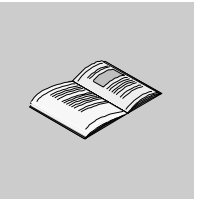
Unity Studio Manager

Step-By-Step Description

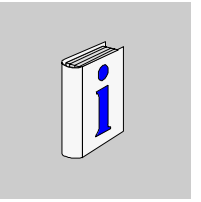
02/2005 V 2.0.1

33003247.00

Table of Contents



About the Book



At a Glance

Document Scope

This documentation contains a step-by-step description for creating a plant in Unity Studio Manager.

Validity Note

This document applies to Unity Studio Manager V2.0.1 using Microsoft Windows 2000 or Windows XP professional, as well as Microsoft Visio 2002 or Visio 2003 and Unity Pro V2.0.2 and XBT-L1000 V4.40, OFS V3.2 and Power Suite V1.5.

Related Documents

Title of Documentation	Reference Number
Unity Pro documentation	-
Unity Studio Manager User Guide	-

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

Step-By-Step Description

1

Introduction

Overview

This chapter contains a step-by-step description for creating a plant in Unity Studio Manager.

What's in this Chapter?

This chapter contains the following topics:

General

Step by Step

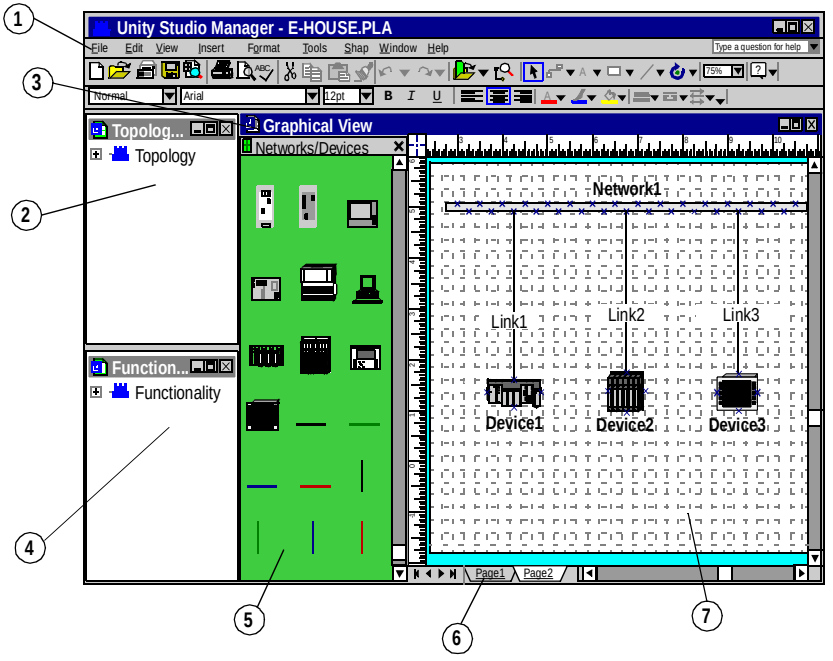
Unity Studio Manager is a tool for supporting the configuration of distributed automation solutions.
A Step-by-Step description of plant creation in Unity Studio Manager now follows.
A simple task has been selected to help your understanding.

Start the Unity Studio Manager

Start the Unity Studio Manager. Select **File** → **New...** to create a new plant and name it E-HOUSE.PLA.

Screen structure

The following illustration shows a typical screen construction for Unity Studio Managers after a new plant has been created. The screen construction can be changed using standard Window functions.



1 Main menu bar

2 Topological view

3 Graphical view

4 Functional view

5 Stencils

6 Drawing sheet tab

7 Drawing sheet

Task

Definition of the Task

The following task should be solved:

- A room is to be heated with warm air using a fan.
- The heating is controlled by a temperature sensor.
- The controller settings are made using an operating panel.
- In addition, the heating must be switched off if the window is open.

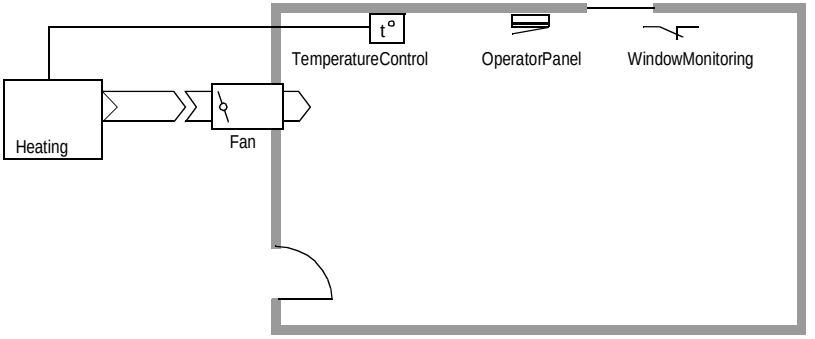
Optional: Sketch the plant

The plant sketch does **not** have to be made in Unity Studio Manager in the **Graphical view**. It is only meant to provide a visualization of the task and can be made in other programs or on paper.

Step	Action
1	Click in the **Graphical view** with the right mouse button on the drawing sheet (**page 1**) tab, and change the name to **Sketch**.
2	Create the sketch below on the **Sketch** sheet.

Optional: Sketch the task

The following shows a sketch of the task.



Structuring the Task in the Functional View

Functional view

You can display the distributed automation system independently from the installed components using the **Functional view**. The distributed automation system is structured in units and displayed in a hierarchical directory tree.
The **Functional view** helps the user divide their complete task into subtasks.

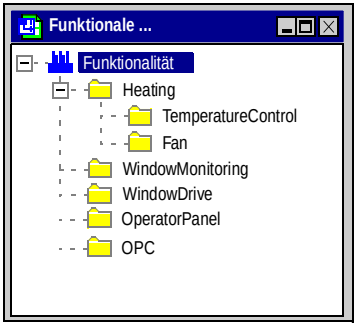
Create a Tree Structure

Create a directory tree for your plant.

