no branch. All of these points are in the ASYS branch.
<tagging_branch>	Not set.	OPC UA tagging point branch name; the default is the OPC UA point branch name.

		In this example, the tagging branch name for all points is the default branch ASYS.
<type>	<int:16> <int:16> <int:16> <int:16> <int:16> <int:16> <f-p:32:8> <f-p:32:8> <f-p:32:8> <f-p:32:8> <f-p:32:8> <f-p:32:8>	Specifies the data type of the local OPC UA server nodes from where the control information is passed.
<flags>	DISCRETE DISCRETE|SBO DISCRETE|TAGABLE DISCRETE|TAGABLE|SBO DISCRETE|SBO DISCRETE|SBO REAL REAL|SBO REAL|TAGABLE REAL|TAGABLE|SBO REAL REAL	Specifies the control flags of the receiving local ICCP control points: **REAL** – The control point is a floating-point number **DISCRETE** – The control point is an integer **SBO** – The control point supports Select Before Operate **TAG** – The control point supports tagging The bar (|) represents an AND. INTEGER|SBO indicates that the value is an integer number and that the control point supports Select Before Operate.
<tagging_point>	Not set.	OPC UA tagging point name. The OPC UA tagging point name is a variable contained in the local OPC UA server. The default is the external OPC UA point name with _TAG appended.
<agent>	AgentS	Label of the OPC UA server agent node to handle the connection with the external OPC UA client.
<ChkBkId>	681 681 681 681 681 681 9 9 9 9 9 9	Expected Check back ID. The expected Check back ID for the first set of control points is 681. The expected Check back ID for the second set of control points is 9.
<dom_scope>	0	Indicates the scope of the variables in the local VCC. The scope in this VCC is VMD-specific as denoted by <dom_scope> of 0. A <dom_scope> of 1 would indicate Domain-specific scope, which refers to the domain specified by <local_dom> in the #My_VCC table.
<timeout>	6000	Specifies the timeout in milliseconds to abort a pending control or set point operation.

<tag_pollclass>	Not set.	Defines the poll period. The available poll periods are as follows: 1 to specify 10 seconds 4 to specify one minute Any number between 4 and 1024 will resolve to the default, which is one minute.
<opc_label>	"DevCtrl_1A_00000" "DevCtrl_1A_00001" "DevCtrl_1A_00002" "DevCtrl_1A_00003" "DevCtrl_1A_ FO1" "DevCtrl_1A_ FO2" "DevSet_1A_00000" "DevSet_1A_00001" "DevSet_1A_00002" "DevSet_1A_00003" "DevSet_1A_ FO1" "DevSet_1A_ FO2"	Specifies the OPC UA control point name. The default OPC UA control point name is the node label. These control points reside on the local OPC UA server.

Repeated Table Headers with Additional Points

At the end of the AppTestA2.csv file, you will find three tables that have the same header information as those in sections: *#Blk4_buflen*, *#Blk4MessageRouter*, and *#Blk4FromIccp*.

Except for the *#Blk4MessageRouter* table, these tables add additional points to the configuration. For more information about these tables, see the previously described tables in the sections mentioned above.

! Information buffer for receipt of block 4 messages from external ICCP server.

```
#Blk4_buflen,        <BufLen>,     <dom_scope>

Info_Buff_1A_16,         16,          0

Info_Buff_1A_64,         64,          0

Info_Buff_1A_256,        256,          0

Info_Buff_1A_1024,       1024,          0

Info_Buff_1A_4096,       4096,          0
```

! Message router to route block 4 messages from external ICCP server to specific OPC UA points subscribed to by external UPC UA client via LEC IFE.

```
#Blk4MessageRouter, <agent>

Blk4FromIccpRouter, AgentS
```

! ICCP block 4 messages from external ICCP server routed to external OPC UA client via LEC IFE.

```
#Blk4FromIccp,    <NS_Suffix>, <branch>,      <InfoRef>,      <LocalRef>,     <agent>,
<opc_label>
```

```
MSG_TEST_1A_1_2,    Messages,          ,          1,            2,         AgentS,
MSG_TEST_1A_1_2

MSG_TEST_1A_X_4,    Messages,          ,          0,            4,         AgentS,
MSG_TEST_1A_X_4

MSG_TEST_1A_5_X,    Messages,          ,          5,             ,         AgentS,
MSG_TEST_1A_5_X
```

Initial Batch File that is Loaded Remotely

This section shows an example of the initial batch file that is loaded by the remotely executed command file. For more information on the mechanism for loading batch files, see Loading of Batch Files and Command Files.

#DBCurrentlyLoaded

#DBCurrentlyLoaded table defines a single static node that identifies the last command file executed whether or not the command file succeeded in loading all of the batch files within it. The command file is stored as part of the configuration, and therefore persists across LEC Server restarts.

#DBCurrentlyLoaded, <value>

DBCurrentlyLoaded, """AppTestA.txt"""

Prototype VCC or Property	Value	In the BatchScriptFilter template, defines or indicates
#DBCurrentlyLoaded	DBCurrentlyLoaded	The label used in referencing this node.
<value>	"""AppTestA.txt"""	The output variable name, indicating the last command file that was executed whether or not it loaded the batch files successfully or not.

Command Files

This section shows an example of the sample command files that can be executed remotely from any one of the OEM's machines.

AppTestA1.txt

```
load file=AppTestA_Loaded.csv

load file=A1, from=AppTestA1.csv, share=False

load file=A2, from=AppTestA2.csv, share=False TestA_Loaded.csv

load file=A1, from=TestA1.csv, share=ileA2, from=TestA2.csv, share=False
```

This example command file loads three batch files:

- **AppTestA_Loaded.csv** contains the #DBCurrentlyLoaded table which defines the variable that returns the name of the last loaded command file. *See Loading of Batch Files for more information*.
- **AppTestA1.csv** contains the tables that define the VCCs, VMDs, association control variables, and device points that allow for the following types of communication:
    - An application's OPC UA server to pass Block 1 and Block 2 data to a remote ICCP client.
    - A remote ICCP client to pass Block 5 controls to the application's OPC UA server.

See the VCC, VMD, Variables, and Node Batch File Definitions for AppTestA1 section for more information on these tables.

- **AppTestA2.csv** contains the tables that define the VCCs, VMDs, association control variables, and device points that allow for the following types of communication:
    - A remote ICCP client to transfer Block 1 and Block 2 data to an application's OPC UA server.
    - An application's OPC UA server to pass Block 5 controls to the remote ICCP client.

See the VCC, VMD, Variables, and Node Batch File Definitions for AppTestA2.csv section for more information on these tables.

AppTestARemove.txt

This file removes AppTestA_Loaded.csv, AppTestA1.csv, and AppTestA2.csv.

