North

Commander Tutorial

This tutorial shows how to use the main features of Commander version 2. It covers Commander's integrate, control, and inform functionality, as well as introducing general North technology such as Zip and the Start Engineering applications.

Work through this tutorial, following the steps marked '💻' using the North Training Pack, containing: ObSys software, Commander, and Zip modules.

This tutorial relates to the Commander v2.0 (hardware v3) (base firmware build 01/06/26), distributed with the July 2026 release of North ObSys.

Contents

Getting started…

What is Commander?

Commander is the smallest of North's building controllers, which also includes ObServer. All the controllers contain North's interface technology, block-based programming language, and easy-to-use information services. Commander can work as a stand-alone controller; linked with Commander Hub for easy remote monitoring; or alongside other North controllers and display systems to create a larger control or monitoring solution.

Interface Technology

Commander includes North's interface technology. Commander can access values from thousands of different systems in a common way, using North drivers. This ability allows Commander to pass data between different systems and enables different sub-systems within a building to be fused together to form a single, coherent system.

Programmable Control

ObVerse is North's block-based programming language. It is available in all North controllers. Although it is easy to use, it provides real flexibility during engineering, allowing the engineer to incorporate design changes with minimal effort. Date and timer functions are standard, along with feedback control and logic.

Information Services

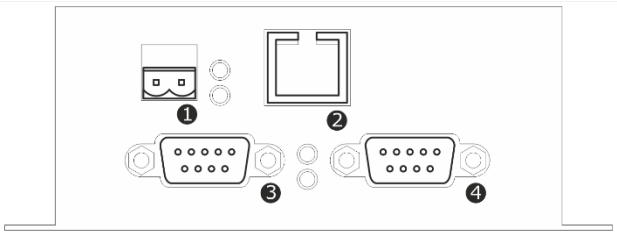
Commander supports North's standard protocol, allowing secure communications with other North products, including powerful engineering tools and display software. Commander generates and serves local web pages automatically, and can link to Commander Hub the cloud-based meeting place for Commanders, these both provide a consistent user display on all browsers. Commander can monitor and inform users about alarm conditions, using email or SMS for example.

You can extend information services, if necessary, by using North's driver technology: values from within Commander can be made available to, say, BACnet and Modbus devices.

Commander Hardware

Commander's black steel case contains a two-board device. The upper board contains the main processor, memory, LAN port, battery and switches. The lower board contains the power regulation and the isolated RS232 ports.

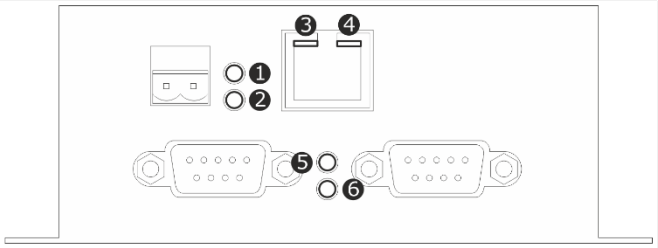
Power and Connectors



❶ PWR		12V-24V AC/DC at 3W
❷ LAN		Isolated 10BASE-T/100BASE-TX Auto-MDIX
❸ COM1		Isolated RS232 port
❹ COM2		Isolated RS232 port

Commander's power connector (PWR) is polarity independent. Use a 12 to 24 Volts, either AC or DC, power supply. Although Commander uses approximately 3VA when running (therefore at 12V it requires approximately 250mA and at 24V it requires approximately 125mA), North recommend you use 6VA supplies. We recommend using a good quality regulated DC power supply, for example 12V DC providing 500mA per Commander.

LEDs



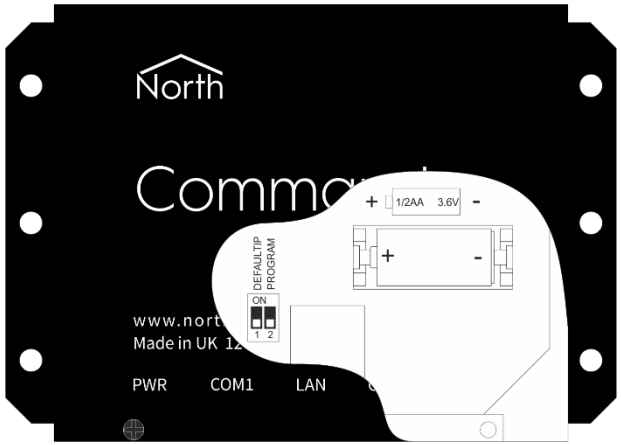
❶ MODE	Green	Off=No power, On=Run, Blink=Default IP or Program mode
❷ SAVE	Green	On=Writing to flash memory
❸ LAN SPEED	Yellow	Off=10 Mbps, On=100 Mbps
❹ LAN LNK/ACT	Green	Off=No link, On=Link, Flash=LAN port activity
❺ COM1 ACT	Red	On=COM1 port activity
❻ COM2 ACT	Red	On=COM2 port activity

Battery and Switches

Remove the lid to access Commander's battery and switches.

When fitted, the battery retains Commander's engineered settings during a power-down. The lithium-ion battery is non-rechargeable but retains its charge for a long time, typically needing replacement after 10 years.

Battery type: 1/2 AA size, 3.6V, 1.2Ah, Lithium Thionyl Chloride (Li-SOCI2)



Setting the **DEFAULTIP** switch to ON restarts Commander in Default IP mode, to use its default IP address of 192.168.192.167 instead of any assigned IP address. This is useful if you do not know the IP address of Commander. It also unlocks the device – pausing security and enabling a DHCP server and Web Server. Setting the switch OFF forces Commander to restart and use its assigned IP address and security settings.

Setting the **PROGRAM** switch to ON (and restarting) allows programming of new firmware into Commander's flash memory.

(Previous hardware versions have a **FACTORY** switch which should always be left OFF).

Battery Safety

CAUTION

Risk of fire, explosion or severe burn hazard:

DO NOT recharge, mechanically crush, disassemble, short-circuit, heat above 100°C, incinerate, or expose contents to water.

Installation

Locate Commander so it is accessible to remove the lid and set the internal switches when required. Leave a clearance of at least 20mm around the device, with 80mm minimum at the cable connection edge.

Choose a safe location away from vibration, shock, and electrical noise; in an area where there is no excessive moisture.

Installation on an even surface

Use the chassis as a template to mark and drill four 4mm holes. Screw the Commander securely either directly on to a wall or within a panel using the mounting holes.

Installation on DIN-rail

An optional DIN-rail mounting kit is available, order code MISC/DINKIT. Attach the two clips to the back of Commander using the bolts provided. Clip on to standard 35 mm DIN-rail, 'top hat' compatible.

Prepare Commander

To prepare Commander for use, follow these steps:

→ Detach the lid by removing the two screws, and sliding it off

→ Set the **DEFAULTIP** switch to ON, to force Commander to a known IP address of 192.168.192.167

→ Connect and apply power, 12 to 24V AC or DC – approximately 3W per Commander

→ Check that the MODE LED is illuminated green to indicate a healthy power, and blinking to show Commander is in Default IP or Program mode

→ Insert the supplied 3.6V battery into the holder in the direction indicated with + symbol, and clip in place.

Installing ObSys, the Engineering Software

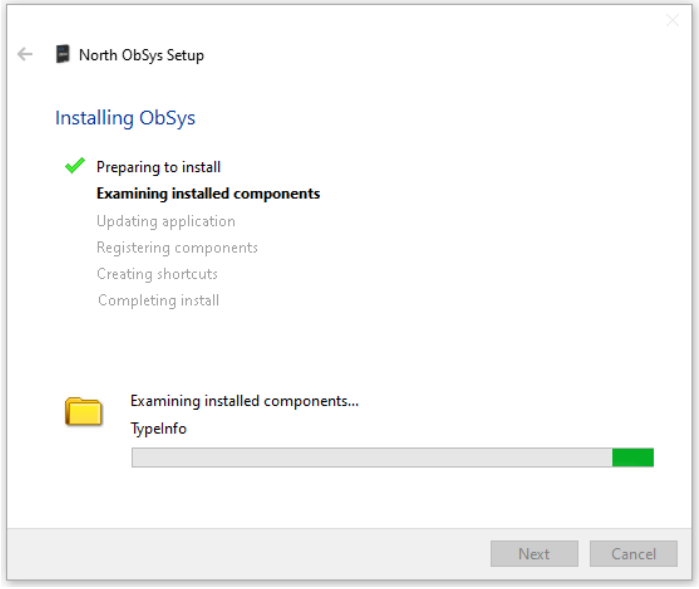
You can engineer all North products using North's ObSys software package. This is a collection of Windows applications, which are installed on a PC - most engineers use a laptop, as ObSys requires little processing power or disk space.

Starting Installation

Download ObSys Setup from the *www.northbt.com* website. Once downloaded, run the auto extracting SETUP.EXE and wait for North ObSys Setup to launch.

ObSys Setup

- To specify what to install, follow these steps:
  - → At the **How do you want to use ObSys?** welcome page, press **To engineer North products.**
  - → Read the Licence Agreement: if you are happy to, select **I accept...**, and press **Install**.



ObSys Overview

ObSys is a package containing several Windows applications. ObServer is the main application. ObView is another important application. Engineering applications needed later also include ObVerse Editor, WebView Editor, and Object Editor.

ObServer, the Communications Router

ObServer, shortened from Object Server, is the communications router of ObSys. Other compatible applications can link to, and communicate via, ObServer. ObServer also supports interface drivers, supplied in ObServer Module files, OSM files, and these can communicate via ObServer. ObServer can also link across physical networks to other North IP Device compatible products.

When Start Engineering, or any other ObView window, is started, it automatically runs ObServer if necessary.

ObView, the Object Viewer

ObView is an application that uses ObServer to communicate. ObView provides a simple way of discovering, showing, and editing values. It also supports more advanced features, such as engineered views, and can support different levels of users with tailored views – but you do not need these when simply engineering other North products.

To start engineering using ObView, follow these steps:

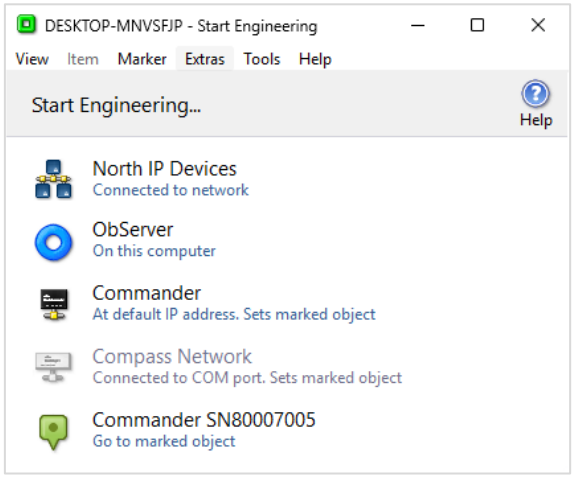
→ From Windows **Start**, select **All Programs > North ObSys > Start Engineering**

Note: ObSys used for engineering purposes typically runs in 12-hour time-restricted mode (it stops after 12 hours and you just need to restart it). We offer a dongle so ObSys will run without stopping - for a fee – which is used for permanent display systems.

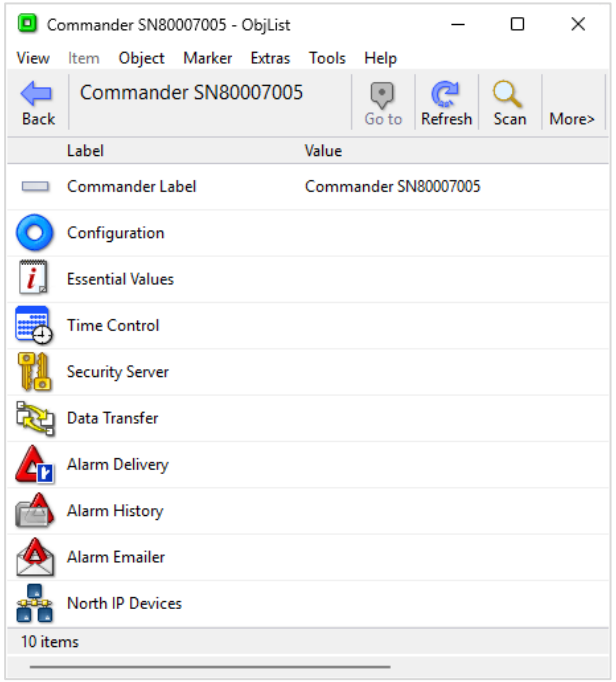
Basic Commander Settings

The first things to do, when setting up a Commander, are to set its label and real-time clock.

Engineering Commander at Default IP Address



- To set Commander to a known IP address, and start engineering, follow these steps:
  - → Connect the PC to Commander's LAN port using Cat 5 or better Ethernet cable. It will detect whether a patch or cross-over cable is used and work with either
  - → Set Commander's **DEFAULTIP** switch to ON. This will force Commander to ignore its specified LAN settings and use the default IP address 192.168.192.167, with a subnet mask 255.255.255.0
  - → If your PC's Ethernet port is set to Automatic (DHCP mode) IP assignment, it can use Commander's DHCP server to get a valid address. Otherwise, set IP assignment to manual with an IPv4 address of 192.168.192.1 and subnet mask 255.255.255.0
  - → From Windows **Start**, select **All Programs > North ObSys > Start Engineering**
  - → From Start Engineering, select **Commander at default IP address** – ObView will scan the network, display Commander, and set the marker to this object. If Commander was not found, this may be because the PC does not yet have an IP address. Wait a moment, check the LAN LNK/ACT LED, then select the option again
  - → ObView will auto-scan and show the top-level of Commander (shown right). You can also press the **Scan** button to manually scan the view
  - → Click on **Configuration** object to open it, and ObView will again auto-scan and show a list of Configuration objects found
  - → Press the **Back** button to go back to the previous view – the top-level objects within Commander. At any time, you may re-run Start Engineering and select Commander (at default IP address) to get to the top-level of Commander.



Changing Commander's Label

When first supplied by North, Commander's label consists of the word Commander followed by its serial number – for example 'Commander SN80002014'.

From the ObView window that shows the top-level of Commander, you can check the current setting of the Commander Label – although it cannot be changed from this page.

To set Commander's label, follow these steps:

→ From the top-level of Commander, click on **Configuration**

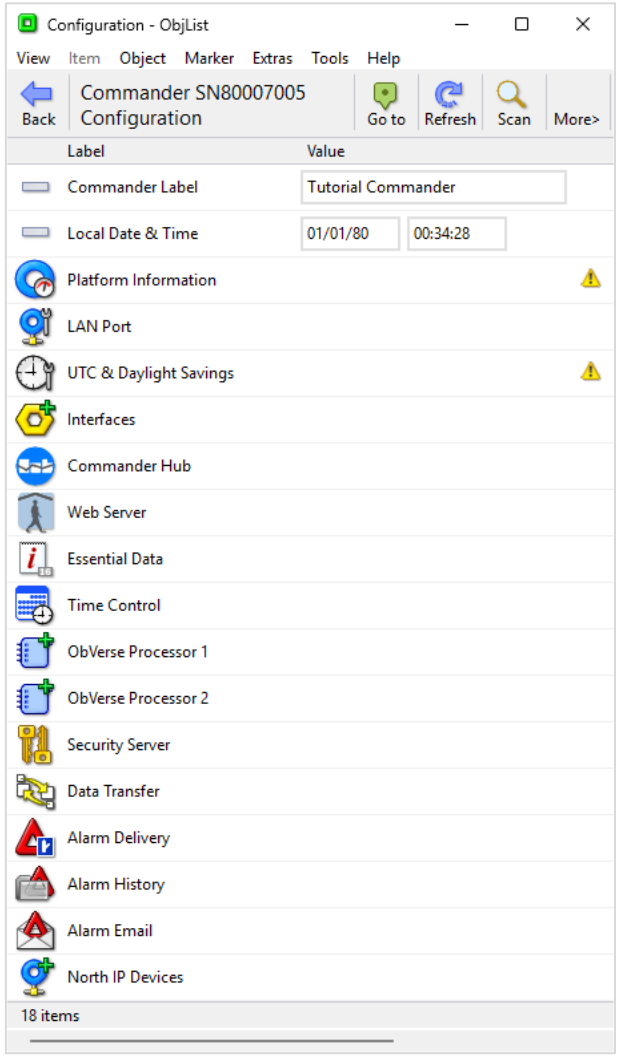
→ Move your mouse over the words 'Commander Label' – the mouse is in the shape of a hand – which means the value can be changed

→ Click the **Commander Label** object (where the mouse is a hand) and the adjust popup window will appear, showing the current value

→ Modify the text in the box, and press **OK** – Commander will only allow text up to 30 characters in length in this Label

→ Press the **Back** button to get back to the top-level of Commander

Notice how some objects have a value to the right of the label. Values that can be viewed, but not adjusted, are shown without a box, values with a box around them can be adjusted – you change the object's value by left-clicking on the box, modifying the value on the popup window, and pressing Ok. Some objects have no value, but instead are container objects (they contain sub-objects). Clicking a container opens that container and clicking on the Back button closes it – we will cover this later.



Setting Commander's Clock

Commander has a battery-backed clock that holds the current time. This is used to time-stamp alarms and values, including data sent to Commander Hub, and ObVerse programming language can also use it. Once set, Commander can maintain an accurate clock by synchronising with an Internet time server.

To set the local time within Commander, follow these steps:

→ From the top-level of Commander, click on **Configuration**

→ Click on each **Local Date & Time** value – an adjust popup window will appear to set the date and time individually. Modify the date and time in the box – notice the format of the date, *dd/mm/yy*. The time format can be *hh:mm* or *hh:mm:ss* – press **OK** when you are finished adjusting. The new date and time will appear on the window if set correctly.

Tip: Select **Extras** menu, **Set Date & Time from PC** to automatically set Commander's date & time with your PC's current data & time.