Step	Action
1	Click in the **Functional view** with the right mouse button on the **functionality** root directory.
2	Add a functional module (FE) via **New** and call it `Heating`. **Note:** Note that when naming functional modules using the naming conventions defined in **Extras** → **Options...** ; an automatic check is carried out during the entry. More information about the naming conventions can be found in the *Online Help, Tools, Options* in the General tab.
3	Click on the right mouse button on `Heating`.
4	Add a Sub FE via **New** and call it `TemperatureControl`.
5	Click on the right mouse button again on `Heating`.
6	Add a second Sub FE via **New** and call it `Fan`.
7	Repeat steps 1 and 2 and enter the name `WindowMonitoring`.
8	Repeat steps 1 and 2 and enter the name `WindowDrive`.
9	Repeat steps 1 and 2 and enter the name `OperatorPanel`.
10	Repeat steps 1 and 2 and enter the name `OPC`.

Representation of the Tree Structure

The plant directory tree should look like this in the **Functional view**.



Drawing the Plant in the Graphical View

Graphical view

In the **Graphical view**, configure the physical structure of your plant on one or more drawing sheets by placing the networks, devices and connections from the respective stencils on these drawing sheets using the **Drag&Drop** method or the **Copy/Paste** function.
If multiple drawing sheets are used, the same device or network instance can also be used on more than one drawing sheet.

Topological view

Parallel to the **Graphical view**, the plant is shown in the **Topological view** as a directory tree. The exact contents of the plant file are shown in the directory tree. In this view, only changes can be made in an existing topology, such as copying and renaming objects or deleting objects from the plant. Additionally, objects from the directory tree can be added to the drawing sheets in the **Graphical view** using the **Drag&Drop** method.

> **Note:** When inserting an object from Topological View into Graphical View using **Drag&Drop**, there is a "forbidden" zone surrounding the Link-Shape. The object cannot be inserted there, and a message appears. You will have to place the object in an open area.

Create New Drawing Sheet

Create a new drawing sheet.

Step	Action
1	Right click in the **Graphical view** on a tab in the drawing sheet.
2	Using the option, **Add page...** create a new drawing sheet with the name **Network**

Draw the Plant

Now draw your plant in the **Graphical view**. The **Graphical view** from your plant is shown below.

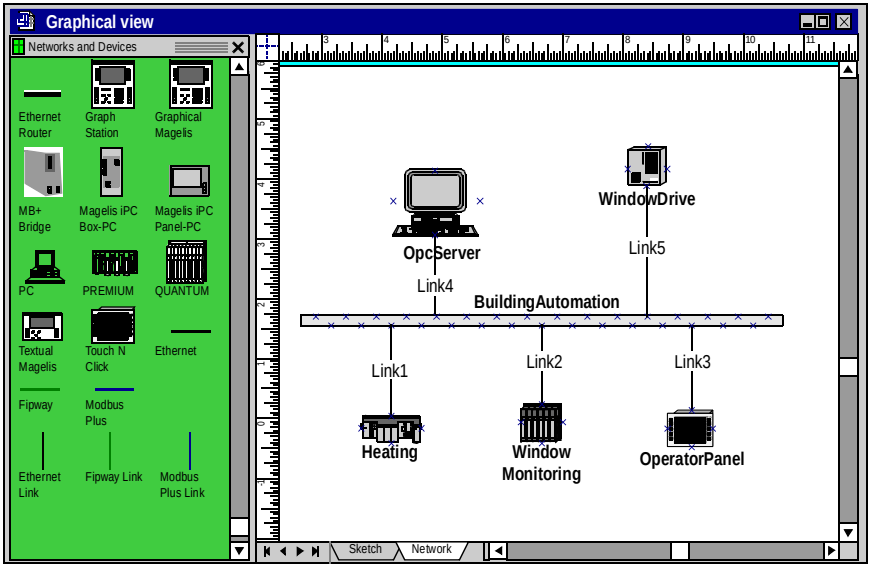
Step	Action
1	Click in the **Networks and Devices** stencil on the horizontal line of the **Ethernet** network, hold the mouse button down and drag the line on to upper area of the **Network** drawing sheet.
2	Double click on the **Ethernet** network in the drawing sheet and change the name to `BuildingAutomation`.
3	Click in the **Networks and Devices** stencil on a **PREMIUM** controller, and drag it to the lower area of the **Network** drawing sheet.
4	Double click on the **PREMIUM** controller in the drawing sheet and change the name to `Heating`.
5	Drag a vertical **EthernetLink** to the drawing sheet above on the controller until the connection of the line and the controller is indicated with a red square.
6	Click on the vertical **EthernetLink**. Click on the upper handle and drag to the horizontal **Ethernet** network. **Result:** A connection is now made between the controller and the Ethernet. **Note:** If you drag the controller in step 3 directly to the horizontal line of the **Ethernet** network, the controller is placed close to the line and is automatically connected with an **EthernetLink**.
7	Drag a **QUANTUM** controller directly to the **Ethernet** network. The controller is placed close to the horizontal network line and is automatically connected with an **EthernetLink**.
8	Double click on the **QUANTUM** controller and change the name to `WindowMonitoring`.
9	Repeat steps 7 and 8 with an **ATV58F** drive and enter the name `WindowDrive`.
10	Repeat steps 7 and 8 with a **TouchNClick** operator panel and enter the name `OperatorPanel`.
11	Repeat steps 7 and 8 with a **PC** and enter the name `OpcServer`.

Delete Objects

If an object is deleted from the **Graphical view** using the **Del** key, it is no longer visible but remains a component of the **Topological view**.
If an object is to be completely deleted from the plant, it must either be deleted in **Topological view** or in **Graphical view** by **right clicking on** → **Delete from Plant**.

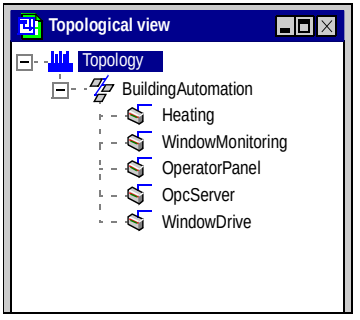
Graphical View Display

This is how the Graphical view of your plant should look.



