Adroit Technologies

Adroit 10

Technical Description

Adroit | MITSUBISHI ELECTRIC FACTORY AUTOMATION

Advanced Worx 112 (Pty) Ltd
20 Waterford Office Park
189 Witkoppen Road (c/o Waterford Drive)
Fourways
South Africa

www.adroittech.co.za

Version	Date	Author	Comment	Approval
1.0	06.11.2017	G Joyce	Original	D. Wibberley
2.0	22.05.2020	J. Duncan	Update Cover and formatting	
3.0	29.06.2020	J. Duncan	Update content and flow	
4.0	03.07.2020	D. Wibberley	Edited	
5.0	08.07.2020	J. Duncan	Added SDS, DBQuery Agent	
6.0	20.07.2020	J. Duncan	Add Accumulator Agent	
7.0	27.08.2020	J. Duncan	Finalise	
8.0	29.09.2021	J. Duncan	QMS	

Contents

1. Overview 7
1.1. Distributed Client-Server Architecture 7
1.2. Object Oriented and Context Driven Solutions 7
1.3. Openness 8
1.4. Scalability and Redundancy 9
1.5. Performance 9
1.6. Product Comparison Matrix 10

2. Main Components and Architecture 11
2.1. Data Layer 11
2.2. Middle Layer – The SmartUI Server 12
2.3. Presentation Layer – The Operator and Designer 12
2.3.1. The Operator 12
2.3.2. The Designer 13

3. Agent Servers (I/O Servers) 14
3.1. Agents 16

4. Agent Types 18
4.1. Primitive Agent Types 18
4.1.1. Text Agent 18
4.2. Basic Agent Types 18
4.2.1. Analog Agent 19
4.2.2. Counter Agent 19
4.2.3. Digital Agent 20
4.2.4. Expression Agent 20
4.2.5. Marshal Agent 22
4.2.6. MultiState Agent 23
4.2.7. String Agent 23
4.2.8. StringList Agent 23
4.2.9. Timer Agent 24
4.3. Advanced Agent Types 25
4.3.1. AgentGroup 25
4.3.2. Alarm Agent 25
4.3.3. AlarmList Agent 27
4.3.4. AlarmManagement Agent 27
4.3.5. Command Agent 28
4.3.6. DBAccess Agent 29
4.3.7. DBQuery Agent (ver 10.0.5) 29
4.3.8. DBLog Agent 30
4.3.9. EventLog Agent 31
4.3.10. EventOutput Agent 31
4.3.11. Multimedia Agent 31
4.3.12. Notify Agent 31
4.3.13. Recipe Agent 32
4.3.14. Scheduler Agent 32
4.3.15. Script Agent 33
4.3.16. Shift Agent 33
4.3.17. Statistical Agent 34
4.4. MIS/MES Agent Types 35
4.4.1. Accumulator Agent 35
4.4.2. OEE Agent and the new Enterprise OEE Agent 35
4.4.3. PID Agent 36
4.4.4. MaxDemand Agent 36
4.5. IT Agent Types 37
4.5.1. Audit Agent 37
4.5.2. DumpConfig Agent 38
4.5.3. ICMP Agent 38
4.5.4. PerfMon Agent 39
4.5.5. SNMPMgr Agent 40

4.6. System Agent Types 41
4.6.1. SystemDataLog Agent 41
4.6.2. Alias Agent 41
4.6.3. DataLog Agent 41
4.6.4. Device Agent 43
4.6.5. Other System Agent Types 44
4.7. Custom Agent and UDT Types 45
4.7.1. User-defined Type 45
4.7.2. Custom Agent Type 45
5. Detailed Architecture – Client Side 47
5.1. SmartUI Server 47
5.1.1. Datasources 47
5.1.2. Management 49
5.1.3. Projects 49
5.1.4. SmartUI Server Clustering 49
5.2. SmartUI Designer 50
5.2.1. The Home Menu: 50
5.2.2. The Agent Configurator 52
5.2.3. The View Menu: 53
5.2.4. The Project Menu: 54
5.2.5. The Design Menu: 55
5.2.6. The Behaviors Menu consists of: 56
5.2.7. Shapes Wizards & Examples Menus 57
5.2.8. Enterprise manager & Toolbox Menus 58
5.2.9. Graphics Library 59
5.2.10. Navigation Templates 59
5.2.11. Toolbox 59
5.2.12. Wizards 63
5.2.13. Templates 64
5.3. SmartUI Designer Advanced Functionality 64
5.3.1. Spiders 64
5.3.2. Available Spiders 66
5.3.3. Scripting 70
5.3.4. Object and Context Model 73
5.4. SmartUI Operator 75
5.5. Secure Mobile Gateway 76
5.6. Security 76
5.6.1. Server Security 77
5.6.2. Allowed Users and Groups 77
5.6.3. Policies 78
5.6.4. Datasource-specific Security 79
5.6.5. Client Security 81
6. Redundant Agent Servers (Active Clustering) 82
6.1. Active Clustering - Foundation 82
6.1.1. Catch-up 83
6.1.2. Real-time Catch-up 84
6.1.3. Replication (Shadowing) 85
6.1.4. Switchover 86
6.2. Client to Server Connection 87
6.2.1. Basic principles of Dynamic UI connections 87
6.3. Configuring Redundant Servers 89
6.3.1. Server Side Configuration 89
6.3.2. Client Side Configuration 91
7. Application Protocol Interface – Gateway to Performance 92
7.1.1. Bulk Engineering Tools - EXCEL 93
7.1.2. Open Interfaces 93
7.2. Management Information and Reporting 94
7.2.1. Report Suite 94
7.2.2. Report Pack 94

7.2.3. Event Reports 97
7.2.4. Process Reports 97
8. Performance Anywhere Client 98
9. Utilities 99
9.1. Agent Browser 100
9.2. File Version 100
9.3. OPC Technology 101
9.3.1. Adroit as an OPC Client 101
9.3.2. Adroit as an OPC Server 101
9.3.3. OPC UA Client 101
9.4. Protocol Driver Monitor 101
9.5. Scanning Monitor 102
9.6. Scheduler 102
9.7. Script Editor 103
9.8. Security Manager 103
9.9. Agent Server Monitor 104
9.10. Tag Monitor 104
9.11. Watches Window 104
9.12. UDT Agent Type Designer 104
9.13. Adtech Licensing Utility 106
9.14. WANLink 106
10. MAPS Process Suite 108
10.1. Overview 108
10.2. MAPS Projects 109