Resetting Commander

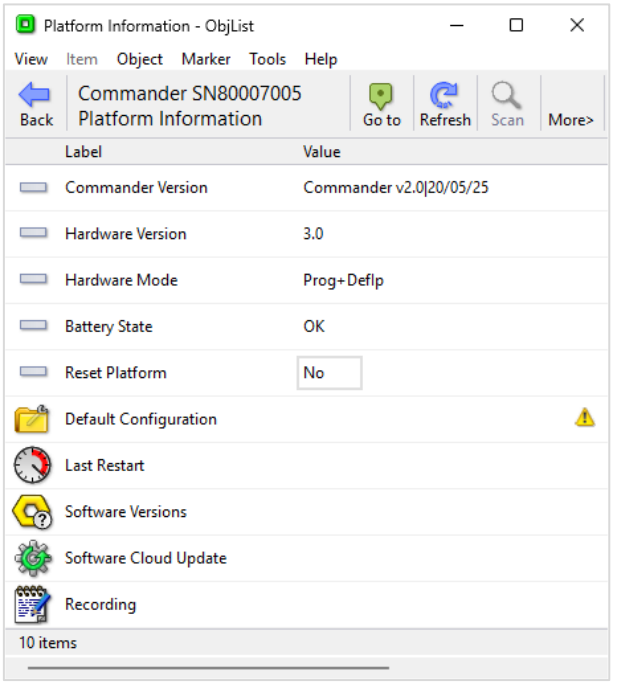
Some changes have no effect until Commander is reset – there are several ways of doing this: if Commander is near, you can power it off and on, you can change its **DEFAULTIP** switch, or you can perform a reset using ObView. If you wish to reset Commander using ObView, follow these steps:

→ Press **Go to** marked object, to go to the top-level of Commander, and then click on **Configuration,** and **Platform Information**. This page shows the version of software in Commander, and other information relating to the Commander hardware itself

→ Click on **Reset Platform**, select 'Yes', and press **Ok** – the Commander will reset itself – press **Refresh** to get the page to re-request all the values – you may have to wait until Commander re-establishes Ethernet connections and obtains an IP address

→ Click on **Last Restart** and notice the **Last Start Date & Time** shows a recent time, **Reset Count** has incremented, and **Reset Reason** shows 'User reset'

→ Press **Go to** marked object, to return to the top-level of Commander.



ObView Window

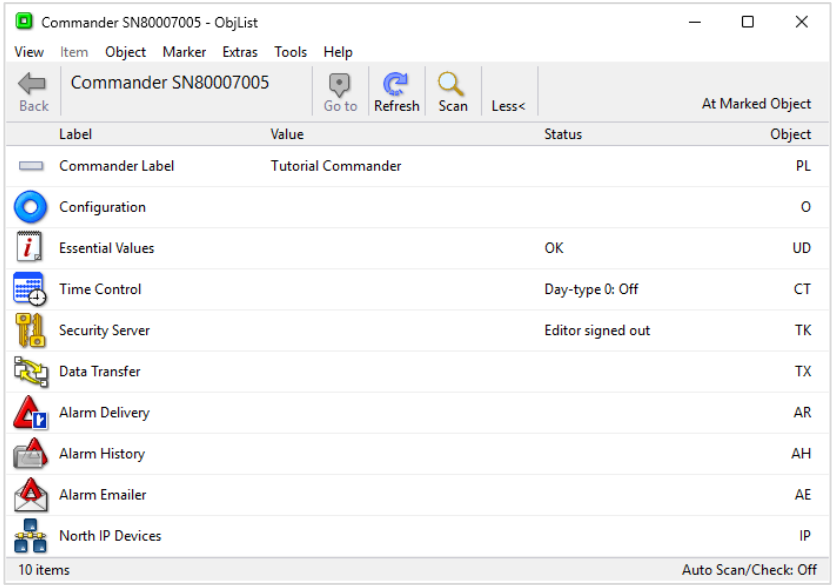
We have just used ObView, the Object Viewer, to do some simple editing of Commander. Before we go further, let us take a quick look around the ObView window itself, so you can understand how the main engineering software works.

Objects within the North system are organised in a tree-structure, in a similar way to files and folders on a PC's drive. ObView allows us to navigate over this tree structure, by displaying information about a particular object, including its sub-objects, and allowing us to move up and down the tree.

Menu

The menu allows access to certain commands and functions to perform on the displayed object – we will learn about these, as we need them.

Toolbar Area



The toolbar area below the menu shows information about the displayed object and allows the most common actions to be triggered quickly:

Back will load the view of the previously opened object (usually of this objects parent.)

The label area shows the label of the device and the displayed object within that device.

Go to will load the view of the Marked Object – in our example, Commander's top-level.

Refresh will reload the page and re-collect its values and statuses.

Scan causes ObView to re-scan the displayed object for sub-objects. You may need to do this after a configuration change. ObView enables the button only on objects that have a variable number of sub-objects.

More increases the window to full width (see image) – showing status and object reference columns. **Less** decreases the window width back to it's compact view – you can still scroll the window right and left.

Object List

Beneath the toolbar is the main area of the window. It contains the list of sub-objects that are within the displayed object.

Each sub-object appears on a single line, starting with an icon and label, followed by a value (if the sub-object has one) or a status (if the sub-object has one), finally ending with an object reference.

If there are too many sub-objects to fit on the main area, scroll-bars appear on the right-hand side of the window.

Left-clicking on a sub-object causes an action depending on the type of sub-object:

If the object has an adjustable value, with a box surrounding the value, then left-clicking on the adjustable value will allow you to adjust the value – we saw this earlier.

If the sub-object has a non-adjustable value, then left-clicking on the sub-object has no effect.

If the sub-object contains other things, then left-clicking on that container will cause ObView to open the container and display its sub-objects – again we have seen this as we navigate around.

Right-clicking on a sub-object shows a popup context menu with various actions that can be performed on or with the sub-object:

If the sub-object is adjustable, then **Adjust** displays the popup window for adjusting it. **Delete Value** simply sets the object to zero or blank.

If the sub-object contains other things, then **Open** opens that container, and displays its sub-objects – this is the default action when you left-click on the object.

Cut Value, **Copy Value** and **Paste Value** perform clipboard actions on the current value.

Summary

In this section you have learnt:

- ✓ Commander hardware – understand power requirements, identify connectors and LEDs, battery safety, internal switches
- ✓ Engineering software – installed ObSys on your PC
- ✓ Basic Commander settings – connected to Commander at default IP address, set the label, set the date & time, restarted Commander
- ✓ ObView Window – identify areas of the window and what they do.

Integrating…

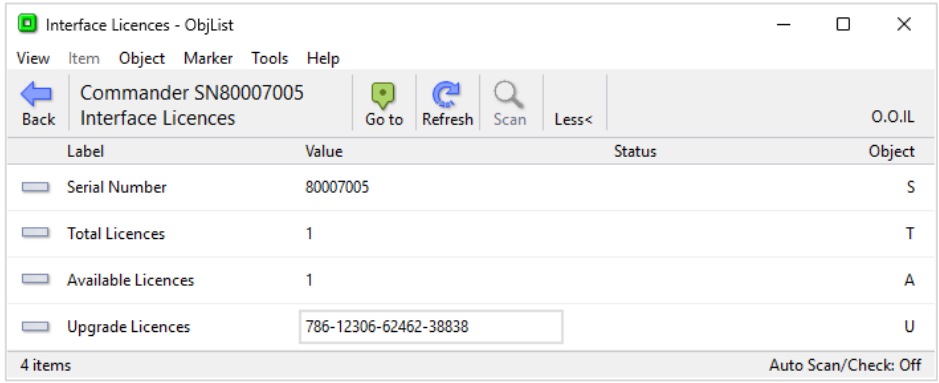
Interfacing Commander to other Systems

One of the roles of Commander is to provide interfacing to third-party systems. It does this using North's interface technology. For each third-party system, North produce a driver, a software protocol-converter, that converts the communications language of that system to North's standard object model. North products are based on different hardware platforms, and so North supply drivers in platform-compatible files.

Drivers for Commander come in Commander Module files, or CDM files. Commander comes with popular CDM files pre-installed. It is possible to install other CDM files and to update the CDM files to the latest version – we cover this later in this tutorial.

Although Commander has many pre-installed drivers, it only supports up to four operating interfaces at one time, and each requires a driver.

Interface Licences within Commander



North supply Commanders with a certain number of interface licences (ILs). These are used to license the use of drivers.

As Commander starts an interface using a driver, that driver borrows an interface licence from Commander – if Commander does not have any available, the driver will not operate, and the interface will not start.

When an interface is no longer required, and the driver stops, the driver returns the borrowed interface licence to Commander – allowing another driver to borrow it, and therefore allowing you to change the interface as you require.

To see the current Interface Licence information within Commander, follow these steps:

→ Open **Configuration, Interfaces, Interface Licences**. Each Commander has a unique **Serial Number**. **Total Licences** shows the number of licences the Commander has - **Available Licences** shows how many are still unused

If you wish to increase the interface licences with a Commander, *telephone North*. North will need details from the Interface Licences page to guide you through adding more.

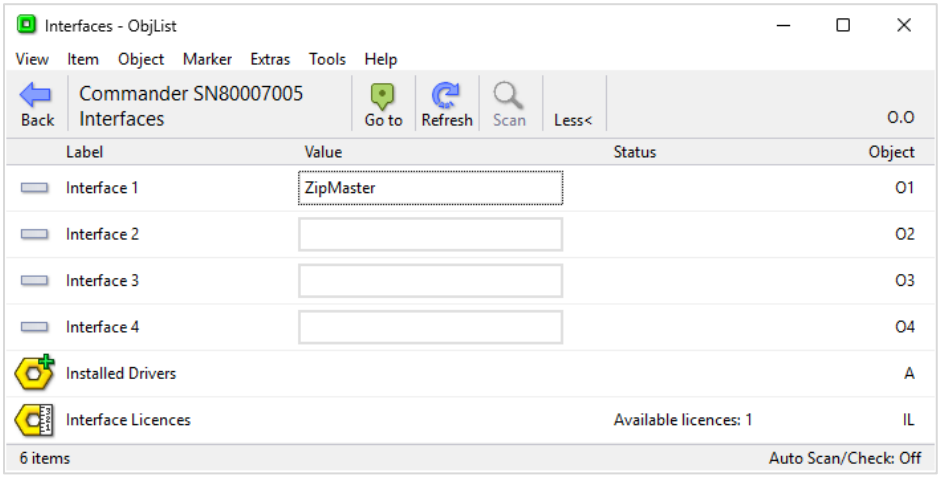
Starting an Interface

Before you start an interface using a driver, you need to find out whether the driver is installed on the Commander, and then specify that the interface use that driver.

In this tutorial, we will set up Interface 1 to use the ZipMaster driver, as we will need this later.

To set up Interface 1 to connect to a Zip system, follow these steps:

→ Open **Configuration, Interfaces** to view the interfaces currently set up – and the drivers they use – if they are all blank, then there are no interfaces. Open **Installed Drivers** to view a list of drivers currently installed in the Commander



→ Scroll up and down and find the **Installed Driver** object that has the value 'ZipMaster'. Copy this value by right-clicking on the object and selecting **Copy Value** from the popup menu. (You can also copy the object's value by clicking to highlight the object, and pressing CTRL+C on the keyboard)

→ Press **Back**, then right-click on **Interface 1**, and select **Paste Value** from the popup menu. This will set the value to 'ZipMaster'. (You can also paste to an object's value by clicking to highlight the object, and pressing CTRL+V on the keyboard)

If there are enough available interface licences available, the driver will operate and will appear in the top-level of Commander (you will need to re-scan that level.)

Stopping an Interface

To remove a driver, and therefore stop an interface, follow these steps:

→ Open **Configuration, Interfaces**. Right-click on **Interface 1** and select **Delete Value** from the popup menu. This will delete the value and therefore stop the driver.

Important: to continue following the tutorial, set Interface 1 to 'ZipMaster' again to start the ZipMaster driver – we will use it next!

Assembling the Training Pack Zip components

This tutorial, and others, are written assuming you have access to a set of Zip modules, which allow you to try out certain features of the products. If you are using the Zip components from the Training Pack, they might need assembling.

Zip components typically require connecting, so that a Zip module can share a NetCard's power supply and Zip network. The components are made up of a circuit board with input and output connectors, and green carrier. Each module is supplied with the correct amount of green carrier for the module. Each NetCard is also supplied with two green carrier endcaps.

To assemble the Zip modules in the Training Pack, follow these steps:

→ Remove the components from the NC12B, M7002, and M7007 boxes, and slide the circuit boards from the green carrier

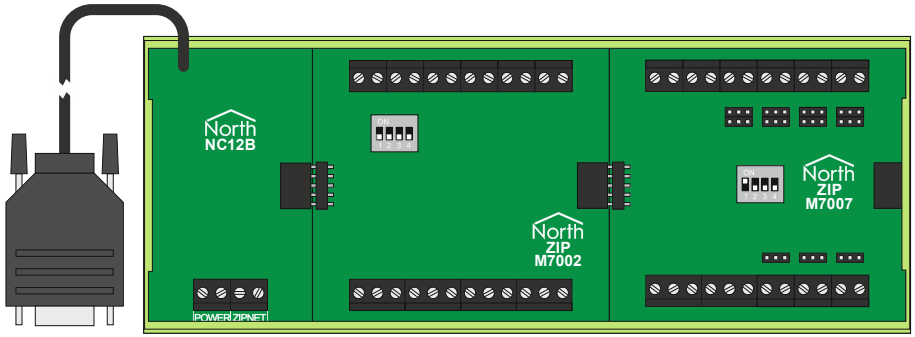
→ Clip together the green carrier, except for one end-cap.

→ Slide into the assembled carrier the NC12B first, then the M7002, then the M7007, ensuring that the 5-pin connection on each module slots into the 5-plug connector of the previous

→ Add the final endcap to hold everything in place. This keeps the circuit boards connected and prevents you touching the underside of any modules when they are in use

→ With the NC12B on the left, set the address switches of the M7002 to Off-Off-Off-Off (address 0) and the M7007 address switches to On-Off-Off-Off (address 1)

→ Connect 12VDC to the NC12B's power connector– the NC12B's green POWER OK LED should light, and the M7002 and M7007 green OK LEDs should blink.



Interface Set up

When Commander starts an interface, the interface normally adds two new objects to the top-level of Commander. We started the ZipMaster interface earlier, so we now need to **Scan** the top-level.

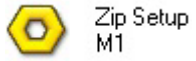
To scan the top-level of Commander, follow these steps:

→ Press **Go to** or **Back** to return to the top-level of Commander

→ Press **Scan**

→ Once finished, ObView shows the objects found, including new objects for the interface.

The first object that the interface adds is the Setup object, which has a yellow hexagonal nut icon (see the icon to the right). This is where values relating to the driver's operation are – things like RS232 port numbers, baud rates, and system labels. The second object added to the top-level of Commander represents the system that can be accessed using the driver.



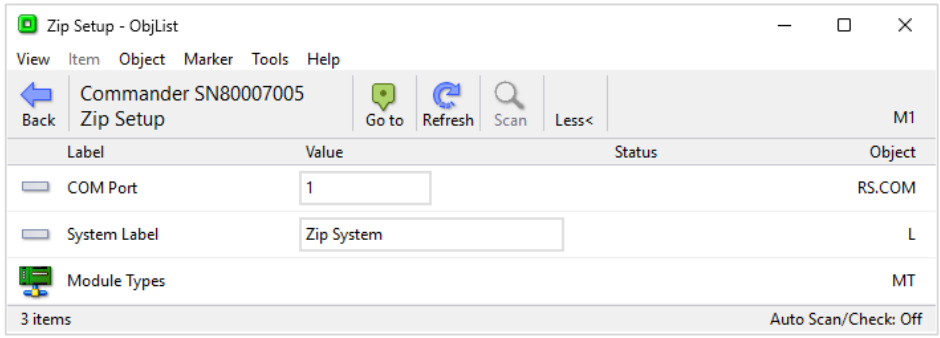
Different drivers have different setup values, which, because of their diversity, are not documented here.

To configure the Zip interface for the Training Pack Zip, follow these steps:

→ Open **Zip Setup** to view the Zip setup objects

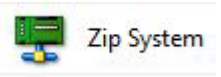
→ Set **COM Port** to '1', to tell the driver to use COM1 on the Commander

→ Use the cable to connect the NC12B to the Commander's COM1 port - the green OK LED of each Zip module should stop blinking and become solid on to indicate that it has a link with the ZipMaster.



The External System

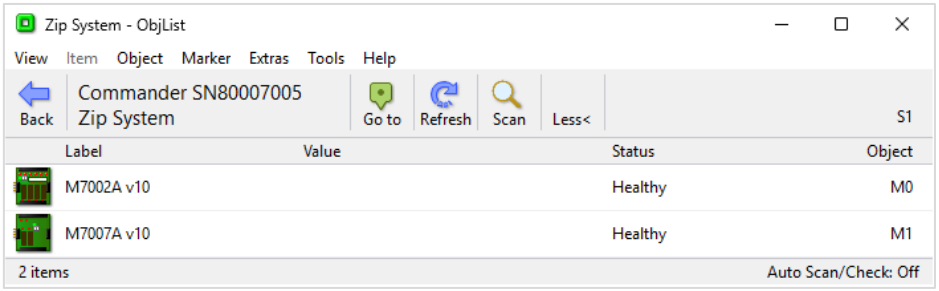
The second new object that is added to the top-level of Commander when an interface starts, represents the external system that can be accessed using the driver. It is usually shown with icon that looks like that system (see the Zip System icon to the right). This is where objects relating to the system are located – devices, values, controllers, sensors, etc. The objects within each system are different, so they cannot be documented here.



To view the Zip modules, follow these steps:

→ Open **Zip System** to view the system's object

→ ObView will auto-scan and show the Zip modules found (shown right). You can also press the **Scan** button to manually scan the view.



What are Objects?

Before we navigate down into the objects within an external system, let us examine these things we call objects. There are two main groups, or classes, of objects – value objects and container objects.

Value Objects

We have seen value objects (objects that have a value) before. Some are read-only, and just provide information, like sensor values; some are adjustable, like set points and labels.

Within North's extensible object model (XOM), each value object also has a type – which indicates the type of value it holds. For example, the type could indicate that the value can only hold whole numbers. Below is a list of the main object types and the values they can hold.

Value Type	Values	Format	Examples
Number	Whole numbers, positive or negative	ddd	1, 50
Float	Numbers with a decimal point	ddd.dddd	12.5, -2.0, 5239.2345
NoYes	A digital state – No or Yes	d Where 0=No, 1=Yes	0, 1
OffOn	A digital state – Off or On	d where 0=Off, 1=On	0, 1
Text	A text value, up to a certain defined length	ccccccc...ccc (up to max chars)	"Some Text"
Date	A calendar date	dd/mm/yy	12/01/11, 25/12/07
DateTime	A moment in time	dd/mm/yy|hh:mm:ss	25/12/11|15:00:00
Enum	An enumerated value, where a number represents something else	ddd Where 0=aaa, 1=bbb, 2=ccc	4
Obj	An object reference	cccccc (0-30 chars)	S1.C1.V
Times	A list of on-off times	hh:mm-hh:mm,hh:mm-hh:mm	07:30-12:00,12:30-17:00
Profile	A list of value change points	hh:mm=v,hh:mm=v,...	07:30=21,17:00=12
IP	An IPv4 address	ddd.ddd.ddd.ddd	192.168.0.4

As examples, the Commander's label is a Text object, and the Commander's clock is a DateTime object.

Each object type has extra parameters that define type-specific things – for example, text objects have a maximum length parameter; number objects have a high value parameter – but we will cover this later.

Container Objects

Container objects are objects that contain sub-objects. We use containers to group things together or repeat things. Again, each has a type, but there is no simple list as there are hundreds of container object types – however you do not need to know or remember them.

