

















GCM Ascend Driver Guide

April 26, 2016



This documents usage of the GCM Ascend driver on the Niagara platform.

QUICK START	3
MODULE INSTALLATION	4
NIAGARA AX.....	4
NIAGARA 4	4
LICENSING	5
PERFORMANCE	6
USE MULTIPLE CONNECTIONS	6
CAREFULLY USE BATCH POLLING.....	6
FINE TUNE WITH TUNING POLICIES.....	7
COMPONENT GUIDE	8
GCMASCENDNETWORK 	8
GCMTUNINGPOLICY 	8
GCMFOLDER 	8
GCMIPDEVICE 	9
GCMSERIALDEVICE 	9
LCMIPDEVICE 	10
LCMSERIALDEVICE 	10
GCMSUBNETDEVICEEXT 	11
GCMSUBNETDEVICE 	11
LCMSUBNETDEVICE 	11
GCMSUBNETFOLDER 	12

GCM Ascend Driver Guide

GCMPOINTDEVICEEXT 	12
GCMBLOCK 	12
GCMVEOGROUP 	13
GCMPOINTFOLDER 	13
GCMPROXYEXT 	14
DRIVER HISTORY	15

Quick Start

- Install the driver module in Workbench. Restart Workbench and install module on the station.
- Open the csi3Gcm palette and paste a GcmAscendNetwork object under the Drivers node in your station.
 - Set the number of threads in the thread pool to at least one more than the number of connected devices.
 - Configure the license object and reboot. This could be the very last thing you do – it is safe to build an unlicensed database.
- Double click the GcmAscendNetwork to enter the device manager view and press the new button to add connections.
 - Enter the credentials needed to log into each device.
 - Configure either the serial port name or the IP host and port.
- In the network's device manager view, double click the device network icon  in a device row to go to the subnet device extension of that device and discover additional devices. Do not add subnet devices for which you have purchased serial or IP connections.
- In the network's device manager view, double click the points icon  in a device row to go the point device extension and discover blocks and points.
- In the block manager view of a point device extension you can discover blocks and View Edit Override (VEO) groups.
- Double click a block to discover and add points.
 - Each point can use one attribute to read, and another to write.
 - If you are experiencing write faults, try changing the writeMethod property on the proxy extension of those points.
 - The write method is either edit or override. This corresponds to which GCM menu is used to write to the attribute. Edit is always used to read attributes.
- Move points from blocks to VEO groups. Invoke the program action on a VEO group to program the VEO. There can be only 20 points in a VEO group.
- Points can be moved out of blocks into point folders but those points can not be batch polled unless the folder is under a block or VEO group.
- Separate points for graphics from points with history extensions – see the performance section of this document.

Module Installation

Niagara AX

Install `csi3gcmAscend.jar` on the engineering tool where Niagara AX Workbench will be run. To install, place a copy of the file in the modules directory of your Niagara AX installation. This is typically `C:\Niagara\Niagara-3.n.nnn\modules`.

Install the module on the target station. Using a Niagara AX Workbench where the module has already been installed, connect to the station's platform service. Go to the Software Manager and install `csi3gcmAscend`.

Niagara 4

Install `csi3gcmAscend-rt.jar` and `csi3gcmAscend-wb.jar` on the computer where Niagara 4 engineering tool (Workbench) will be run. To install, place a copy of the files in the modules directory of your Niagara 4 installation. This is typically `C:\Niagara\Niagara-4.n.nnn\modules`.

Install the modules on the target station. Using a Niagara 4 Workbench where the modules have already been installed, connect to the station's platform service. Go to the Software Manager and install `csi3gcmAscend-rt` and optionally `csi3gcmAscend-wb` (if any views of the driver will be accessed with a browser).

Licensing

Every serial or IP connection must be licensed. Unlicensed connections will operate in demo mode for 2 hours. After demo mode expires, the station must be restarted to reestablish a connection, but it is otherwise safe to build a database.

Licensing is managed on an object in your database. The licensing object is located on the property sheet of the GcmAscendNetwork. It has the following properties.

- Connections – The number of serial or IP connections.
- Product Code – Text automatically generated by the driver that is needed to generate a license key.
- License Key – Where the key to validate the license must be entered.

