North

ObSys Tutorial – Part 1

ObSys is a software package that allows a Windows PC to be used for integrating, controlling and managing different systems within a building.

Part 1 of the tutorial shows how to use ObServer, the low-level element of ObSys.

This document relates to ObSys version 2.0

Contents

Getting started…

What is ObSys?

ObSys is a software package that allows a Windows PC to be used for integrating, controlling, and managing different systems within a building.

ObSys can be installed on a variety of PC types, allowing the engineer to build different solutions: embedded PCs become integration controllers, desktop PCs become displays for engineers, and rack-mount servers become massive data loggers.

The core of ObSys is ObServer – low-level software that runs behind the scenes. ObServer integrates with third-party systems to form a single coherent system.

ObSys also includes display and analysis applications – which can access information from any connected third-party systems: ObView is a general-purpose graphical user interface, Alarm Manager receives and displays alarm messages, and Data Manager logs data for display and analysis.

This tutorial is split into two parts:

- Part 1 covers ObServer and how to engineer it with ObView
- Part 2 covers using Alarm Manager, Data Manager, and creating your own views using ObView

This tutorial uses the North Training Pack as an example of a building system.

What is ObServer?

ObServer is the most powerful of North's family of building controllers. ObServer contains North's interface technology, cause-and-effect language, and easy-to-use web services. ObServer can work as a stand-alone controller, or alongside other ObSys applications; becoming part of a larger control or monitoring solution.

Interface Technology

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

Programmable Control

ObVerse is North's block-based programming language. It is available in all instances of ObSys, and 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

ObServer supports North's standard protocol, allowing communications with other North products, including powerful engineering tools and display software. ObServer also generates and serves standard HTML web pages automatically - these remove the need for graphical design and provide a consistent user display. ObServer can also monitor and inform users about alarm conditions using email or SMS for example.

You can also extend information services using, for example, BACnet and Modbus.

Installing ObSys, the Engineering Software

You can engineer all North products using North's ObSys software package. ObSys 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

If you downloaded and saved the ObSys set-up package from North's website, *www.northbt.com*, then run the self-extracting file.

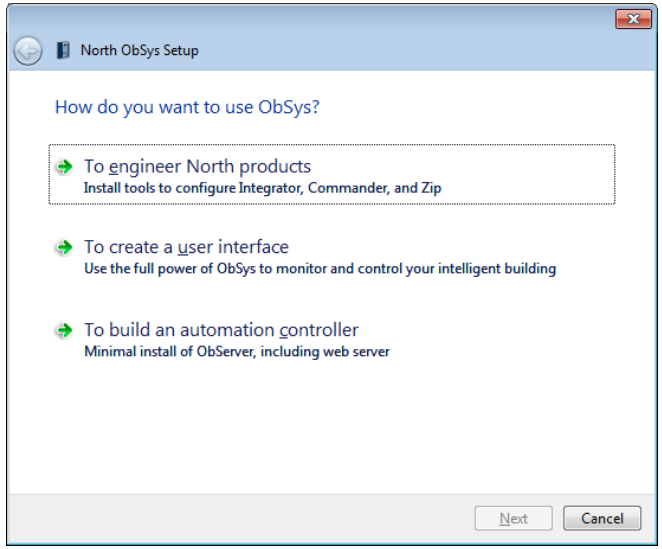
If you have ObSys on a CD-ROM or flash drive, just insert it into your PC – the 'autorun' feature will start the installation automatically. If installation does not start automatically, you can run SETUP.EXE manually from the drive.

ObSys Setup

To specify what to install, follow these steps:

→ At the **How do you want to use ObSys?** welcome page, press **To create a user interface**

→ After reading the Licence Agreement, if you are happy to, select **I accept...**, and press **Install**.



Introduction to ObSys

ObSys is a package containing several Windows applications. ObServer is the main application. ObView is another important application. Further engineering applications needed later are: ObVerse Editor, WebView Editor, and Object Editor. This part of the tutorial covers using ObServer – it also shows how to use ObView to engineer ObServer.

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. These can communicate via ObServer. ObServer can also link across physical networks to other XOM-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 for discovering, viewing, and editing information within ObServer. You can think of it as your way of communicating with ObServer. It also allows the creation of site-specific views; these can provide simpler views for the different users of the system.

To start engineering using ObView, once ObServer is already running, follow these steps:

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

ObServer Overview

Before we connect to real inputs and outputs, a quick understanding of ObServer is needed.

ObServer is the core application of ObSys. It runs in the background on a PC. It normally starts when the PC is started and runs until the PC is shut down.

💻 To start viewing ObServer objects on your PC, follow these steps:

→ From Windows **Start**, and select **All Programs** > **North Building Technologies** > **Start Engineering**

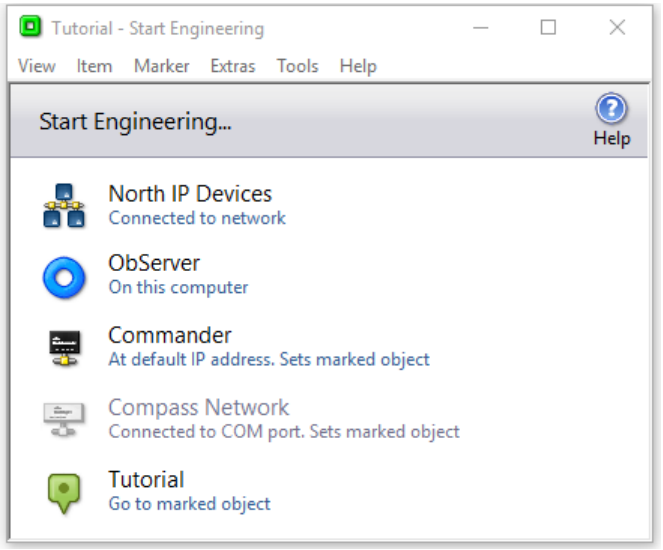
→ From Start Engineering, left-click on **ObServer** – ObView will then display the top-level object in ObServer

→ Left-click on **Configuration** to open that sub-object, and ObView will show a list of the last Configuration objects it found

→ Press **Scan**, and ObServer will re-scan the sub-objects within Configuration - once finished, ObView shows the new sub-objects found

→ Press **Back** to go back to the previous view - the top-level objects within ObServer.

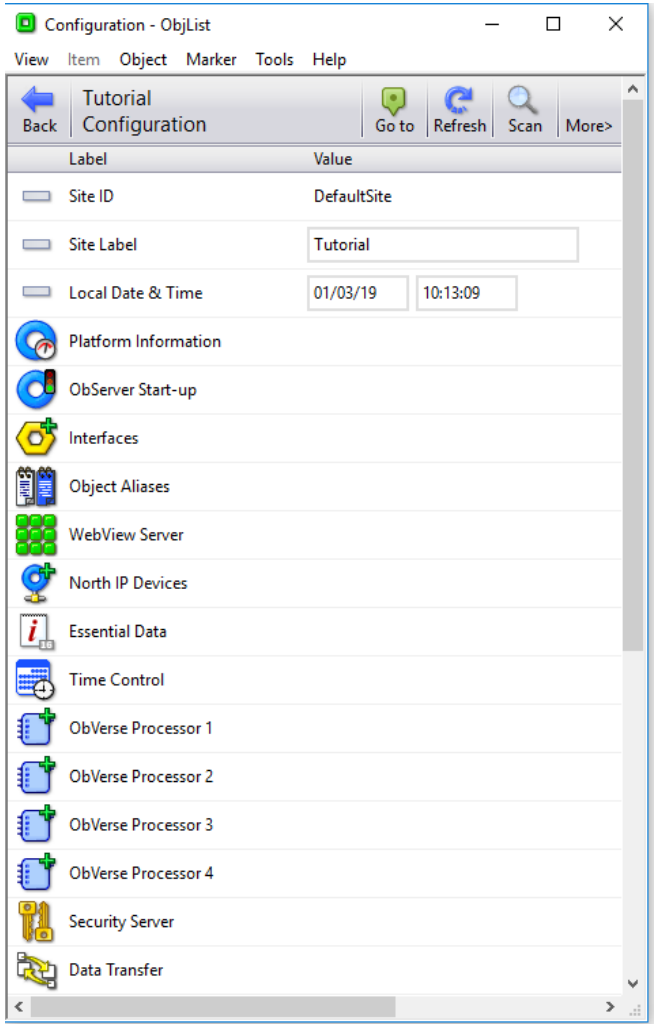
At any time, you may re-run Start Engineering and select ObServer to get to the top-level of your ObServer



Changing ObServer's Site Label

When first installed, ObServer's site label is set to 'DefaultSite'.