You have already seen and used container objects – the top-level of Commander is a container object; the Configuration object is a container object; the Platform Information object is a container object.

Container objects split into two main types: fixed and variable.

Fixed container objects always contain the same sub-objects. Fixed container objects never need scanning to discover what sub-objects they contain. It is possible to backup data from an object and restore it to another object of the same type because they always contain the same sub-objects. It is also possible to generate object-specific views that can be re-used, and object-specific processes. In summary:

- Always contain the same sub-objects, wherever they are
- Do not need scanning
- Backups can be re-used elsewhere, as the sub-objects are the same

Variable container objects contain sub-objects that are site-specific. There is no predefined list of sub-objects for each object - each site is usually different. The sub-objects may 'stabilise' over time, or they may change regularly. The top-level object of Commander is a variable container object, and when you change the interfaces within Commander, you change the top-level's sub-objects – adding an interface normally adds two sub-objects. You can scan variable container objects to discover the sub-objects they contain. In summary:

- Different sites have different sub-objects
- Need scanning when you first see them, and when their contents have changed
- Backups cannot be used elsewhere, as the number of objects changes

Within any container object, each sub-object must have a unique identifier, to allow us to refer to that sub-object – we call this its sub-object reference, and it is normally a few characters of text.

Object References

When a task in a device needs to read the value of an object, say to calculate something, we need to be able to specify which object it should read. We call this 'referencing an object', and we call the identifier of the object its 'reference'.

Object references are made up of the unique sub-object references, which are appended, (using a period, or full-stop, to separate the parts,) depending on the route from the task to the object.

For example: Consider a route through Commander, with its sub-object **Configuration** (sub-object reference O); its sub-object **Platform Information** (sub-object reference PI); its sub-object **Last Restart** (sub-object reference LR); its sub-object **Reset Count** (sub-object reference RC). The complete reference for the Reset Count within Commander is **O.PI.LR.RC** – if a task within Commander needs to find the count of resets, it would read the value of the object O.PI.LR.RC

Notice how the reference describes the route to the object – go to these sub-objects, read this sub-object.

Relative References

This concept of the route to the object being held in the reference makes the references extendable. When another device needs to read the count of resets within a Commander, it needs a reference that describes both the route to the Commander and the route to the object within that Commander. All North products that support IP networks have a sub-object called the **IP Network** (IP), which in turn has sub-objects that represent known IP addresses (say CDIP could represent IP address 192.168.192.167). This would make the object reference to the count of resets **IP.CDIP.O.PI.RC**

To see different object references, follow these steps:

→ From **Start Engineering**, select **Commander** at its default IP address

→ Press **More** to increase the ObView window to full width. ObView puts the object reference of the current object just above the word 'Object' in the column title bar, and sub-object references below it.

→ Open **Configuration, Platform Information, Last Restart** to see the objects within – the object reference to this object has become O.PI.LR

→ Right-click on **Reset Count**, and select **Copy Object Reference** from the popup menu – the object reference (O.PI.LR.RC) has been copied to the clipboard

→ Run Windows Notepad, and select **Paste** – the object reference appears in the document.

Transferring Values

All North products support the concept of a transfer – a task that can transfer the value from one object to another. These transfers form the most basic integration between systems.

Commander has 500 transfers built-in. You can use each to transfer values from any object to any other object. The source we read the value from, and the destination we write the value to, can be internal to Commander, or in one of the external systems connected to Commander – and we specify these using object references. Be careful - if you try to transfer a time-type object to a no-yes-type object, the result might not be what you were expecting!

Reading from the Source Object

Transfers happen in two parts – reading the value, then writing it if necessary.

The first part of a transfer is the reading of the source object. As well as specifying the reference of the source object, we need to specify how often we want to read it.

💻 To set up Transfer 1 to read from Zip modules on Interface 1, follow these steps:

→ **Go to** the top-level of Commander

→ Open **Data Transfer** (to view the 500 transfers,) then **Transfer 1**

→ Set **Source Object** to 'S1.M0.DI1.S' – System one, Module zero, Digital input one, State. If this successfully reads, the **Value** will change to 0 or 1, representing the open or closed contact; if the **Source Read Fails** rises, then the transfer cannot read the object, and something is wrong[1]

→ Leave the **Source Read Rate** as 'ASAP'. The transfer reads the object as fast as possible – try changing the state of the input and watch the Value change.

Although the default rate is ASAP (as soon as possible), choose the read rate carefully, as some older systems may struggle to perform if they are being constantly read from. Consider also: if you have 100 reads within transfers, and each read takes 1 second (due to slow communications say), it will take 100 seconds to read them all!

Note: If the transfer source reading fails, check the Source Object reference is correct, and the communications route to the object is working.

Writing to the Destination Object

The second part of a transfer is the writing of the value to the destination.

Every time the transfer reads the value, it compares the new value against the current value it holds. If the value has changed, the transfer writes the new value to the destination object. If the value has not changed, there is usually no need to re-write it.

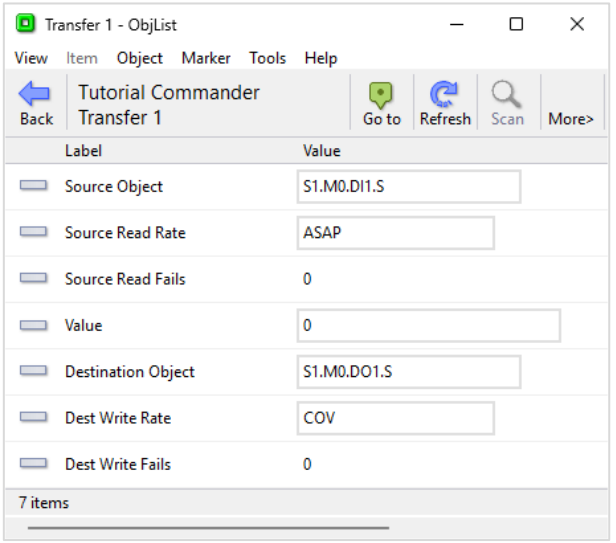
Some systems, however, can forget values if they lose power, and so they may need a periodic re-write of the value, to correct the loss of memory after a power fail.

Similarly, some integration designs require a periodic re-write of a value to reset a temporary local adjustment.

The **Destination Object** specifies where to write the value. The **Dest Write Rate** allows you to specify the periodic rewrite rate – however if you set it to 'COV' (change of value) the write will only happen when the value changes (or on power-up of Commander).

To set up Transfer 1 to write a value to the Zip modules on Interface 1, follow these steps:

→ Set **Destination Object** to 'S1.M0.DO1.S' – System 1, Module 0, Digital output 1, State – if the object reference is correct, and communications route to the destinations is working, the Value will be written to the relay - therefore the relay will follow the state of the digital input 1.



Remember that the destination write occurs whenever the value changes, the read rate will dictate how fast the destination responds to source changes.

The Zip system remembers states and values through power-cycles, and therefore you can leave the **Dest Write Rate** as COV, unless some other task is causing temporary adjustments that need resetting.

Tip: when you are editing lots of transfers, or want to see them as all in one list, you can view them in a table view. Open **Data Transfer**, then from the menu select **Extras**, **View as Table**.

What is Essential Data?

Essential Data is the name of Commander's value database and can perform useful tasks.

Although it is not necessary to collect values into a database within Commander before they are used, there are benefits:

- A value can be collected once, to save other tasks each collecting it individually
- Once collected, a value is immediately available from the database, rather than a task or user having to wait for slow communications
- Associated values, such as labels and limits, can be assigned in a consistent way

If it is used, Essential Data provides data for a wide range of other services, which may be used:

- Essential Data can hold a value and read or write it elsewhere
- Essential Data can check whether a value is within an acceptable range, and generate alarm messages if necessary
- Essential Data can build historic logs of the value
- Essential Data values are uploaded to Commander Hub, part of North's cloud services
- Commander can automatically make Essential Data available on web pages
- Some drivers make Essential Data available on other protocols – BACnet/IP, Modbus, Zip

Pages and Objects

Essential Data stores data in a two-layer structure that consists of pages and objects.

Commander's Essential Data has pages. Each page has a label, which could refer to a location, say 'Living Room', or function, for example 'Heating'. If a page has no label set up, other tasks cannot access it.

Each page has objects. Each object has a label, value type, a value, and access rights, alongside other properties. If an object has no label, other tasks cannot access it.

When Commander shows Essential Data as a web page, this page/object structure provides hierarchy. However, the page/object structure has little effect on data collection.

Simple Data Storage

Commander can use Essential Data for simply storing values. This needs the least set-up – just a label, a value type, and the actual value. As an example, an engineer could write ObVerse strategy to read a heating setpoint from an Essential Data object.

To set-up a simple Essential Data page, follow these steps:

→ Open **Configuration, Essential Data** – to see some general information, along with the list of pages to edit

→ Open **Page 1**, to see the settings for Page 1, along with a list of objects within Page 1

→ Set the page's **Label** to 'Zip Input 2'

To then set up simple value storage, follow these steps:

→ Open **Object 1**, to see the setting for object 1 on page 1

→ Set **Label** to 'Count', **Type** to 'Number', and **Current Value** to '20' – the Value Last Updated date/time object changes, to reflect when the value was set

We have now created storage for a value within Essential Data. The engineer can always change the value within the Configuration section, as we have just done – but this is only really for engineering the pages and objects – other tasks should access the values from the top-level of Commander.

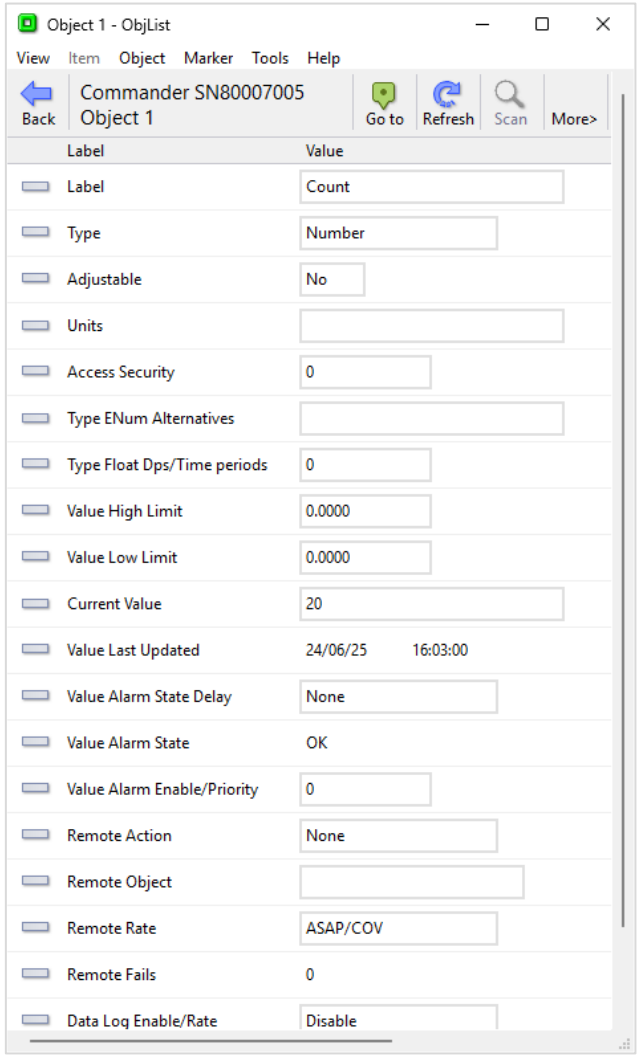
To see which pages and objects are available within Essential Data, follow these steps:

→ **Go to** the top-level of Commander

→ Open **Essential Values** – this is the 'public' side of the database, and by default, has this friendlier label that appears on the web pages

→ You may need to **Scan** the list of pages you have created

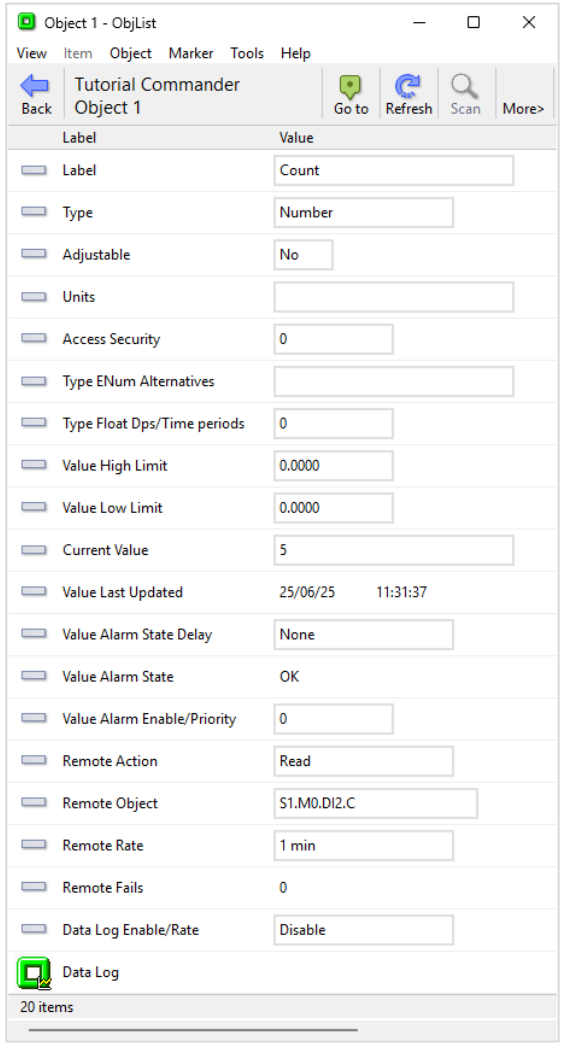
→ Open **Zip Input 2** to view the objects within the page - if necessary, **Scan** the list of objects – in this example, a single 'Count' from above



Data Collection

Using Essential Data purely as data storage is a contrived example. However, data collection is probably the most important use, and it builds on data storage.

Rather than just storing a value, an Essential Data object can be set up to collect its value periodically from another remote object – which could be within Commander, from one of its interfaces, or from another North device.



To add an Essential Data object with data collection, follow these steps:

→ Open **Configuration, Essential Data**

→ Open the page and object that will be modified for data collection – we will use the **Zip Input 2** page, and the **Count** object

→ Set **Remote Object** to 'S1.M0.DI2.C' – i.e. the count of the times the digital input changes from 0 to 1

→ Set **Remote Rate** to '1 minute' – we will have the data collected every minute

→ Set **Remote Action** to 'Read' – we will read the remote object – and the Current Value should change to the correct count of the digital inputs.

→ Connect a simple switch between the **C** and **I** terminals of DI2 of the Zip M7002, and open and close the switch a few times to cause the digital input's count to rise. Press **Refresh** to see the Current Value change.

The Remote Rate requires a little thought. Although in an ideal world we would collect the data constantly, this can cause communications bottlenecks both within Commander, and on systems connected to Commander. So, before setting the Remote Rate, consider:

- How fast the value might change - outside air temperatures change quite slowly
- How fast the data will be seen from within the Essential Data – for example, the web pages may only refresh every minute
- How many other values are collected from the same external system - if there are 100 values being collected, and the external system communications are slow and only allow one read per second, then it will take 100 seconds, or approximately 2 minutes, to collect all the values

Data Distribution

As well as collecting data, the engineer can use Essential Data to distribute data – i.e. writing values from Essential Data to other objects.

Adjusting Object Values

Normally Essential Data spends its time reading a remote object, and perhaps showing it to a user.

Sometimes, however, we need to allow the user to change the value, and have Essential Data write the value back to the remote object. This is the normal operation of Essential Data – if the object is adjustable.

When the Remote Action is set to 'Read' and the object is made adjustable, Essential Data spends most of the time reading the remote value. When the user adjusts the value within Essential Data, Essential Data attempts to write the new value to the remote object, before going back to reading it. If the write succeeds, the next read will collect the new value. However, if the write fails for whatever reason (value not allowed, or communications are too busy), Essential Data just goes back to reading it.

Limiting Adjustments

When adjusting values, it would be useful to limit the range that the user can set the value to. You do this with the Value High Limit and Value Low Limit objects.

💻 To put an adjustable, limited value into Essential Data, follow these steps:

→ Open **Configuration, Essential Data**

→ Open **Page 2**, and adjust the page's **Label** to 'Test Changes'

→ Open **Object 1**, and adjust the object's **Label** to 'Limited'

→ Set **Type** to 'Number', **Adjustable** to 'Yes', and **Value High Limit** to '10'

If the high and low limits are both set to zero, then limits are not checked when the user makes adjustments.

The web server allows users to adjust Essential Data, and it is safer to put limits on these adjustments, than leave them unchecked.

Only Writing

Sometimes we only need to write a value to an object – when the user changes it, we write it. Essential Data supports this with its Remote Action set to 'write'. In this mode, Essential Data never reads the remote object. You can decide whether to write the value only when it changes, or when it changes and periodically after the change. As we said previously during Transfers, this periodic writing can be useful.

To put Essential Data in control of a relay, follow these steps:

→ Open **Configuration, Essential Data, Test Changes, Object 2**

→ Set **Label** to 'Relay', **Type** to 'OffOn', and **Adjustable** to 'Yes'

→ Set **Remote Object** to 'S1.M0.DO3.S', **Remote Rate** to '1 minute', and **Remote Action** to 'Write'. Every minute (or whenever something adjusts this Essential Value), the value will be written to the State of DO3 on M0 – i.e. the third relay on the Zip M7002 in the Training Pack.

You can adjust the value from the top-level of Commander, by navigating to Essential Values – remember you may need to scan and refresh as the number of pages and objects has changed.

If you set the Remote Rate to ASAP/COV, the value will be written only when it is changed – be careful that the remote object cannot be modified from elsewhere, or things will get very confusing!

Summary of the Essential Data set-up so far in the tutorial:

Page	Object	Use	Adjustable
1	1	Zip Input 2 - Count	No
2	1	Test Changes - Limited	Yes
2	2	Test Changes - Relay	Yes

Tip: when you are editing lots of Essential Data objects, or want to see all the objects on a page in one list, you can view them in a table view. Open a page, then from the menu select **Extras**, **View Objects as Table**.