AppTestARemoveAll.txt

This file removes all of the batch files that have been loaded by AppTestA1.txt and AppTestA_Loaded.txt as well as the header batch file AppTestAHeader.csv.

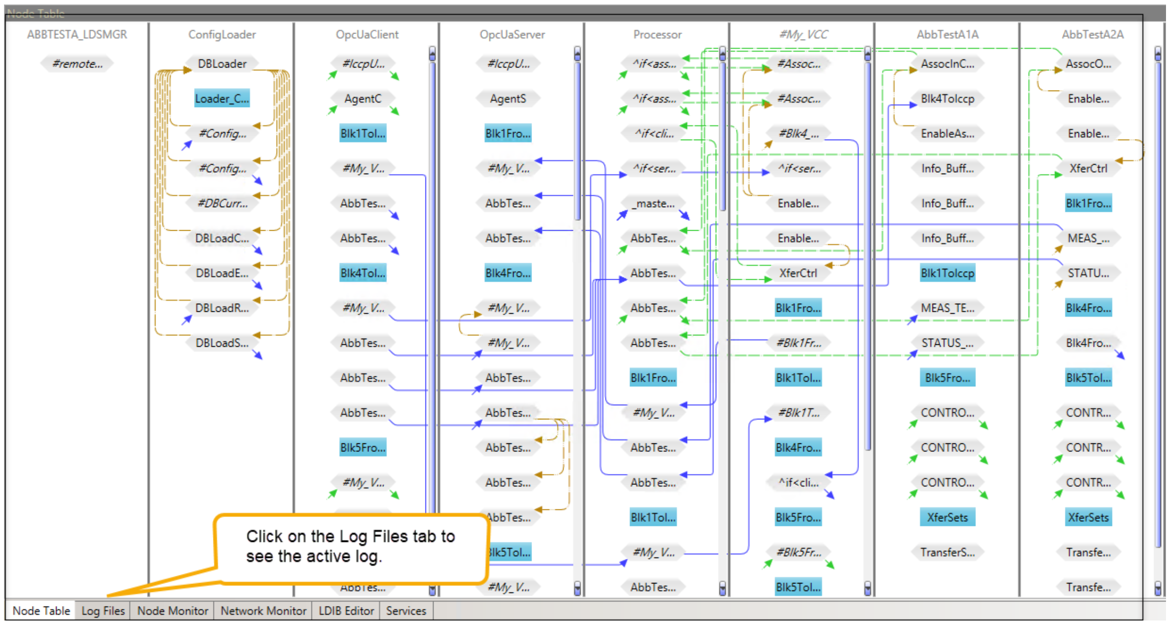
Note: It is unlikely you will need to use this command file once the IFE is in production.

Fault Tracing with LEC Configuration Manager

To start LEC Configuration Manager, click on the Oracle Utilities LEC Configuration Manager icon from the Windows Start screen.

The default layout will be displayed as shown in Figure 13.

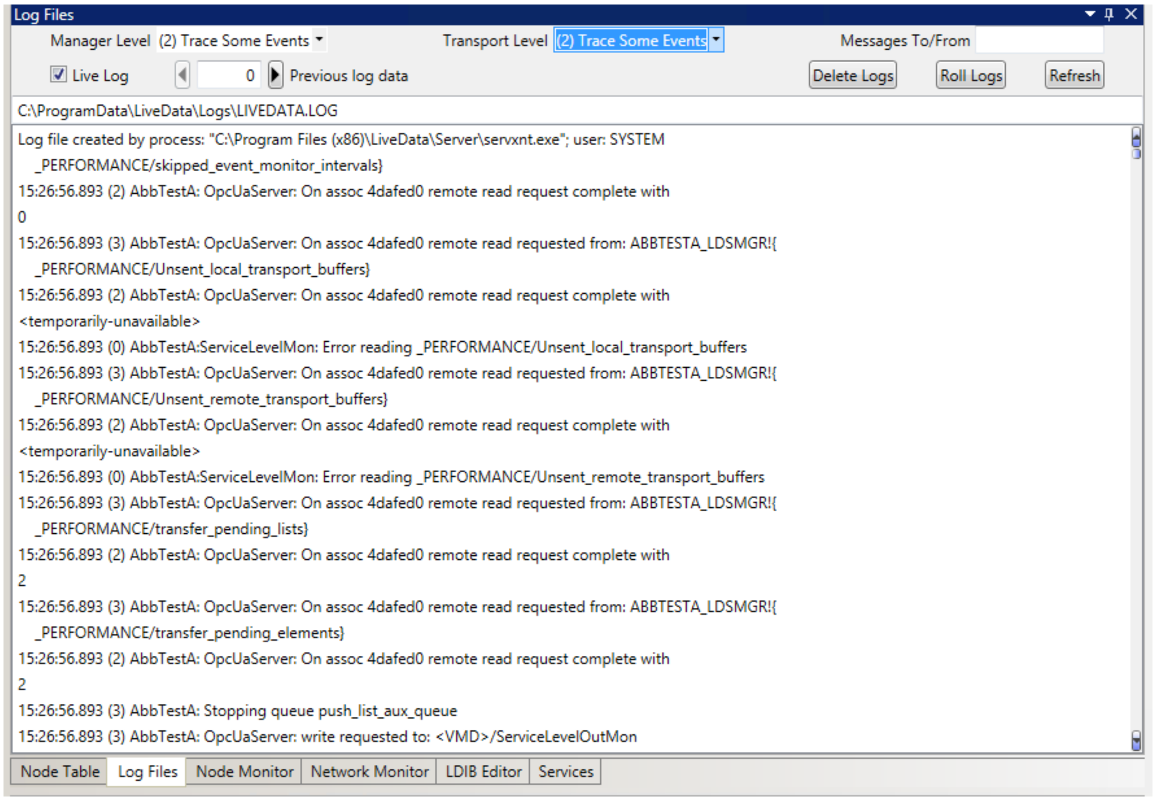
Figure 13: Click on the Log Files tab of the Node Table



Log Files

The Log Files tab shows a running log of the activity in your configuration as shown in Figure 14.