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



To set your ObServer's site label, follow these steps:

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

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

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

→ Set the text in the box to 'Tutorial', and press **Ok** – ObServer 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 ObServer

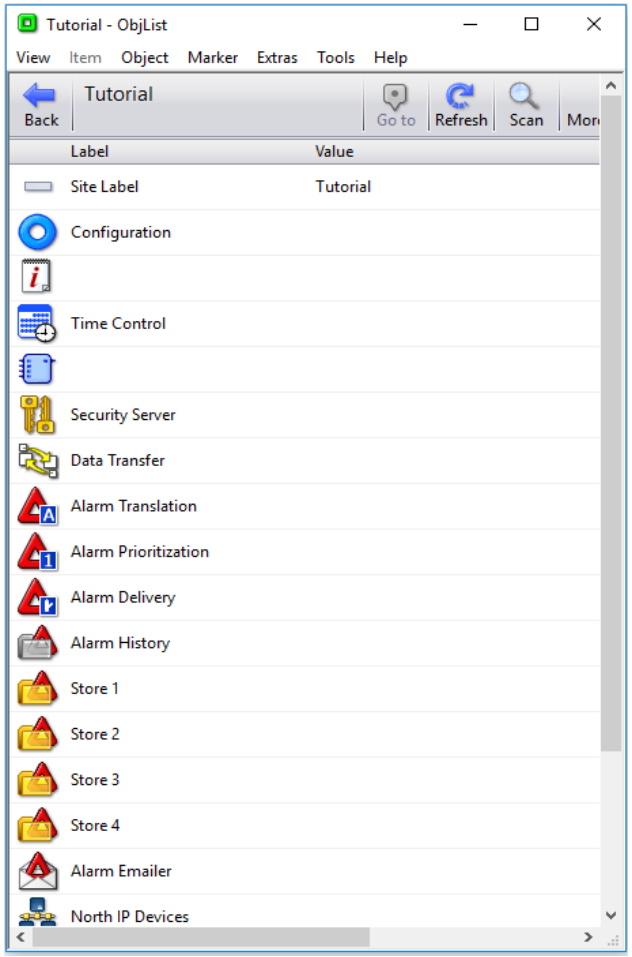
Notice how some objects have a value to the right of the label. Values that can be viewed, but not adjusted, are shown without a box, values with a box around them can be adjusted - you change the object's value by left-clicking on the box, modifying the value on the popup window, and pressing Ok. Some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

Note: The Site ID allows us to identify the name of the site when we are connecting to an instance of ObServer from another North device. The Site ID will only change when this instance is scanned from another machine; it will remain as DefaultSite for the time being.

ObView Window

We have just used ObView, the Object Viewer, to do some simple editing of ObServer. 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 your computer's hard drive. ObView allows us to navigate over this tree structure, displaying information about an 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 common actions to be triggered quickly:

Back will load the view of the previously opened object.

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, the top-level object in ObServer.

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

Scan causes ObView to re-scan the displayed object for sub-objects. ObView enables the button only on objects that have a variable number of sub-objects.

Less decreases the window width – you can still check the status and object references by scrolling the window right and left. **More** increases the window to full width.

Main Area

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 followed by a Status (if the sub-object has one), followed finally by 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 menu with various actions that can be performed on or with the sub-object:

If the sub-object has a value, then **Copy Value** copies the current value to the clipboard, ready to be pasted elsewhere.

If the sub-object is adjustable, then **Adjust** displays the popup window for adjusting it; **Cut Value** copies the current value and sets the object to zero or blank; **Paste Value** pastes the value in the clipboard to this object; **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.

Integrating…

Interfacing ObServer to other Systems

One of the roles of ObServer 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 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 ObServer come in ObServer Module files, otherwise known as OSM files. ObServer comes with popular OSM files pre-installed. It is possible to install other OSM files and to update the OSM files to the latest version – we cover this later in this tutorial.

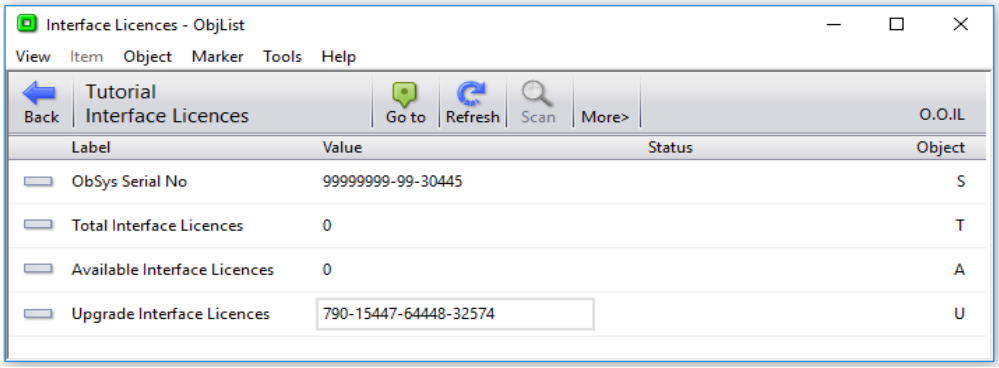
Although ObServer has many pre-installed drivers, it only supports up to ten operating interfaces at one time, and each requires a particular driver.

Interface Licences within ObServer

North supply ObServer with a certain number of interface licences (ILs), normally 1. These are used to license the use of drivers.

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

When an interface is no longer required, the engineer can stop the driver. This causes the driver to return the borrowed interface licence to ObServer – allowing another driver to borrow it, and therefore allowing you to change the interface as you require.



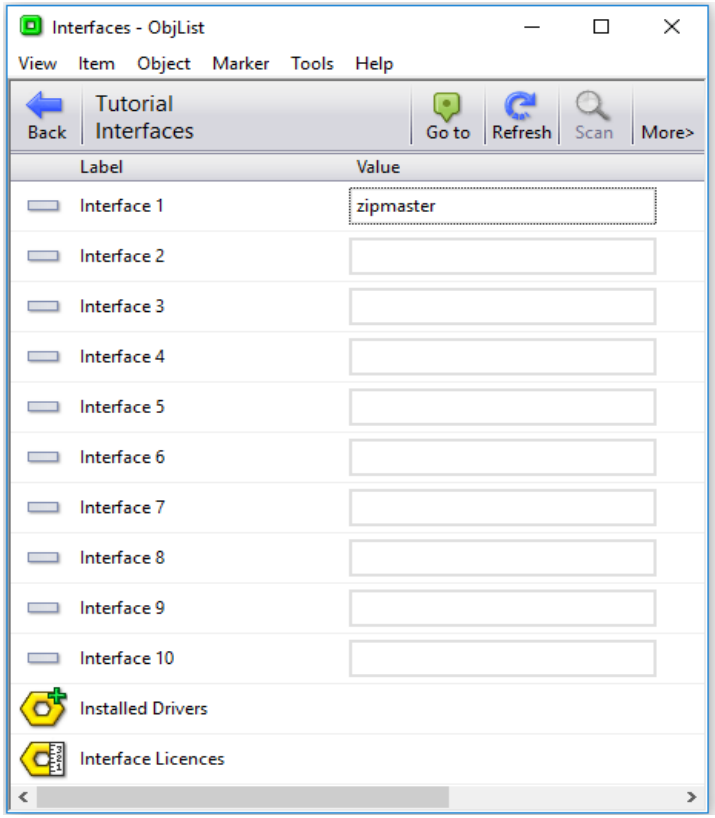
To see the current Interface Licence information within ObServer, follow these steps:

→ Open **Configuration**, **Interfaces**, **Interface Licences**. Each ObServer has a unique Serial Number. Total Licences shows the number of licences the ObServer has - Available Licences shows how many are still unused. If you wish to increase the available interface licences in ObServer, more can be purchased from North. Telephone North with details from the Interface Licences page and a representative will guide you through adding more.

Starting an Interface

Before you start an interface using a particular driver, you need to find out whether the driver is installed on the ObServer, 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. ZipMaster does not require any Interface Licences and can be run freely in ObServer and in Commander.



- 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 ObServer
  - → 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 ObServer - you will need to re-scan that level. Remember to press **Back** twice to return to the top-level of ObServer.

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 **Cut Value** from the popup menu. This will delete the value and therefore stop the driver.

Important Point: If you are following the tutorial, set Interface 1 to 'ZipMaster' to start the ZipMaster driver – we will use it later.

Interface Set up

When ObServer starts an interface, the interface normally adds two new objects to the top-level of ObServer – you will therefore need to go back to the top-level of ObServer and **Scan** the top-level.

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 ObServer 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.

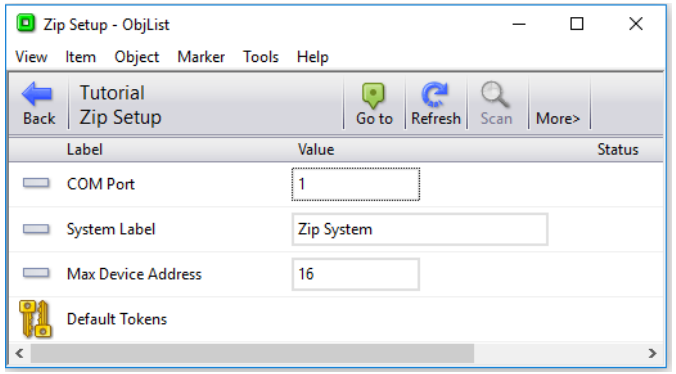
To configure the ZipMaster interface for the North Training Pack, follow these steps:

→ Open **Zip Setup**, (you may need to Scan the top-level object if you have just added the interface) to view the Zip setup objects

→ Set **COM Port** to '1', to tell the driver to use COM1 on the ObServer. Your PC may use a different COM port depending on which ports are available. You can normally check which COM port should be used using Windows' Device Manager (accessible from Control Panel)

→ Connect 12VDC to the NC12B's power connector on the assembled North Training Pack - the NC12B's 'POWER OK' led should light, and the M7002 and M7004 OK LEDs should flash

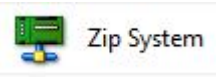
→ Use the cable to connect the ObServer's COM1 port to the NC12B's 9-way port - the OK LED of each Zip module should light solid to indicate that it has a link with the ZipMaster



Note: The North Training Pack consists of an NC12B Network Card, an M7002 module (6 digital input, 4 relay out) at address 0, and an M7004 module (6 universal input, 4 universal output) at address 1, along with the RS232 cable to connect to a Commander or ObSys running on your PC.