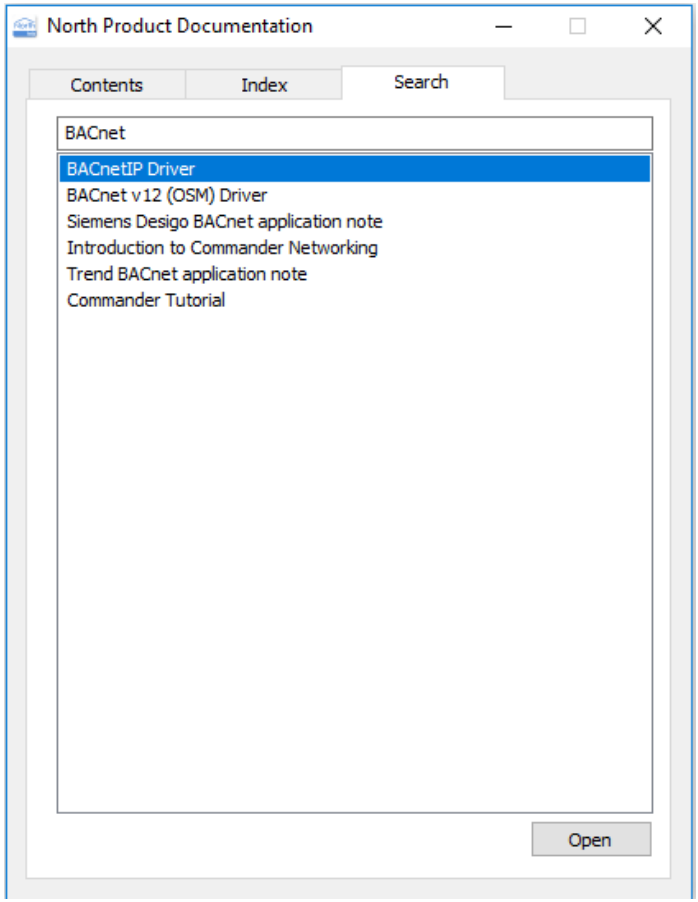
BACnet/IP and Modbus Server

Some North drivers expose the Essential Data to other systems – making it easy to set Commander as a server.

Two examples of these types of driver are BACnet/IP and Modbus. You must start the interface as any other, but once set up it will work automatically, making the Essential Data available to other systems.

North Product Documentation

Each North driver is documented with details of the equipment it can connect to, how the driver is used, and what objects are available. You will find driver manuals in the North Product Documentation app.



To view the documentation for the BACnet/IP driver, follow these steps:

→ From Windows **Start**, select **All Programs > North ObSys > Product Documents**

→ Click on the **Search** tab; in the **search documentation** box, type 'BACnet'

→ The app shows a list of documents that it has found related to 'BACnet'

→ Double-click on **BACnetIP Driver Manual**, and the PDF document opens.

Connecting Commander to a LAN

Up to this point, all engineering within this tutorial has been done with your PC directly connected to Commander, effectively creating a ‘mini-LAN’ – with Commander unlocked in Default IP mode.

However, Commanders can sit together on a LAN and collect information from other devices, provide information to other devices, as well as serving web pages to users on that network.

Although Commander is secure by default, we must consider the security risks of putting it onto a LAN, as we could be connecting it to other unknown networks, including (indirectly) the internet.

Commander supports many different protocols on Ethernet. Each of these is both a strength and a weakness – a strength when building a powerful integrated system; a weakness because each protocol could provide a point of attack by malicious actors. Therefore we must consider the benefits and reduce any risks of starting an interface and making a protocol available.

The most powerful protocol supported by Commander is the built-in North IP Device protocol. This protocol is used when North devices talk to each other: when Engineering Software talks to a Commander; when one Commander talks to another Commander; and when an ObSys display talks to a Commander.

So, before we can connect Commander to a LAN, we will need to: set an authentication key for North IP Device access, determine the IP addressing method used on the LAN, and then set Commander’s LAN port address.

Securing North IP Device Access

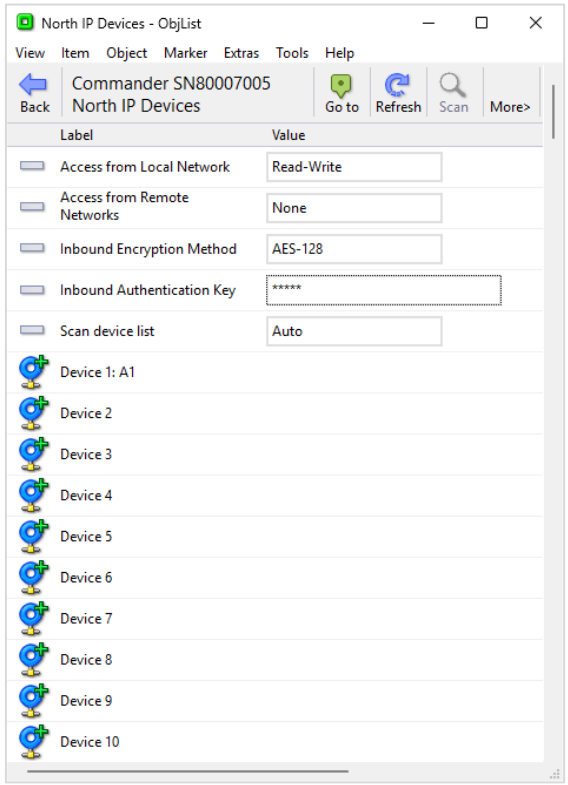
Each North device has control over what can talk to it. Access to the device, in this case Commander, is secured with these settings:

- Authentication key – encrypts messages and only allows access from devices that know the key
- Access from local network – controls whether devices on the same LAN can read and write values
- Access from remote networks – controls whether devices on a different LAN can read and write values.

Authentication Key

Commander first needs to be set with an authentication key (similar to a password) that other devices must know before they can send read and write requests to Commander. The key is never actually shown, and by default is set with a random key, but is used to encrypt messages. The receiving Commander decrypts the message to confirm the sender knows it's key.

Remember to document any authentication keys you set. For ease, you may decide to use the same authentication key for all devices on a site. You should not use the same key for more than one site or project.



To set the Commander's authentication key, follow these steps:

→ **Go to** the top-level of Commander; open **Configuration, North IP Devices** to view the current settings.

→ First, check **Inbound Encryption Method** is set to 'AES-128'. This means any incoming messages must be encrypted using AES-128 before the Commander will process them.

→ Next, set the **Inbound Authentication Key** to something ideally more than 12 characters. REMEMBER THIS VALUE! The Commander always shows ***** for the value. Once you set the Authentication Key the device will immediately use both Encryption Method and Authentication Key values – which may stop you talking.

Note: These settings are not used when Commander is unlocked in Default IP mode.

If you have older North devices on the network that cannot be upgraded, set the **Encryption Method** to 'None'.

Network Access

Commander can also limit access from North devices on different networks. Access from local and remote networks can be restricted: read-and-write, read-only, or none.

By default, Commander is set to allow read-and-write access from the local network and no access from any remote networks (such as via a gateway).

💻 To limit Commander's access by network, follow these steps:

→ **Go to** the top-level of Commander; open **Configuration, North IP Devices** to view the current settings.

→ Check **Access from Local Network** is set to 'Read-Write'. This will allow North devices on the local network, including engineering software, to both read and write values within the Commander.

→ If access from any remote networks is required, set the **Access from Remote Networks**. For example, setting this to 'Read-only' will allow North devices on any remote networks to read values but not change them.

IP Addressing Methods

Commander supports three methods of IP addressing on the LAN Port: Default, Static, or Dynamic. Check with your network administrator about how the Commander addressing should be done.

Default IP Addressing forces the IP address to 192.168.192.167 with subnet mask of 255.255.255.0, and no gateway address – this is useful for quick engineering, or if the Commander's IP address has been forgotten.

Don't connect Commander to a working network when it is using Default IP addressing.

When in Default IP mode, Commander's DHCP server is enabled to simplify Engineering Software setup.

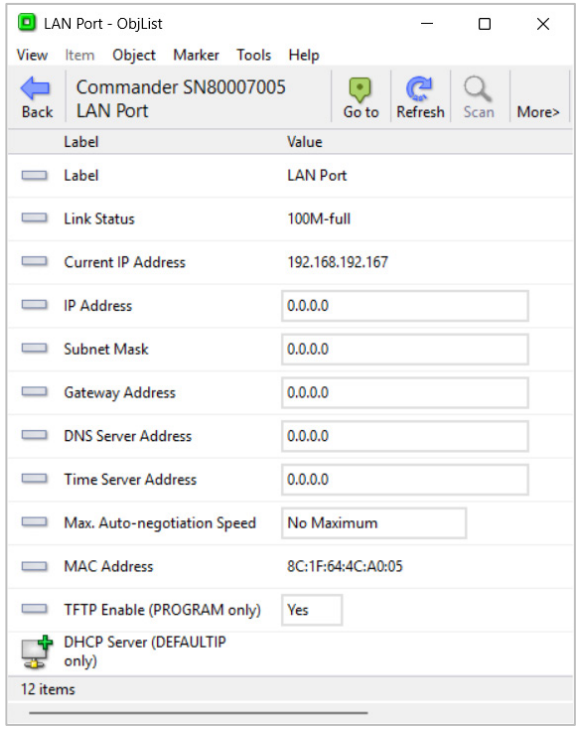
Static IP Addressing (or fixed IP address) forces Commander to use the specified IP address and subnet mask – you must also tell it DNS and NTP server addresses as required. This can be useful as users and other devices will know exactly which IP address the Commander is available at. You will need to agree with the IT department the address to use, so they don't allocate it for other devices

Dynamic IP Addressing instructs Commander to request its IP information, including IP address and subnet mask from the network's DHCP server – the server normally also supplies DNS and NTP server addresses. Dynamic addressing is simpler that static addressing, but the IP address given by the DHCP server to Commander may change after a period. Ask your network administrator to create a reservation.

When using dynamic addressing, Commander supplies the Commander Label to the DHCP server.

Setting LAN Port Address

Whilst Commander is in Default IP mode, set the IP addressing to match your network – either to use dynamic addressing or to use a static IP address. You'll need to restart Commander to use any new LAN port settings.



- To set the Commander LAN Port to use dynamic addressing, follow these steps:
  - → **Go to** the top-level of Commander, open **Configuration, LAN Port** to view the current settings. The **Current IP Address** is the address that Commander is using right now: '192.168.192.167 is the default IP address
  - → The **IP Address** object holds the address that you wish the Commander to use – set it to '0.0.0.0' to tell Commander to use Dynamic IP Addressing and use DHCP to get its IP settings
  - → Change the Commander's **DEFAULTIP** switch to OFF – it will now restart using the IP and security settings you specified
  - → Connect the Commander to your network switch or router using Cat 5 cable or better, and Commander will attempt to contact the DHCP server.

- Or, to set the Commander LAN Port to a static IP address, follow these steps:
  - → **Go to** the top-level of Commander, open **Configuration, LAN Port** to view the settings
  - → Set **IP Address** to a known IP address on your LAN, and set the **Subnet Mask** for your LAN – in this document, we will use 192.168.2.150 and 255.255.255.0 – you may need to ask your network administrator
  - → If known, set the **Gateway Address** to enable access to other networks
  - → Optionally, set **DNS Server Address** to resolve domain names to IP addresses, and **Time Server Address** to access accurate world time. They can be on the local network, or if you have a gateway address set, they can be on an external network. If not set, Commander will attempt to use Internet based servers.
  - → Change the Commander's **DEFAULTIP** switch to OFF – it will now restart using the IP address and security settings you specified
  - → Connect the Commander to your network switch or router using Cat 5 cable or better.

The ObView window that you used to set the Commander's new IP address will no longer get values from Commander because it is trying to talk to the Commander at the old IP address – so **Close** the ObView window, ready to Start Engineering again later.

Engineering Commanders across a Network

Once you have connected Commander to your network, you may also need to prepare your PC for working on the same network, by setting an IP address, etc.

Tip: If you are not sure where to find your PC's network settings, from **Start Engineering**, select **Extras**, **Check Network Connections**.

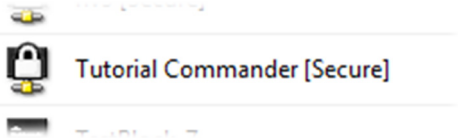
Scanning for North devices

Once you have set Commander's IP and security settings, and connected it to the same network as your PC, you can engineer Commander across that network.

To start engineering Commander across your network, follow these steps:

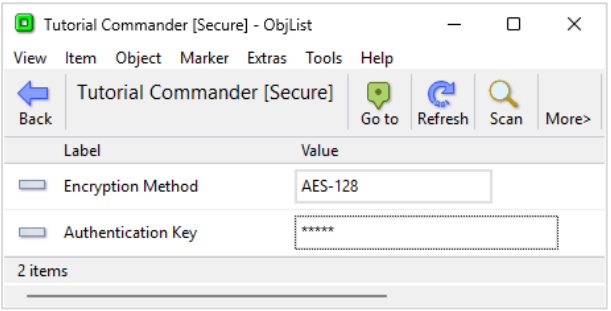
→ From Windows **Start**, select **All Programs > North ObSys > Start Engineering**

→ From the **Start Engineering** page, select **North IP Devices.** Any previously found devices are shown



→ Press **Scan** to look for all North devices on the network. The list will show both devices you can talk to, and any that are secure

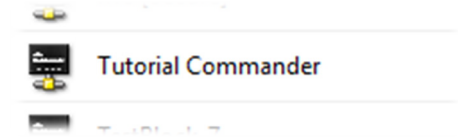
→ Navigate to your Commander, it will have the label you set at the beginning of this tutorial, followed by '[Secure]'. Set the **Encryption Method** and **Authentication Key** to the values set earlier in that device (these values will then be saved securely in your Engineering Software for future use)



→ Navigate back to **North IP Devices** and press **Scan** again to find the devices you can talk to

→ Navigate to your Commander. ObView will display a list of the last objects it found in a device at that address. You may need to press **Scan** to see the contents of the Commander.

→ You may set the Marker to this object, by selecting from the menu **Marker, Set to this object**. This sets the Marked Object to the new address of the Commander.



Tip: If you cannot communicate with the Commander, go back and set Commander to its Default IP mode, then connect locally to find what went wrong!

Tip: If Commander disappears after setting the authentication key, you may have entered the wrong key! Select **Extras**, **North IP Configuration** then navigate to the device and re-enter the key.

Communicating between Commanders

Once a Commander is on a network, it can communicate across the network with other Commanders – transferring values or sending alarms, for example.

To tell Commander to find other North devices on the IP network, follow these steps:

→ From the **Start Engineering** page, select **North IP Devices** then your Commander

→ Select **North IP Devices** and press **Scan** to look for all North devices on the network your Commander can see. You should see ObSys Engineering software running on your PC

→ You may need to set the **Encryption Method** and **Authentication Key** for any devices that require them.

You can now navigate down these devices – the first time you access them you may need to scan and refresh to see the objects they contain. The power of XOM means you may use any of the objects you find in transfers, or other tasks within Commander.

Saving your changes

Before we leave this section of the tutorial, we should cover saving the changes you have made to a Commander.

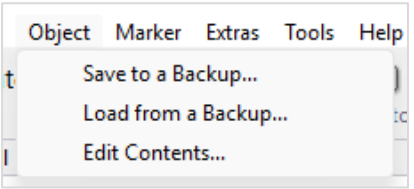
Whenever you make changes to the setup of a Commander (or any other equipment for that matter), you should save your changes. This is good engineering practice.

Saving to a Backup

The Engineering Software can back up any object and its sub-objects. It only backs up objects that can be written – any read-only objects are not saved.

Backup files, with the file extension '.obs', are text files, and as such can be viewed and edited.

- To save the settings within a Commander to a backup file, follow these steps:
  - → Navigate to the top-level of the Commander
  - → From the menu, select **Object, Save to a Backup…**
  - → Select the folder into which the backup is to be stored – by default, the selection is the folder for the specific Commander within North's TypeInfo folder:
    C:\ProgramData\North Building Technologies\ObSys\TypeInfo\DefaultSite\IP\<*IP Alias*>
  - → Press **Save** to start the backup process. Commander and its sub-objects will be saved to the backup file.



- To save any object to a backup file, follow these steps:
  - → Navigate to the object you wish to back-up, for example, **Configuration**, **Essential Data, Page 1**
  - → From the menu, select **Object**, **Save to a backup…**
  - → Select the folder, (you may change the default file name if you wish), and press **Save**.

Note: when saving a backup, the backup application, ObSave, will only backup items it knows about – for example, if you have not scanned a Zip network, ObSave will not save it.

See the later section *Default Configuration* about saving your work to non-volatile flash memory within Commander.

Loading from a Backup

Once saved, backup files can be loaded into the original Commander, or anywhere else that has the same object structure (like another Commander). Object backups can also be loaded into other objects of the same structure.

For example, a backup from one Essential Data Page can be loaded into a different Essential Data Page within Commander or another.

To Restore an object, follow these steps:

→ Navigate to the object you wish to restore with the same object type as the original backup. For example, **Configuration**, **Essential Data, Page 8**

→ From the menu, select **Object, Load from a Backup…**

→ From the Open window, find the '.obs' file to load. Let's use the 'Blank.obs' file, this will delete the object values

→ Press **Open** – the loading starts.

Remember, when loading a backup, the loaded data may change Commander's operation – for example, if the IP Address is set, Commander will operate at that new IP address.

Export to a CSV

When documenting a configuration, it is sometimes useful to save all objects including the read-only ones. The Engineering Software can export an object and its sub-objects to a CSV file.

CSV files can be opened by a text editor, or spreadsheet application such as Excel.

To export the software summary of a Commander, follow these steps:

→ Navigate to the top-level of the Commander, then **Configuration**

→ From the menu, select **Extras, Export Software Summary to CSV…**

→ Select the folder location and document name for the exported CSV file, click **Next**

→ The export process starts. Once created, click on **Open document** to view the file.

Summary

In this section you have learnt:

- ✓ Interfacing Commander to other systems – understand interface licences, how to start and stop an interface, connecting an external system then accessing its setup and system objects
- ✓ Objects – understand value objects, container objects and object references
- ✓ Transferring values – using transfers to read an object then write the value to another object
- ✓ Essential Data – using Commander's value database, understand its page and object structure, configuring an object to collect or write data, allowing adjustment
- ✓ Connecting Commander to a LAN – secure North IP device access, understand IP addressing methods, set the LAN port address, accessing Commander across a network
- ✓ Saving you changes – how to save and load a backup of Commander's configuration.