Topological View Display

This is how the Topological view of your plant should look.



Edit Object Properties

Object Properties Tab

The object properties sheet enables you to display and configure object properties. The properties can differ according to the object.

Opening the Properties Sheet

Select an object in the **Graphical view** or the **Topological view,** mark an object and by **right clicking and selecting** → **Properties...** open the object properties sheet.

Editing the Object Properties

Now edit the object properties for your plant.

Step	Action
1	Right click on the Network `BuildingAutomation` and by selecting **Properties...** , open the object properties sheet.
2	Double click on the **X-Way Network Number** row in the **value** column and enter `1`. Enter the following values: ● **Default Subnet Mask**: `255.255.255.0` ● **Default Gateway Address** `1.0.0.255` Close the window using **OK**.
3	Right click on the `OperatorPanel` and by selecting **Properties...** open the object properties sheet.
4	Double click on the **Type** in the **Value** column and select the drop-down menu `XBT-FC064610`. Close the window using **OK**.
5	Right click on the `Heating` and by selecting **Properties...** open the object properties sheet.
6	Double click on the **Type** in the **Value** column and select the drop-down menu `TSX P57 204MV01.00`. Close the window using **OK**.
7	Open the object properties tab for the connection between `Heating` and `BuildingAutomation` and enter the following values: ● **Name**: `LinkHeating` ● **Type**: `TCP/IP 10/100 regular PREMIUM` ● **Address**: `1.0.0.1` ● **X-Way Station number**: `1` Close the window using **OK**.
8	Right click on the `WindowMonitoring` and by selecting **Properties...** open the object properties sheet.
9	Double click on the **Type** in the **Value** column and select the drop-down menu `140 CPU 651 60V01.00`. Close the window using **OK**.

Step	Action
10	Open the object properties tab for the connection between `WindowMonitoring` and `BuildingAutomation` and enter the following values: • **Name**: `LinkMonitoring` • **Type**: `TCP/IP 10/100 regular QUANTUM` • **Address**: `1.0.0.2` Close the window using **OK**.
11	Open the object properties tab for the connection between `OpcServer` and `BuildingAutomation` and enter the following values: • **Name**: `LinkOpc` • **Type**: `Ethernet PC Link` • **Address**: `1.0.0.3` • **Specific ID**: `1` • **X-Way Station number**: `3` • **Host Name**: `<Your Hostname>` Close the window using **OK**.
12	Right click on the `WindowDrive` and by selecting **Properties...** open the object properties sheet.
13	Double click on the **Type** line in the **Value** column and select `ATV58F*U18N4` from the drop-down menu. Close the window using **OK**.
14	Open the object properties tab for the connection between `WindowDrive` and `BuildingAutomation` and enter the following values: • **Name**: `LinkDrive` • **Address**: `1.0.0.4` Close the window using **OK**.

Missing Object Properties

Note: Up to this point, all of the object properties for your plant have not been set. This is included to help understand the description of Unity Studio Manager automatic error analysis, which is discussed later in this step-by-step description.

Configure and Assign Applications

Application view

The Application view provides the following features:

- Creation of applications.
- Assignment of devices to applications.
- Creation of project files for individual applications.
- Assignment of project files to the individual applications.

Opening the Application View

The application view is opened using the main menu bar **View** → **Application View**.

Creating Applications

Now perform the following steps for your plant in the **Application view**.

Step	Action
1	Click in the empty field under**Name** and enter `Heating`.
2	Click in the empty field under **Type** and select `UnityPro` from the drop-down menu.
3	Click in the empty field under **Device** and select `Heating` from the drop-down menu.
4	Right click in the empty field under **File** and select**create "Heating.STU"**.
5	Wait until the file name and path specification appears in the field. **Note:** This process can take some time.
6	Now repeat steps 1 to 5 for `WindowMonitoring`.
7	Click in the empty field under**Name** and enter `OperatorPanel`.
8	Click in the empty field under **Type** and select `XBT-L1000` from the drop-down menu.
9	Click in the empty field under **Device** and select `OperatorPanel` from the drop-down menu.
10	Right click in the empty field under **File** and select **create "OperatorPanel.DOP"**.
11	Wait until the file name and path specification appears in the field. **Note:** This process can take some time.
12	Click in the empty field under **Type** and select `OFS` from the drop-down menu.
13	Click in the empty field under **Device** and select `OpcServer` from the drop-down menu.
14	Click in the **Name** field and change it to `OpcApplication`.
15	Open the properties sheet for this application using the right mouse button.
16	Double click on the **Server Location** line in the **Value** column and select `<YourHostname>` from the drop-down menu. Close the window using **OK**.
17	Select the application `WindowDrive`.
18	Right mouse click in the empty field in **File** and select **Create "WindowDrive.58f"**.
19	Wait until the file name and path specification appears in the field. **Note:** This process can take some time.

Attach file...

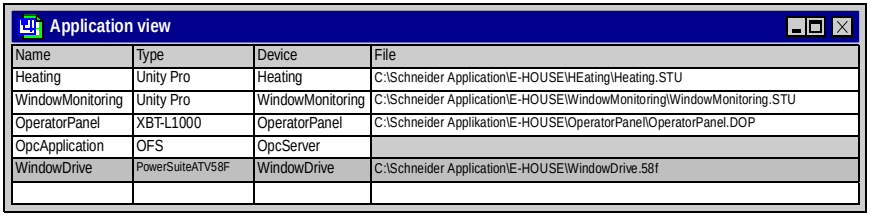
If a project file (*.STU, *.DOP, etc.) already exists, it can be selected using **Attach file...** .

Create file...

Use **Create file...** to create the file after the names and storage location are entered.

Application View Display

This is how the Application view of your plant should now look.