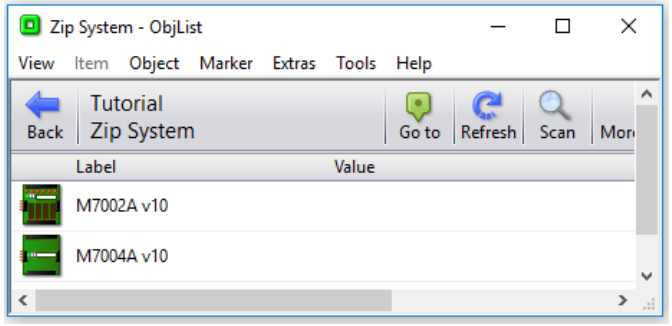
The External System

The second new object that is added to the top-level of ObServer when an interface starts, represents the external system that can be accessed using the driver. It is usually shown with an 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 we cannot document them here.



To view the Zip modules, follow these steps:

→ Open **Zip System** to view the system's object – you may need to **Scan** to see any Zip modules on the network



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 give 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

As examples, the ObServer's label is a Text object, and the ObServer'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 always contain sub-objects. We use container objects 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. You don't need to know or remember them all however.

You have already seen and used container objects – the top-level of ObServer 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 number and types of 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, to allow you to view a fixed list of values in a device for example. 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

A Zip M7001 object is a good example of a fixed container – it always has eight digital inputs

Variable container objects contain site-specific sub-objects. 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 ObServer is a variable container object, and when you change the interfaces within ObServer, you change the top-level sub-objects; adding an interface normally adds two sub-objects to the list. 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 they are 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 ObServer, 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 LS); its sub-object **Reset Count** (sub-object reference RC). The complete reference for the Reset Count within ObServer is **O.PI.LS.RC** - if a task within ObServer 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 reference being the route to the object makes the references extendable. When another device needs to read an object within an ObServer, it needs a reference that describes both the route to the ObServer, and the route to the object within that ObServer. 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:

→ **Start Engineering** ObServer: 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. Click **'More'** on the toolbar to display this information if it is not shown already

→ Open **Configuration**, **Platform Information**, **Last Restart** to see the objects - the reference to this object is O.PI.LS – then right-click on **Reset Count**, and select **Copy Object Reference** from the popup menu – the object reference (O.PI.LS.RC) has been copied to the clipboard

→ Run Windows Notepad, right-click in the main text area, and select **Paste** from the popup menu – and the object reference appears in the document

Transferring Values

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

ObServer has 1000 transfers built-in. You can use each to transfer a value 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 ObServer, or in one of the external systems connected to ObServer – 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 North Training Pack on Interface 1, follow these steps:

→ **Start Engineering** ObServer

→ Open **Data Transfer** (to view the 1000 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.

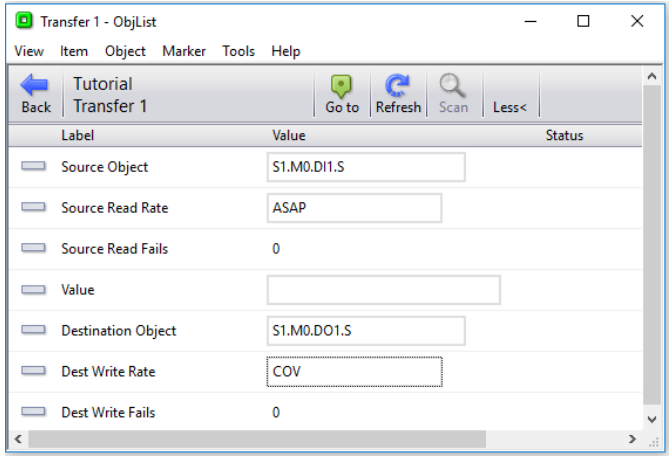
Some systems, however, can 'forget' values if they lose power, and so they may need a periodic re-write of the value, to correct the loss of memory after a power fail.

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 actually changes (or when ObServer is started).

To set up Transfer 1 to write a value to the North Training Pack on Interface 1, follow these steps:

→ Set **Dest Write Object** to 'S1.M0.DO1.S' – system one, module zero, digital output one, 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 Value object shows the Source Object's value, whether it has been transferred to the Destination Object or not.

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.

What is Essential Data?

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

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

- A value can be collected once, to save other tasks 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:

- 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
- ObServer can automatically make Essential Data available as 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.

By default, ObServer's Essential Data has 80 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.

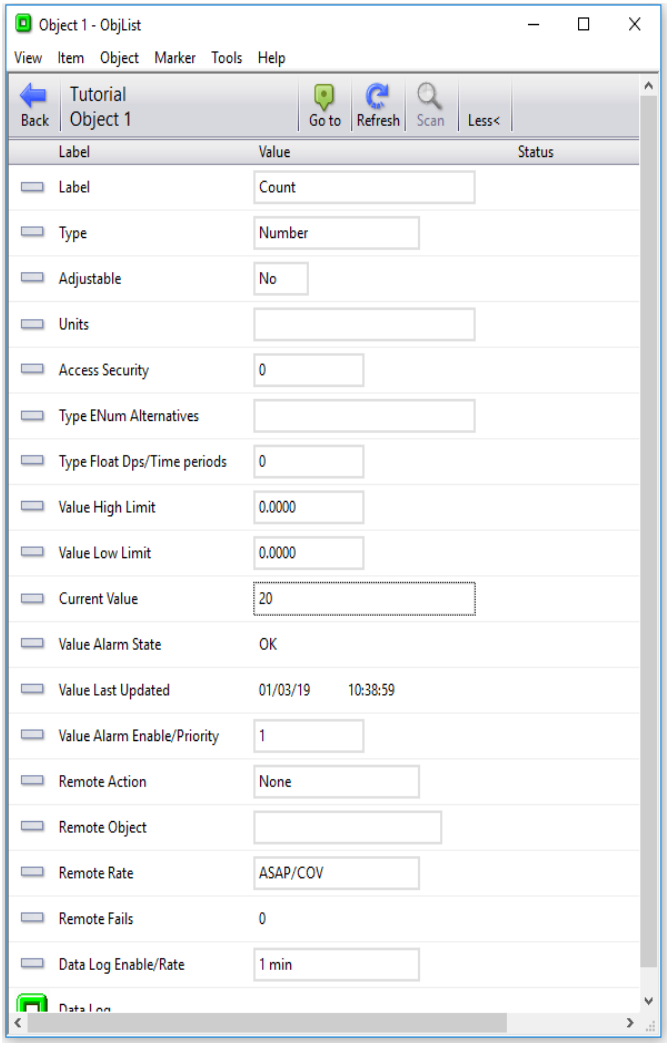
By default, each page has 16 objects. Each object has a label, value type, a value, and access rights, alongside other properties. If an object has no label, other tasks cannot access it. When ObServer shows Essential Data as a web page, this page/object structure provides hierarchy. However, the page/object structure has little effect on data collection.

It is possible to change the number of pages and objects in Observer. This may be useful if you wish to organise Essential Data values into groups for engineering and/or display purposes. It is not possible to change the maximum number of Essential Data objects in total; the maximum number is 1280 in ObServer.

If you require more than 1280 objects, consider using additional North devices, or North's Extra Data Driver. Please contact North Support for more information.

Simple Data Storage

ObServer 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 cause-and-effect strategy to read a heating setpoint from a particular 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 object, 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'
  - → Press **Refresh** to get ObView to re-read all the object values, and notice that 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 ObServer.

- To see which pages and objects are available within Essential Data, follow these steps:
  - → Press **Go to** to get to the top-level of ObServer
  - → Open **Essential Values** – this is the 'public' side of the database, and by default, has this friendlier label that appears on the web-pages
  - → **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. 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 ObServer, 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** screw 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, in reality this can cause communications bottlenecks both within ObServer, and on systems connected to ObServer. 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 Adjustable is set to 'Yes', 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 **Value 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.

Simply Writing?

Sometimes we just 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 M7002 of the North Training Pack.