Figure 14: Log Files



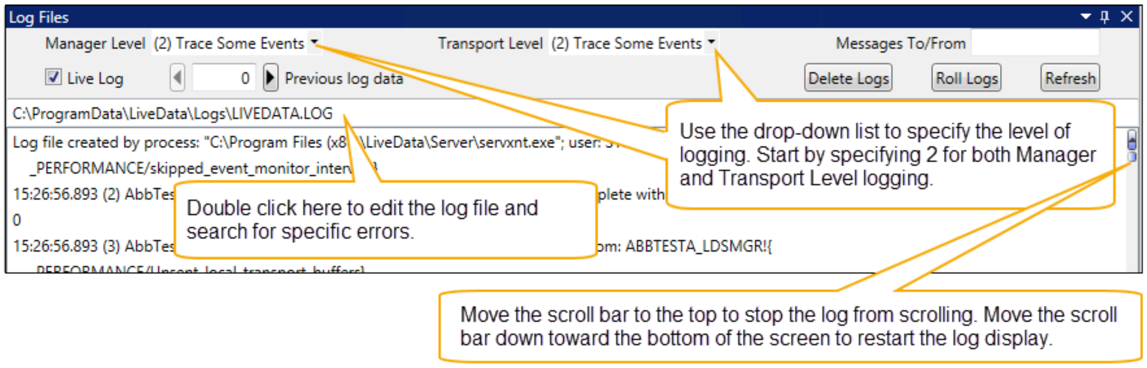
Manager Level (2) Trace Some Events
Transport Level (2) Trace Some Events
Messages To/From
☑ Live Log 0 Previous log data

```
C:\ProgramData\LiveData\Logs\LIVEDATA.LOG
Log file created by process: "C:\Program Files (x86)\LiveData\Server\servxnt.exe"; user: SYSTEM
   _PERFORMANCE/skipped_event_monitor_intervals}
15:26:56.893 (2) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read request complete with
0
15:26:56.893 (3) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read requested from: ABBTESTA_LDSMGR!{
   _PERFORMANCE/Unsent_local_transport_buffers}
15:26:56.893 (2) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read request complete with
<temporarily-unavailable>
15:26:56.893 (0) AbbTestA:ServiceLevelMon: Error reading _PERFORMANCE/Unsent_local_transport_buffers
15:26:56.893 (3) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read requested from: ABBTESTA_LDSMGR!{
   _PERFORMANCE/Unsent_remote_transport_buffers}
15:26:56.893 (2) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read request complete with
<temporarily-unavailable>
15:26:56.893 (0) AbbTestA:ServiceLevelMon: Error reading _PERFORMANCE/Unsent_remote_transport_buffers
15:26:56.893 (3) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read requested from: ABBTESTA_LDSMGR!{
   _PERFORMANCE/transfer_pending_lists}
15:26:56.893 (2) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read request complete with
2
15:26:56.893 (3) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read requested from: ABBTESTA_LDSMGR!{
   _PERFORMANCE/transfer_pending_elements}
15:26:56.893 (2) AbbTestA: OpcUaServer: On assoc 4dafed0 remote read request complete with
2
15:26:56.893 (3) AbbTestA: Stopping queue push_list_aux_queue
15:26:56.893 (3) AbbTestA: OpcUaServer: write requested to: <VMD>/ServiceLevelOutMon
```

From the top of the Log Files screen, events and data are shown in reverse chronological order staLECng with the most recent at the top of the log. You can use the scroll bar to the right side of the panel to see the beginning of the log by scrolling down to the bottom of the screen.

Figure 15 shows how you can change Transport and Manager Level logging and edit the active log to search for specific errors.

Figure 15: Change the Trace Level and Edit the Active Log File



```
C:\ProgramData\LiveData\Logs\LIVEDATA.LOG
Log file created by process: "C:\Program Files (x8 \LiveData\Server\servxnt.exe"; user:
  _PERFORMANCE/skipped_event_monitor_interv
15:26:56.893 (2) AbbTes                                   plete with
0
15:26:56.893 (3) AbbTes                                   om: ABBTESTA_LDSMGR!{
```

Use the drop-down list to specify the level of logging. Start by specifying 2 for both Manager and Transport Level logging.

Double click here to edit the log file and search for specific errors.

Move the scroll bar to the top to stop the log from scrolling. Move the scroll bar down toward the bottom of the screen to restart the log display.

Moving the scroll bar to the top of the screen temporarily stops the scrolling display of messages.

Transport and Manager Levels

Log file information is divided into two types: Transport and Manager.

The transport layer is responsible for end-to-end communication over a network. It provides logical communication between application processes running on different hosts within a layered architecture of protocols and other network components. Transport Level logging displays information from the transport layer and other lower protocol layers. You can specify the amount of information that you would like to receive from least to most by selecting (0) Default, (1) Diagnostics, (2) Trace Some Events, (3) Trace All Events, and (4) Trace All Data from the Transport Level drop-down list.

The manager level displays information from the upper protocol layers: Session, Presentation, and ACSE. You can specify the amount of information that you would like to receive from least to most by selecting (0) Default, (1) Diagnostics, (2) Trace Some Events, (3) Trace All Events, and (4) Trace All Data from the Manager Level drop-down list.

Note: For both Transport and Manager Levels, start with Level 2 logging to trace some events so that you capture.

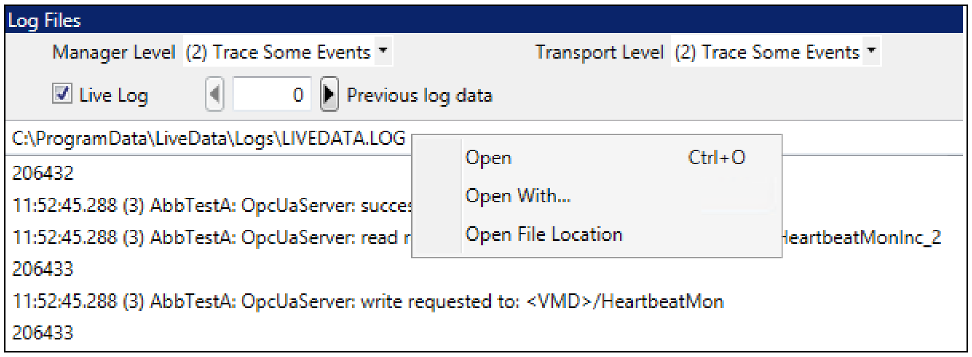
Editing a Log File

You might see an error as the log is scrolling that you would like to look at more closely. You can search for the error or a VMD or node by searching the log file in an editor.

When you open a log file in an editor, LEC Server stops writing to the file unless you use an editor that allows LEC Server to continue writing to the file after you have opened it for inspection. One such editor is Notepad++. If you would like to use an editor other than Notepad, you first must install it on your system.