Name	Type	Device	File
Heating	Unity Pro	Heating	C:\Schneider Application\E-HOUSE\HEating\Heating.STU
WindowMonitoring	Unity Pro	WindowMonitoring	C:\Schneider Application\E-HOUSE\WindowMonitoring\WindowMonitoring.STU
OperatorPanel	XBT-L1000	OperatorPanel	C:\Schneider Applikation\E-HOUSE\OperatorPanel\OperatorPanel.DOP
OpcApplication	OFS	OpcServer	
WindowDrive	PowerSuiteATV58F	WindowDrive	C:\Schneider Application\E-HOUSE\WindowDrive.58f

Note: The path specification for the project files depends on your Unity Studio Manager default settings.

Assign Functional Modules to Applications

Functional view

Earlier in the chapter, *Structuring the Task in the Functional View, p. 11* you structured your entire task in functional modules.
Now you must assign the functional modules to the corresponding applications, so that the correct structures are created when generating in the respective programs (e.g. Unity Pro or XBT-L1000).

Assigning Applications

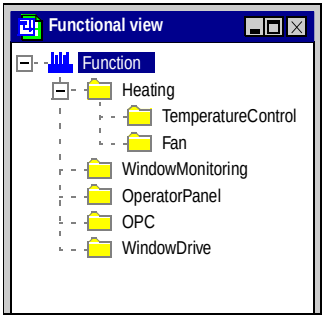
Assign the functional modules to the applications

Step	Action
1	Expand the directory tree in the **Functional view**.
2	Right click on the `Heating` and by selecting **Properties...** open the properties sheet for this functional module.
3	Double click on the **Application** line in the **Value** column and select `Heating` from the drop-down menu.
4	Close the window using **OK**.
5	**Result:** The icons for `Heating`, `TempertureControl` and `Fan` change as shown below.
6	Also repeats steps 1 to 4 for `WindowMonitoring`, `OperatorPanel`, `OPC` and `WindowDrive`.
7	**Result:** The icons for `WindowMonitoring`, `OperatorPanel`, `OPC` and `WindowDrive` change as illustrated below.

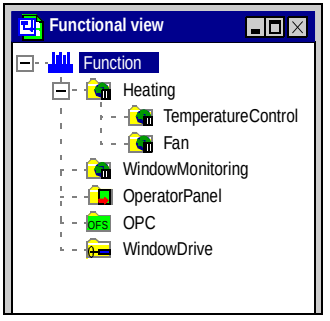
Representation of the Tree Structure

This is how your directory tree should look before and after making the assignments.

Before Mapping



After Mapping



Configuring communication variables

Communication View

In **Communication View**, all definitions are made for variables that should be communicated. This view is opened using the main menu bar **View** → **Communication view**.
The following variables can be configured in the **Communication View**:

- Publisher and Subscriber Variables
- Modbus Client and Modbus Server Variables

Configuring Global Data

Global data

Global data is identified as the set of cross-system variables, in which data is exchanged between the individual applications. This exchange is carried out cyclically. Different data types are provided for this.
Global data is used to make the connections between the application variables, which send their contents (Publisher), and the variables in one or more other applications which receive the contents of these variables (Subscriber).

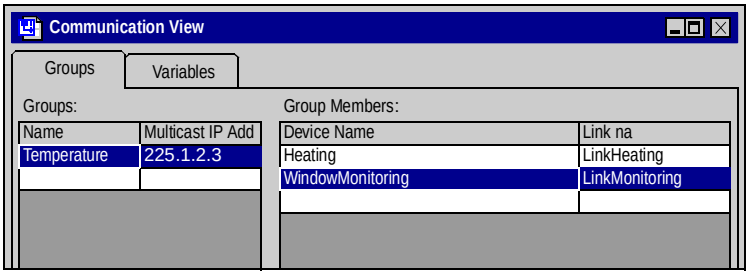
Configuring Groups

Now configure a group of variables for your plant in the **Communication View**.

Step	Action
1	Open the **Communication View** and click on the **Groups** tab.
2	Click in the empty field under **Name** and enter `Temperature`.
3	Click in the empty field under **Multicast IP address** and enter `225.1.2.3`.
4	Now assign these group devices as group members. Click in the empty field under **Device name** and select `Heating` from the drop-down menu. **Note:** Only devices / links which also support global data are offered in the **Device name** and **link name** lists.
5	Click in the empty field under **Link name** and select `LinkHeating` from the drop-down menu.
6	Click in the next empty field under **Device name** and select `WindowMonitoring` from the drop-down menu.
7	Click in the next empty field under **Link name** and select `LinkMonitoring` from the drop-down menu.

Appearance of the Group tab

This is how the **Group** tab should now look in your plant.