You can adjust the value from the top-level of ObServer, by navigating down 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, 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

Communicating between ObServer PCs

Once ObServer is connected to a LAN, it can communicate across the network with other PCs running ObServer, as well as Commanders – transferring values, or sending alarms, for example. However, you must first set up ObServer's IP Alias table with the IP addresses and optional security-key. An alias is just a short sub-object reference for the device, rather than always having to use its IP address.

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

→ Open **Configuration**, **North IP Devices**

→ Set **Autofill List** to 'Scan now'– ObServer will find other North products on the LAN (using the same Subnet Mask), and then press **Refresh** – each will have been given a reference automatically. You can change this if you wish (keep the reference short) or disable ObServer talking to a particular device by deleting its reference

→ **Go to** the top-level of ObServer, and open **North IP Devices**

→ **Scan** to see the devices at the IP addresses– you should see ObSys Engineering software running on your PC

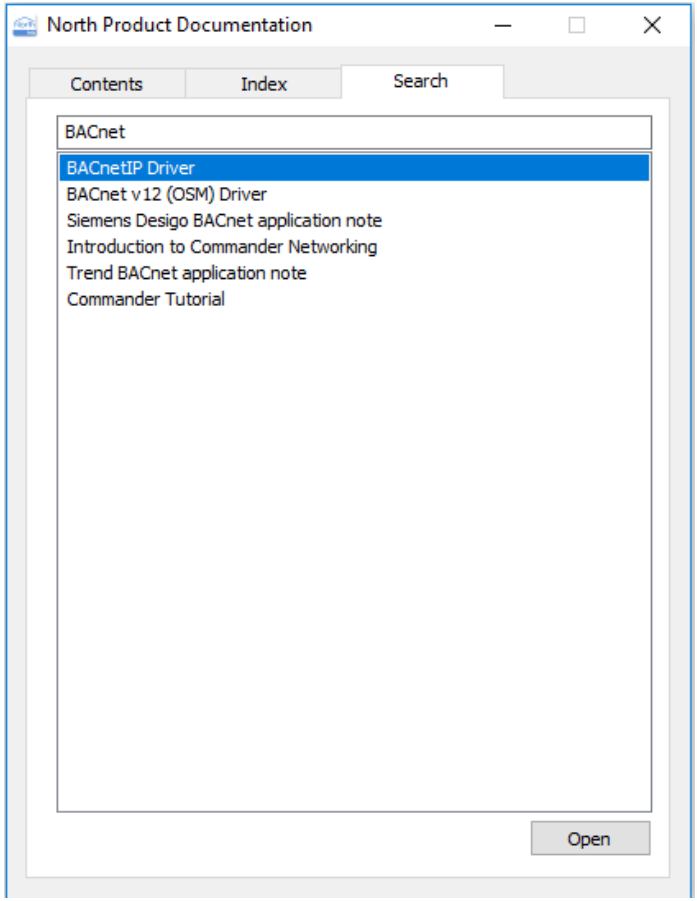
You can now navigate down these devices – the first time you access them you might need to scan and refresh to see the objects they contain. The power of XOM means you may use any of the objects you find in transfers, or other tasks within ObServer.

BACnet/IP and ModbusTCP

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

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

You need to refer to the document about the particular driver to learn how to set up and use it.



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 for** box, type 'BACnet' and press **Search** - the Help Assistant shows a list of documents that it has found containing the word 'BACnet' in the title

→ Double-click on the **BACnetIP Driver**

→ The Help Assistant minimises to the toolbar, and the document appears.

Loading from a Backup

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

For example, a backup from an Essential Data Page in Commander can be loaded into an Essential Data Page within ObServer.

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…**

→ Select the folder and '.obs' file, and press **Open** – the loading starts.

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

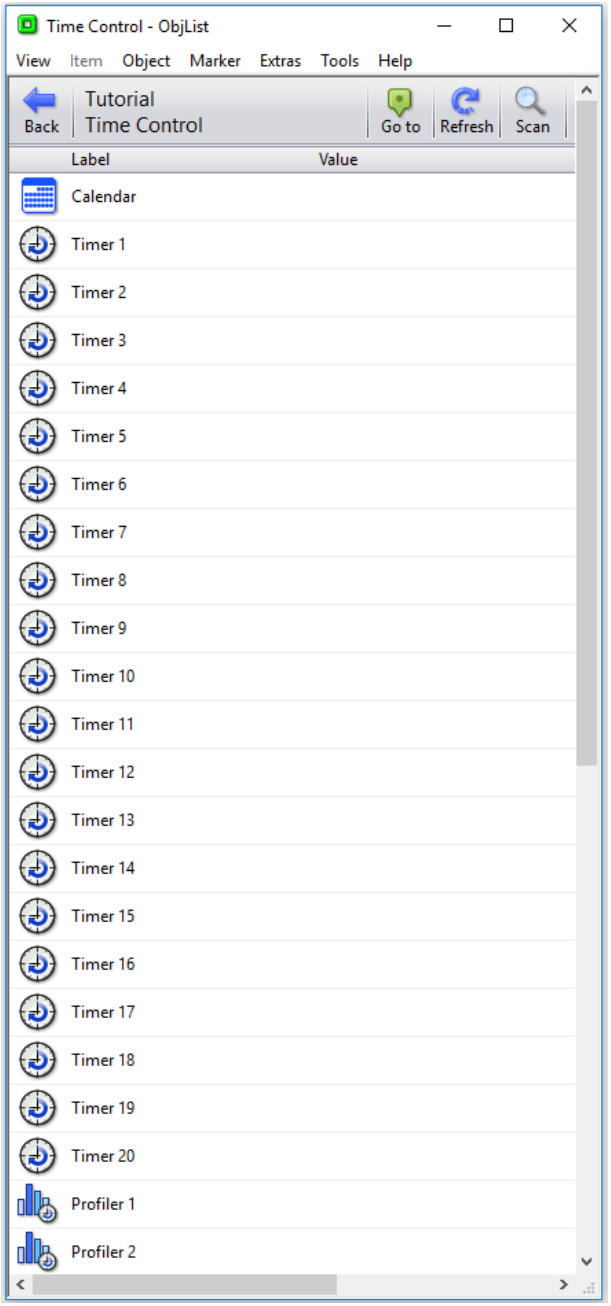
Controlling…

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 still keep the occupants happy. ObServer supports timed control with its Calendar, Timer and Profiler tasks.

ObServer'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.

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



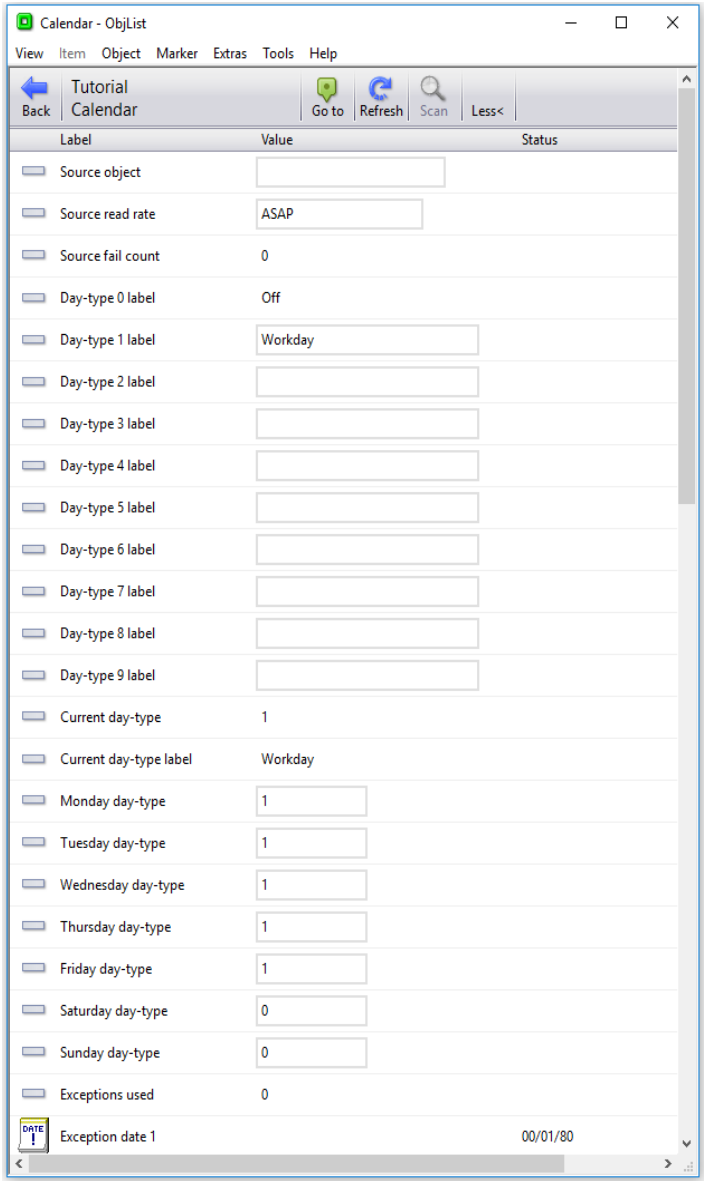
The Calendar and Today's Day-Type

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

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

If you are calculating the day-type in ObServer, 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 ObServer, and open **Time Control** - if necessary, press **Scan**, to see the calendar and a number of 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