Table of Figures

Figure 1 Designer – Start Page 8
Figure 2 Layers 11
Figure 3 Typical Analog Agent 12
Figure 4 Operator – Display Example 12
Figure 5 SmartUI Designer 13
Figure 6 Analog Agent, Attribute and Tags 14
Figure 7 Alarming - Setting Alarm Limits 15
Figure 8 Analog Agent 16
Figure 9 Typical application where an Expression Agent can be used 20
Figure 10 OEE Agent 36
Figure 11 Systeminfo – Configuration 44
Figure 12 Systeminfo - Running 44
Figure 13 Diagnostics and Tools 46
Figure 14 Smart Data Services 48
Figure 15 Designer – Detailed Display Example 50
Figure 16 Home Menu 50
Figure 17 Configurator 52
Figure 18 View Menu 53
Figure 19 Project Menu 54
Figure 20 Design Menu 55
Figure 21 Behaviors Menu 56
Figure 22 Shapes Wizards & Examples Menu – Electrical Symbols 57
Figure 23 Shapes Wizards & Examples Menu – Hardware 57
Figure 24 Shapes Wizards & Examples Menu – Process Symbols 57
Figure 25 Shapes Wizards & Examples Menu – Situational Awareness Symbols 57
Figure 26 Shapes Wizards & Examples Menu – Static Graphics Symbols 57
Figure 27 Shapes Wizards & Examples Menu – Templates 58
Figure 28 Shapes Wizards & Examples Menu – Wizards 58
Figure 29 SmartUI Designer Left Tabs 58
Figure 30 SmartUI Designer Right Tabs 58
Figure 31 Project Navigation Template 59
Figure 32 AdroitHistoricalAlarmViewer 61
Figure 33 Event Viewer 62
Figure 34 Spider Workspace 65
Figure 35 Script Editor 72
Figure 36 Designer – Object Model 73
Figure 37 Active Clustering 82
Figure 38 Add a New Adroit Datasource 83
Figure 39 Catch-up 83
Figure 40 Real-time Catch-up 84
Figure 41 Replication 85
Figure 42 Switchover 86
Figure 43 Semi-automatic UI switchover 87
Figure 44 Semi-automatic UI switchover 88
Figure 45 Semi-automatic UI switchover 88
Figure 46 SmartUI Configuration Editor 90
Figure 47 SmartUI Editor – Cluster Settings 91
Figure 48 Application Protocol Interface 92
Figure 49 Bulk Engineering Tools - EXCEL 93
Figure 50 Report Suite 94
Figure 51 Audit Activity Report 95
Figure 52 DBLog Daily Summary Report 95
Figure 53 Multiple Line Trend Report 96
Figure 54 Historical Incidents Report 96

Figure 55 Performance Anywhere Client 98
Figure 56 Utilities 99
Figure 57 Agent Browser 100
Figure 58 File Version 100
Figure 59 Scanning Monitor 102
Figure 60 Scheduler – All users 102
Figure 61 Agent Server Monitor 104
Figure 62 Watches Window 104
Figure 63 UDT Agent Type Designer 105
Figure 64 Adtech Licensing Utility 106
Figure 65 WANLink 106
Figure 66 S88/S95 Extended Physical Model 109

1. Overview

Adroit is the product and result of the experience gained in supplying thousands of Supervisory Control and Data Acquisition (SCADA) and Human Machine Interface (HMI), and other real-time industrial software and information systems to customers in over 25 countries over a period of more than 30 years.

It represents a combination of the key requirements for a secure, network and Internet-centric software toolkit that can be readily be tailored to meet the needs of a wide range of applications and industries. Simultaneously it still preserves a common basic underlying design approach.

The critical need for application future proofing, portability and growth has been fully understood and integrated right from the start. It is one of the reasons why the Microsoft Windows Operating Systems, robust, portable 32- and 64-bit multiprocessor platforms were chosen on which to design Adroit. Since Adroit strives to keep pace with the ever-changing technology, the base software is supported on most Windows OS's and Server platforms.

Adroit is a good example of Windows Distributed Network Applications Architecture (DNA) – the application architecture of choice for Microsoft Windows. It has three clearly discernible tiers or layers: data layer, a middle layer containing the rules or logic for interconnecting the other two outer layers and the presentation layer.

1.1. Distributed Client-Server Architecture

Adroit is highly-scalable. From stand-alone systems to typical larger Adroit configurations that involve many Agent Servers (I/O Servers) and Operators running on several different physical computers connected in a network – local or remote, employing Smart Client technology – with transparent and seamless communications taking place between presentation and data tiers of the DNA model.

Adroit may therefore be succinctly described as an open, portable, object-based set of process automation tools and components, structured around the Windows DNA architecture, and hosted on operating systems from the post Microsoft Windows operating system family.

1.2. Object Oriented and Context Driven Solutions

The real-world works in objects and the value lies in the context in which those objects are used. For example, a pump is made up of various signals or attributes that define the pump and the way it operates within a process. Real-time signals may include pump *running, pressure, and flow signals.* Fixed attributes might include Manufacturer, Model, and Power Rating but equally valuable is the physical context in which the pump has been deployed. A Plant, Process Area, Process Unit etc. as defined for example within an ISA 88/95 standard.

In Adroit it is possible to configure your own templates that include real time and static attributes. By creating a physical hierarchy for a process and deploying instances of these templates within that hierarchy allows the Adroit system to build all the Agents, Scanning, Alarming required automatically and seamlessly as you configure your system.

This allows for greater levels of standardisation; higher quality solutions and the result allows users to search and filter for any objects by the static attributes assigned during engineering. Reaching all 22kW pumps, made by Mitsubishi within the Main Process area is easy and intuitive, using the context view

control that can be simply dropped into an Adroit configuration. Adroit saves you time and money on engineering and maintenance, a perfect choice for your SCADA / HMI.