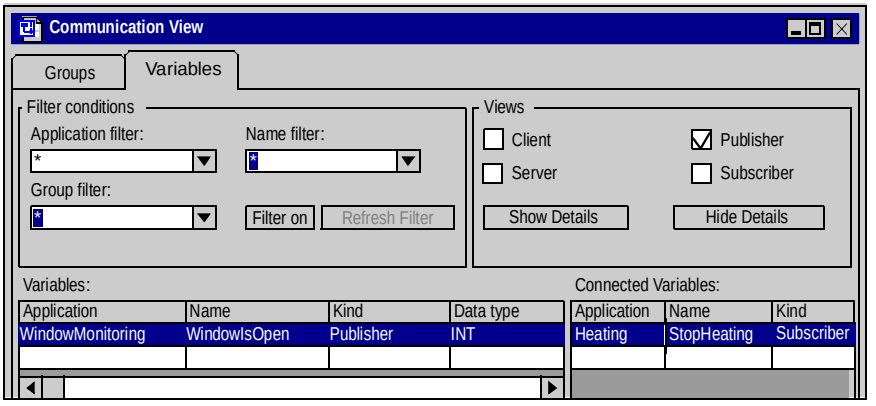
Configuring Global Data

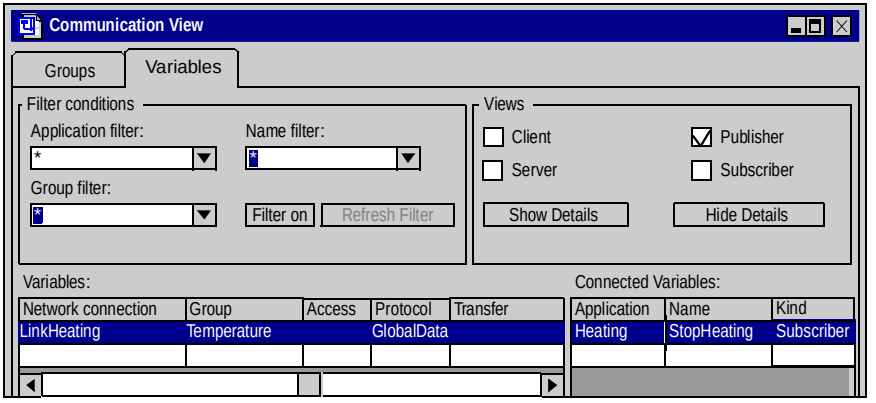
Now configure the variables for your plant in the **Communication View**.

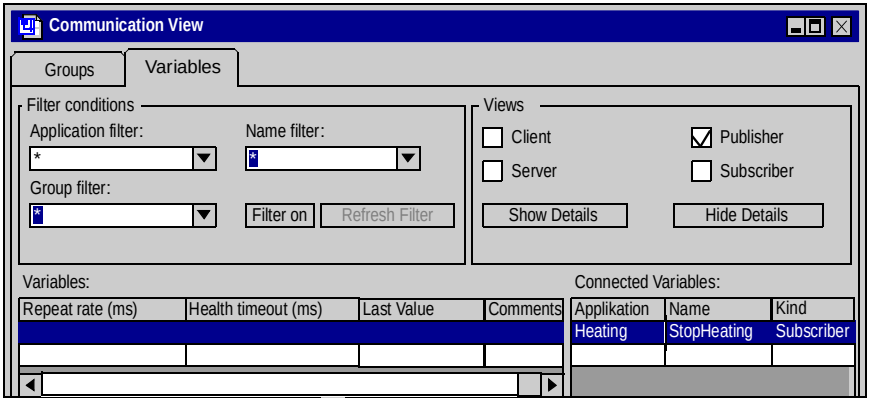
Step	Action
1	Open the **Communication View** and click on the **Variables** tab.
2	Activate, in the area **Views**, the check box **Publisher**.
3	In the area **Variables** click in the empty field under **Application** and select from the drop-down list `WindowMonitoring`.
4	Click in the empty field under **Name** and enter the name `WindowIsOpen` for the Publisher variable.
5	Click in the empty field under **Type** and select `Publisher` as a variable type from the drop-down menu.
6	Click in the empty field under **Type** and select `INT` as the variable type from the drop-down menu.
7	Click in the empty field under **Link** and select `LinkHeating` from the drop-down menu.
8	Click in the empty field under **Group** and select `Temperature` as group from the drop-down menu.
9	Click in the next empty field under **Protocol** and select the communication protocol `GlobalData` from the drop-down menu.
10	In the area **Assigned Variables** click in the empty field under **Application** and select from the drop-down list `Heating`.
11	Click in the empty field under **Name** and enter the name `StopHeating` for the Subscriber variable.
12	Click in the empty field under **Type** and select `Subscriber` as a variable type from the drop-down menu.

Appearance of the Variables tab

This is how the **Variables** tab should now look in your plant.







Configuring PLC to PLC communication variables

Now configure the PLC to PLC communication in the **Communication View**.

Step	Action
1	Open the **Communication View** and click on the **Variables** tab.
2	In the **Views** area, activate the **Client** and **Server** check boxes.
3	In the area **Variables** click in the empty field under **Application** and select from the drop-down list `WindowMonitoring`.
4	Click in the empty field under**Name** and enter `GlassBroken` as name.
5	Click in the empty field under **Type** and select `Server` as a variable type from the drop-down menu.
6	Click in the empty field under **Type** and select `WORD` as the variable type from the drop-down menu.
7	In the area **Assigned Variables** click in the empty field under **Application** and select from the drop-down list `Heating`.
8	Click in the empty field under**Name** and enter `BrokenGlassAlarm` as name.

Configuring PLC to I/O communication

Now configure the PLC to I/O communication in the **Communication View**.

Step	Action
1	Open the **Communication View** and click on the **Variables** tab.
2	In the **Views** area, activate the **Client** and **Server** check boxes.
3	In the **Variables** area, mark the predefined variable `ParameterTableIn`.
4	In the area **Assigned Variables** click in the empty field under **Application** and select from the drop-down list `Heating`.
5	Click in the empty field under**Name** and enter `FanIn` as name.
6	In the **Variables** area, mark the predefined variable `ParameterTableOut`.
7	In the area **Assigned Variables** click in the empty field under **Application** and select from the drop-down list `Heating`.
8	Click in the empty field under**Name** and enter `FanOut` as name.

Analyzing the Entered Data

Analyze the entered data

The data entered is analyzed via **Operations** → **Analyze** for your Plant.

> **Note:** This method always analyzes **all** plant data which could take a long time. The **Application view** enables you to analyze the entered data for **individual** applications.

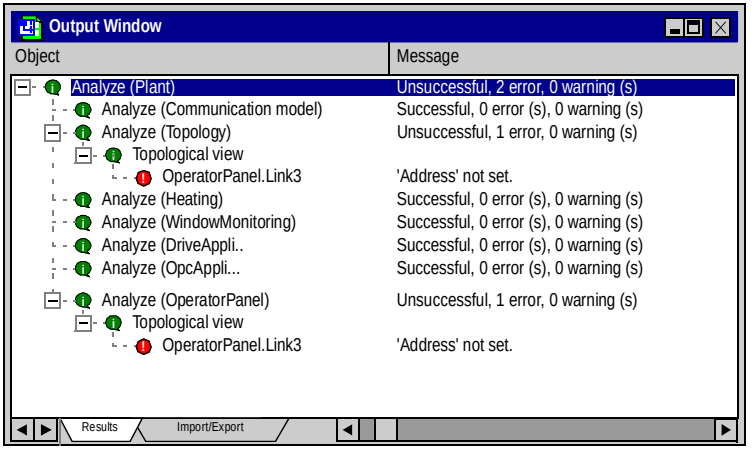
Start analysis

Now start the analysis of the entered data.