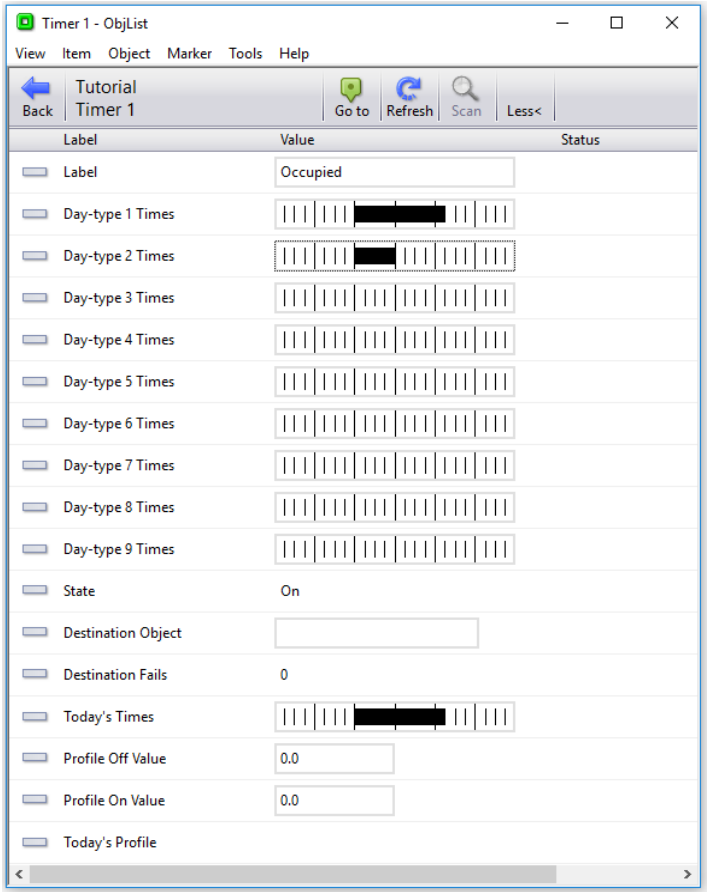
→ You can see the Current Day-Type near the top of the object list, along with a Current Day-Type Label

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. However, the user can view and adjust the Calendar using the web pages.

Timers

ObServer 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 for 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

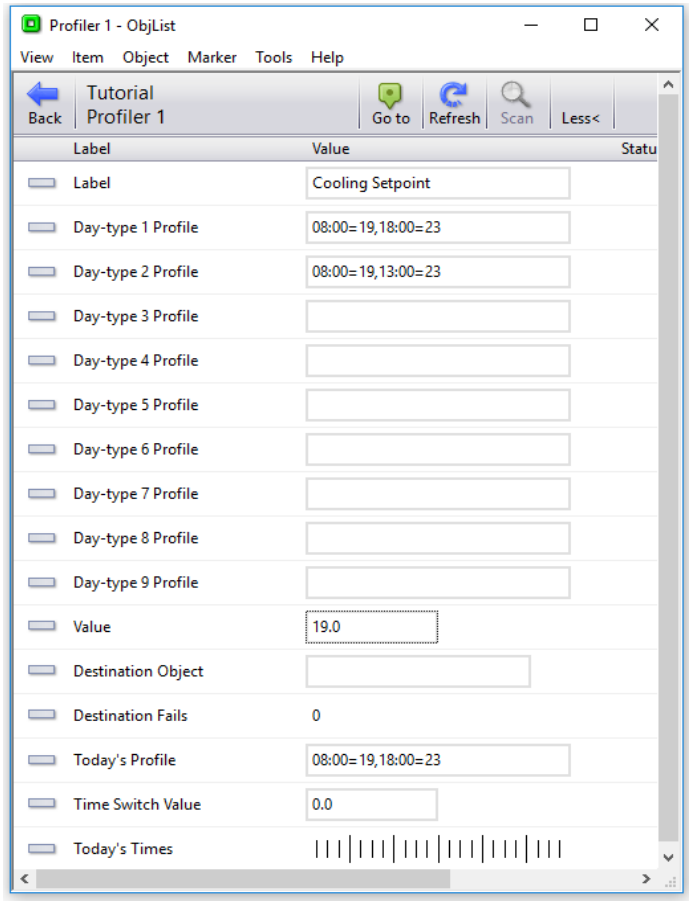
- To send the state to another object, follow these steps:
  - → Adjust the timer's **Destination Object** to the object reference 'S1.M0.DO2.S' (The second relay output of the Training Pack)
  - → After the next on/off time change, the relay turns on or off depending on the timer state – if the object 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

ObServer uses profilers to control analogue values over time. 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 and 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 that of the change-point.



- To set a profiler, follow these steps:
  - → Open **Time Control, Profiler 1** to see the profiles, labels, etc.
  - → Adjust **Label** to 'Cooling Setpoint'
  - → Adjust **Day-type 1 Profile**, and change the value to '8:00=21,17:00=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 the profiles using the web pages.

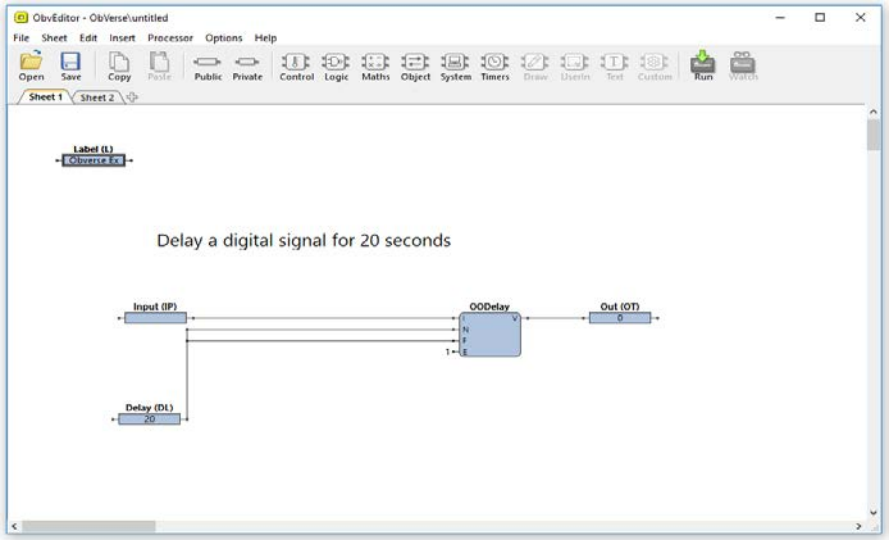
Introduction to ObVerse Programming

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 particular modules and links them together to perform a desired strategy.

ObVerse strategy runs in an ObVerse processor within a device. ObServer has four ObVerse standard processors, and whenever ObServer is running, 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.

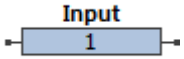
To start the ObVerse Editor, follow these steps:

→ 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 is 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 ObServer.

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

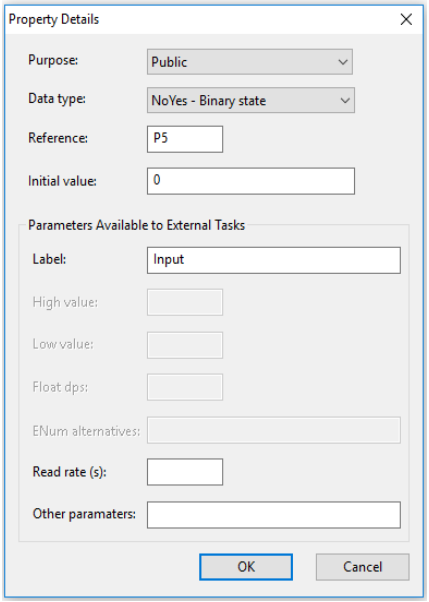
→ Hover the cursor over the Label property to display the tooltip for the property - this shows it's reference and the type of value associated

→ Double-click on the Label property, and set its **Value at Start** 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

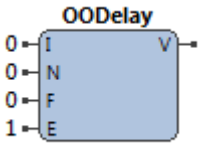
Each property you create holds a particular type of value – an Off/On value perhaps, or a Text label:

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

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!

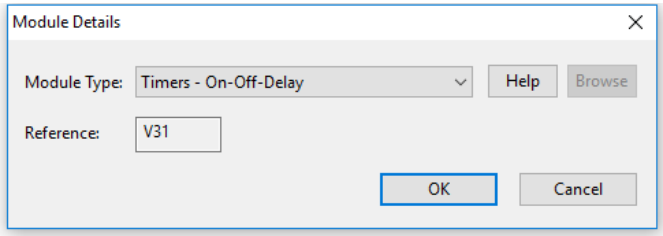
Modules have their inputs connected on the left-hand side, and the outputs on the right. It therefore makes sense that you place your ObVerse strategy inputs on the left-hand side of the page, and its outputs on the right-hand side – calculations will therefore occur naturally from left to right.