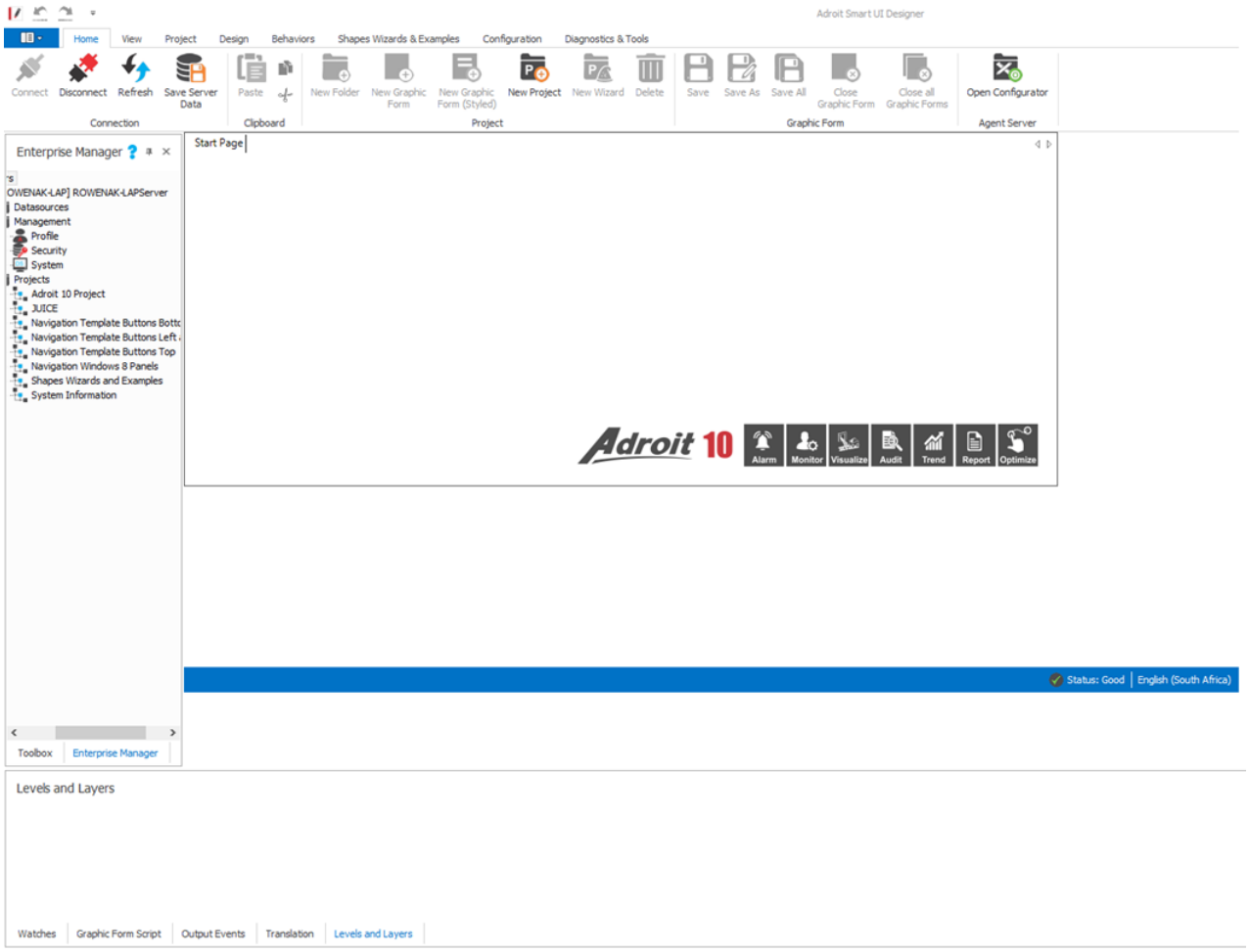


Figure 1 Designer – Start Page

1.3. Openness

Adroit is a product with "Open Automation Technology" built into its core, that is realised through the various open interfaces of both the Agent Server, SmartUI Server and Operator components.

In a control system, users want data everywhere and all the time. With this in mind, the various open interfaces, built on known software standards, mean every property of every object can be exposed to 3rd party applications and components running on computers anywhere that have secure access to a suitably licenced Agent Server (I/O Server) or the Middleware Layer the SmartUI Server (SUIS).

Supporting Web-Services, OPC, OLE DB, SNMP, Scripting and other entry points makes Adroit into an ideal enterprise level shop-floor visualisation solution that will support any vision for a connected enterprise.

1.4. Scalability and Redundancy

Adroit is scalable – users can add power and resources as required, building the system from a single stand-alone computer to a very large installation that may encompass multiple sites – with very little effort. Scalability is implemented by adding Agent Server scanning ability or additional remote client connectivity to the network.

For example, consider a stand-alone installation consisting of a single computer where both the Operator and Agent Server would be resident. As the application grows, the Operator could be hosted on a different computer forming a physically separate "client" in the distributed network architecture. This would allow more computational resources to be dedicated to the Agent Server, an expansion strategy that neither compromises the integrity of existing data nor requires reworking of the existing configuration.

Redundancy is supported at the Agent Server (I/O Server) level to give comfort where it is critical to ensure data is always available from the control (PLCs) network. This is achieved through implementing the Active Clustering technology. It is extremely easy to achieve and has proved its value to many clients running critical process control systems. Redundancy and load balancing is also support at the SmartUI Server level where higher loads because of large numbers of Operators might be connected to a central set of Agent Servers.

1.5. Performance

Adroit handles large I/O counts with ease. Benchmarked tests were done by configuring a database containing 50,000 I/O points:- 16,000 analogue and 32,000 digital.

Various performance measurements were taken using the Windows Performance Monitor tool. The Agent Server containing 50,000 I/O points was loaded up in less than 2½ minutes.

It is a generally accepted rule that under normal plant operating conditions one can expect an average of about 1% of the total I/O to be changing every second. For a plant with 48,000 I/O this represents around 260 server updates per second. Using relatively inexpensive single CPU hardware less than 20% average processor loading was measured.

1.6. Product Comparison Matrix

	MAPS	Adroit	Adroit Lite	Adroit Lite & IPC Bundle	Adroit Ignite	OPC/Edge Gateway
Scanned Point Sizes	30, 75, 150, 300, 750, 1 500, 2 500, 5 000, 10 000, 15 000, 20 000, 25 000 & 50 000 (also available 5 000 Bundles)	30, 75, 150, 300, 750, 1 500, 2 500, 5 000, 10 000, 15 000, 20 000, 25 000 & 50 000 (also available 5 000 Bundles)	30, 75, 150, 300, 750 & 1 500	300	8, 30, 75, 150, >150	8, 30, 300, 1500, 5000, >5 000
Local Design Client	Included as standard on local AS	Included as standard on local AS	Included as standard on local AS	Included as standard on local AS	Included as standard on local AS	Included as standard on local AS
Client Type	Single or 5 x View Client Bundles	Single or 5 x View Client Bundles	2 x Single View Client **ONLY**	2 x Single View Client **ONLY**	2 x Single View Client **ONLY**	OPC Client Connection Licensed
Ø Designer	✔	✔	✔	✔	✔	–
Ø Operator	✔	✔	✔	✔	✔	–
Ø Performance Anywhere	✔	✔	✔	✔	✔	–
Ø Adroit Air	✔	✔	✔	✔	✔	–
Redundancy (Hot Standby)	Supported (75% of Agent Server prices)	Supported (75% of Agent Server prices)	✘	✘	✘	✘
Distributed/Proxy Architecture	✔	✔	✘	✘	✔	✔
Group Wide License	✔	✔	✔	✔	✔	✔
Technology Agreement	Based on Scanned Points	Based on Scanned Points	Based on Scanned Points	Based on Spanned Points	Based on Agents used	Based on Spanned Points
Agent Types	All Available	All Available	All Available*, **EXCEPT** for: -	All Available*, **EXCEPT** for: -	All Available*, **EXCEPT** for: -	All Available*, **EXCEPT** for: -
Ø Primitive	✔	✔	Frame; Integer; Real	Frame; Integer; Real	Frame; Integer; Real	Frame; Integer; Real
Ø Basic	✔	✔	MultiState	MultiState	Marshall; MultiState	MultiState
Ø Advanced	✔	✔	Alarm Management; ARec; Recipe; Scheduler; ShifAdv	Alarm Management; ARec; Recipe; Scheduler; ShifAdv	Alarm Management; ARec; Recipe; Scheduler; ShifAdv	Alarm Management; ARec; Recipe; Scheduler; ShifAdv
Ø MIS/MES	✔	✔	MaxDemand; OEE_adv; OEE; PID	MaxDemand; OEE_adv; OEE; PID	MaxDemand; OEE_adv; OEE; PID	MaxDemand; OEE_adv; OEE; PID
Ø IT	✔	✔	Audit; DumpConfig; ICMP; SNMPMngr	Audit; DumpConfig; ICMP; SNMPMngr	Audit; DumpConfig; ICMP; SNMPMngr	Audit; DumpConfig; ICMP
Ø System	✔	✔	✔	✔	✔	✔
Ø Other	✔	✔	Custom; UDT	Custom; UDT	Custom; UDT	Custom; UDT
Datasources	Adroit; Adroit Air; OLEDB; EventLog; GIS; ObjectModel; ServerScripts; MAPS; Simulation	Adroit; Adroit Air; OLEDB; EventLog; GIS; ObjectModel; ServerScripts; MAPS; Simulation	Adroit; Adroit Air; OLEDB; EventLog; GIS; ObjectModel; ServerScripts; MAPS; Simulation	Adroit; Adroit Air; OLEDB; EventLog; GIS; ObjectModel; ServerScripts; MAPS; Simulation	Adroit; EventLog; ServerScripts; Simulation	–
Interfaces	C API; Web Services; OLE; Active X	C API; Web Services; OLE; Active X	C API; Web Services; OLE; Active X	C API; Web Services; OLE; Active X	C API; Web Services; OLE	C API; Web Services; OLE
Operating Software - OS	Windows 10 Enterprise	Windows 10 Enterprise	Windows 10 Enterprise	Windows 10 Enterprise	Windows 10 Enterprise	Windows 10 Enterprise
Operating Software - Other	.Net framework 4.6.2	.Net framework 4.6.2	.Net framework 4.6.2	.Net framework 4.6.2	.Net framework 4.6.2	.Net framework 4.6.2
FED Drivers	Over 100 available	Over 100 available	Over 100 available	Over 100 available	Over 100 available	Over 100 available
Licensed Drives	Allen Bradley; Siemens EtherNet and DNP3.0	Allen Bradley; Siemens EtherNet and DNP3.0	Allen Bradley; Siemens EtherNet and DNP3.0	Allen Bradley; Siemens EtherNet and DNP3.0	Allen Bradley; Siemens EtherNet and DNP3.0	Allen Bradley; Siemens EtherNet and DNP3.0
			*Marshall Agent requires 16 Scanpoints	*Marshall Agent requires 16 Scanpoints		