Determine the number of connections needed. It is safe to build an unlicensed database so that you may determine your requirements through actual use. Enter the number of connections in the appropriate property of the license object.

Copy the value of the “Product Code” property that is automatically generated after entering the number of connections. You can highlight the value and copy it by pressing CTRL-C. Send the product code to your CSI³ representative. They will respond with a text string for you to enter in the “License Key” property.

You must restart the station after changing the “License Key”.

The exact text and case of the product code and license key are critical. Do not send screen shots. Highlight the text, copy it using CTRL-C and paste into an email.

Performance

Unlike fieldbus protocol drivers, this driver integrates with an ASCII terminal HMI. It is not a fast driver. Under ideal conditions, it can read or write a point in about two and a half seconds per connection. If the fieldbus network is exceptionally busy or the controllers are poorly programmed, performance can be slower.

Use Multiple Connections

Use one connection for graphics only. A network can have 100000 points in the Niagara sub-tree, but it will only need to poll points being viewed on a graphic. Do not trend these points, do not export them with another driver, and do not link their output.

Use additional connections for trending, export and logic. If a network has 100 permanently subscribed points, it will take $100 * 2.5s$, or 250s, or 4.2 minutes to poll each point. If those points are divided among two networks, the time gets cut in half and so forth.

Carefully Use Batch Polling

Polling multiple points at once is faster than individual polling. There are two point containers that perform batch polling: Blocks and View Edit Override (VEO) groups. VEO Groups can only batch poll. Blocks have a batchPoll property and it is true by default.

You do not want to batch poll when only a single point needs its value. In fact, batch polling may be slower. It is best to test which is faster: enable the "Poll Time" column in the point manager to monitor the performance of each method.

There are risks:

- It can impact the performance of the fieldbus network.
- It can max out CPU utilization of related devices.
- It might not be faster than individual point polling.

Fine Tune with Tuning Policies

Learning about and utilizing Niagara Tuning Policies (and poll rates) can marginally improve the performance of your application. The most important thing to remember is *not to poll points faster than necessary*.

Component Guide

GcmAscendNetwork

There can be only one GcmAscendNetwork in your station and it must be located under the standard “Drivers” node (/Config/Drivers). It has no physical correspondence with the underlying system.

The following properties are unique or have special importance:

- **Thread Pool** – This controls the number of threads used to execute all actions of all objects in the network. There should be one thread per connected device and at least one additional thread for other tasks.
- **Tuning Policies** – Policies such as when to read and write values are configured here.
- **License** – Discussed in another section.

GcmTuningPolicy

Tuning policies are used by point extensions to determine certain behavior such as when to read and write. There is a default tuning policy in the GcmAscendNetwork but additional policies can be pasted into “Tuning Policies” in the property sheet of the GcmAscendNetwork.

The default GCM tuning policy differs from the default Niagara tuning policy in that “Write On Start” is set to false.

The following properties are unique or have special importance:

- **Ignore Write Equal Out** – Do not write when the write value equals the current output.

GcmFolder

This is for organizing connection level objects.

GcmIpDevice

This represents a GCM that the driver communicates with via a serial connection tunneled over IP.

The following properties are unique or have special importance:

- **Username** – The username required to log into the device.
- **Password** – The password for the given username.
- **Timeout** – How long the device pause at any given time and after which the driver gives up waiting for a response.
- **Inter Char Delay** – How long to pause between sending characters of text.
- **Ip Host** – IP address or host name.
- **Ip Port** – IP port number.
- **Subnet** – See GcmSubnetDeviceExt.
- **Tunnel** – If enabled, this will accept TCP/IP connections on the configured port and relay any data between the Jace and the device. Do not leave the tunnel enabled when it is not needed, any incoming connection from the internet can disable normal communications.

GcmSerialDevice

This represents a GCM that the driver communicates via a serial connection.

The following properties are unique or have special importance:

- **Username** – The username required to log into the device.
- **Password** – The password for the given username.
- **Timeout** – How long the device pause at any given time and after which the driver gives up waiting for a response.
- **Inter Char Delay** – How long to pause between sending characters of text.