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 H**elp –** 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 ObServer'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
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

Timers

Counter
Latch
On-Off-Delay
Profile
Pulser

Control

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

Object

Alarm
Object-Read
Object-Write

System

System-Information

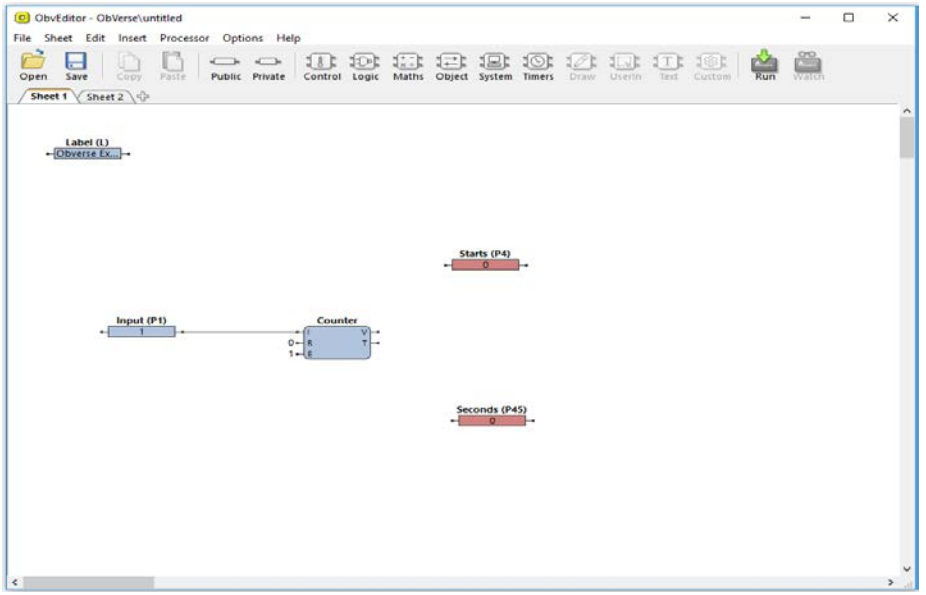
Module Inputs

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 of 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, 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 isn't clear which value it would use at which time. However, several different module inputs can connect to a single property and use the value.

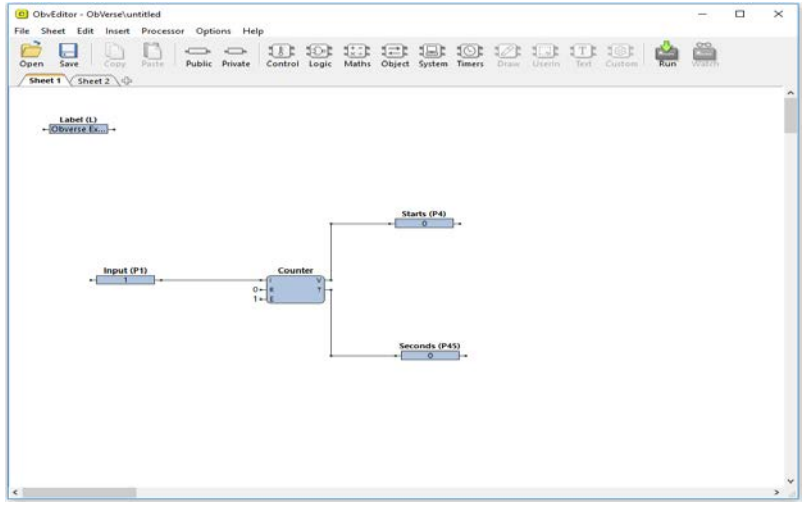
If you connect a module input to a property of the wrong type, then the module will attempt to convert between the two types – for example, if a number input is connected to a text property, the ObVerse Processor will try to extract a numeric value from the start of the text.

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



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 top area of a property, or the top area of 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 top 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 top 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

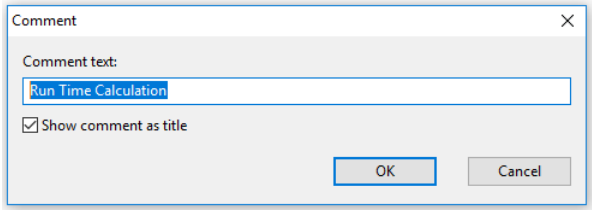
Adding comments to ObVerse strategy makes it easier for others (or even yourself) to see what you were trying to do with the ObVerse.

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 to Disk

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

💻 To save the current ObVerse strategy, follow these steps:

→ From the editor menu, select **File > Save** or click the disk button on the toolbar

→ If you have never given the ObVerse a filename, Save will ask you to specify one. For the **Filename**, enter 'ObVerseExample' and click **Save**

The ObVerse will be saved to disk 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.

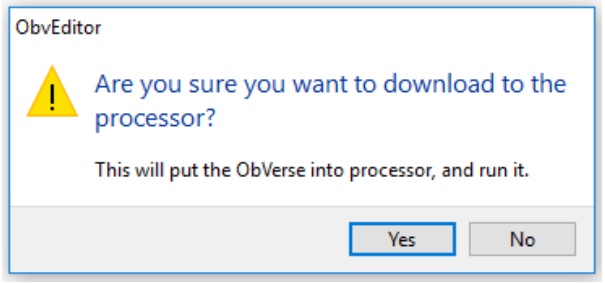
ObVerse Processors

Now you have written your first chunk of ObVerse strategy, you need to practice running and watching, and see the way other tasks within ObServer can access the properties within the ObVerse processor.

Running your ObVerse Strategy in Processor

While you are editing ObVerse strategy, the 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 am 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 ObServer, 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 ObServer as value objects. The ObVerse processor itself appears as a container object in the top-level of ObServer, and the input and output properties appear within that container object.

Other tasks within ObServer 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 ObServer

→ 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

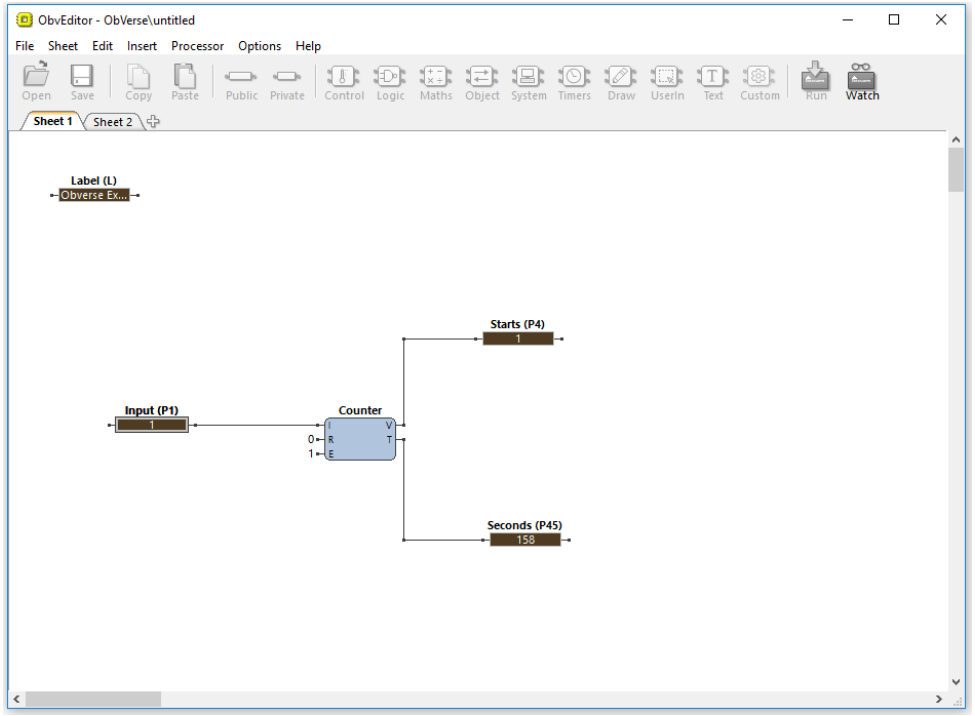
After downloading the ObVerse strategy, you can use the editor to watch what is happening in your ObVerse processor - the editor shows the live data from each property. When watching, it is only possible to adjust property values; objects cannot be moved, added or deleted. Be careful adjusting output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

Informing...

Simple Web Pages

Once ObServer is on a LAN, its built-in web-server (if enabled) automatically becomes available to others on the LAN. ObServer will serve the Essential Data pages and objects, showing the values it last received. Users can adjust these values, as well as view and adjust to Calendar and Timers.

To view ObServer'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 ObServer - or use the loopback address of '127.0.0.1' - and press **Enter**, Observer's web page then appears

Notice how ObServer uses the Essential Data pages (and objects) to build a menu structure on the left side of the page. The menu also allows access to the Calendar, and the Alarm History – we will cover these later in the tutorial.

ObServer builds the web pages with simple HTML, and so you can use any browser, including those on mobile phones and tablets.

(If you are browsing with a mobile device, such as a smart phone, you may choose the 'Mobile site' link, which displays pages more suitable for smaller screens)

It is possible to enable and disable certain web pages – this needs knowledge of ObServer's security features.