2. Main Components and Architecture

2.1. Data Layer

Adroit offers users the unique ability to securely integrate both Real-Time and Transactional data (Databases) and WebService data and even IoT data into a rich and functional user interface. This is done through creating connections from the SmartUI Server into the various Datasources that make up the data layer.

Sources of Data can include:

- Agent Server (I/O Server)
- OLE DB Databases (external to Adroit)
- Web Services
- IoT Endpoints (Azure)

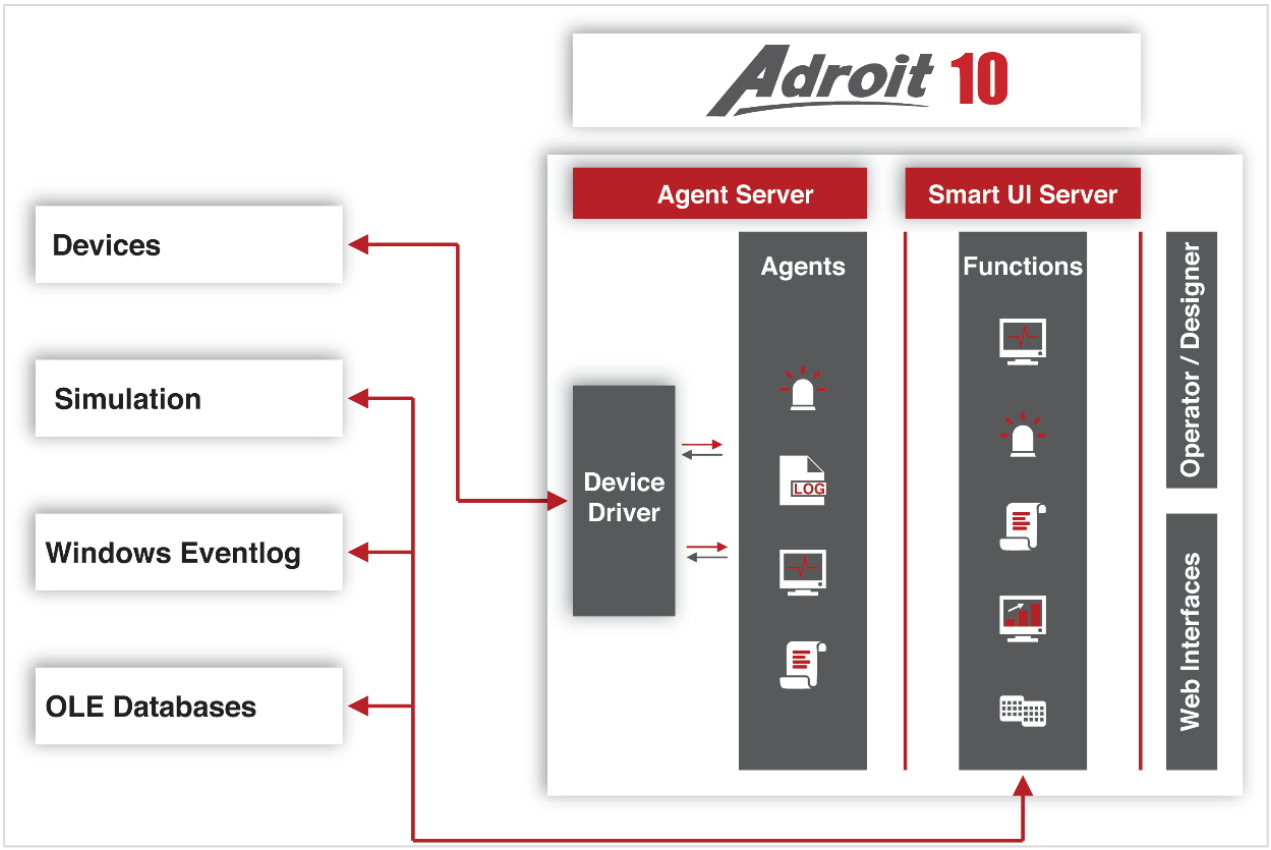


Figure 2 Layers

A more detailed description of each Datasource is described later however to summarise:

In Adroit, the I/O Server part of this architecture is called an *Agent Server*. Being object oriented, Adroit does not deal in primitives but delivers value from the start through the concept of intelligent objects called Agents. Agents are intelligent objects because, in addition to containing "state" information, as do conventional database records, they also embody "behaviour". A more detailed examination of the implications and meaning of this is presented under the sub-heading Agents later in this document.

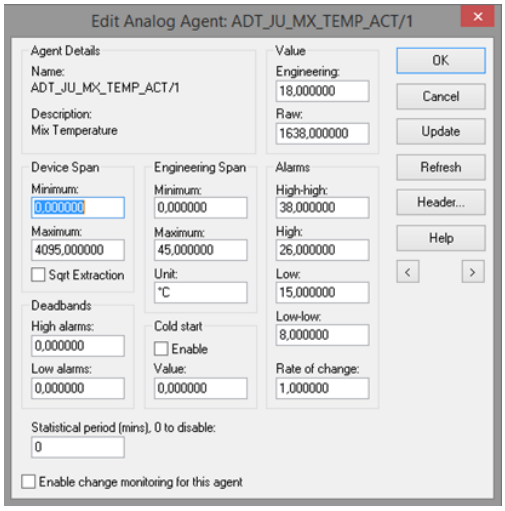


Figure 3 Typical Analog Agent

The Agent Server is responsible for scanning data, pre-processing, logging and alarming and interacting with external systems such as databases and third-party applications wishing to use the data available from the plant. Other sources of data can be included in an Adroit system and include data from Web-Services, OLE DB databases, IoT platforms etc.