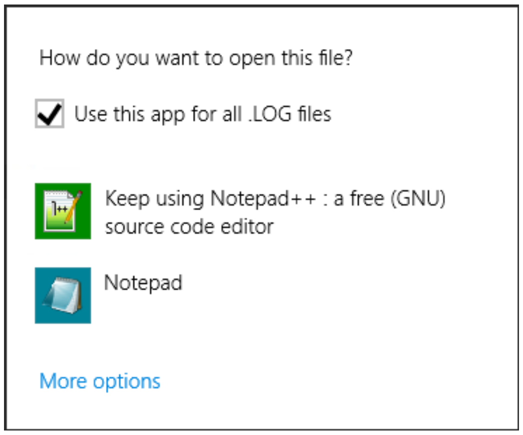
To select an alternative editor:

1. Right-click the log file's pathname and select **Open With** from the drop-down list.



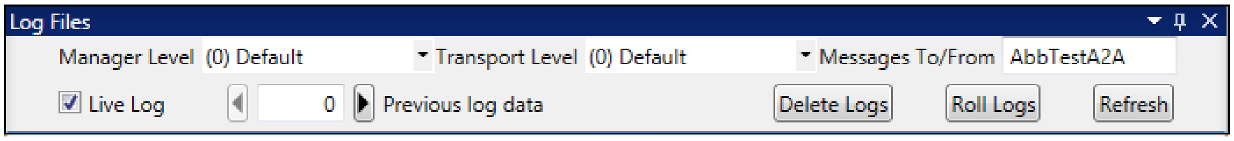
```
C:\ProgramData\LiveData\Logs\LIVEDATA.LOG
206432
11:52:45.288 (3) AbbTestA: OpcUaServer: succes
11:52:45.288 (3) AbbTestA: OpcUaServer: read r                    HeartbeatMonInc_2
206433
11:52:45.288 (3) AbbTestA: OpcUaServer: write requested to: <VMD>/HeartbeatMon
206433
```

2. Select the editor you want to use. If it does not appear in the list, click **More options**. If the desired editor still does not appear, click **Look for another app on this PC** at the bottom of the list to search for an editor.


How do you want to open this file?
Use this app for all .LOG files
Keep using Notepad++ : a free (GNU) source code editor
Notepad
More options

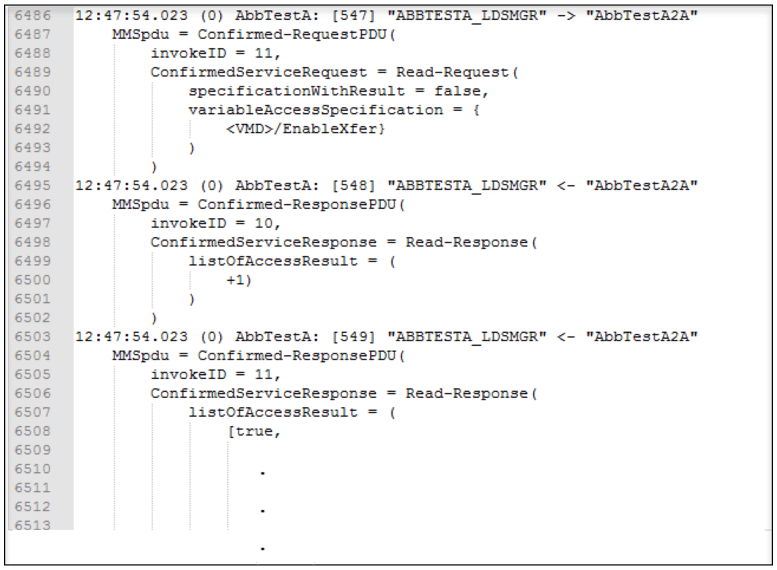
Examine Messages to and from a Remote Peer VMD

You can see formatted MMS message logging to and from a remote peer by specifying the name of the peer in the **Messages To/From** field as shown below.


Log Files
Manager Level (0) Default
Transport Level (0) Default
Messages To/From AbbTestA2A
Live Log
0
Previous log data
Delete Logs
Roll Logs
Refresh

Note: you can use a wildcard (*) to get information from all remote peers.

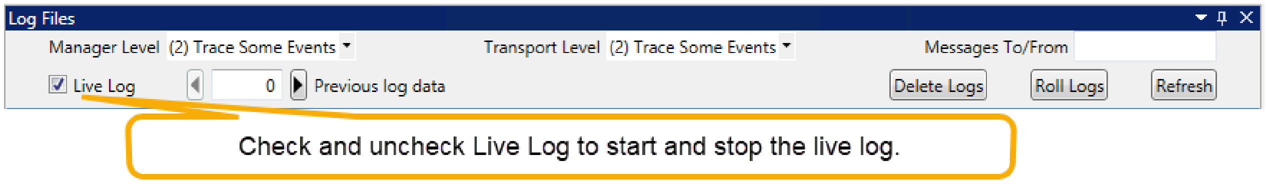
By typing the name of a remote VMD, you can receive detailed messages from and to the specified remote peer as shown on the next page.



```
12:47:54.023 (0) AbbTestA: [547] "ABBTESTA_LDSMGR" -> "AbbTestA2A"
    MMSpdu = Confirmed-RequestPDU(
        invokeID = 11,
        ConfirmedServiceRequest = Read-Request(
            specificationWithResult = false,
            variableAccessSpecification = (
                <VMD>/EnableXfer)
            )
        )
12:47:54.023 (0) AbbTestA: [548] "ABBTESTA_LDSMGR" <- "AbbTestA2A"
    MMSpdu = Confirmed-ResponsePDU(
        invokeID = 10,
        ConfirmedServiceResponse = Read-Response(
            listOfAccessResult = (
                +1)
            )
        )
12:47:54.023 (0) AbbTestA: [549] "ABBTESTA_LDSMGR" <- "AbbTestA2A"
    MMSpdu = Confirmed-ResponsePDU(
        invokeID = 11,
        ConfirmedServiceResponse = Read-Response(
            listOfAccessResult = (
                [true,

                        .

                        .

                        .
```

Stop and Start the Live Log

You can stop or start the Live Log by unchecking or checking the box next to **Live Log**.



Check and uncheck Live Log to start and stop the live log.

Note: The log is only live when **Live Log** is checked, and you have scrolled to the end of the log.

Delete Logs

You can delete all log files and start with a new log by clicking **Delete Logs** from the top-right of the Log Files panel. While debugging a configuration, it is helpful to stop the service, delete the existing log files, and restart the service to see if there are any issues when the configuration first starts.

Roll Logs

Alternatively, you can roll the contents of the active log in LIVEDATA.LOG to another file named LD_00001.LOG by clicking on the **Roll Logs** button in the top-right corner of the Log Files panel.

The contents that were in LD_00001.LOG would be rolled into the file named LD_00002.LOG. The number of log files that you can have is configurable as is the size of a log file.

Note: Since LEC Server actually rolls the log files, LEC Server must be running in order to use this function.

For more information, contact My Oracle Support.

Refresh Logs

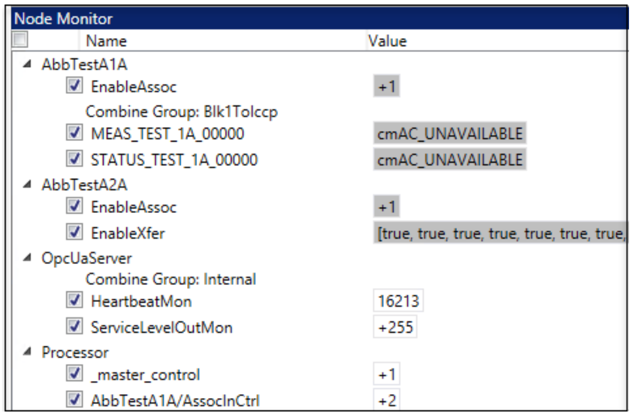
If you or another engineer has edited the log levels in the LDIB.ini file (see LDIB Editor), you can update the log levels by clicking **Refresh** in the right-corner of the Log Files panel.