Step	Action
1	Start the analysis via the main menu bar **Operations** → **Analyze.** **Result:** Since there are still errors an **output window** opens that displays the corresponding errors and warnings for the respective objects.
2	Click the right mouse button in the **output window** and open the entire directory tree using **Expand all**.

Representation of the output window

The output window should now look like this.



> **Note:** Behind **Analyze** there is a display in brackets which shows which area is being analyzed. The **bold** font shows which area has an error.

Error Action

Correct the error displayed.

Step	Action
1	Double-click on the error `OperatorPanel.Link3` (white exclamation mark in a red circle). **Result:** Automatic jump in the **Graphical view** to the network connection without an address.
2	Open the object properties sheet of `Link3`and enter the following value: • **Name**: `LinkOperatorPanel` • **Address**: `1.0.0.5`
3	Close the window using **OK**.

Start analyze again

Start the analysis once again.

Step	Action
1	Start the analysis again via the main menu bar **Operations** → **Analyze**.
2	The **output window** indicates that the analysis is running, and then that the analysis has been completed successfully.
3	Save your plant.

Generating the Plant

Structure generation

The **Operations** → **Generate** function transfers the structure and data for all Applications in your Plant to the project.

Note: This method always generates **all** structures that could take some time. The **Application view** enables you to start **individual** applications.

Start generation

Now start the generation of the application structures and data.

Step	Action
1	Start the generation via the main menu bar **Operations** → **Generate.** **Result:** Because all errors are resolved after the analysis, the generation runs until complete which may take some time. **Note:** If not all the errors were resolved, the generation is cancelled and an error message output window appears. You have to solve this error. More information can be found in the *Error Action, p. 26* section.
2	The **output window** shows that the individual operation parts (analyze, consistency check, generate) are running and then that the generation was successfully completed.
3	Click the right mouse button in the **output window** and open the entire directory tree using **Expand all**. You can now see a list of all the generation steps.
4	Save your plant.

Opening and Modifying the Unity Pro Project

Editing the Unity Pro Project

You can now open the Unity Pro project generated in Unity Studio Manager and make modifications.
The data modified in the Unity Pro project is then transferred back to the Unity Studio Manager (**Update from Project**).

Open HEATING.STU

Now open the Unity Pro project HEATING.STU as described below.

Step	Action
1	Open the Application view in Unity Studio Manager.
2	Click on the right mouse button in the **Name** column on `Heating`.
3	Use **Launch tool** → **Unity XL** to start the Unity Pro program. **Result:** Unity Pro is started and the HEATING.STU project is opened.

Open the Functional View

Open the **Functional view** in Unity Pro.

Step	Action
1	If still visible, open the Project-Browser via **Tools** → **Project-Browser**.
2	Select the Functional view via **View** → **Functional view**.
3	Expand the directory tree in the **Functional view**.

Changing the name and adding a functional module

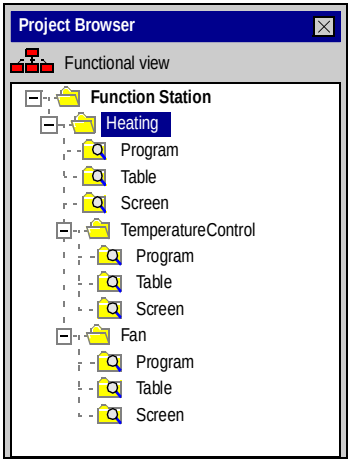
Change the name of the ventilator and add another one.

Step	Action
1	Click the right mouse button on `Fan` and in **Properties** open the properties sheet for this functional module.
2	Change the name to `FanRoom11` and close the window with **OK**.
3	Click the right mouse button on `Heating` and then on **New Functional Module...**. **Result:** The **Create...** window is opened.
4	Enter the name as `FanRoom12` and close the window with **OK**.

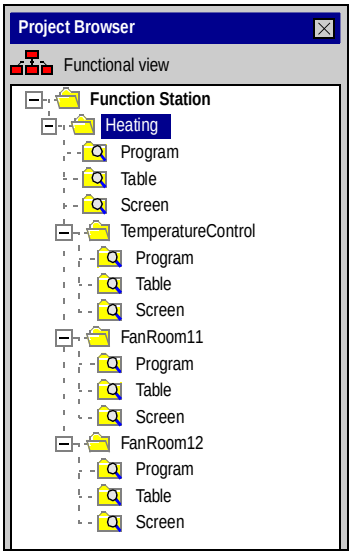
Representation of the Functional View

The plant directory tree of your **Functional view** should look like this before and after the modification in Unity Pro.

Before Changes



After Changes



Save and close

Save the changes.

Step	Action
1	Save the HEATING.STU project.
2	Exit Unity Pro.

Check Consistency

Consistency check

The menu **Operations** → **Check Consistency** is used to check consistency between all applications and your plant in Unity Studio Manager.

Note: Always use this method to check the consistency of all the applications in your plant, and under certain circumstances, this could take a long time. You can start checking individual applications in the **Application view**.

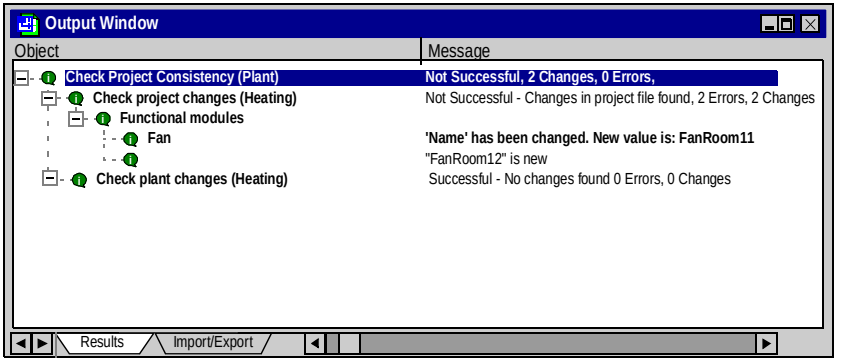
Start consistency check

Go to the Unity Studio Manager and check the consistency of your plant.