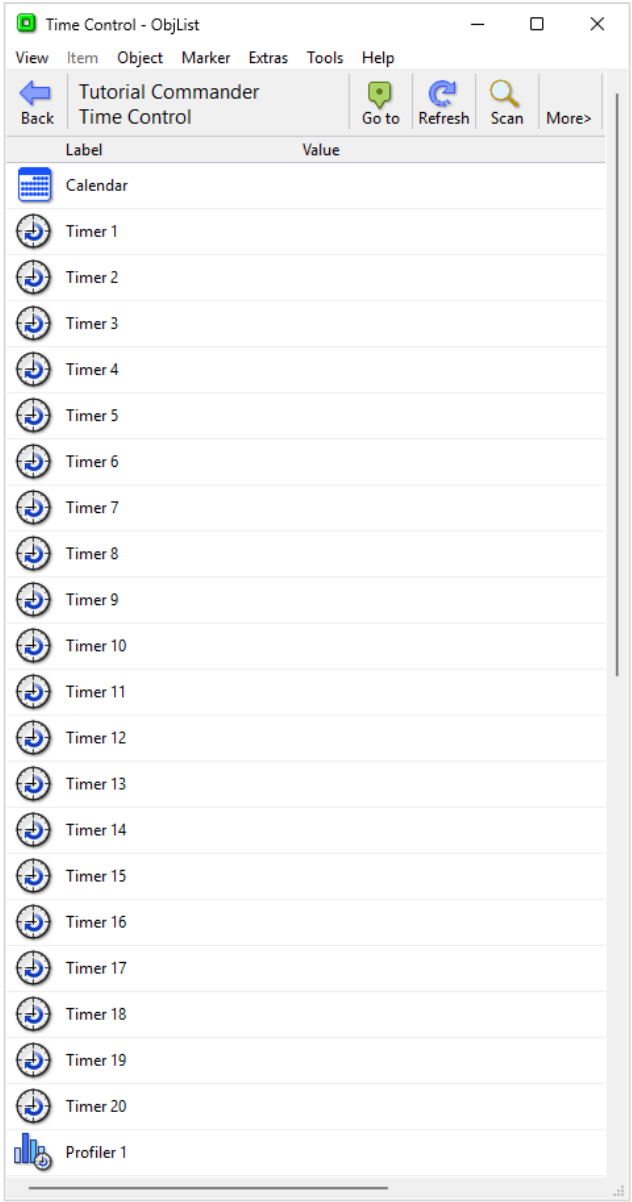
Controlling…

Time Control

Timers and profilers allow users to control when things happen in the day, and when they do not. You can save energy and keep the occupants happy. Commander supports timed control with its Calendar, along with its Timer and Profiler tasks.

Commander's single calendar determines today's day-type, and its twenty timers and profilers use today's day-type to determine which of their values to set based on time periods.

Commander supports ten different day-types: Commander uses one of them as day-type 'off', leaving you nine to define and use yourself. They are numbered 0 (off) and 1 – 9.



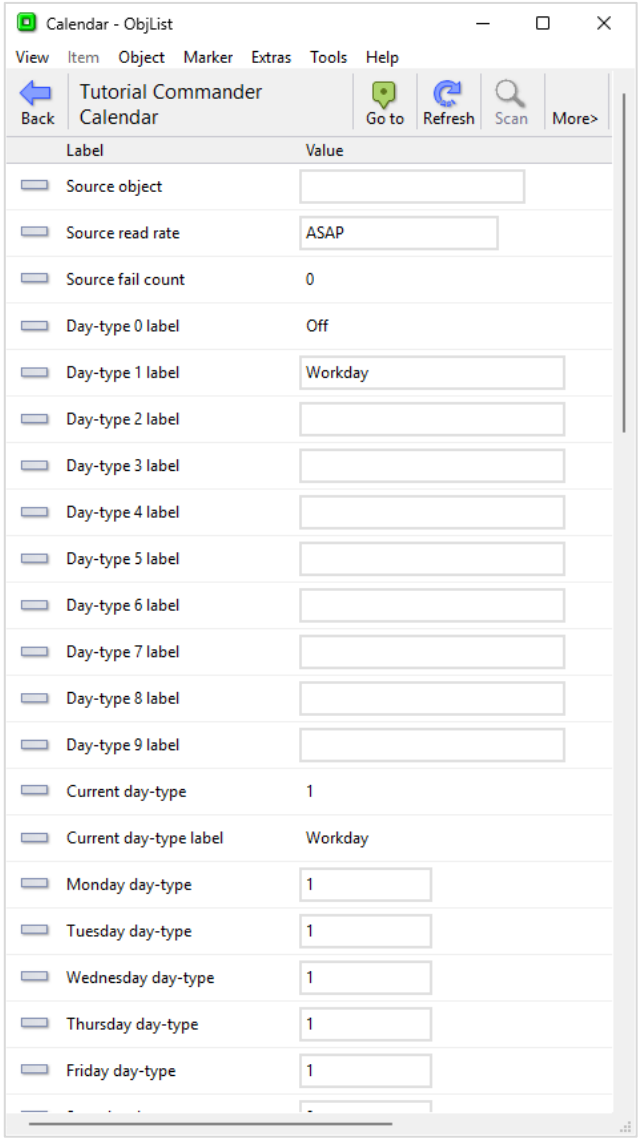
The Calendar and Today's Day-Type

The Commander's calendar determines today's day-type. It does this either by requesting it from some other calendar in a different North device, or by calculating the day-type itself.

If you have a 'master' calendar elsewhere on the system, Commander's Calendar can request the day-type from that master – you need to specify the object reference of the master calendar's day-type.

If you are calculating the day-type in Commander, it works in the following way: The calendar determines whether today's date is an exception date – if so that exception day-type is used – otherwise the day-type based on the day-of-week is used.

The calendar re-calculates the day-type every minute, based on the day-types and the exception dates.



To set up a standard week within the calendar, follow these steps:

→ **Go to** the top-level of Commander, and open **Time Control** – you may need to press **Scan** to see the calendar and timers

→ Open **Calendar**. You can now see the objects that make up the calendar, including the day-type labels, the day-of-week day-types, and the exception dates and day-types

→ Adjust **Day-type 1 Label** to 'Workday'

→ For each of the weekdays Monday to Friday, adjust the Day-type to '1' (Workday). This means that weekdays are all workdays, unless the date is an exception date

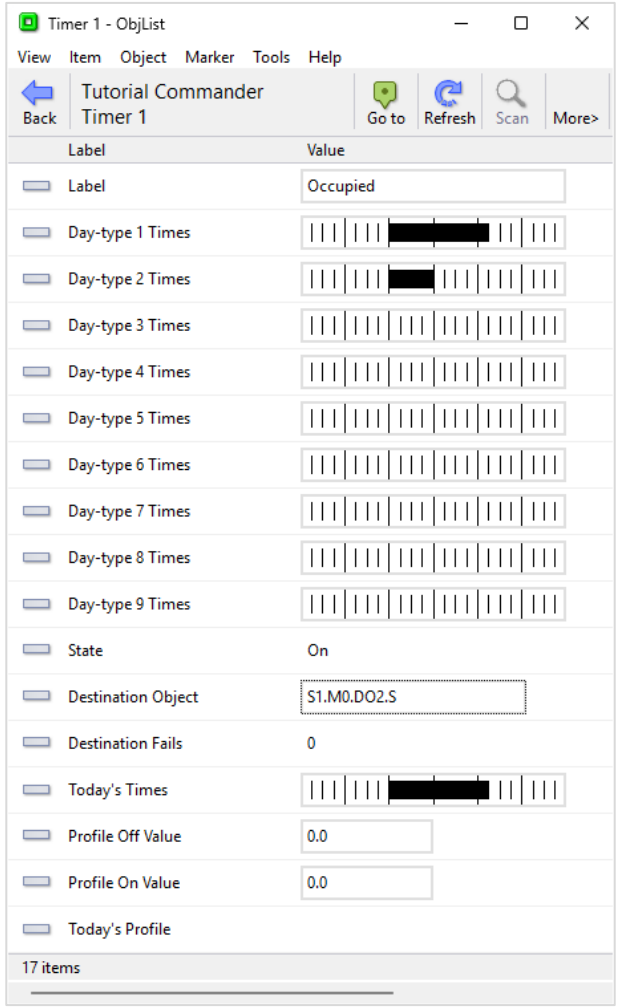
→ You can see the Current Day-Type near the top of the object list, along with a Current Day-Type Label

This object list view of the calendar is good for engineering – imagine the power of ObVerse strategy modifying the day-types for the week ahead perhaps.

Timers

Commander uses timers to control off/on processes. Each timer produces an off or on state, which can be accessed by other tasks.

Each timer has on-off times for each of the possible day-types and uses those on-off times on days that have that day-type. The timer re-calculates the state of the timer every minute.



To set a timer, follow these steps:

→ Open **Time Control, Timer 1** to see the times, label, etc.

→ Adjust **Label** to 'Occupied'

→ Adjust **Day-type 1 Times**, and change the **Period 1 - Start** value to '8:00' – either type in the box, or click-and-drag the slider – then the **End** value to '17:00'

→ Repeat the last step with **Day-type 2 Times**, putting in '8:00' to '12:00' – we will use this as a half day

Notice the State object near the bottom of the object list, and the Today's Times – which is useful for transferring to other things that have an 'on-off times' object.

To send the state to another object, follow these steps:

→ Adjust the timer's **Destination Object** to the object reference 'S1.M0.DO2.S' (The second relay output of the Training Pack)

→ After the next on/off time change, the relay turns on or off depending on the timer state – if the object reference is wrong, the Destination Fails count will rise

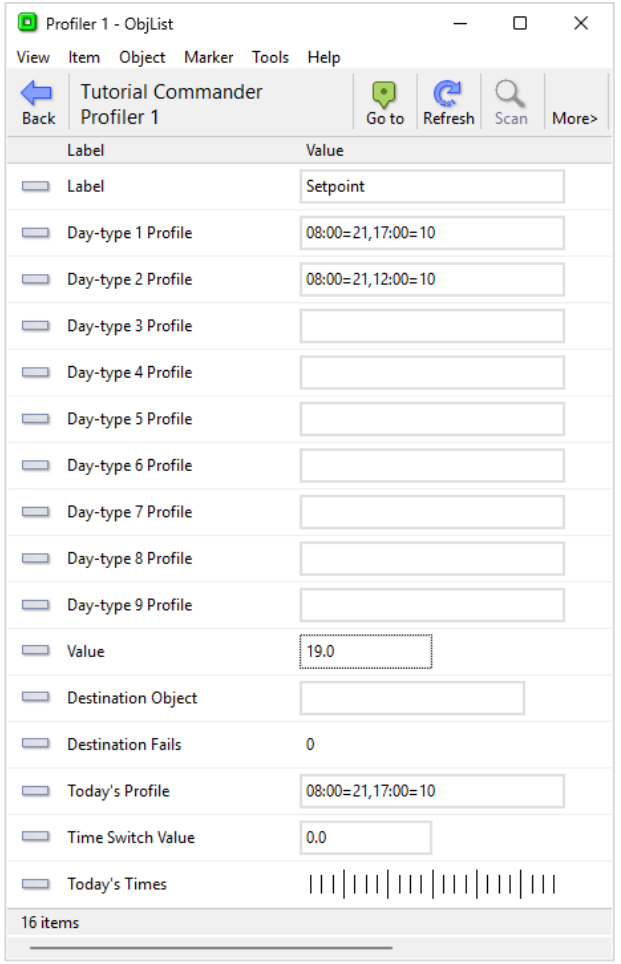
This object list view of timers is good for engineering. However, the user can view and adjust the Timers and on-off times using the web pages.

Profilers

Some control solutions require different values to be written to an object at different times of the day. For example, room control during certain periods of the day requires different setpoints. Commander uses profilers to do this.

Each profiler produces a value, which can be accessed by other tasks.

Each profiler has a list of change-points for each of the possible day-types. It uses those change-points on days that have that day-type. Every minute, the profiler checks whether the current time matches a change-point time, and if so, sets the value to the change-point value.



To set a profiler, follow these steps:

→ Open **Time Control, Profiler 1** to see the profiles, labels, etc.

→ Adjust **Label** to 'Setpoint'

→ Adjust **Day-type 1 Profile**, and change the value to '8:00=21,17:00=10' – this means at 8:00 we want the value set to '21', and at 17:00 we want the value set to '10'

→ Repeat the last step with **Day-type 2 Profile**, putting in '8:00=21,12:00=10'– we will use this as a half day

The Value object will change only when the current time matches the time in a change-point. If the Destination Object is specified, then when the value changes it will be written to the object.

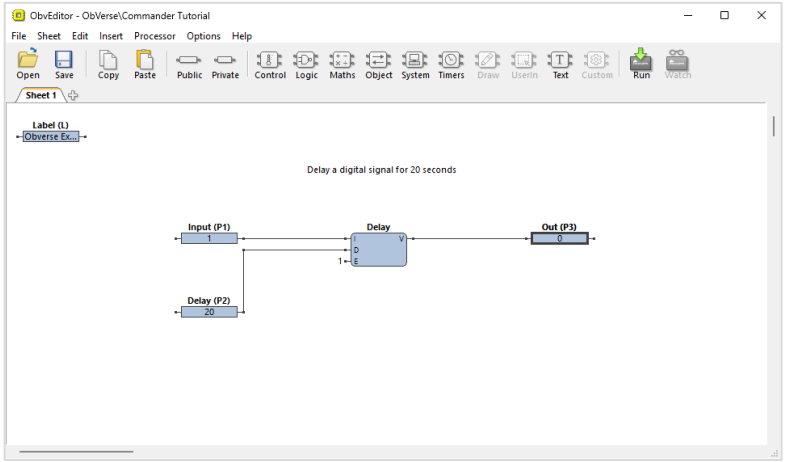
To make a temporary value change, follow these steps:

→ Change the **Value** object to '19'. The Value will remain at 19 until the next change-point.

Notice the Today's Profile object near the bottom of the object list which is useful for transferring to other things that have a 'time-value profile' object.

This object list view of profilers is good for engineering. However, the user can view and adjust a profile from Essential Data using the web pages.

Introduction to ObVerse Programming



Sometimes you need to do more than simply transfer a value from one object to another – you need to calculate something, delay something, or perform more complex functions on a value. North provides this flexibility with ObVerse, a cause-and-effect programming language.

ObVerse consists of a range of modules. The engineer selects modules and links them together to perform a desired strategy.

ObVerse strategy runs in an ObVerse processor within a device. Commander has two ObVerse standard processors, and whenever Commander is powered-on, these processors run their ObVerse strategy continuously.

You can create and edit ObVerse strategy using North's ObvEditor application, installed as part of the ObSys software. ObvEditor provides drag-and-drop graphical editing of ObVerse, uploading and downloading of ObVerse strategy, and run-time monitoring of the strategy within the processor.

ObVerse Basics

ObVerse strategy is made up from properties, modules, and comments.

ObVerse properties are value objects as we have met before. They are containers for storing data values and carry a value from one module to another or between the processor and other tasks in the system.

ObVerse modules are used to perform an operation on one or more input values and calculate a value. Properties are linked to the inputs and outputs to store these input and output values.

Comments in ObVerse are short pieces of text used to explain ObVerse strategy and make it easier for others to understand.

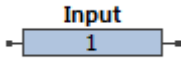
The image above shows ObVerse strategy that takes a digital value and adds a delay to it: it has three properties (the narrow blue items with sharp corners), a module (the blue item with rounded corners), and a comment in heavier text.

To start the ObVerse Editor, follow these steps:

→ Open **Configuration, ObVerse Processor 1**.

→ Open **ObVerse** to run the ObVerse Editor - this will also associate the editor and the processor. When the editor starts, it shows pre-defined ObVerse strategy – which contains an input property called Label.

ObVerse Properties



An ObVerse property is a value object and holds a value. When you add a property, you are adding value storage. You choose the object's reference within the ObVerse strategy, the type of value it holds, and whether the property is public (other processes and tasks can access it) or private (its value is only accessible by other modules within the process). If the property is public you can also specify a label, and an initial value – the value of the property after initially downloaded, but before other tasks write to it. If the property is private the label and initial value are not configurable.

The public property shown to the right has been labelled 'Input' and has an initial value 1.

The property is drawn as a rectangle, with the initial value shown inside. If the property is public, its label is shown above the property. A 'tooltip' will show the property's reference and type of value it holds. The short lines on either side of the property show what connects to (and therefore uses), the property's value - a left-hand line may be connected to the single module that calculates and sets the value in the property; a right-hand line may be connected to one or modules that use the value. In most cases the property will first be shown red meaning it is not connected to anything (and therefore has an error.)

All ObVerse strategy should have a Label property (L) – a text input property – it is automatically used as a title for the main ObVerse processor object that appears within the top-level of Commander.

To set the label of our example ObVerse strategy, follow these steps:

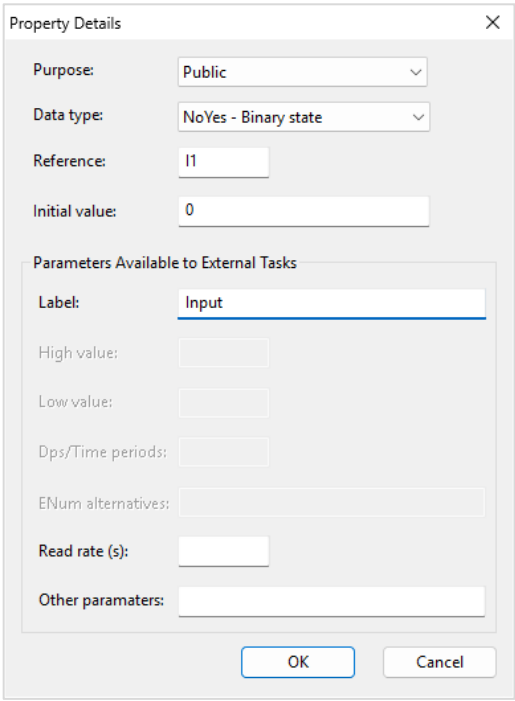
→ Hover the cursor over the **Label property** to display the tooltip for the property - this shows it's reference and the type of value associated

→ Double-click on the Label property, and set its **Initial value** to 'ObVerse Example'

To add a public property to our example ObVerse strategy, follow these steps:

→ In the editor window, select **Public** on the toolbar. Select **NoYes** as the type of value needed, and the cursor will change to show a property…

→ Click in the editor window (wherever you would like the property to be placed), and the Property Details window appears. Set the **Reference** to 'I1', **Initial value** to '0' and the **Label** to 'Input', then click **OK**. The initial value of the property is now shown in the property itself. If you have made a mistake double-click the property again to edit its details.



Property Purpose and Type

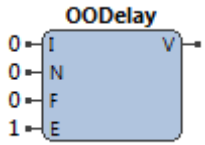
When you create a property, you must select the purpose of each property you create:

Purpose	Use
Public	Public property values allow other tasks to read and write to their values – within ObVerse the properties have connectors on both the left and right hand side
Private	Private property values cannot be seen by other tasks - they can only be seen within this ObVerse. Commonly used to store a value between modules, they always create their own reference and label

Each property you create holds a particular type of value – an Off/On value perhaps, or a Text label:

Type	Holds
NoYes	Binary state: 0 or 1, where 0 means No, 1 means Yes
OffOn	Binary state: 0 or 1, where 0 means Off, 1 means On
Num	Whole numbers (no decimals)
Float	Floating point numbers with decimal points
Obj	Object References
ENum	Enumerated values 0 to n, where you determine the meaning of each
Text	Any text string up to 30 characters
DateTime	Date and time
Times	List of on-off times
Profile	List of time-value pairs

ObVerse Modules

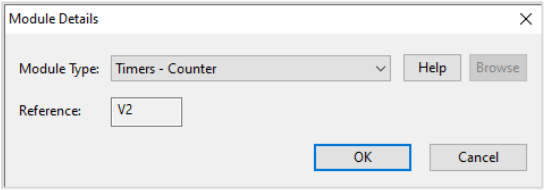


An ObVerse module takes inputs, performs an action, and produces outputs – and you can connect these inputs and outputs to the properties you have created. Modules come in a variety of functions, including control, logic, maths, timers, object access – the list goes on!