- **Serial Config** – The default settings other than the port name do not need to be changed (9600,8,1,N).
- **Subnet** – See GcmSubnetDeviceExt.
- **Tunnel** – If enabled, this will accept TCP/IP connections on the configured port and relay any data between the Jace and the device. Do not leave the tunnel enabled when it is not needed, any incoming connection from the internet can disable normal communications.

LcmIpDevice

This represents a LCM that the driver communicates with via a serial connection tunneled over IP.

The following properties are unique or have special importance:

- **Username** – The username required to log into the device.
- **Password** – The password for the given username.
- **Timeout** – How long the device pause at any given time and after which the driver gives up waiting for a response.
- **Inter Char Delay** – How long to pause between sending characters of text.
- **IpHost** – IP address or host name.
- **Ip Port** – IP port number. **Tunnel** – If enabled, this will accept TCP/IP connections on the configured port and relay any data between the Jace and the device. Do not leave the tunnel enabled when it is not needed, any incoming connection from the internet can disable normal communications.

LcmSerialDevice

This represents a LCM that the driver communicates with via a serial connection.

The following properties are unique or have special importance:

- **Username** – The username required to log into the device.
- **Password** – The password for the given username.

- **Timeout** – How long the device pause at any given time and after which the driver gives up waiting for a response.
- **Inter Char Delay** – How long to pause between sending characters of text.
- **Serial Config** – The default settings other than the port name do not need to be changed (9600,8,1,N).
- **Tunnel** – If enabled, this will accept TCP/IP connections on the configured port and relay any data between the Jace and the device. Do not leave the tunnel enabled when it is not needed, any incoming connection from the internet can disable normal communications.

GcmSubnetDeviceExt

This represents the network of devices available by tunneling through another device.

GcmSubnetDevice

This represents a GCM that the driver communicates with by tunneling through another device.

The following properties are unique or have special importance:

- **Username** – The username required to log into the device.
- **Password** – The password for the given username.
- **Subnet** – See GcmSubnetDeviceExt.

LcmSubnetDevice

This represents a LCM that the driver communicates with by tunneling through another device.

The following properties are unique or have special importance:

- **Username** – The username required to log into the device.
- **Password** – The password for the given username.

GcmSubnetFolder

This is for organizing subnet level objects.

GcmPointDeviceExt

This represents the tree of points available on the parent device.

GcmBlock

This represents a block in either a GCM or LCM.

The following properties are unique or have special importance:

- **Address** – Block type : block name.
- **Batch Poll** – An optimization that polls all attributes of a block at once rather than individually. No two Niagara points in the same batch polled block can have the same read attribute.

The following actions are unique or have special importance:

- **Force Update** – Add a poll message to the urgent queue.

GcmVeoGroup

This represents a View/Edit/Override group in either a GCM or LCM.

Discovery assumes GCMs have 10 possible VEO groups, and LCMs have only 1 VEO group. If it is possible to have more on your device, you can manually paste them from the csi3gcmAscend palette.

Manually add up to 20 points to a VEO group. Discover points normally and then move them into the VEO group. Once the VEO group has all of the desired points, invoke the “Program” action on the VEO group.

The following properties are unique or have special importance:

- **Program State** – Status information about a Program VEO action invocation.

The following actions are unique or have special importance:

- **Force Update** – Add a poll message to the urgent queue.

GcmPointFoldeer

This is for organizing point level objects.

GcmProxyExt

This is a point extension in the NiagaraAX architecture. Points with this extension represent points on the remote device. Points with this extension must be in GcmPointFolde, GcmBlock or GcmVeoGroup.

The following properties are unique or have special importance:

- **Block** – The block of this attribute.
- **Read Attribute** – The attribute name in the parent block to poll.
- **Write Attribute** – The attribute name in the parent block to write to.
- **Write Mode** – Attributes can be edited or overridden which correspond block edit and block override menus on the GCM. If the attribute is overridden, the override time on the GCM will be set to over 190 years. The override time will be cleared when the Niagara point outputs null. The default value is override but many points must be edited. If points fault on write, try changing this property.
- **Write Pending** – True when a write message is on the queue. Returns to false after the write is complete.
- **Value** – The unparsed valued read from the system.
- **Device Facets** – These facets are used to map GCM values into Niagara. For Boolean points, it is important that the trueText and falseText be set.