Node Monitor

After you have a working configuration, you can start the configuration and examine the contents of each node in the Node Monitor.

To examine the nodes in your configuration:

1. Click the **Node Monitor** tab at the bottom of the Node Table.
2. Click **Start service** in the LEC Configuration Manager header if LEC Server is not already started.
3. Click each of the checkboxes next to the nodes whose values you want to monitor.



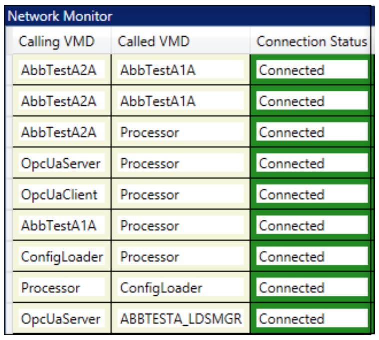
Node Monitor

Name	Value
AbbTestA1A	
✓ EnableAssoc	+1
Combine Group: Blk1Tolccp	
✓ MEAS_TEST_1A_00000	cmAC_UNAVAILABLE
✓ STATUS_TEST_1A_00000	cmAC_UNAVAILABLE
AbbTestA2A	
✓ EnableAssoc	+1
✓ EnableXfer	[true, true, true, true, true, true, true,
OpcUaServer	
Combine Group: Internal	
✓ HeartbeatMon	16213
✓ ServiceLevelOutMon	+255
Processor	
✓ _master_control	+1
✓ AbbTestA1A/AssocInCtrl	+2

Network Monitor

To examine the connections between VMDs/VCCs in your configuration, click the **Network Monitor** tab at the bottom of the Node Table.

The Network Monitor shows the status of each VMD connection.



Network Monitor

Calling VMD	Called VMD	Connection Status
AbbTestA2A	AbbTestA1A	Connected
AbbTestA2A	AbbTestA1A	Connected
AbbTestA2A	Processor	Connected
OpcUaServer	Processor	Connected
OpcUaClient	Processor	Connected
AbbTestA1A	Processor	Connected
ConfigLoader	Processor	Connected
Processor	ConfigLoader	Connected
OpcUaServer	ABBTESTA_LDSMGR	Connected

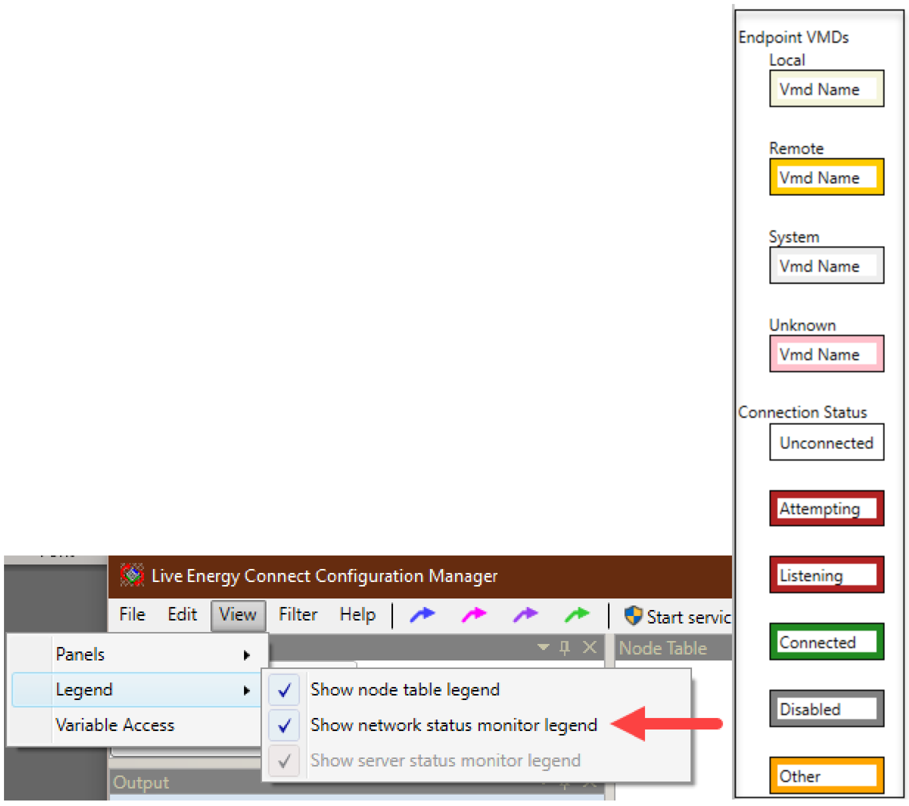
The Network Monitor panel shows the MMS association statuses: Listening, Attempting, Connected, and Disabled.

- **Listening** means the device is waiting for an inbound connection.

- **Attempting** means that the device is attempting to make an outbound connection
- **Connected** means that the two devices are connected.
- **Disabled** means that the device's ability to connect has been turned off.

To display a legend of the different types of endpoint VMDs and Connection Statuses:

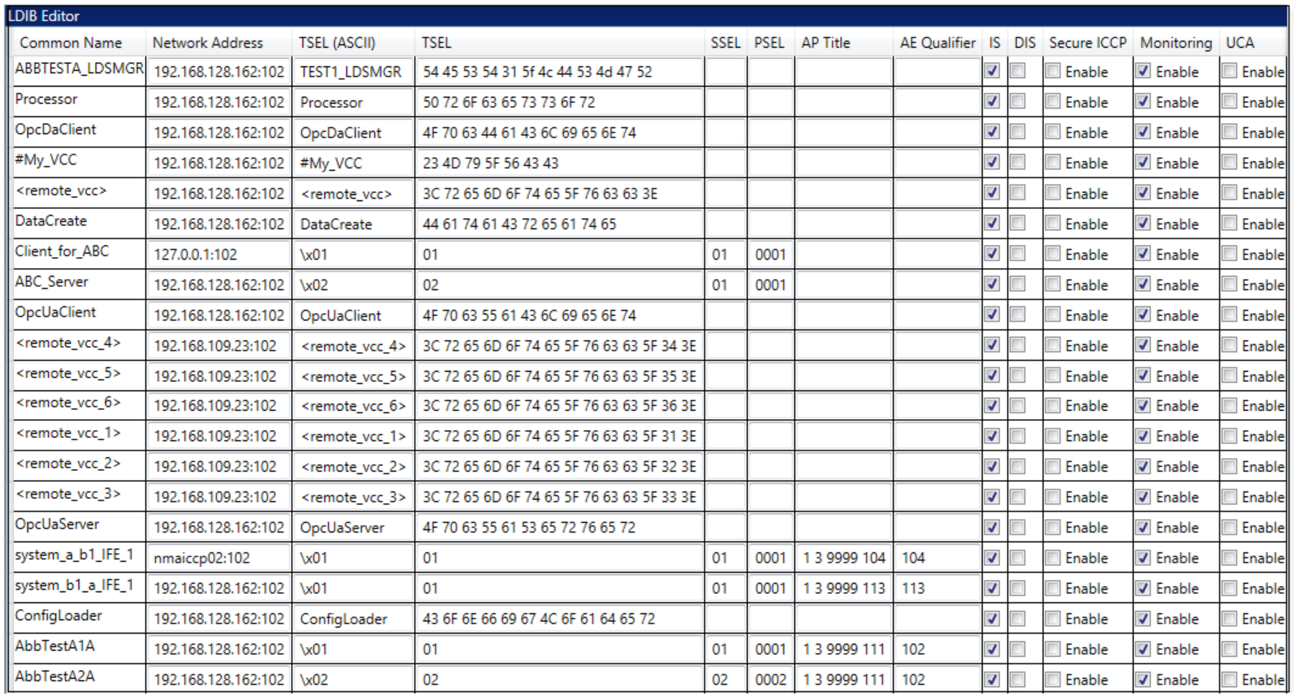
1. Click **View** and then **Legend** from the command bar.
2. Check **Show network status monitor legend**.



Note: VMDs in the example are local and are connected because the VMD names are outlined in light green and the connections are outlined in forest green as shown in the legend.

LDIB Editor

The LDIB Editor allows you to check the network addresses and other parameters and make any changes that are necessary.


LDIB Editor

Common Name	Network Address	TSEL (ASCII)	TSEL	SSEL	PSEL	AP Title	AE Qualifier	IS	DIS	Secure ICCP	Monitoring	UCA
ABBTESTA_LDSMGR	192.168.128.162:102	TEST1_LDSMGR	54 45 53 54 31 5f 4c 44 53 4d 47 52					✓	☐	☐ Enable	✓ Enable	☐ Enable
Processor	192.168.128.162:102	Processor	50 72 6F 63 65 73 73 6F 72					✓	☐	☐ Enable	✓ Enable	☐ Enable
OpcDaClient	192.168.128.162:102	OpcDaClient	4F 70 63 44 61 43 6C 69 65 6E 74					✓	☐	☐ Enable	✓ Enable	☐ Enable
#My_VCC	192.168.128.162:102	#My_VCC	23 4D 79 5F 56 43 43					✓	☐	☐ Enable	✓ Enable	☐ Enable
<remote_vcc>	192.168.128.162:102	<remote_vcc>	3C 72 65 6D 6F 74 65 5F 76 63 63 3E					✓	☐	☐ Enable	✓ Enable	☐ Enable
DataCreate	192.168.128.162:102	DataCreate	44 61 74 61 43 72 65 61 74 65					✓	☐	☐ Enable	✓ Enable	☐ Enable
Client_for_ABC	127.0.0.1:102	\x01	01	01	0001			✓	☐	☐ Enable	✓ Enable	☐ Enable
ABC_Server	192.168.128.162:102	\x02	02	01	0001			✓	☐	☐ Enable	✓ Enable	☐ Enable
OpcUaClient	192.168.128.162:102	OpcUaClient	4F 70 63 55 61 43 6C 69 65 6E 74					✓	☐	☐ Enable	✓ Enable	☐ Enable
<remote_vcc_4>	192.168.109.23:102	<remote_vcc_4>	3C 72 65 6D 6F 74 65 5F 76 63 63 5F 34 3E					✓	☐	☐ Enable	✓ Enable	☐ Enable
<remote_vcc_5>	192.168.109.23:102	<remote_vcc_5>	3C 72 65 6D 6F 74 65 5F 76 63 63 5F 35 3E					✓	☐	☐ Enable	✓ Enable	☐ Enable
<remote_vcc_6>	192.168.109.23:102	<remote_vcc_6>	3C 72 65 6D 6F 74 65 5F 76 63 63 5F 36 3E					✓	☐	☐ Enable	✓ Enable	☐ Enable
<remote_vcc_1>	192.168.109.23:102	<remote_vcc_1>	3C 72 65 6D 6F 74 65 5F 76 63 63 5F 31 3E					✓	☐	☐ Enable	✓ Enable	☐ Enable
<remote_vcc_2>	192.168.109.23:102	<remote_vcc_2>	3C 72 65 6D 6F 74 65 5F 76 63 63 5F 32 3E					✓	☐	☐ Enable	✓ Enable	☐ Enable
<remote_vcc_3>	192.168.109.23:102	<remote_vcc_3>	3C 72 65 6D 6F 74 65 5F 76 63 63 5F 33 3E					✓	☐	☐ Enable	✓ Enable	☐ Enable
OpcUaServer	192.168.128.162:102	OpcUaServer	4F 70 63 55 61 53 65 72 76 65 72					✓	☐	☐ Enable	✓ Enable	☐ Enable
system_a_b1_IFE_1	nmaiccp02:102	\x01	01	01	0001	1 3 9999 104	104	✓	☐	☐ Enable	✓ Enable	☐ Enable
system_b1_a_IFE_1	192.168.128.162:102	\x01	01	01	0001	1 3 9999 113	113	✓	☐	☐ Enable	✓ Enable	☐ Enable
ConfigLoader	192.168.128.162:102	ConfigLoader	43 6F 6E 66 69 67 4C 6F 61 64 65 72					✓	☐	☐ Enable	✓ Enable	☐ Enable
AbbTestA1A	192.168.128.162:102	\x01	01	01	0001	1 3 9999 111	102	✓	☐	☐ Enable	✓ Enable	☐ Enable
AbbTestA2A	192.168.128.162:102	\x02	02	02	0002	1 3 9999 111	102	✓	☐	☐ Enable	✓ Enable	☐ Enable

To see the LDIB Editor click the **LDIB Editor** tab at the bottom of the Node Table.

If you have loaded a new or changed batch file, click **Refresh** to see the current values.

If you need to change the IP address of a VMD, you can change it in the **Network Address** column and then click **Apply** from the bottom of the screen

If you think your VMD or VCC addressing parameters are incorrect, you can check under the TSEL, SSEL, PSEL, AP Title, and AE Qualifier columns to see what they are set to in your configuration, make any necessary changes, and apply your changes by clicking **Apply**.

> **Note**: These changes will override values that you set if you or someone else loaded one or more batch files. However, you can restore these values if you reload the original batch file(s).