Modules have their inputs connected on the left-hand side, and the outputs on the right. It therefore makes sense that you place your ObVerse strategy inputs on the left-hand side of the page, and its outputs on the right-hand side – calculations will therefore occur naturally from left to right.

The module shown to the right is an On-Off-Delay module – which delays a digital signal when it turns on and off. It takes four inputs (I, N, F, and E) and calculates one output (V). It has its own label above, OODelay.

- To add a module to the ObVerse strategy, follow these steps:
  - → Select the function of the module you would like to add from the toolbar, in this example we will add a Counter module so click the **Timers** button.
  - → There are several modules which come under the Timers category - select **Counter.** The cursor will now change to the shape of a module…
  - → Click in the editor window to the right of the property we have just added, and the Counter module will now be added to the window
- To see what a module does, follow these steps:
  - → Double-click a module on the ObVerse page, in this case we will double-click the **Counter** module itself and select **Help –** and the *ObVerse Manual* will open showing the help for the Counter module – you can see from the help that the Counter module takes three inputs, and calculates two outputs – a count of 0-to-1 changes on the I input, and a count of seconds that the I input is set to On
  - → **Close** the documentation when finished, and **Cancel** the Module Details window



- To add two properties ready for the module's outputs, follow these steps:
  - → Click the **Public** button on the toolbar then select **Num** from the sub-menu
  - → The cursor will now change to a property… click in the editor window somewhere to the right of the Counter module we have just added

→ The Purpose, Data Type and Reference have already been filled in for us by the editor; change the Reference to 'O1' and **Label** to 'Starts' and click **OK**

→ Repeat the process to add another **Public Num** property, setting the **Reference** to 'O2' and **Label** to 'Seconds'.

Module Types

Here is a list of some of the modules supported by Commander's ObVerse standard processors. For a complete list, and to find out how a module is used, refer to the *ObVerse Manual: Standard Processor* document.

Maths

Add
Subtract
Multiply
Divide
Modulus-Remainder
Multiple-Add
Average
Minimum
Maximum
Byte-To-Bits
Bits-To-Byte
Num-To-Bit/Bit-To-Num
Random
Rescale
Smooth
Square-Root
Linearize
Usage-Over-Period

Logic

Logical-And
Logical-Or
Logical-Inverse
Equal
Gate
Greater
Less
Logical-Exclusive-Or
Multiple-Logical-And
Multiple-Logical-Or
Select

Control

Feedback
Hysteresis
Lead-Lag
Optimum-Start-Stop
Proportional-Integral-Derivative
Raise-Lower

Text

Text-Equal
Text-In-Text
Text-Join
Text-Length
Text-Split

Timers

Counter
Date-Pulse
Last-Change
Latch
On-Off-Delay
Profile-Value
Pulser
Times-State

System

System-Information

Object

Alarm
Object-Read
Object-Write

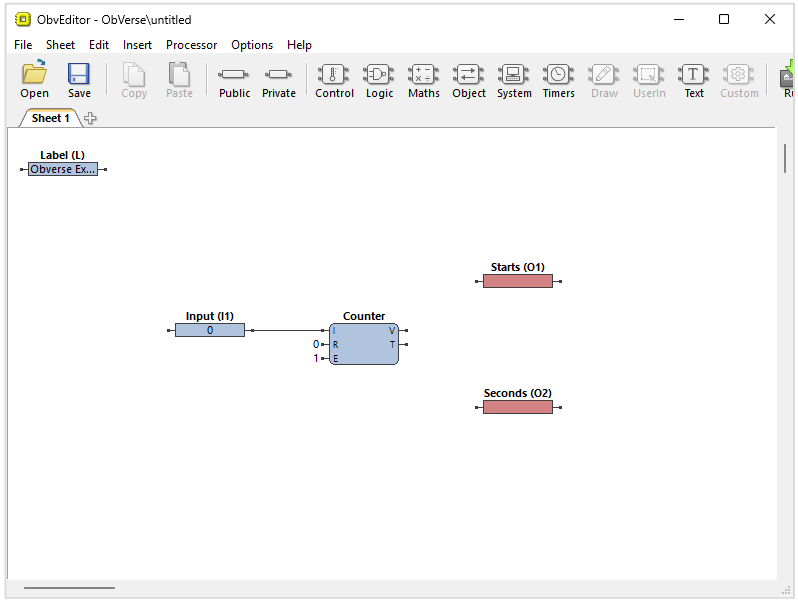
Module Inputs

Each module input – the lines on the left side of the module - may be used in one of several ways: as a constant value; connected to a property; or connected one of the device's Essential Values.

If you want the input to remain constant throughout the process, you can set the input to a constant value – the module will always see the input as the value you specify. When you first insert a module, all its inputs are set to pre-determined constants.

- To view the label of a module input, follow these steps:
  → Move the cursor over the module input, on our example the R input of the Counter module, and a tooltip will show the label of the R input, Reset, and the current constant value, 0

- To change the constant value of a module input, follow these steps:
  → Move the cursor over the module input, on our example the R input of the Counter module, to see the full label of the R input, Reset, and the current constant value, 0
  → Double-click on the **module input**, to see the value adjustment window
  → Change the value to that required, in our case '0', and press **OK**



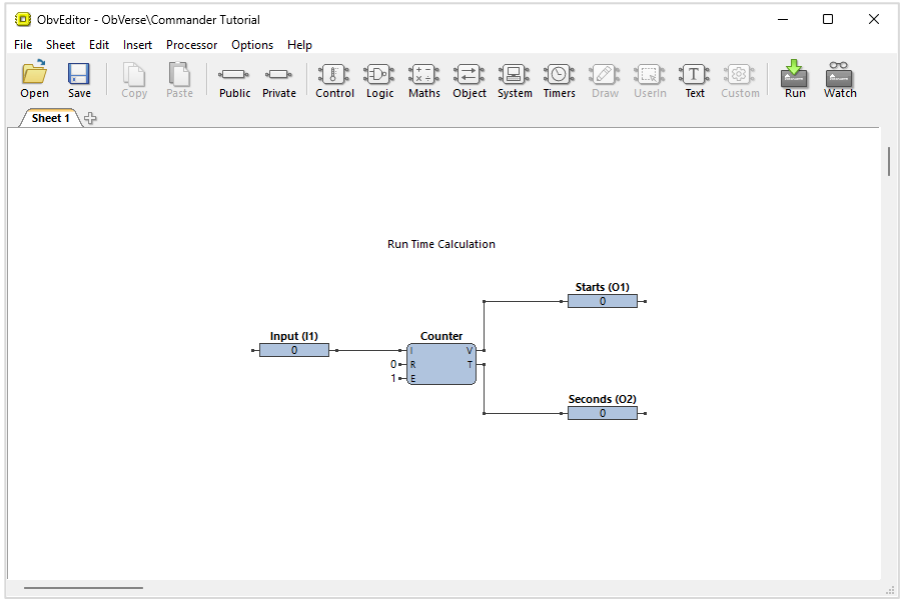
- To connect a module input to a property, follow these steps:
  → Move the cursor over the **module input**, on our example the I input of the Counter module, to see the full label of the I input, Input, and the current constant, 0
  → Click-and-drag the cursor to the property, in our case the Input (I1) property we created earlier – the connecting line will appear, and then is drawn thicker when the link becomes valid. The connecting line will remain, showing the link has been made

Module inputs can only connect to property outputs. They cannot be connected to property inputs or other modules directly. If you try to connect two modules, the editor will automatically place a private property between the modules you wish to connect. This property can be edited further if you wish.

You can also connect module inputs to properties by right-clicking on the module input, and from the popup-menu, select **Connect to Public** or **Connect to Private** – the list of input or output properties appears, and you select the one you want.

Each module input can only connect to one property – otherwise it is not clear which value it would use at which time. However, several different module inputs can connect to a single property and use the value.

If you connect a module input to a property of the wrong type, then the module will attempt to convert between the two types – for example, if a number input is connected to a text property, the ObVerse processor will try to extract a numeric value from the start of the text.



Module Outputs

A module's purpose is to calculate its outputs, which are available via the connectors on the right-hand side of the module. The outputs from the module usually connect to properties – where the module puts the actual values. If you do not need a particular output value, you do not connect it to a property.

- To view the label of a module output, follow these steps:
  - → Move the cursor over the module output, in our example the V of the Counter module, to see the full label of the V output, Count, appear in the tooltip
- To connect a module output to a property, follow these steps:
  - → Move the cursor over the module output, on our example the V output of the Counter module
  - → Click-and-drag the cursor to the property, in our case the Starts (O1) property – the connecting line will appear and go thicker when the cursor drags over a valid link – and release it. The connecting line will remain, showing the link is in place. Do the same for Counter's output T to Seconds (O2)

Module outputs can only connect to properties, and not to other module inputs. If you try to connect a module output directly to a module input the editor will place a private property between them in the same way as described earlier.

You can also connect module outputs to properties by right-clicking on the module output, selecting **Connect to Public** or **Connect to Private** from the popup-menu, and when the list of output properties appears, selecting the property.

Each module output can only connect to one property.

If you connect a module output to a property of the wrong type, then the module will attempt to convert between the two types – for example, if a number output is connected to a NoYes property, the ObVerse processor will attempt to convert – by checking if the number is zero (i.e. No) or non-zero (i.e. Yes).

Moving an Item

Once your ObVerse strategy is laid out, the next task is to tidy it up - and leave it in a state that you would wish to find it! By click-and-dragging on the shaded area of a property or a module, you can move the item to wherever you wish on the page – or even onto another page or sheet (see below).

💻 To move a module, follow these steps:

→ Click-and-drag the shaded area of the Counter module, and position it so that the connecting line from the Input property to the module is straight – then release the mouse-button

💻 To move several items at once, follow these steps:

→ Click-and-drag a bounding rectangle around the two output properties O1 and O2 to select them – they will both have thick surrounding lines to show they are selected

→ Click-and-drag the shaded area of one of the selected properties to move both together – release when you have the items where you want them

Working with Pages and Sheets

ObVerse strategy can be created over a number of pages, sheets or a combination of both. Pages and sheets aid the organisation of large amounts of ObVerse.

Click and drag the editor window scroll bars and you will see dotted lines spaced over the entire window space – these show boundaries between A4 landscape pages. Connection lines between modules and properties over different pages will be shown as long as the input lies to the right of an output.

The Sheet tabs in the top-left hand corner of the editor window show which sheets are currently available, the highlighted tab shows which sheet is currently being edited in the main window. Modules and properties can be connected between sheets in the same way as between pages, although the connections are displayed differently: When a connection is made between a module and a property over different sheets the reference of the connected module or property will be shown on the input or output.

💻 To add another sheet to the editor window, follow these steps:

→ Click on the '+' icon next to the latest sheet tab, the next numbered sheet will now be added

Adding Comments

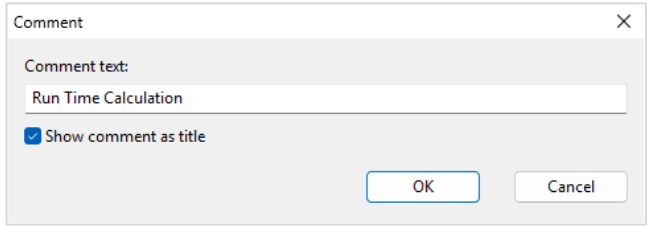
Adding comments to ObVerse strategy makes it easier for others (or even yourself) to see what you were trying to do with the ObVerse.

ObVerse comments come in two sizes, regular and title.

💻 To add a comment to our ObVerse strategy, follow these steps:

→ Right-click on the screen above the module, and select **Insert Comment...** from the popup menu

→ In the **Comment Text**, enter the comment 'Run Time Calculation', select **Show as title**, and press **OK** – the comment appears in bold text



Again, you can move the comment by click-and-dragging it to where you wish.

Saving ObVerse Changes

After changing ObVerse strategy, you can save a copy to a local drive, for backup purposes.

To save the current ObVerse strategy for the first time, follow these steps:

→ From the editor menu, select **File > Save As...**

→ Save As will ask you to specify a **File name** for the ObVerse file. Enter 'ObVerse Example' and click **Save**

→ The ObVerse will be saved to the drive in the 'TypeInfo\ObVerse' folder within the ObSys Application Data folder.

The folder in which the ObVerse file is saved is related to the ObVerse type that is reported by the ObVerse when it is scanned – and is shown in the title bar of the editor window. If the window title bar of the editor shows *xxxx**yyyy*, then the ObVerse file will be stored in 'TypeInfo*xxxx**yyyy*.obv' within the ObSys Application Data folder.

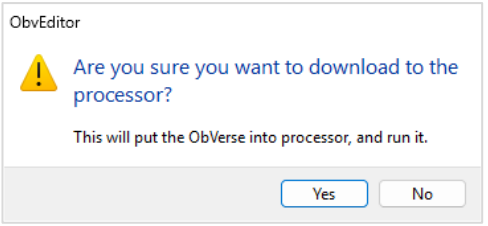
Tip: When saving a Commander to a Backup, either from the top-level of Commander or the Configuration object, the ObVerse strategy will be included.

ObVerse Processors

Now you have written your first chunk of ObVerse strategy, you need to practice running, watching and seeing how other tasks within Commander access the properties within the ObVerse processor.

Running your ObVerse Strategy in Processor

While you are editing ObVerse strategy, the ObVerse processor continues to run the strategy already in it (even if it has none). You need to put the new strategy in the processor to allow it to run.



To put the new ObVerse strategy into the processor and set it running, follow these steps:

→ On the editor toolbar, select **Run**. When asked 'Are you sure…', click **Yes**

→ If the file needs saving, you will be asked 'Do you want to save', press **Yes** – after the file has been saved it will be put in the processor and will run automatically

When the editor puts strategy into the processor, it sends it an item at a time, and so you might see a downloaded item count. This depends on the size of the strategy, and the speed of the link between the editor and the processor.

Manually uploading ObVerse Strategy from the Processor

After you have downloaded and tested the strategy, the editor may be closed. Remember that the ObVerse strategy was stored in two places – a file on the PC and in the ObVerse Processor.

When an ObVerse processor object is selected in Commander, the editor window opens and automatically displays the strategy which is running in the processor. This is worth bearing in mind if you lose the original ObVerse file, or you need to see the running strategy within a processor.

To manually upload the ObVerse strategy currently running in the processor:

→ From the editor window, select **Processor** from the dropdown menu then click **Get ObVerse** to display the last strategy which was downloaded into the processor.

Accessing Property Values from Elsewhere

Once the ObVerse strategy is running, the public properties (not the private properties) become available within Commander as value objects. The ObVerse processor itself appears as a container object in the top-level of Commander, and the input and output properties appear within that container object.

Other tasks within Commander can read from and write to the public properties - for example, transfers.

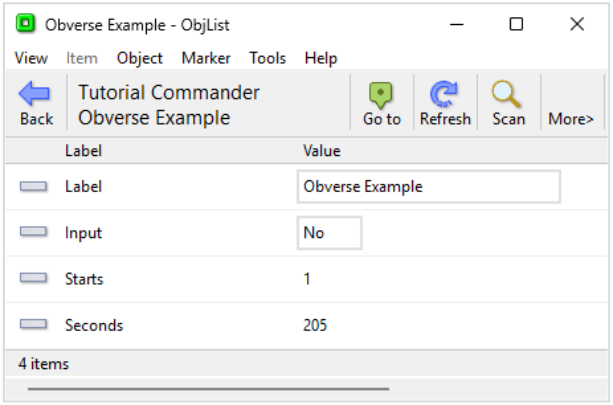
To modify an input property using ObView, follow these steps:

→ Navigate to the top level of Commander

→ You may need to **Scan** to see the new ObVerse Example objects added by the ObVerse processor

→ Open **ObVerse Example** – again you may need to **Scan** to see its sub-objects

→ Set **Input** by clicking the object's value and selecting 'Yes'– the property is changed, and the ObVerse strategy continues counting seconds – ObView will normally refresh the values every 5 minutes, so you will need to press **Refresh** to see the values changing more regularly

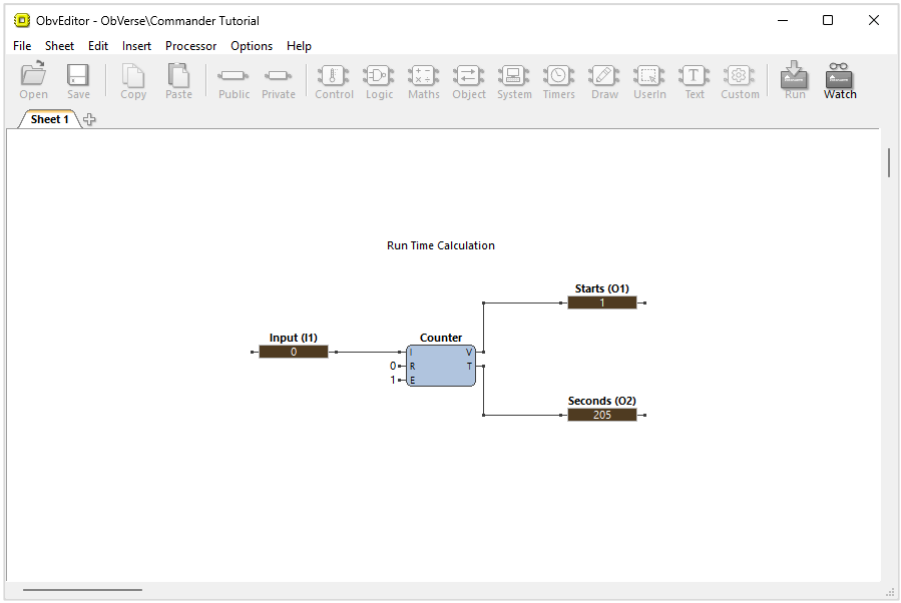


Watching ObVerse Run

After downloading the ObVerse strategy, you can use the editor to watch what is happening in your ObVerse processor - the editor shows the live data from each property. When watching, it is only possible to adjust property values; objects cannot be moved, added, or deleted. Be careful adjusting output and private property values as they will normally be overwritten when modules recalculate.

To start Watching, follow these steps:

→ From the editor window, select the **Watch** button on the toolbar



To see a current value, if the value is too large, follow these steps:

→ **Move** the cursor over the property, and the tooltip will show the full Current Value

To change a property value, follow these steps:

→ Double-click on the property, and the Value dialog box appears.