The following actions are unique or have special importance:

- **Auto** – The auto action on the proxy extension, not the point, will clear an override on the GCM.

Driver History

April 26, 2016: Module version 3.8.0

- Module update for N4 conversion.

January 30, 2012: Module version 3.6.36.1

- Fixed status propagation to subnet devices.

October 7, 2011: Module version 3.6.36

- Points could get stuck in write pending.

November 12, 2010: Module version 3.5.25

- Points would not poll after writes if the point was not subscribed. Now the driver simulates a read with the value that was correctly written.
- When batch poll was set to true on GcmBlocks, points would not write until the block was viewed in either a graphic or the point manager view.
- ABNORMAL LOST COMM now results in poll fault.

November 11, 2009: Module version 3.4.51

- Point discovery of Alarm blocks could miss several attributes if the message text of the alarm wraps to a second line.

February 19, 2009: Module version 3.3.27

- Added IgnoreWriteEqualOut to the GCM tuning policy.
- Once a point was released, no subsequent writes could take place.
- Added the telnet tunnel to connection objects.

October 8, 2008: Module version 3.3.23

- Blocks would batch poll even when batchPoll was set to false.

February 18, 2008: Module version 3.3.13.2

- License change, driver no longer distinguishes between types of connections.
 - Existing installations will need to be relicensed.

January 11, 2008: Module version 3.3.13.1

- Write messages in won't throw an error if the device is no longer connected.
- Connection status now propagated to subnet devices.

December 26, 2007: Module version 3.3.13

- Point discovery could cache results from one Jace and show them in another.

October 29, 2007: Module version 3.2.16.10

- Added writePending to GcmProxyExt
- Added daily polling statistics to GcmBlock and GcmVeoGroup.
- No urgent poll messages until connection object attaches. This eliminates the huge queue of urgent messages at station start.
- Added force update to VeoGroup and GcmBlock.

August 31, 2007: Module version 3.2.16.9

- Point reading now only uses the edit menu, even if the write mode is override.
- Fixed a problem when write mode was edit.
- If the write mode changes from override to edit, an auto message will be written to the GCM to clear the override.
- The driver has always attempted to read an attribute immediately after writing it. Often, attributes go into the ABNORMAL LOST COMMUNICATIONS state after a write. This was causing read failures. Now when that happens, readOk will be called with the successful write value.

August 7, 2007: Module version 3.2.16.8

- Fixed – The last release broke point discovery.
- Made programState property on VeoGroup persistent so there is an easily accessible record of when a group was programmed.

August 6, 2007: Module version 3.2.16.7

- Fixed – could not log onto LCMs under subnet GCMs.
- Added programState property on VeoGroup to monitor the status of a VEO programming message.
- Major stability improvements.
- Performance optimizations.

July 13, 2007: Module version 3.2.16.6

- Noise filtering introduced an issue that could short circuit message timeouts and unnecessarily cause read failures. Most notably, some users could not log into the GCM.

June 14, 2007: Module version 3.2.16.5

- Less leniency
- Fix VEO Poll and Program
- Timeout can now be longer than 60 seconds (!)

June 11, 2007: Module version 3.2.16.4

- Enhancement – some noise filtering and more leniency when parsing responses from the system.

May 30, 2007: Module version 3.2.16.3

- Bug Fix – Point discovery was failing in the cleanup phase of discovery, after everything had been learned. Unexpected data is now ignored in that last phase.

May 17, 2007: Module version 3.2.16.2

- Bug Fix – Reattachment would not actually open and close serial or IP connections. IP connections would never be able to reconnect to remote devices after going down. It is not known if this affected serial connections.
- Bug Fix – Write faults would occur for attributes that did not take override times.
- Bug Fix – Fix writes to FLO2 block attributes.

May 4, 2007: Module version 3.1.30.1

- Enhancement - Messages now have three attempts before they fail. Because of this, the default timeout has been dropped to 7 seconds.
- Updated the known units to include all 13 predefined units.
- Bug Fix – If devices went {down}, status remained down even after a successful reconnect.
- Bug Fix – Licensing was broken and would say the license was valid even when it was not.

April 28, 2007: Module version 3.1.30

- First release.