2.2. Middle Layer – The SmartUI Server

The SmartUI Server is responsible for managing access to the data layer, graphic forms, and management of user profiles governing access rights etc.

2.3. Presentation Layer – The Operator and Designer

2.3.1. The Operator

The Operator is Internet-enabled and built on top of the Microsoft .Net framework allowing for greater flexibility and more custom programming functionality.

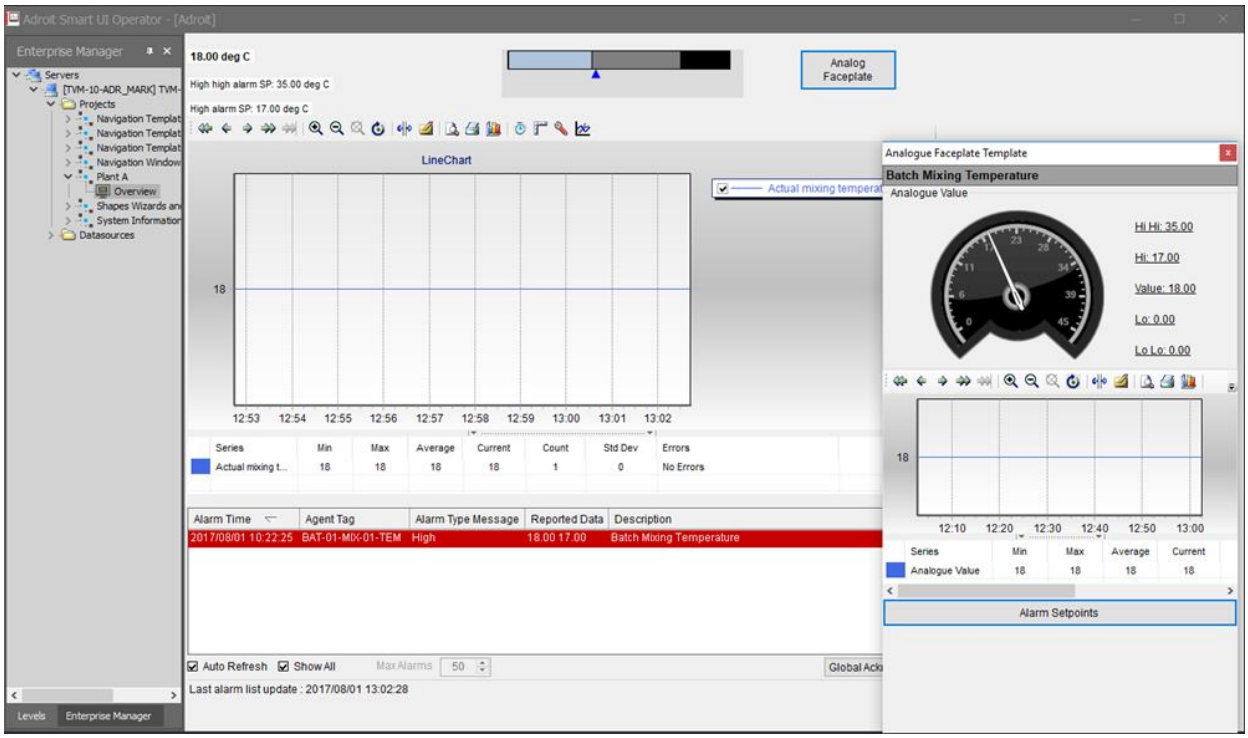


Figure 4 Operator – Display Example

2.3.2. The Designer

The Designer application is the entry point into building any Adroit solution. A secure log-on and rights applied allows users to manage what capability and user or group of users has on the system. In addition, the Designer is the engineering environment in which users create access to data, manage users and groups and build graphic forms to be displayed in the Operator.

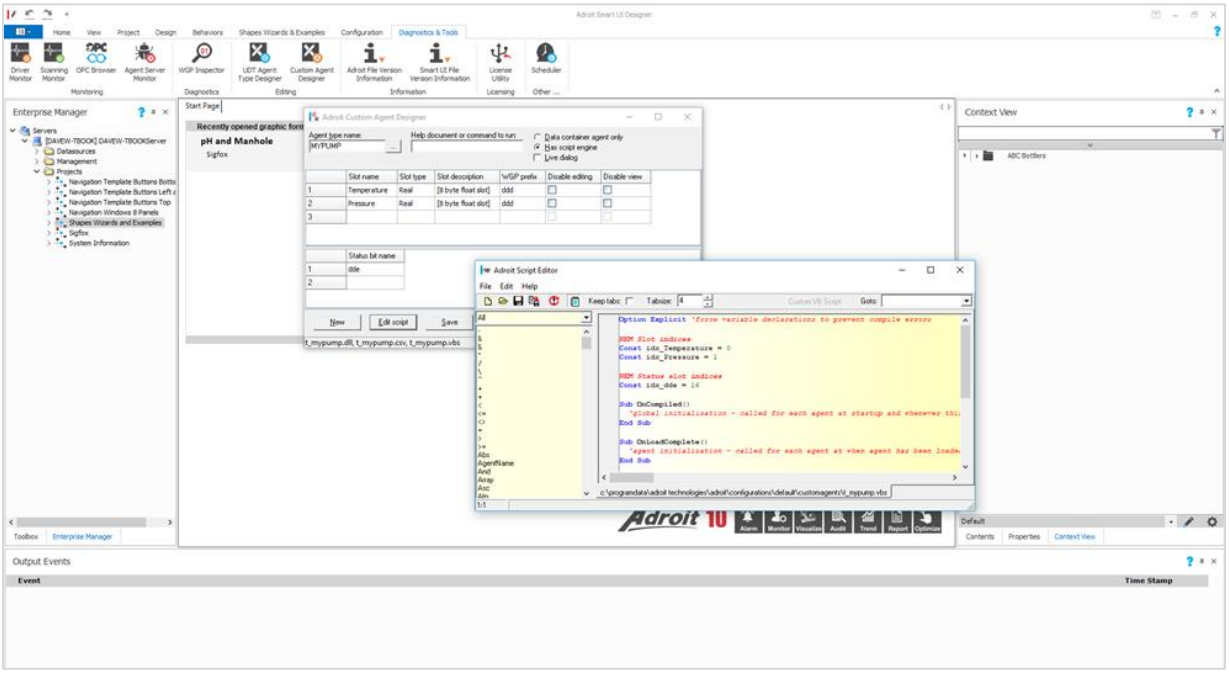


Figure 5 SmartUI Designer

The graphical plant display windows within the Operator are themselves composed from *objects*, in this case display objects known as *picture elements*. In addition to state information, such as size, location, colour, etc., display objects may also be configured to have a wide variety of *animation behaviours*. By connecting presentation layer animation behaviours to plant values represented by Agents within Agent Servers, many different animated display effects can be configured that, in real-time, depict your plant in exactly the way that you require.

3. Agent Servers (I/O Servers)

The Agent Server, in automation terms is also known as the real-time I/O server. It is essentially a collection or repository of intelligent objects known as Agents. It is important to understand that most HMI systems talk of Tags but the object architecture of Adroit adds a layer of intelligence to this legacy approach.

Agents are called "Agents" because, instead of only containing data like simple database records or Tags as used by many other vendors, they also usually contain the ability to operate or act on their own data, driving its values and to read and write to other objects, and so influence them. Making Adroit very flexible to the User.