Controlling Access with the Security

Once ObServer is on a LAN, you must start to consider security – to control who can and who cannot view and modify values.

North products (including Commander and ObSys) use security databases 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', he/she presents 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.

ObServer's security database is called **Security Server**

By default, North pre-configure the database with a single User with ID 'admin', with a password of 'admin'. This user has full privileges in all areas. We recommend changing this!

Security Areas and Levels

Within each security system there are eight security areas. Security areas could be actual areas in a building, but are normally functional areas: for example, ‘environmental control’ and ‘North engineering’ areas would allow each user to have different privileges when controlling set points and engineering ObServer.

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, his/her 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=None, 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 – she can open the door. User B has privilege level 5 in area 1 – he cannot open the door. User C has privilege level 1 in area 1 – she cannot open the door.

The example continues: on the same ObServer, 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.

Enabling Users

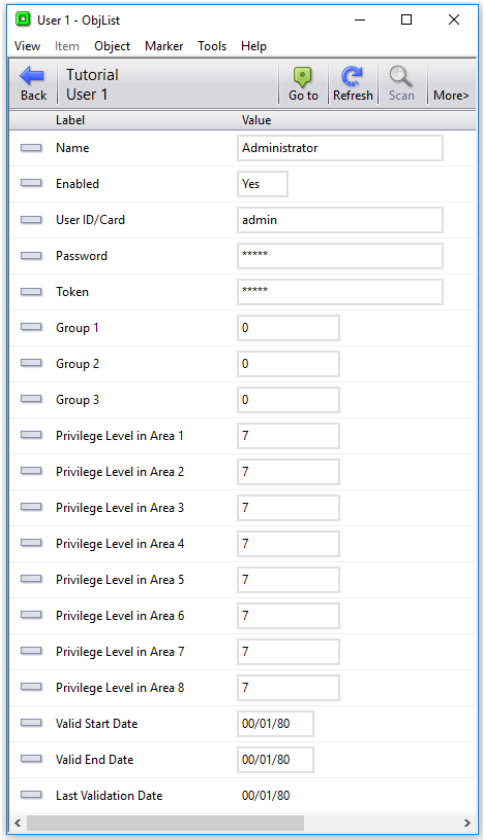
As well as holding the user's own privilege level in each of eight areas, the security database holds the user's name, and whether the user is enabled – if disabled, the user will never be validated.

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

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 he is a member of, along with his own privilege levels.

Each user has a UserID (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.

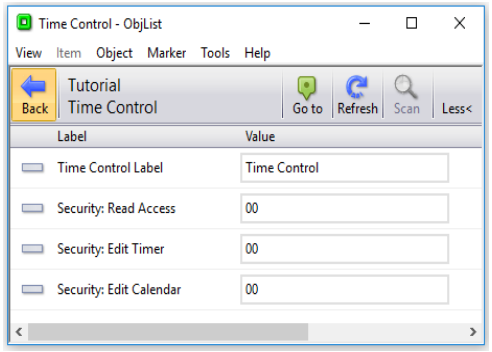
Whenever an item checks a coded token with the security database, the database remembers the date and time of the successful validation – this allows you to see when the user was last validated.



Specifying Access Security

Once you have added users to the database, the only thing left is to assign the area and minimum privilege level to control access to particular features.

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 particular feature - such as seeing the value, or adjusting the value, or viewing the page. The two-digit value is made up of the 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 value is 00, then no security checks are made.



Adding Users

Each security database has extra security to protect against any 'back-door' ObView access – called the Editor Password. You must enter this single password into the Editor Sign In object before any major changes can be made to the user information. Once signed in, you can change the Editor Password to something more memorable.

By default, the Editor Password is blank. We recommend you change this!

- To enable Editor Access, follow these steps:
  - → **Go to** the top level of ObServer, and open **Security Server**
  - → Set **Editor Sign In** to the current Editor Password – the default setting for this is blank! You can check editing is allowed – the **Editor Access Allowed** object will show 'Yes'

- To add a low-level user, follow these steps:
  - → Check the **Editor Access Allowed** object reads 'Yes', and open a free **User** entry – we will use **User 2**
  - → Set **Name** to 'Basic User', **Enabled** to 'Yes', and **UserID** to 'BU'. Leave the **Password** blank – in other words no password yet
  - → Set **Privilege Level in Area 3** to '7' – this is the one area where this user has power

- To add a medium-level user, follow these steps:
  - → Check that the **Editor Access Allowed** object reads 'Yes', and navigate to a free **User** entry – we will use **User 3**
  - → Set **Name** to 'Medium User', **Enabled** to 'Yes', and **UserID** to 'MU'. Leave the **Password** blank – in other words no password yet
  - → Set **Privilege Level in Area 1** to '7', and **Privilege Level in Area 2** to '7'
  - → Set **Privilege Level in Area 3** to '4' – this is the one area where this user has only medium power

Summary of User and Privileges set up so far in the tutorial:

User	Area1	Area2	Area3	Area4	Area7	Area6	Area7	Area8
admin	7	7	7	7	7	7	7	7
BU	0	0	7	0	0	0	0	0
MU	7	7	4	0	0	0	0	0

Enabling Access Security on Web Pages

Now you have some users, you can enable Web Server access security, which will force the web server to request the UserID and password from the user (done within the user's browser).

💻 To enable general security on ObServer's web server, follow these steps:

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

→ Set **Method** to 'North User Token'. This means a User's details are checked against the local Security Server when accessing values on the local Web Server.

→ Test by using your browser to view the ObServer's web page – you should need to sign in now – use the new 'MU' with a blank password

Important Point – your browser will remember your UserID and password, so changing these when testing can be quite awkward. The most common way is to sign out from ObServer, and close and restart your browser.

💻 To sign out of ObServer and get your browser to forget the UserID it is remembering, follow these steps:

→ On the web page in the browser, select the option at the top left of the screen, shown as the current user's name **Medium User** – this will show the Change Password option and the Sign out option

→ Select **Sign Out**, which forces the browser to re-request the user and password

→ Restart the browser

You should now be able to sign in as either 'BU' (basic user) or 'MU' (medium level user), or as the default 'admin'. By default, users can only view calendars and timers if they have privilege level 1, or above, in area 1 – BU has no privileges in area 1, so cannot view calendars.

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 particular page – we will use **Zip Input 2**

→ Set **Access Security** to '31' – this means that our example users 'BU' and 'MU' can view the page

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

→ Open the required object – we will use **Test Changes**, object **Limited** – an adjustable value we set up early

→ Set **Access Security** to '35' – this means that our example user 'BU', who has level 7 in area 3 can adjust the value, but user 'MU' who has level 4 in area 3 cannot adjust the value

Once Essential Data pages have Access Security set, the web server builds the left-hand menu for the current user – not showing pages that the user cannot access with his/her privileges.

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

Action	Access Level	User 'BU'	User 'MU'
View Calendar	11	No	Yes
Edit Calendar	14	No	Yes
View Page 1	31	Yes	Yes
Adjust Page 2 Object 1	35	Yes	No

Alarms

Normally, 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 an object's value, it is better that the object tells the user when it changes. North call these object-to-user 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 – this is 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 alarms are sent between object and user, they are sent in text form, with the | separating the different parts.

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

The Point Name is normally 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) shows 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 delivery, because each user wants the messages delivered in a different way, to different places.

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

ObServer delivers all alarms, including those generated by Essential Data, in the same way.

By default, 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 destination, which is set up by default, is the Alarm History. Other common destinations are the Alarm Stores (seen in the next section of the tutorial) and the Alarm Emailer.

The engineer can set up other destinations, including Commanders and ObSys software, SMS, and printers.

Alarm History

ObServer has an Alarm History, to which the Alarm Delivery module sends alarms. The Alarm History is a rolling record of the last 500 or so alarms. 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 ObServer's **web page** in your browser. The top page may show the last few alarms that have occurred recently

→ Select **Alarms** from the menu on the left, and from within its sub-menu, select **Alarm History**

→ Select the order that the server sorts the alarms using the **Change Order** drop-down box

→ Press **Clear List** to wipe all alarms from the history

Generating Alarms using Essential Data

Some external systems connected to ObServer send alarms or 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 alarm messages. As seen before, Essential Data is set up to read the values of external systems; by setting value high and low limits, Essential Data 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, Page 1**, **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 value is 0 it is ok, when 1 it goes out-of-limits
  - → Set **Remote Object** to 'S1.M0.DI2.S' – i.e. the state of the digital input
  - → 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 Current Value should change to the correct state
  - → 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 **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 the Alarm Delivery, which delivers the alarm to the Alarm History,
  - → On the web browser, view the ObServer's Alarms page to see the alarm

Generating Alarms from Zip Modules

Using a Zip system as an example external system, you can see how alarms from Zip work. 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 ObServer, and open **Zip System**, **M7002A v10 (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 flash to indicate 'loss of comms', and the alarm will be generated, and be delivered to the alarm history – you can see this on the Alarms web page.

Routing and Filtering

By default, ObServer delivers all alarms to one destination – the 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.