The following are descriptions of other columns:

- **IS** - International Standard is mutually exclusive with DIS. Select either IS or DIS.
- **DIS** - Draft International Standard is mutually exclusive with IS. Select either IS or DIS. Secure
- **ICCP** - Uncheck the box in order to disable Secure ICCP.
- **Monitoring** - Check the box to enable monitoring. Uncheck it to disable monitoring.
- **UCA** - Check the box in order to disable the Utility Communications Architecture.

LDIB File for LEC Server's OCX ActiveX MMS Client Interface

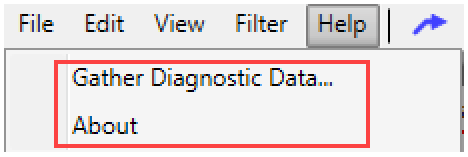
If your configuration needs to communicate with the LiveData MMS Client ActiveX Control facility, and you change an address or any of the networking parameters of a VMD in LDIB, you need to use the Write LDIB.ini button. Clicking **Write LDIB.ini** exports your VMD network addressing records into the requisite INI file format, which the LiveData MMS Client ActiveX Control facility requires to resolve VMD names to network addresses. Note that you do not have to click this button for changes to prototype nodes, nodes, or connectors.

Another use of the **Write LDIB.ini** button is to generate the LDIB.ini file, which allows you to validate your configured VMD addressing information.

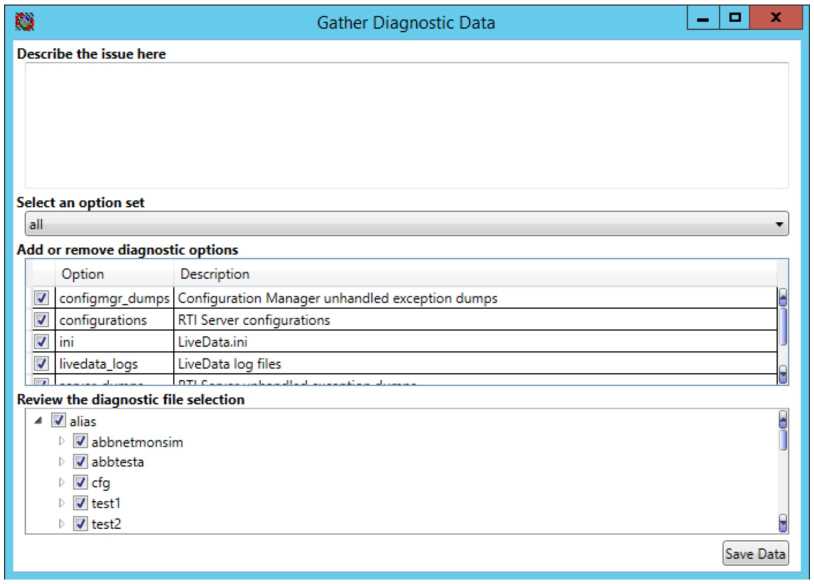
Clicking on the **Write LDIB.ini** button will replace the existing LDIB.ini with one that matches the LDIB for the current configuration. Existing entries in LDIB.ini will be overwritten.

Gather Diagnostic Data

There are two options under the Help menu: **Gather Diagnostic Data** and **About**.



You can use **Gather Diagnostic Data** if LEC Server crashes or runs into other problems. If you select **Gather Diagnostic Data**, LEC Configuration Manager will display a form in which you can select all or some of the configuration, initialization, log, and dump files that you want LEC Configuration Manager to compress into a ZIP file. You can then send this ZIP file to Oracle Utilities Professional Services for further examination.



Gather Diagnostic Data

Describe the issue here

Select an option set

all

Add or remove diagnostic options

	Option	Description
✓	configmgr_dumps	Configuration Manager unhandled exception dumps
✓	configurations	RTI Server configurations
✓	ini	LiveData.ini
✓	livedata_logs	LiveData log files

Review the diagnostic file selection

- ✓ alias
  - ✓ abbnetmonsim
  - ✓ abbtesta
  - ✓ cfg
  - ✓ test1
  - ✓ test2

Save Data

The form contains four labeled sections labeled as follows:

- **Describe the issue here:** Always provide a description of the issue.
- **Select an option set:** This section provides a drop-down list of sets of files that you can select to send to have compressed: all, dump_files_only, or log_files_only.
- **Add or remove diagnostic options:** This section allows you to add or remove specific files.

- **Review the diagnostic file selection:** This section allows you to review the files that you have selected and deselect any that you think are not necessary.

After reviewing your selections, click on the Save Data button to generate, name, and save the compressed .zip file. After generating the ZIP file, you can send it to Oracle Utilities Professional Services or to another team member at your company who could help debug the issue.

Glossary of System Components

This section is intended to be a reference for clarification on terms used throughout this document.

Batch Files

The batch files define the OPC UA client and server virtual devices, the OPC UA client and server agents, the local and remote VCCs, their properties, and their nodes/points as well as the association variables and client-side transfer sets.

Heartbeats

Application's Heartbeat: Each application OPC UA server writes to a heartbeat variable in an OPC UA server within an LEC Server instance. The application's OPC UA server updates the value of the variable every 10 seconds. The local OPC UA client monitors this heartbeat value, and if it is not updated in 30 seconds, the instance of LEC Server will go offline.

LEC's Heartbeat: The OPC UA server in each LEC Server instance has a heartbeat variable that is updated every 10 seconds. Each external OPC UA client monitors this heartbeat variable, and if the heartbeat is not updated in 30 seconds, the client OPC UA application will failover to another instance of LEC Server instance.

IFE (ICCP Front End)

LEC Server is configured to act as an IFE for an OEM's application and handle associations between the application's OPC UA clients and servers and devices of remote ICCP peers' devices. The processing engine behind each instance of LEC IFE is LiveData LEC Server.

Local ICCP Client

See ICCP Client (LEC).

Local OPC UA Client

See OPC UA Client (LEC).

Local OPC UA Server

See OPC UA Server (LEC).

Network Manager SCADA/EMS

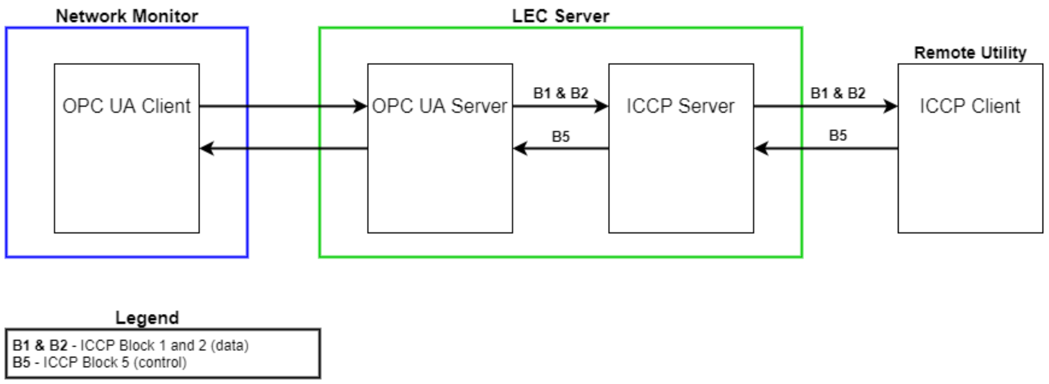
In this context, the OEM's application interacts with LEC Server instances and controls which LEC Server instances are in Active or Passive mode.