Agents are classified by function into a number of Agent types which can be used to implement a wide variety of system related, as well as application-specific tasks. For example, the Analog Agent type is used to handle the signal conditioning and alarm checking requirements of a process variable.

For each Agent type there may be zero, one or many Agent instances, (tags) Each time you add a tag you are creating a new Agent instance, however, Agent instances are simply referred to as Agents, whereas the Agent + Attribute is to Adroit a Tag.

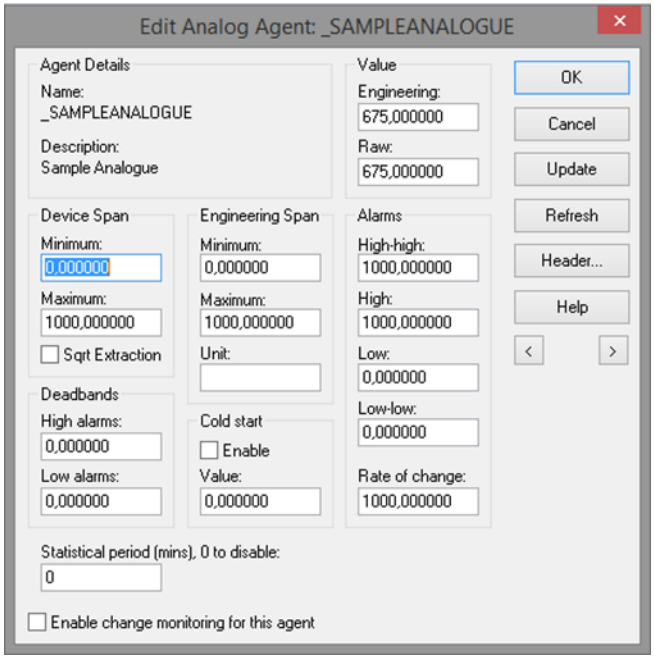


Figure 6 Analog Agent, Attribute and Tags

Agent Servers provide consistent mechanisms for creating and destroying Agents and efficient means of accessing the information contained within them. There are also no practical limits to the number of different Agents, nor indeed types of Agents, that can be contained within an Agent Server.

Agent Servers can concurrently interact with as many clients as may be required. This feature is facilitated by *Agent level locking* which prevents concurrent access to any single Agent instance but permits concurrent access to data contained within different Agents.

The Agent Server database or .WGP file stores almost all of the data and configuration of the Agent Server, including the added Agents, their values and their alarming, logging and scanning configuration. Configuration of the Agent Server is done on-line and live and for this reason, it is essential that this file,

along with the other Adroit project files, be frequently backed up to ensure that the integrity of this data is preserved.
WGP files provide the following two additional features to assist with their regular back up:

- Automatic backup of WGP file on each save: To further ensure that changes made to the WGP file are safeguarded between the regular backups, WGP files can now be automatically backed up each time they are saved.
- Version information saved in the WGP file: In addition to providing an audit trail of changes for debugging and troubleshooting purposes, this versioning information can also be used to assist in locating the required backup file when recovering lost changes.

Agent Servers can also be audited, in other words logging the Agent Server's sessions and selected activities performed during these sessions to an OLE DB database. In addition, by using the Audit Agent you can log the changes of values that are important to the running of your process, such as set point values and/or alarm limits.

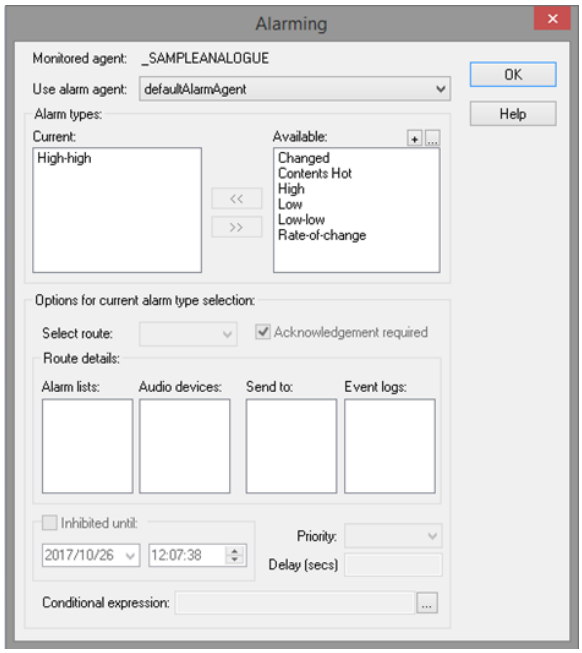


Figure 7 Alarming - Setting Alarm Limits

Generally, you will audit the Agent Server to provide a means of recourse in the event of a problem occurring to your process. Examining the audit logs you can determine any of the following user actions:

- When the Agent Server was stopped and started;
- What Agents have been added or removed from your installation;
- What Documents have been saved (Graphic Forms);
- Which audited tags (as specified in the Audit Agent), have changed in value (Setpoint changes);
- Who the logged-on user was when these audited events occurred.

Agent Servers allow clients to connect to them by using TCP sockets over and above the legacy named pipes:

- Sockets give you better performance when a remote client (running on another computer), connects to an Agent Server.
- Named pipes give you better performance when a local client (running on the same computer), connects to an Agent Server.

3.1. Agents

Adroit configurations are made up of one or more collections of intelligent, cooperating objects known as *Agents*. An Agent is said to be "intelligent" because, in addition to containing "state" information as would a conventional database "record" or "structure", it also embodies "action".

Consider for a moment the Analog Agent type. Each Analog Agent contains, amongst other things, the following data values, also often referred to as *attributes, slots,* or *properties* depending on your preferred terminology.

Here is a dialogue box representing an Analog Agent. As you can see the name of the Agent is CO_PA1_ANA_PH01 and the slots are the attributes of the Agent.

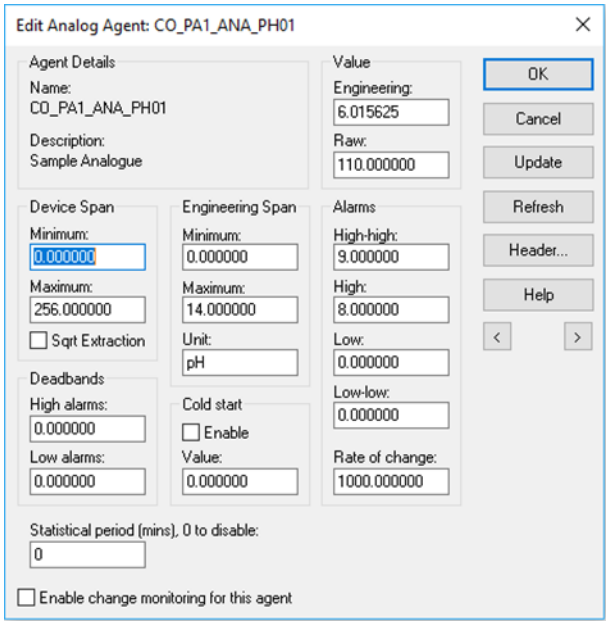


Figure 8 Analog Agent

Value	Description
rawValue	used to hold the raw or unconditioned value read-in from a front end device
rawMin	device minimum value
rawMax	device maximum value
engMin	engineering, i.e. conditioned or scaled, minimum value
engMax	engineering, i.e. conditioned or scaled, maximum value
value	conditioned or scaled value of the Analog

The Agent's "state" at any instant in time, is the collective value of these, and any other attributes. Because our Agent is of type Analog, it has the inherent knowledge to transform any value coming into its *rawValue* slot into a scaled or conditioned value for its *value* slot, using the *rawMin, rawMax, engMin, and engMax* scaling parameters.