→ When asked 'Are you sure…' select **Yes**

→ Change the value to '1', and press **Ok** – the value changes, and the editor shows how that value affects the strategy - in our case the Starts property increases, and the Seconds counter starts to rise

To stop Watching, follow these steps:

→ From the editor window, click **Watch** on the toolbar a second time

Note: Watching ObVerse strategy running within a processor can put stress on the communications path between the editor and the processor, and so should be used only when necessary.

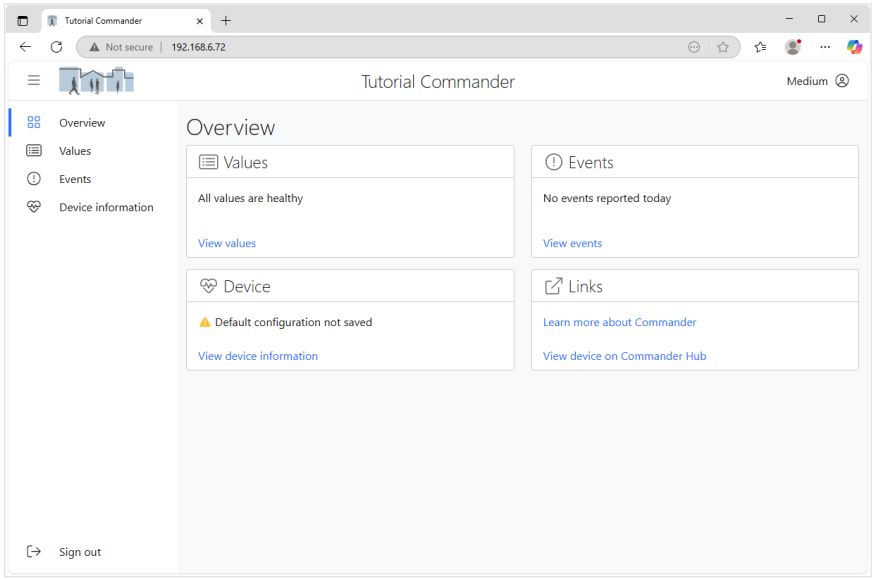
Summary

In this section you have learnt:

- ✓ Time control – using the calendar and setting a day-type, using timers to control off/on processes, using profilers to control analogue values
- ✓ ObVerse Programming – understand the basics of an ObVerse processor and its strategy, using ObVerse editor to add modules and properties, how to use the ObVerse Manual
- ✓ ObVerse Processors – running strategy in a processor, accessing public properties, debugging using watch.

Informing…

Local Web Pages



Commander can provide pages using a built-in web server.

Commander will serve the Essential Data pages and objects, showing the values it last received, which users can adjust.

By default, the web server is not enabled. So, we first need to enable it. We can control access to the web server from local and remote networks, with view and adjust, view only, or no access.

To enable the web server on the local network, follow these steps:

→ Navigate to **Configuration**, **Web Server**

→ Set **Access from Local Network** to 'View and Adjust'

→ If you also need to view the web pages from remote networks, set the **Access from Remote Networks** as required.

Now the web server is enabled, we can view it from a web browser. Commander builds simple responsive html pages, so you can use any modern browser, including those on mobile devices.

To view Commander's web pages, follow these steps:

→ Start your preferred web-browser on your engineering PC – this must be on the same network if you have been engineering

→ In the **address bar**, enter the IP address of the Commander. In this tutorial we earlier set the Commander to '192.168.2.150'

Tip: A shortcut to the web page is available from the top-level of Commander. Click **Extras**, **Open in browser**.

→ Commander's web page appears.

The website is arranged into the areas: Overview – showing summary information; Values – to view and adjust values from Essential Data; Events – from Alarm History (we will cover this later in the tutorial); and Device information.

It is possible to enable user sign-in and disable certain web pages – this needs knowledge of Commander's security features covered later in this tutorial.

Commander Hub

Commander Hub extends the power of Commander by providing easy remote viewing and editing, long-term recording of building data, and management of shared access.

Commander Hub is part of North's cloud-based services. When enabled, all Essential Data values and Alarm History events (which we will cover later) are sent to North's cloud service. It requires access to the Internet.

By enabling the link to Commander Hub, you agree to have read and understand the *Terms of Use* and *Privacy Policy* for Commander Hub.

To enable a link from your Commander to Commander Hub, follow these steps:

→ Open **Configuration, Commander Hub**
→ Set **Enable Link** to 'Yes'
→ **Link to Hub** should change to 'Ok', or report any issues
→ Wait for **Registration Code** to update, this may take up to 15 minutes after you have enabled the link
→ You will need the **Serial Number** and **Registration Code** to add the device to My Hub.

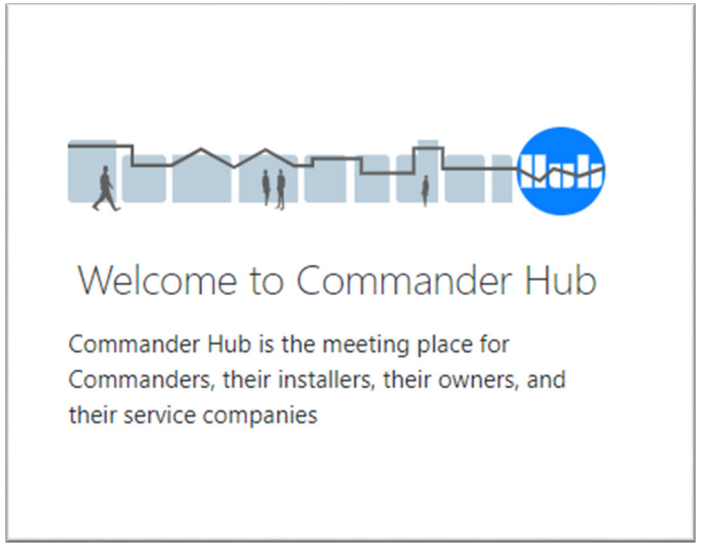
To add your Commander device to My Hub, follow these steps:

→ Visit *www.cmdrhub.com*

Tip: A shortcut to Commander Hub is available from the **Extras** menu, select **View on Cmdr Hub**

→ If you do not already have an account, you will need to register
→ From the **My Hub** page, select **Options**, **Add device…** and enter the information requested.

If a Commander has already been added to another user's account, give them your Hub username and ask them to share access with you.



Alarm Events

Typically, when a user or task wishes to know the value of an object, say a digital input, they can read it. This method works well when the values are needed only occasionally.

There are times, however, when instead of a user monitoring a value, it is better that the object tells the user when it changes. North call these object-to-user event messages 'alarms'.

Within any North system, alarms have the same basic format, made up of six pieces of information:

- System Name – the system name assigned by the engineer
- Point Name – the unique point name within the System
- Condition – the condition or state that the Point is now at
- Priority – the importance of the alarm message, set by the engineer, or by the system
- Date – the date that the Point went to the Condition
- Time – the time that the Point went to the Condition.

When alarm events are sent between object and user, they are sent in text form, with the '|' character separating the different parts.

The System Name relates to the system that sent the alarm. It could be a Commander Label if the Commander generated the alarm. It could be a system label of an interface running within Commander, say a fire alarm system.

The Point Name is usually generated by the system, although the engineer might be able to specify some point names.

The Condition informs the user of the new condition the Point is in. Each time the condition changes, an alarm is sent.

The Priority is a single digit, in the range 1 (the highest priority), to 9 (the lowest), and is sometimes specified by the engineer. Although particular priorities have no official meanings, the table (shown right) indicates how they are generally used.

Priority	Meaning	Default Colour
1	Life-critical – someone may get injured	Red
2	Property-critical – something may be destroyed	Orange
3	Important – needs some action	Yellow
4 to 9	Information only	Blue/Grey

The Date and Time refer to the instant the condition occurred (if the time was known – some objects do not know the actual time).

Generation and Delivery

Alarms must travel from the object that generated them to the user. The easy part is the generation. The most complex part is the display part because each user wants the messages delivered in a different way, to different places.

Commander can help with the generation. If the external systems cannot send alarms, Essential Data has some features which can generate alarms itself.

Commander delivers all alarm events, including those generated by Essential Data, in the same way. All alarms are sent to the Alarm Delivery task, which will deliver them onwards, perhaps based on priority and/or content, to several destinations.

One Alarm Delivery destination, which is set up by default, is to the Alarm History. The engineer can set up other destinations in other places, including to other Commanders and ObSys software, emails messages, SMS, and printers.

Alarm History

Commander has an Alarm History, to which the Alarm Delivery module sends alarms. The Alarm History is a rolling record of the last 100 or so alarm events. This provides a simple way of testing alarms, as well as a record.

To view the alarm history using a web browser, follow these steps:

→ Open the Commander's **web page** in your browser. The **Overview** page may indicate if any events were reported today

→ Select **Events** from the menu on the left

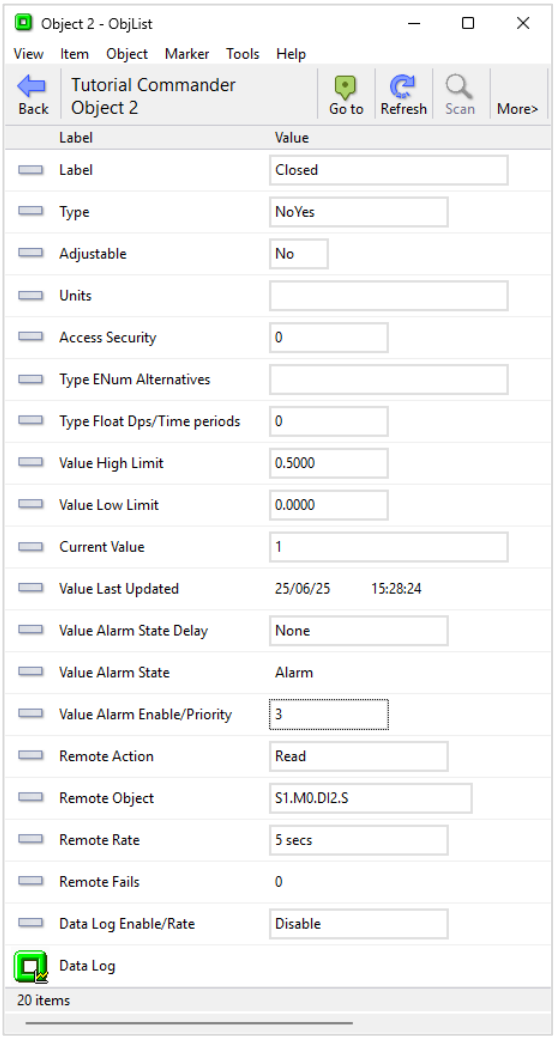
→ A list of alarm events are shown, with the most recent first.

Alarm events that are put into the Alarm History are also sent to Commander Hub, if enabled within the Commander.

Generating Alarm Events using Essential Data

Some external systems connected to Commander send alarm events. Fire Alarm systems, originally designed to print alarms, have made alarm sending the main part of the link to North. Other systems based more on values and states, like Modbus, never send alarms.

Essential Data can generate alarms on behalf of any system that cannot send its own. As seen before, Essential Data is set up to read the values of external systems. By setting the value high and low limits, it can also monitor the value being read, and work out when the value is 'out-of-limits'.



To set up an Essential Data object to generate alarms, follow these steps:

- → Open **Configuration, Essential Data**
- → Open the **Zip Input 2** page, and **Object 2**
- → Set **Label** to 'Closed', **Type** to 'NoYes', and **Current Value** to '0'
- → Set **Value Low Limit** to '0', and **Value High Limit** to '0.5' – when the value is 0 it is ok, when it is 1 it goes out-of-limits
- → Set **Remote Object** to 'S1.M0.DI2.S' – i.e. the state of the Zip digital input 2
- → Set **Remote Rate** to '5 seconds' – the state will be collected every 5 seconds
- → Set **Remote Action** to 'Read' – the object will be read – and the Current Value should change to the correct state
- → Using the switch we connected earlier, switch the digital input and leave for 5 seconds. The **Value Alarm State** will change to reflect whether the **Current Value** is within limits – but no actual alarm is sent because the Alarm Priority is 0.

To set the Alarm Priority, and therefore enable sending of alarms, follow these steps:

- → Set **Value Alarm Enable/Priority** to '3' – this is a non-critical alarm
- → Switch the digital input, to change the **Value Alarm State**, and cause the alarm to be sent to Alarm Delivery, which delivers the alarm to Alarm History
- → On the web browser, view the Commander's Events page to see the alarm.

Generating Alarm Events from Zip Modules

Zip is an external systems that can send alarm events. For example, the ZipMaster driver checks communications with the Zip modules and can generate alarms when a module stops replying to requests.

To set up a Zip Module communications alarm, follow these steps:

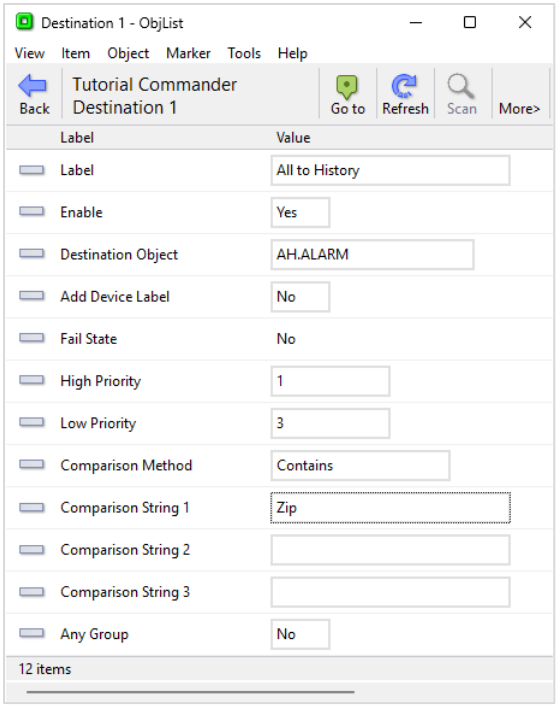
→ **Go to** the top level of Commander, and open **Zip System**

→ Open **M7002A v10** (object M0)**, Module Information**

→ Set **Alarm Priority** to '2' – module alarm messages, including communication alarms, are now enabled

→ Disconnect the RS232 cable from the Zip NC12B – the modules will blink to indicate 'loss of comms', a Zip System 'Comms Fault' alarm event will be generated and delivered to the alarm history – you can see this on the Event web page

→ Reconnect the RS232 cable and a 'Comms Ok' alarm event will be sent.



Routing and Filtering

By default, Alarm Delivery is configured to deliver all alarms to one destination – Alarm History.

You can use Alarm Delivery to filter alarms based on the alarm priority or contents: Alarm Delivery will only pass alarms that meet your criteria; these are sent onwards to the destination.

The alarms events so far have been priority 2 and 3, so you could modify the filter to only deliver priority 1 to 3 events and discard lower priorities.

To send only alarms with priority 1 to 3 to the Alarm History, follow these steps:

→ **Go to** the top level of Commander, and open **Alarm Delivery**

→ Open **Destination 1: All to History**

→ Set **High Priority** to '1', and **Low Priority** to '3'.

Both events also contain the text 'Zip', so you could modify the filter to only deliver these.

To send only alarms that contain the word 'Zip' to the Alarm History, follow these steps:

→ Set **Comparison Method** to 'Contains', and **Comparison String 1** to 'Zip' – remember that the comparison string is case-sensitive.

Other Alarm Destinations

Below are listed other possible alarm destinations – this document only briefly describes them, as they require additional equipment or knowledge that you might not have to hand.

Email

North's Alarm Email driver is standard within Commander. Once set up with the IP address of an SMTP server, alarms can be sent to one of several groups of people using either standard or HTML emails. If users have their corporate emails sent to mobile devices, then alarms will be passed to remote engineers.

Printer

North's Printer driver supports alarm printing to a serial printer – this type of printer will print single lines, rather that the whole page printers, and therefore can provide not only a way of seeing alarms as they occur, but also a paper record of all alarms.

The Printer driver supports the ESC/P Epson colour printer codes.

SMS

North's GSMSMS driver connects to a SIM-enabled GSM modem. GSM users are added to the driver setup, and alarms can then be sent to any of the users, or to an 'on-call' user. The alarm then arrives at the users GSM phone as a text message.

Other North devices

ObSys supports **Alarm Store**, where alarms are held waiting user silencing and acknowledgement. Both also support alarm translation, and re-prioritisation. North's **Alarm Manager** application can be used to view alarms from several Alarm Stores, and provide a very easy-to-use, powerful interface.

User Access with the Security Server

Let's take a few moments to consider security. Earlier we enabled Commander's web server, and could control access from local and remote networks. Sometimes we need to control not just if access to a feature is available, but who can and who cannot view and modify values.

North products (including Commander and ObSys) contain a security database to authenticate users; remote 'areas' ask a particular security database for information about a user when that user requires access.

Imagine a virtual 'door'. It works as follows:

- When a user wishes to enter an 'area', they present ID (and optional password) to the 'door'
- The door encrypts the credentials, passes them to the security database, asking for the user's privileges
- The Security Server returns the user's name and current privilege levels
- The door decides whether the user can enter the area, and 'opens' if the user is allowed.

Rather than just one privilege level, security databases hold privilege levels in eight different areas for each user – you must tell the door which area it controls access to, and what the minimum privilege level the user must have in that area before he can pass. This means that a user can have lots of privilege in one area, and yet only a little in another.

Commander's security database is called **Security Server**, and by default it has no users or groups set up.

Security Areas and Levels

Within each Security Server there are eight security areas. Each security area could be an actual area in a building, but is typically a functional area: for example, ‘environmental control’ and ‘North engineering’ areas would allow a user to have different privileges in controlling set points and engineering Commanders.

There can be many doors on a system. Each lock controls access to some functionality, and the engineer assigns each door to one of the eight areas. The engineer also assigns the minimum privilege level needed within that area - zero means no privileges required - seven is the highest security.

The engineer gives each user a privilege level in each of the eight areas. The level is in the range zero to seven, seven being the most powerful. When a user wishes to pass a door, their privilege level in the door’s area is checked against the minimum required for that area – and then either allowed to pass or rejected.

The engineer must decide the use of the eight areas. The engineer must also decide the power of the privilege levels. Most systems use only a few levels per area: 0=No access, 1=Guest, 2=User, 7=Administrator.

As an example, imagine a door into a secure room. The secure room is in area 1. The door needs a user to have a minimum privilege level of 6 in area 1 before it will open. The door has a card-reader that checks users with a security database before unlocking the door. User A has privilege level 7 in area 1 – they can open the door. User B has privilege level 5 in area 1 – she cannot open the door. User C has privilege level 1 in area 1 – he cannot open the door.

The example continues: on the same Commander, a temperature set point can be viewed by all, but users need a minimum privilege level of 2 in area 1 to adjust – therefore Users A and B can adjust the set point, but User C cannot.

To open Security Server, follow these steps:
→ **Go to** the top level of Commander, and open **Security Server**
→ Notice settings for **Area** and **Privilege level** labels.