- **Active Mode:** If the Network Manager SCADA switches an IFE to be in Active mode, the IFE is turned on and each of its virtual devices are set up to make an association with specified remote virtual devices.
- **Passive Mode:** If the Network Manager SCADA switches an LEC Server instance to be in Passive mode, the instance is/remains turned off, preventing any associations from being made.

OPC UA Client (OEM Applications)

Each OPC UA client is under the control of the Network Manager SCADA. An OPC UA client will subscribe to telemetered data as well as status and statistical data from active and standby LEC Servers. For telemetered data, the OPC UA client will have an active subscription only to the active LEC Server instance. For statistical data, the OPC UA client will have active subscriptions to all LEC Server instances, regardless of their state, active or standby. In addition to subscribing to telemetry data, an OPC UA client will subscribe to a heartbeat variable from the active LEC Server instance.

Figure 15: An Application's OPC UA client Receiving Block 1 and 2 Data from the local OPC UA Client



The conformance block data that the application's OPC UA client receives from the remote VCC server is shown in Figure 15. This data is received by the LEC VCC client, and then processed in the LEC Processor so that it is compatible with the OPC UA data types.

In response to this data, the application's OPC UA client will send control data back to the VCC server. This data is processed into ICCP-compatible data by the LEC Processor.

OPC UA Agent

The OPC UA Agent is a node within the LEC Server instance that can enable or disable and monitor the OPC UA connection to the application's system.

OPC UA Server (LEC)

The OPC UA server is a virtual device (resource mapper) that transmits telemetry data (Block 1) from an LEC processor virtual device to an application's OPC UA client. The processor virtual device translates ICCP data into an OPC UA compatible form, having received the data from an LEC VCC that received that data from a remote ICCP server.

The OPC UA server also serves a role in transmitting control directives from the application's OPC UA client to the LEC Processor as shown in Figure 15.

Telemetry Data

The telemetry data received by the application's OPC UA client will provide the following statistical information:

- Number of points
- Last update time
- Number of points with bad quality as set by the remote system
- Number of points with bad quality as set by the local system
- CPU loading
- Memory consumption
- Uptime

OPC UA Tag

Tagging is the mechanism through which OPC UA clients can protect utilities technicians and equipment (usually during maintenance) by hanging a "red tag" (logically, not physically) on a device to prevent the device from executing subsequent operations. OPC UA clients can set, clear, and monitor tags in remote ICCP servers. Setting, clearing, and monitoring are all Block 5 operations.

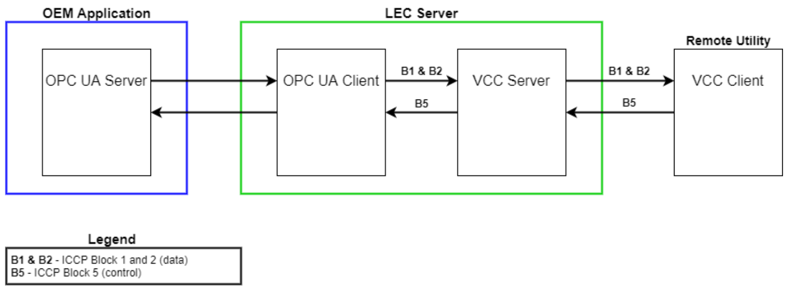
ICCP Client (LEC)

The ICCP client represents an ICCP virtual device, also known as a VCC in LEC Server. In this configuration, the VCC receives telemetry data (Block 1) from a remote VCC server. This data is processed in the LEC Processor so that it is compatible with the OPC UA data types as shown in Figure 3. The VCC client also serves a role in transmitting control directives (Block 5) from the processor to the remote VCC server also shown in Figure 15.

OPC UA Client (LEC)

The OPC UA client is a virtual device that receives telemetry data from an application's OPC UA server. It then transmits this data to an LEC processor virtual device, which transforms this data into ICCP Block 1 and/or Block 2 data and transmits it to the local ICCP server. The ICCP server then transfers the ICCP data to a specified remote ICCP client. In response to the transmitted data from the application's OPC UA server, the remote ICCP device transmits control directives (Block 5) through the reverse path (VCC Server -> Processor -> OPC UA client virtual device -> application's OPC UA server) as shown in Figure 16.

Figure 16: OPC UA Server to OPC UA Client (LEC) to Processor to VCC Server to Remote ICCP Devices



Inbound Association Control (VccAssocInControl): The Inbound Association Control node enables and disables the inbound ICCP association from another VCC.

Outbound Association Control (VccAssocOutControl): The Outbound Association Control node enables and disables the outbound ICCP association to another VCC.

Connection

This term refers to the underlying connection between a local and remote VMD or local and remote VCC.

Sessions

All OPC UA communications are made possible through sessions. In the IFE, the number of sessions per connection is configurable and specified in the node named AgentC of the OpcUaClient VMD and in the node named AgentS of the OpcUaServer VMD. Both AgentC and AgentS are based on the OpcUaAgent template in LEC Configuration Manager. Note that these values are set in the #IccpUaServerAgent and #IccpUaClientAgent tables in the sample header batch file AppTestAHeader.csv.

Subscription

This term describes a set of one or more points selected by the OPC UA client that the OPC UA server periodically monitors for the existence of some condition, and for which the OPC UA server sends Notifications to the client when the condition is detected. In the IFE, the number of subscriptions per connection is configurable and specified in the node AgentC of the OpcUaClient VMD and in the node AgentS of the OpcUaServer VMD. Both AgentC and AgentS are based on the OpcUaAgent template in LEC Configuration Manager. Note that these values are set in the #IccpUaServerAgent and #IccpUaClientAgent tables in the sample header batch file.

VCC

A VCC is an ICCP VMD.

VMD

A **VMD** is a container of nodes. Each VMD is associated with a specific type of communications protocol or interface. Most VMD types are intended to handle communications in and out of LEC Server using a particular communications protocol or interface, such as ICCP, DNP-3, Modbus, database access, or others. Thus, a VMD can map a device from the outside world to Oracle Utilities LEC's internal variable model, which allows LEC Server to capture, transform, and route data to other devices, systems, or applications in a form that the other device, system, or application can understand. Each instance of a VMD has a network address