Step	Action
1	Go to the Unity Studio Manager.
2	Start the consistency check via the main menu bar **Operations** → **Check Consistency**. **Result:** An **output window** opens that displays corresponding messages for the respective objects.
3	Check whether the name change and addition of a new Functional module that you have made in Unity Pro is displayed in the **output window**.

Display of the consistency check output window

The consistency check **output window** should look like this.



Update from Project

Update changes

You have made changes to a Unity Pro application that was generated in Unity Studio Manager. You can now update your plant in Unity Studio Manager from the Unity Pro application.

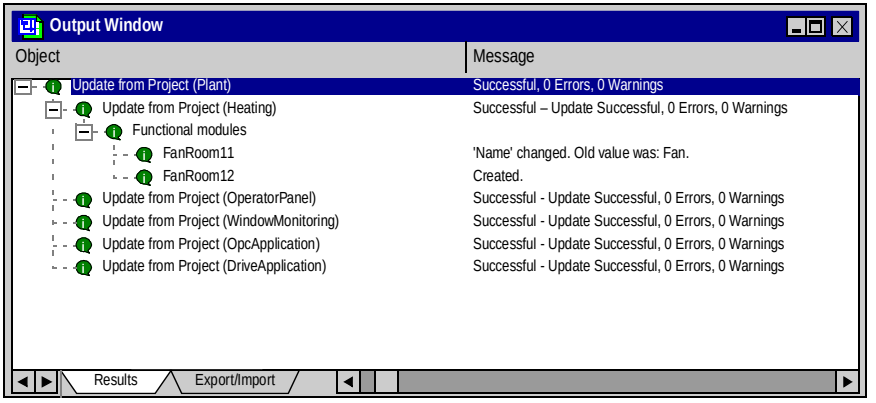
Update from project

Update your plant with the data from the Unity Pro project.

Step	Action
1	Start the update via the main menu bar **Operations** → **Update from Project**. **Result:** An **output window** opens to indicate that the update is running and then that the update has been successfully completed.
2	Check whether the **output window** shows that the name change and addition of a new Functional module has been also carried out in your plant Unity Studio Manager.

Representation of the update output window

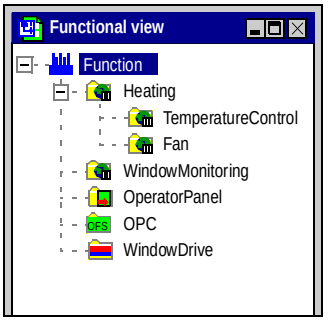
The update **output window** should look like this.



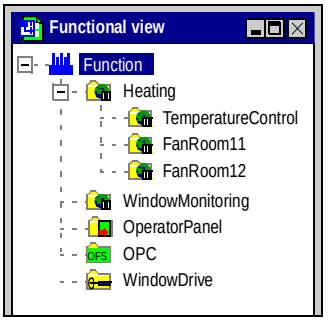
Representation of the Functional View

The plant directory tree of your **Functional view** should look like this before and after the update in Unity Studio Manager.

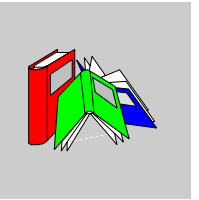
Before Update



After Update



Glossary



A

Access Type — The access type on communication variables can only be defined for Modbus Server Variables.
The following access types are possible on Modbus Server variables:
- R/W (read/write)
- Read
- Write

Application — An application models, on the Unity Studio Manager level, the aspects of a program that are essential for management in a distributed environment.

Application file — An application file contains all the programming and configuration information of a project.

Application Manager — An Application Manager is a software adapter between the Unity Studio Manager Server and the tool-specific server. It converts the Unity Studio operations into operations of the tool-specific server.

Application view — Applications are listed, configured and assigned to devices in the Application view.

Application-specific tool — An application-specific tool is a program that is used to edit a specific application.

C

Catalog
A catalog defines the number of object types and rules that can be used when configuring a plant. Examples of object types are network types, device types and application types.

Communication variables
Communication variables are:
- Publisher/Subscriber Variables
- Modbus Clients/Modbus Server Variables

Communication View
In the Communication View, the cross-system variables of the applications are defined and communications connections between them are planned.

D

Data type
The data type determines the number of permitted values for this variable and the operations that can be used with these values. Unity Studio Manager uses elementary and derived data types as defined in IEC 61131-3.

Device
A device is an independent physical unit that is limited by its interface, and can call one or more specific functions in a defined context.
Devices are e.g. PLC or operating and display units. A device is connected to a network via a network connection, and may also be connected to several networks.

Drawing sheet
The plant topology is configured on one or more drawing sheets in graphical form.

F

Functional module
A functional module is a logical collection of one or more functions of the distributed automation system.
The functional module is the structural element that is used to create the hierarchical functional view. The functional module can contain additional functional modules.

Functional view
The Functional View shows the functions of a plant in a hierarchical structure.

G

Gateway address The Gateway address is the IP address of the "Default Gateways", i.e. the device to which all messages are sent whose destination is not recognized. The Gateway address must be either "0.0.0.0" (i.e. no Gateway) or be located in the same subnet as the network connection. The Gateway address for a network connection or network is usually assigned by the system administrator.

Global data Global data is any cross-system variable whose values can be exchanged between the distributed applications in a plant.

Graphical view The Graphical view is the graphical display of the physical structure of the distributed automation solution of a processing or manufacturing system.
The individual objects such as networks, network connections and devices are configured here.

H

HMI Human Machine Interface: Operation and display device

I

Instance (Visio) An instance is the occurrence of a Master Shape, that has been dragged from a stencil on to the drawing sheet.