Similarly, a value change in any one of the scaling parameters would cause the conditioned value to be updated accordingly. The sum total of all these rules, or inherent knowledge built into an Agent type, is known collectively as the Agent's "action".

An Agent's "state" can therefore be defined as the collective value of all its data attributes at an instant in time, and its "action" can be defined as the collection of expectations as to how it will respond to externally generated stimuli, such as, for example, a change to one or more of its data attributes.

Agents have both *generic* and *type-specific* action and attributes. Generic action and attributes is something that all types of Agents possess, whereas type-specific action and attributes, as the name suggests, is something that pertains only to Agents of the specific type.

Some of the generic actions possessed by all Agent types are, for example, the ability to *create, destroy, save,* and *load* instances. Examples of generic attributes are *name, description, status,* etc. The above partial description of an Analog Agent highlights some of the type-specific action and attributes of the Analog Agent type.

As will be explained in more detail later, Adroit makes use of different Agent types to implement a wide variety of system related, as well as application-specific tasks. The benefit of implementing everything as Agents in Adroit is that the standard mechanisms developed for displaying, scanning, logging, alarming, eventing, etc., of process values can also be used to monitor and control Adroit itself. This has proved to be of enormous benefit from an adaptability and flexibility point of view and will continue to pay dividends as Adroit continues to evolve in the future.

One of the generic Agent attributes is a 32-bit status word that is used by the Agent to store binary information about its state. The first 16 bits of this word are common to all Agent types (i.e., Agent Bad, Log Inhibit, etc.), while the last 16 bits are type-specific, for instance the Analog Agent uses the first type-specific bit to indicate a Low Alarm condition.

It is also critical to understand that any Tag (Agent+Slot) can be in itself, scanned, alarmed, and logged which makes Adroit an incredibly powerful platform for any application. In real life this means that, by way of example that the Analog Alarm limits could be scanned to PLC addresses without having to create another Tag as often required in competitive HMI products, saving time and license costs.

4. Agent Types

Adroit has been expanded over the years to solve many problems facing the modern control engineer. With this in mind we have classified Agent types into 6 different groups:

- Primitive Agent types
- Basic Agent types
- Advanced Agent types
- MES/MIS/IoT Agent types
- IT Agent types
- System Agent types
- Custom Agent and UDT Types

The reason for this is to de-clutter and focus the available functional Agent Types for the user.

4.1. Primitive Agent Types

Primitive Agents are used for gathering, handling and processing data to and from the physical external world, however they offer limited value add and are limited in functionality when compared to even the Basic Agents.

- The *Boolean, Integer, Real, Text and Frame* Agent types make up this group. They are essentially single value containers and can't be alarmed, data can be scanned in from or out to external devices.
- A *Frame* Agent stores up to 32 indexed strings, so that the value field of this Agent is set to the string that corresponds to the current index value.

4.1.1. Text Agent

One of the simplest types of Agent is the Text Agent, which stores a 79-character text string that can be defined at start-up, and then updated on request. A text string may also be specified as a *cold start* value.

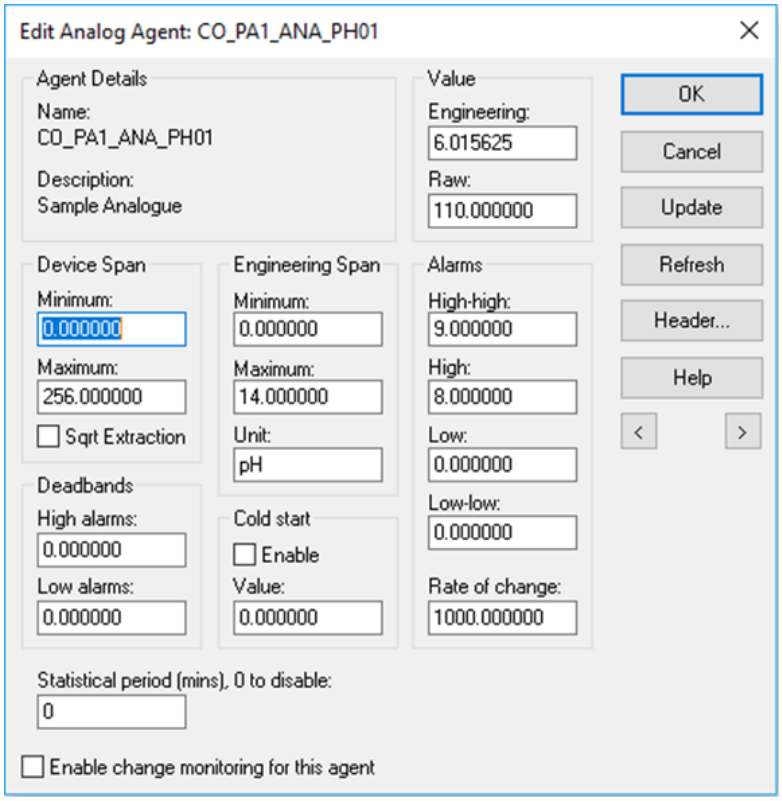
4.2. Basic Agent Types

Basic Agents are used for gathering, handling and processing data to and from the physical external world. Together these form the basic building blocks of a system. For instance:

- the *Analog*, *Digital, Counter, Date, Expression, Marshal*, *MultiState, String, StringList* and *Timer* Agent types perform some conditioning or conversion process on data normally scanned in from or out to external devices.
- The *Counter* Agent type performs the counting and averaging of discrete values.
- The *String, StringList* Agent type provides a means of indexing text.
- The *Expression* and *Timer* Agent types order and transform information for calculation and control purposes.
- The *Date* Agent type holds time and date information broken down into the year, month, day, hour, minute, second and millisecond constituents.