User Records

As well as holding the user's own privilege level in each of eight areas, the security database holds the user's name, and whether the user is enabled – if disabled, the user will never be validated.

Each user has a User ID (or card number) and a Password – together these form a coded Token. It is the coded token that is passed around a system – the password is never seen.

You may make a user a member of up to three groups. Each Group has an enable object, along with group privilege levels for the eight areas. The privilege levels for the user will now be an amalgamation of any groups they are a member of, along with their own privilege levels.

Optionally, a temporary user can also be given start and end dates – if these are set up, the security database disables the user automatically if today's date is not within the range specified.

Whenever an application attempts to authenticate a token with the security database, the database remembers the date and time of the successful validation – this allows you to see when the user was last validated.

Enable Editor Access

The security server requires unlocking before any significant changes can be made to the database. Before we can add a user record, you must first sign-in using the Editor Password. By default this is blank, and we recommend you change this on first use.

To enable Editor Access, follow these steps:

→ **Go to** the top level of Commander, and open **Security Server**

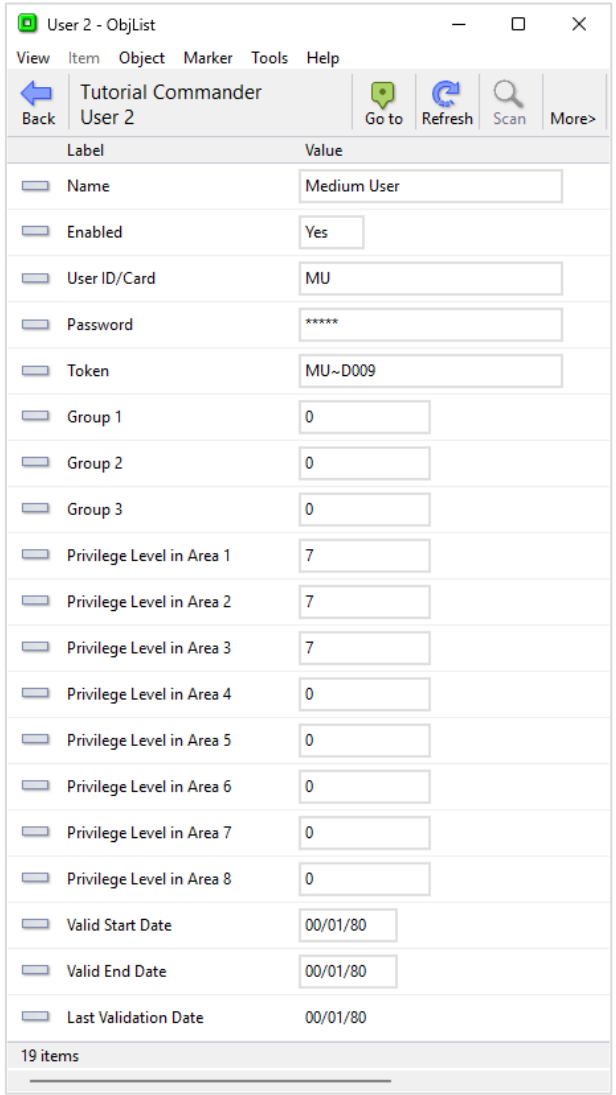
→ Notice **Editor access allowed** is 'No', so editing is not allowed and we need to sign in

→ Set **Editor sign in** to the current Editor Password. The default setting for this is blank, so delete the current value and press **OK**

→ **Editor access allowed** object should now show 'Yes'.

Adding Users

Let's create two users each with different privilege levels. In the following examples, we will use area 1 for controlling web access.



- To add a low-level user, follow these steps:
  - → Open the next available **User** entry – we will use **User 1**
  - → Set **Name** to 'Basic User'
  - → **Enable** to 'Yes'
  - → **User ID/Card** to 'BU', and **Password** to a memorable phrase
  - → Set **Privilege Level in Area 1** to '2' – this is the one area where this user has power.

- To add a medium-level user, follow these steps:
  - → Navigate to the next available **User** entry – we will use **User 2**
  - → Set **Name** to 'Medium User'
  - → **Enable** to 'Yes'
  - → **User ID/Card** to 'MU', and **Password** to a memorable phrase
  - → Set **Privilege Level in Area 1** to '7'
  - → Set **Privilege Level in Area 2** to '7'
  - → Set **Privilege Level in Area 3** to '7' – three areas are set to a high privilege level.

Here is a summary of User and Privileges set up so far in the tutorial:

User	Area1	Area2	Area3	Area4	Area7	Area6	Area7	Area8
BU	2	0	0	0	0	0	0	0
MU	7	7	7	0	0	0	0	0

Tip: when you are editing lots of User objects, or want to see all the users in one list, you can view them in a table view. Open **Security Server**, then from the menu select **Extras**, **View Users as Table**.

Enabling User Sign-in on Web Pages

Once you have added users to Security Server, you can enable Web Server sign-in which will only give authenticated users access. When a user views a web page, the server will prompt for their username and password.

💻 To enable sign-in on Commander's web server, follow these steps:

→ Navigate to **Configuration, Web Server, Security**

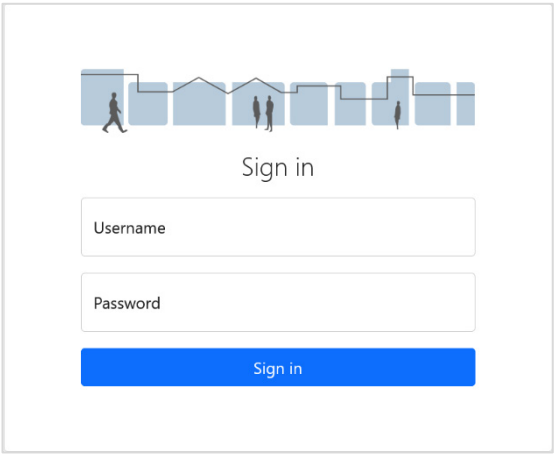
→ Set **Require User Sign-in** to 'Yes'. This means a user's details are checked against the local Security Server when accessing pages on the local Web Server.

💻 To sign in on Commander's web server, follow these steps:

→ Using your browser to view the Commander's web page, you will be prompted to sign in

→ You should now be able to sign in with username 'BU' (basic user) or 'MU' (medium level user), with the password set earlier

→ Select **Sign Out** to end your session.

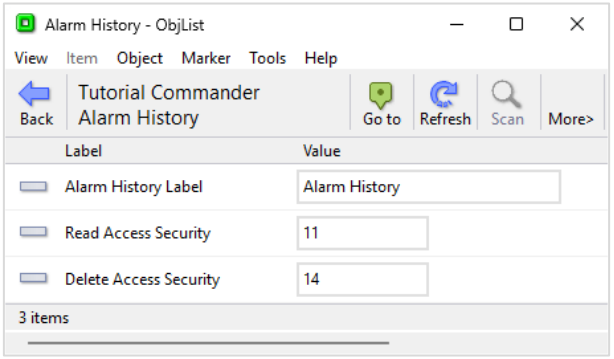


Specifying Access Security

Many features within Commander can be protected by specifying a security area and minimum privilege level required to access the protected feature.

An item that supports security will have one or more Access Security objects, each of which has a two-digit value. Each object controls the access to a feature – such as reading the value, adjusting the value, or viewing the page, etc.

The two-digit access security value is made up of the security area digit (1-8), followed by the minimum privilege level (1-7). For example, if the minimum privilege level is 6 in area 2, then the two-digit value is '26'. If the access security value is '00', then no security checks are required.



Access Security in Essential Data

Using Access Security in Essential Data, you can control who can view Essential Data pages within their browser, and control who can adjust each adjustable object.

To set the viewing Access Security on an Essential Data page, follow these steps:

→ Open **Configuration, Essential Data**, and then the page – we will use **Zip Input 2** page

→ Set **Access Security** to '11' – this means that our example users 'BU' and 'MU' can both view the page.

To set the adjusting Access Security on an Essential Data object, follow these steps:

→ Open the required adjustable object – we will use **Test Changes** page, object **Limited**

→ Set **Access Security** to '15' – this means that our example user 'MU', who has level 7 in area 1 can adjust the value, but user 'BU' who has level 2 in area 1 cannot adjust the value.

Summary of Web Server access set-up so far in the tutorial:

Action	Access Security	User 'BU'	User 'MU'
View Page 2: Test Changes	11	Yes	No
Adjust Page 2 Object 1	15	Yes	Yes

Summary

In this section you have learnt:

- ✓ Local web pages – turn on the web server to access values from Essential Data
- ✓ Commander Hub – how to enable the link to Hub, North's cloud-based service
- ✓ Alarm events – understand alarm events, their generation, delivery, and using alarm history
- ✓ Security Server – understand how to control user access with security areas and levels, adding users, then controlling access to web pages and essential data.

and Finally…

Default Configuration

Commander's firmware and CDMs are held in permanent flash memory. This memory is written to when the firmware is updated.

Commander's configuration is held in battery-backed memory – if the battery expires (or is removed) when the external power is removed, then the configuration is lost.

To counter this loss, it is possible to save Commander's configuration to permanent memory – similar to *saving to a backup*, covered in the first part of this tutorial. Once saved, whenever the configuration in battery-backed memory is lost, it is reloaded from the default configuration in permanent memory.

💻 To save Commander's current configuration as the default configuration, follow these steps:

→ Set the internal **PROGRAM** switch to ON and restart Commander

→ The MODE LED will blink, to show Commander is in Program mode

→ From the top-level of Commander, open **Configuration, Platform Information, Default Configuration**.

→ Set **Save Configuration As** to 'DF1' to cause the configuration to be written to permanent memory. Commander will light the SAVE LED and save the configuration (which takes 20-30 seconds), then restart.

→ Once online, **Saved Configuration** shows the label 'DF1' and **Saved Time** will update.

Removing all Data from Commander

At the end of the Commander's useful life, before you recycle it, you can wipe both the default configuration and battery-backed memory.

If you only remove the battery, Commander will reload the default configuration from permanent memory when re-powered.

To remove all data from Commander, follow these steps:

→ Set the internal **PROGRAM** switch to ON and restart Commander

→ The MODE LED will blink, to show Commander is in Program mode

→ From the top-level of Commander, open **Configuration, Platform Information, Default Configuration**

→ Set the **Save Configuration As** to 'blank' (case sensitive) – this takes 20-30 seconds, after which time the **Saved Configuration** shows an empty value ('')

→ Remove the battery from Commander

→ Disconnect the power from the Commander for at least 1 minute.

Updating Commander's Firmware

When supplied, the Commander hardware contains the main system firmware, along with several popular North drivers. North provide other less-used drivers in separate CDM files, which you need to install before you can use them. North also release updated firmware which you may need to install.

To allow Commander's firmware to be updated, it must be put in Program mode by physically enabling the PROGRAM switch. This means that firmware cannot normally be modified accidentally or maliciously.

You can set Commander to update its firmware from either North's cloud-services if it has Internet access, or locally from a PC using TFTP.

We recommend *taking a complete backup* of the Commander's configuration before updating any firmware.

Cloud Update

If Commander has Internet access (and therefore not in Default IP mode), it can be instructed to check and update its firmware to the latest released version available from North's cloud-based servers.

Two different update options are available:

- In-use only – the quickest option, updates the system firmware and started interfaces
- All factory installed – updates the system firmware and all *drivers installed at production time*.

To add a driver that has not been previously installed, add the driver name to an available Interface before updating the 'in use' firmware.

💻 To update Commander using North's cloud service, follow these steps:

→ Set the internal **PROGRAM** switch to ON, and re-power Commander

→ The MODE LED will blink, to show Commander is in Program mode

→ From the top-level of Commander, open **Configuration, Platform Information, Software Cloud Update**

→ Set **Check and Update Software** to either 'In use only' or 'All factory installed' to have Commander update the firmware. DO NOT REMOVE THE POWER during this operation

→ The **Progress** object shows how the update is progressing. The Commander may restart several times during this time. Once finished, the **Check and Update Software** returns to 'Off', and **Progress** shows how many files were updated.

TFTP Update

Commander supports local updates using the Trivial File Transfer Protocol (TFTP). Use TFTP when no Internet connection is available, or a driver is unavailable via cloud update.

TFTP is one of the standard IP protocols. TFTP requires a client (your PC) and a server (Commander).

TFTP is disabled by default, on both your PC and Commander.

Enable TFTP on your PC

Before you can use the TFTP client on your PC, it must first be enabled.

To enable TFTP client on Windows, follow these steps:

→ From Windows **Control Panel**, select **Programs** then **Turn Windows features on or off**

→ Select the check box next to **TFTP Client** to enable it, then click **OK**. You may require administrator privileges.

Tip: A shortcut Is available to enable TFTP on your PC. Open **Configuration**, **Interfaces**. Select **Extras**, **Enable TFTP Client on Windows**.

Locating the CDM Files

North distributes drivers for Commander in CDM files – these files are made to work in certain areas of Commander's memory – called banks. You can only load one CDM file per bank, so you must choose which CDM to load in each bank.

ObSys setup installs CDM files within the CDMs folder of the ObSys Program Files folder – for example "C:\Program Files (x86)\North Building Technologies\ObSys\CDMs".

Tip: A shortcut to the CDMs folder Is available. Open **Configuration**, **Interfaces**. Select **Extras**, **Open CDM folder**.

Installing or Updating a CDM

Once you have chosen the CDM file, enable TFTP in Commander and send the file.

If you need to update the system firmware, download three files: *Bank 1.cdm*, *CmdrBase.bin*, then *CmdrExt.bin*.

The easiest way to install or update a CDM is to use the supplied batch file with Commander in Default IP mode.

To load a CDM in Default IP mode, follow these steps:

→ Set the internal **PROGRAM** switch to ON, then enable TFTP by setting **DEFAULTIP** switch ON. Commander will re-start

→ The MODE LED will blink, to show Commander is in Program and Default IP mode

→ Open the CDMs folder. A shortcut is available from **Configuration**, **Interfaces**. Select **Extras**, **Open CDM folder**

→ Within the CDMs folder, you will find a Windows batch file called *Update Commander at default IP address*. To update Commander, simply drop a CDM file on to this file.

→ Once Commander has received and checked the file, Commander will write the CDM to flash memory (lighting its SAVE LED), and when complete, will restart, and the MODE LED will blink to show it is in Program mode again

→ Repeat the TFTP command for any other files to send

→ Set the internal **PROGRAM** switch to OFF

→ Re-power Commander.

Alternatively, you can install or update a CDM when Commander is connected to a network.

💻 To load a CDM to Commander on a network using your TFTP client, follow these steps:

→ Set the internal **PROGRAM** switch to ON, and re-power Commander

→ Enable TFTP in Commander from the top-level of Commander, open **Configuration, LAN Port** and set **TFTP Enabled** to 'Yes'

→ The MODE LED will blink, to show Commander is in Program mode

→ On your PC start **Terminal** app

→ At the prompt, type the following to change to the CDMs folder:

```
cd "\Program Files (x86)\North Building Technologies\ObSys\CDMs"
```

→ Type the following TFTP command to send a file, for example:

```
TFTP –i 192.168.2.150 PUT "Bank3 H 200611 OmronPLC.cdm"
```

→ Once Commander has received and checked the file, Commander will write the CDM to flash memory (lighting its SAVE LED), and when complete, will restart, and the MODE LED will blink to show it is in Program mode again

→ Repeat the TFTP command for any other files to send

→ Close the Terminal window

→ Set the internal **PROGRAM** switch to OFF

→ Disable TFTP in Commander by setting **TFTP Enable** to 'No'

→ Re-power Commander.

Pre-Installed CDMs

Below is a list of the factory installed CDMs installed in Commander. Some CDMs contain several drivers. This list is subject to change.

Bank	CDM
1	ZipMaster, Compass, plus system modules
2	Modbus
4	BACnetIP
9	AlmEmail, GSMSMS, SNMPTrap, TextOut, Printer
17	ExtraData
18	DataSync, JSONData, JSONNotify, MQTT, SG, Telnet
19	AlmSet, DeviceCheck, StateCheck, CsvRead, XmlRead
20	Advanced, ColtOPV, Morley, Notifier, Protec, ZitonZP
21	DALI, Helvar, iLight, LutronQS, Luxmate, PhilipsLM
22	Carel, Daikin, MitsubishiG50, PanasonicVRF, CCNDP2, MitsCity
23	EIB, KNXIP, LonSLTA, Mbus
24	DraytonWiser, EnOcean, HeatmiserNeo
25	Meaco, Hue, Kentec, KentecTaktis, LutronHW
26	TrendIQ, Cylon, DavisWeather, JciMetasys, Heatmiser
27	APC, GeistPDU, PowerOne, OCPP, Eltek

Zero Interface Licence Drivers

Most drivers for interfacing to other external systems need one Interface Licence (IL) to before they will start. However, the following drivers require zero Interface Licences:

ZipMaster	Compass	Modbus	BACnetIP	GSMSMS
JSONData	SnmpTrap	SG	DeviceCheck	AlmSet

Internal Switch Summary

PROGRAM switch

When the internal **PROGRAM** switch is set ON, and Commander re-powered, the Commander enters Program mode, where the following flash memory functions are enabled:

- Commander can be updated with new firmware, either using Software Cloud Update or TFTP service
- Saving the Commander's current configuration as the Default Configuration becomes possible
- Interface Licences can be added

DEFAULTIP Switch

When the internal **DEFAULTIP** switch is set ON, Commander automatically restarts and enters Default IP mode, where the following happens:

- Commander operates with a static IP address of 192.168.192.167, and network mask of 255.255.255.0
- DHCP server is enabled, and will assign an IP address to any connected PC
- TFTP server is enabled (also requires Program mode)
- Within Commander's North IP Devices, the Encryption Method and Authentication Key is temporarily disabled. This allows an engineer to access the Commander without knowledge of this security setting (as long as they have physical access to the Commander)
- Commander enables the Web Server, with view and adjust access, and Require Sign-in is temporarily disabled. This allows access to all web pages and adjustment of any adjustable values
- Changing network settings and Commander Hub link on the Web server is enabled.

Congratulations on completing this tutorial. You have learnt about North's controller, Commander. You have powered-on Commander and used ObSys software to configure it. Following the steps, you have also configured the range of integration and control features available.

As you start to use Commander on your projects, you may have questions about some of the features we have covered in this tutorial. We recommend you take a look at the *Commander Manual*. This describes in detail the different areas of function within Commander, and the Object Specifications section provides help on all of Commander's objects.

Next Steps

Next, learn more about North's Zip input-output system by following the steps in the *Zip Tutorial* using the training pack.

Find tutorials, product and driver manuals, and application notes in the North Product Documentation app.

If you require help, contact support on 01273 694422 or visit *www.northbt.com/support*

North Building Technologies Ltd
+44 (0) 1273 694422
support@northbt.com
www.northbt.com

Author: TM
Checked by: JF

Document issued 25/06/2026.