M

Macro (Visio) A macro is a high performance program unit in Unity Studio Manager, which is created in Visual Basic for Applications (VBA). Macros are used to automate repetitive process steps.

Master shape A Master Shape is a shape in a stencil, that you can use repeatedly when creating your plant. If you move a shape from a stencil into your plant drawing sheet, an instance of the Master shape is created there.

Modbus Client Variable	In regard to the data exchange, the Modbus Client device issues a precisely defined read or write command to a Modbus Server device, and thus triggers the data exchange. In this way, the contents of the Modbus Client Variable are written to the Modbus Server device, or the Modbus Client Variable is updated from the variables that are read.
Modbus Server Variable	In regard to the data exchange, the Modbus Server device presents this variable for writing or reading access of a Modbus Client. If this writing or reading command is then issued by the Modbus Client device, data exchange takes place between the Modbus Client and the Modbus Server.
Multicast IP Address	An address that enables a special group of network connections within a subnet to be addressed.

N

Network	A network connects devices that can then communicate with each other via a common protocol. In the Graphical view (topology), a network is an instance of a network type.
Network connection	A network connection is the connection between devices and networks within a drawing sheet. It is represented by a line.
Network node	A network node is a device, that is connected to the network with a unique address.

O

Object	Generally an element of a plant, such as a device, network, application etc, is designated as an object.
Object type	The object type classifies objects that have logical and physical relationships, so that underlying hardware and software tools can be clearly allocated. Examples of object types are network types (e.g. Ethernet), device types (e.g. 140 CPU 671 60/V01.00 in the Quantum device family) and application types (e.g. Unity Pro). All objects types that can be instanced in your plant are listed in the **Network and Devices** catalog.

P

P&IDs	Piping and Instrumentation Diagrams
Plant	Plant is the designation for the sum of all devices, networks, programming and configuration information that describes the automation of a process control or manufacturing system.
PLC	Programmable Logic Controller
Program	A program is created from elements of the IEC 61131-3 programming language. It implements an application.
Project	A project includes all programming and configuration information that can be loaded in an individual device of a plant.
Project file	The project file contains all the programming information that describes an application.
Properties tab	The properties/attributes of the individual objects are displayed or defined in a properties tab.
Protocol	The Communication Protocol defines the rules that must be followed by all communication participants. The protocol contains the following rules: ● Application Protocol ● Rules for specifying the lengths ● Rules for designation of variables on the network ● Rules for designation of data types on the network The communication protocol for Publisher/Subscriber communication is **Global Data**. The Communication Protocol for the Modbus Client/Modbus Server communication is **Modbus**.
Publisher variable	The publisher variable (sender) prepares the data for one or more subscriber variables (recipient).

R

Routing

Routing enables a selective connection between two or more communication partners via network boundaries.

S

Shape

A shape is an object created using Visio drawing tools, or an instance of a master shape, that was dragged from a stencil onto the drawing sheet.

Stencils

A stencil is a collection of Master-Shapes, that are assigned to a specific pattern. Stencils in Unity Studio Manager are assigned to catalogs. They include all topological object types of a catalog.

storage area (%MW / %IW)

For every device with E/A scanner, a % MW and a %IW- area must be set These storage areas are used by the Unity Studio Manager for the generation of the I/O-scanner.
When you close the Search / Replace window, the following data is saved:

- Modbus Client Variables for the I/O-scanner
- Modbus Clients/Modbus Server Variables
- I/O-scanner device control block if Device Control Enabled = **True**

The following information is generated in the %IW storage area:

- I/O-Scanner Default-block (IO ScannerHealthBlockSize)

Subnet mask

The subnet mask is used if IP nets are divided into IP subnets (IP Subnetting). The subnet mask defines the number of bits that are used for IP subnetting. The values possible for the subnet mask depends on the IP address class.
The subnet mask for a network connection or network is usually given by the system administrator.
Example: IP address = 129.10.64.0 (class B address, i.e. the first 16 bits are used for the net ID). If you select the subnet mask = 255.255.240.0 (corresponds to bit sequence 11111111 11111111 11110000 00000000), this also specifies that an additional 4 address bits can be used for the subnet ID.

Subscriber variable

A subscriber variable receives data that was prepared by the publisher variables assigned to it.

T

Third Party Client Program item from third party clients that operate the Unity Studio Manager server.

Tool Tools are software applications for data entry and processing that can be started in Windows with the corresponding parameters, such as Unity Pro, Word etc.

Topology / Topological view The logical assignment of the individual objects such as networks and the devices connected to them in a distributed automation solution for a process control or manufacturing system, are configured in a tree structure in the topological view.

Transfer The transfer type defines how the variable is transferred from one application to the other.
There are:
- I/O Scanner
- Global data

U

USM **U**nity **S**tudio **M**anager

V

Variable A variable is used for data exchange between the individual software sections. They can have different values depending on the variables data type.

Variables in the Communication View

Variables in the Communication view are cross-system variables that applications use to exchange information via the network in accordance with the communication relationship configured. Exchange is carried out cyclically.
These variables specify the subset of the application data that is to be exchanged between the programs.
There are Publisher variables as the sender, and Subscriber variables as the receiver. When the publisher and subscriber variables are connected with each other, they are known as Global data.
Modbus Client/Modbus Server variables are used, whereby the Modbus Server presents a variable and the Modbus Client device writes to this variable or reads it.

Variables Type

Each variable that is involved in a data exchange has the property **Type**, which contains the variable rule.
The following variables types are available:

- Client Read
- Client Write
- Server
- Publisher
- Subscriber

View

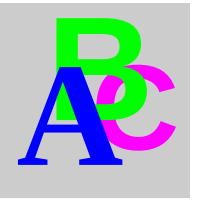
The views show the data and structures of the plant from different perspectives. The plant is configured in the different views.

X

XML format

XML (eXtensible Markup Language) is an expandable description language used for structured display and saving of data in a text file.
XML enables data exchange between software components regardless of their platforms.

Index



S