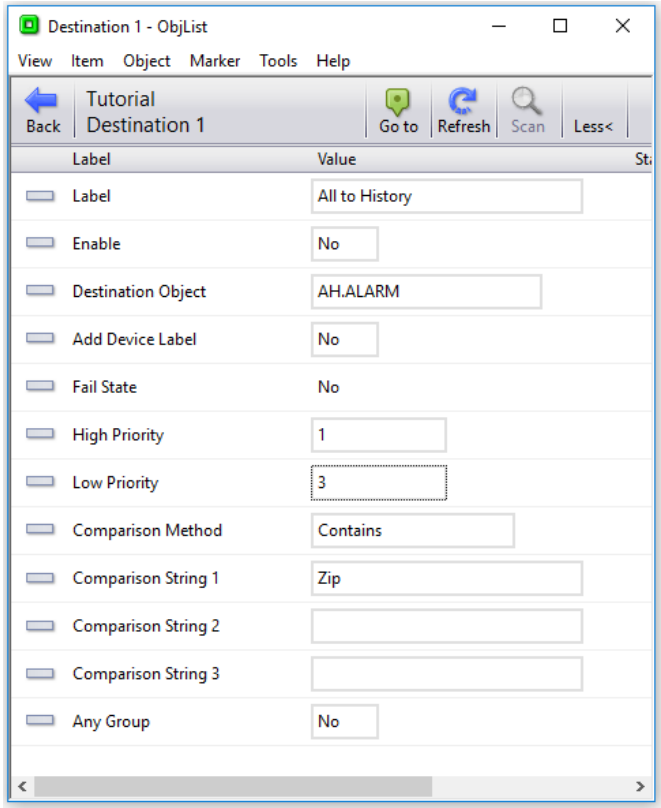
To send only alarms with priority 1 to 3 to the Alarm History, follow these steps:

→ **Go to** the top level of ObServer, and open **Alarm Delivery**, **Destination 1**

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

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



Alarm Stores

Alarm Stores are another type of destination that Alarm Delivery could send alarms to. ObServer has four stores available, which may be used to hold alarms of different types, ready for the end-user to see, action and delete. An Alarm Store is only a storage device – other things are used to view the alarms currently within the store, such as Alarm Manager application or a web-browser.

As new alarms are sent to an Alarm Store, it holds them as unacknowledged 'sounding' alarms – Alarm Manager can even play sounds to attract the end-user's attention (Alarm Manager is covered in ObSys Tutorial - Part 2).

The end-user can mark the alarm as acknowledged and 'silenced'. The time this occurred is recorded, along with the name of the current user (which it gets from the Security Centre.)

When the end-user is happy the alarm is no longer needed in the store, they can 'delete' it. This removes it completely from the Alarm Store.

If the end-user doesn't delete alarms, eventually the store will fill with alarms, and will not accept any more new alarms – the end-user therefore must accept responsibility for deleting alarms when they are no longer needed.

Alarm Store works particularly well when several viewers are used at the same time – when one user 'silences' an alarm, the change automatically appears on the other viewers.

💻 To have alarms sent to Alarm Store 1, follow these steps:

→ Navigate to the **top-level** of ObServer, and open **Alarm Delivery**, **Destination 2**

→ Set **Label** to 'All to Alarm Store 1', **Enable** to 'Yes'

→ Set **Destination Object** to 'AS1.ALARM'

Viewing an Alarm Store using web pages

One way of viewing, accepting, and deleting alarms in an Alarm Store is to use ObServer's web pages.

💻 To view an alarm in a store, follow these steps:

→ First, switch the digital input 2 as we did before, and cause the alarm to be sent to the Alarm Delivery and onwards to the Alarm Store and the Alarm History

→ Open the ObServer's **web page** in your browser. The top page may show the last few alarms that have occurred recently

→ Select **Alarms** from the menu on the left, and from within its sub-menu, select **Alarm Store 1**

💻 To acknowledge that you have noticed the alarm, follow these steps:

→ Press **Silence** on the alarm to be silenced – and the little sounder icon disappears from the alarm

→ Any other users currently viewing this alarm store will also see that the alarm has been silenced

💻 To delete the alarm from the Alarm Store, follow these steps:

→ Press **Delete** on the alarm to be deleted – and the alarm disappears from the list. The alarm will be placed in the store's archive if available.

A user can leave a text-note against an alarm – perhaps indicating why the alarm occurred. Other users that view the same alarm store will see the note against the alarm.

The ObSys Tutorial - Part 2 shows how to perform these functions using Alarm Manager

Other Alarm Destinations

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

Email

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

Printer

North has two drivers for printing alarms. The 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.

The AlmPrint driver supports alarm printing to a **Windows page-based printer** – it can send alarms to a local or networked printer. It can be set up to print anything from a single alarm to a whole page of alarms. It also has an object that will force the module to print the entire contents of its alarm buffer.

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. The GsmSms driver also supports bi-directional communications using text messages, allowing a user to both request current information, as well as change certain objects.

Telnet

Sometimes it is necessary to talk to ObServer without using ObView or web pages – ObServer has a Telnet server that can provide simple text-based access to any object values within, or outside, ObServer. By default, Telnet is enabled.

ObServer supports several services within the Telnet session – Query/Response, and IP-Configuration.

- To enable Telnet client on Windows Vista and Windows 7, 8 and 10, follow these steps:
  - → From Windows **Control Panel**, select **Programs** (or Programs and Features in Vista) then **Turn Windows features on or off**
  - → Select the check box next to **Telnet Client** to enable it, then click **OK**
- To check the state of ObServer's Telnet service, follow these steps:
  - → Open **Configuration**, **Telnet Setup**
  - → Check the **User/Password** is set to 'NBT' – up to 7 letters only (this object is configurable and if changed should be noted for the opening of a telnet session)
  - → Check **Telnet Enabled** is set to 'Yes'

IP Configuration

The Telnet client application runs within the Windows Command Prompt window. TELNET.EXE takes one parameter – the IP address of the Telnet server.

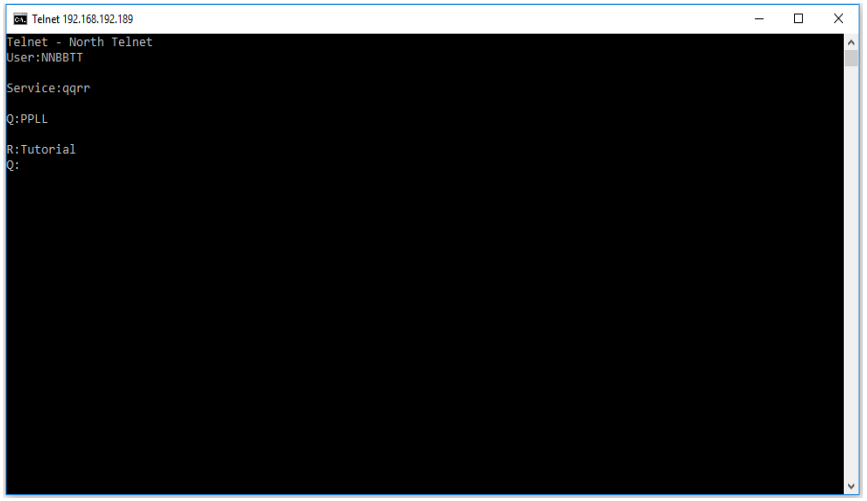
- To connect to ObServer's IP Configuration telnet service, follow these steps:
  - → Within Command Prompt window, type the command line:

```
telnet 127.0.0.1
```

  - → At the User: prompt, type 'NBT', and press ENTER
  - → At the Service: prompt, type 'ipc', and press ENTER – ObServer responds with the current IP configuration
  - → At the Service prompt, press ENTER – this means you have no more service requirements – and ObServer closes the telnet session

Query/Response

ObServer's Telnet server supports a simple query/response protocol. You can use this manually, or can be it used from an external device.



To view the use the Query/Response service, follow these steps:

→ Within Command Prompt window, type the command line:

```
telnet 127.0.0.1
```

→ At the User: prompt, type 'NBT', and press ENTER

→ At the Service: prompt, type 'qr', and press ENTER – this enters the Query-Response service

→ At the query prompt Q: type 'PL' and press ENTER – this asks for the value of the ObServer object PL – the ObServer responds 'R:<value>', and a new query prompt

→ At the prompt, type 'O.PI.RC' and press Enter – this asks for Configuration-Platform Information – Reset Count. ObServer sends the response with the current Reset Count

→ At the prompt, type 'O.PI.RC=0' and press enter – this asks for the same value to be adjusted to 0

→ At the prompt, type 'O.PI.RC' and press Enter – to see the new value

→ At the prompt, press ENTER – this states that you have no more queries to make

→ At the Service prompt, press ENTER – this indicates you have no more service requirements – and ObServer closes the telnet session

Notice that you can both read and adjust values.

Next Steps

In this tutorial you have learnt about the core of North's ObSys software, ObServer. You have used ObSys software to configure it. Following the steps, you have also configured the range of integration and control features available.

You will find these same integration and control features available in another North device, Commander.

Next, learn more about North's ObSys applications by following the steps in the *ObSys Tutorial – Part 2*.

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 31/01/2020.