Some of the most useful Basic Agents are described below and are listed in Alphabetical order, we are not covering every Agent here and trust once the user grasps the Agent concept they will be able to navigate, select and work with the correct Agent for the application:

4.2.1. Analog Agent

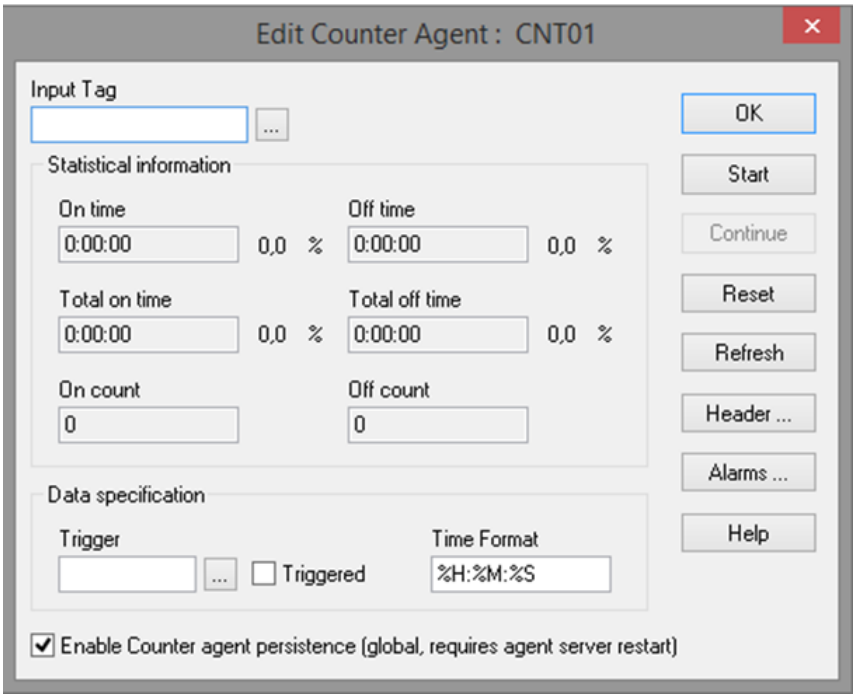


An Analog Agent stores a floating-point value that has been scaled according to user defined scaling parameters. This scaled value is checked by the Agent against the low-low, low, high, high-high and rate-of-change alarm settings. Any infringement of the alarm limits causes the Agent to set the appropriate bits in its status word.

Square root extraction for the linearization of flow signals, alarm deadband settings and cold-start initialization options are also settable by the user. Analog Agents provide a change monitoring feature which if enabled, logs an event and sets the Changed status bit when the value slots changes.

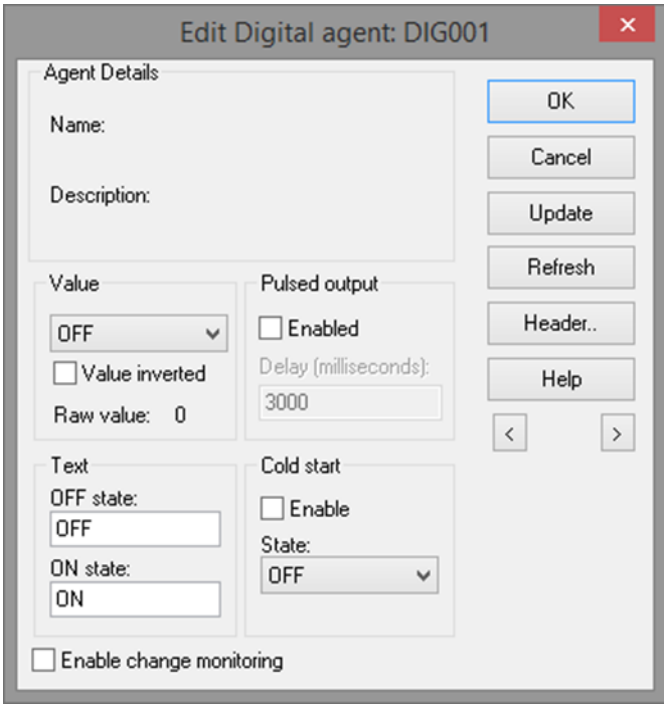
4.2.2. Counter Agent

A Counter Agent is a simple counter and average calculator which takes data from any other Agent's Boolean slots and calculates the time spent on and the time spent off, counts the number of on and off events, and totalizes the on and off time. Each of these values can be individually alarmed (by clicking the **Alarms...** button). This allows you, for example, to generate an event when a motor has started more than a required number of times.



This Agent is particularly useful for monitoring production lines, counts in a line, asset management. An example might be a Pump Running Signal as the digital input to drive the Agent, a setpoint of the number of run hours and an Alarm generated once that setpoint has been reached. All done within a single intelligent object, no complex PLC coding to achieve this.

4.2.3. Digital Agent



A Digital Agent stores a digital (ON/OFF) value as well as the text descriptions for each of the two discrete states. Value inversion, cold-start initialization, alarm infringement checking and a pulsed output option are provided.

With the pulsed output option enabled, the Agent maintains its ON state until a configurable delay has elapsed. As the pulsed output option is applied to the scaled value, inverting the value inverts the effect of the pulsed output. Digital Agents provide a change monitoring feature which, if enabled, logs an event and sets the Changed status bit when the value slots changes.

4.2.4. Expression Agent

The Expression Agent type is an extremely powerful mechanism for implementing user-defined calculations. These calculations can be used for a number of purposes.
Some of the more obvious uses are:

- For constructing control strategies by combining and connecting expressions into arbitrarily complex networks, and for scanning out one or more of the resulting values;
- For enabling or disabling of alarming;
- For enabling logging on a section of plant when a process value exceeds a certain threshold;
- For creating sophisticated animation effects by linking the output of an expression to a display object animation behaviour; and
- For creating relatively simple plant simulations by connecting a network of expressions together.

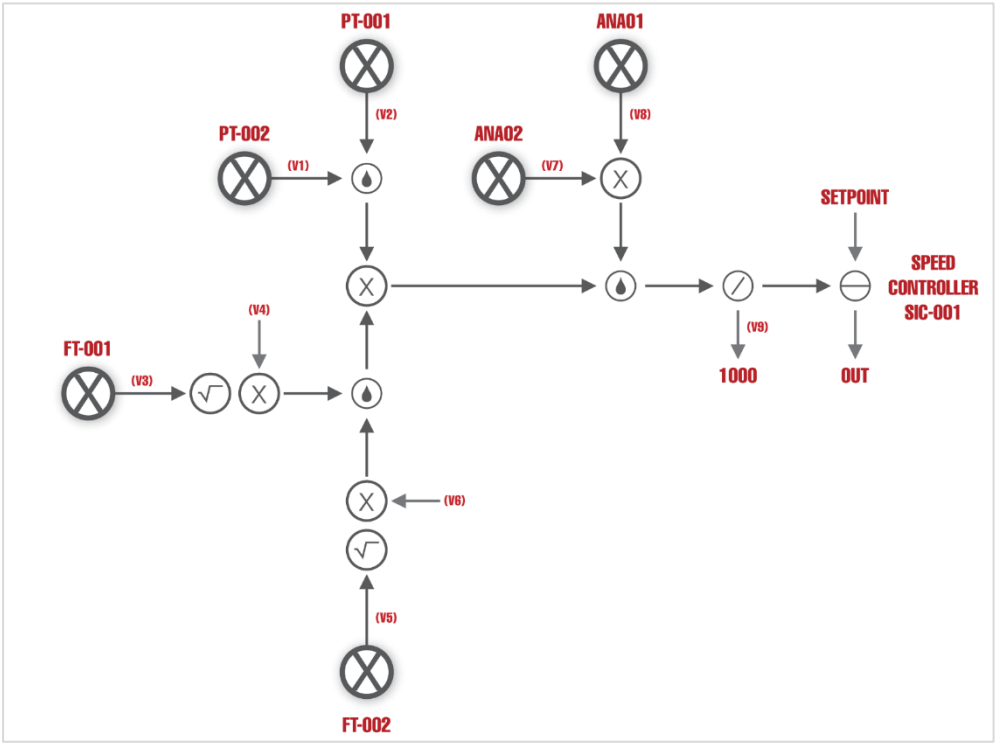
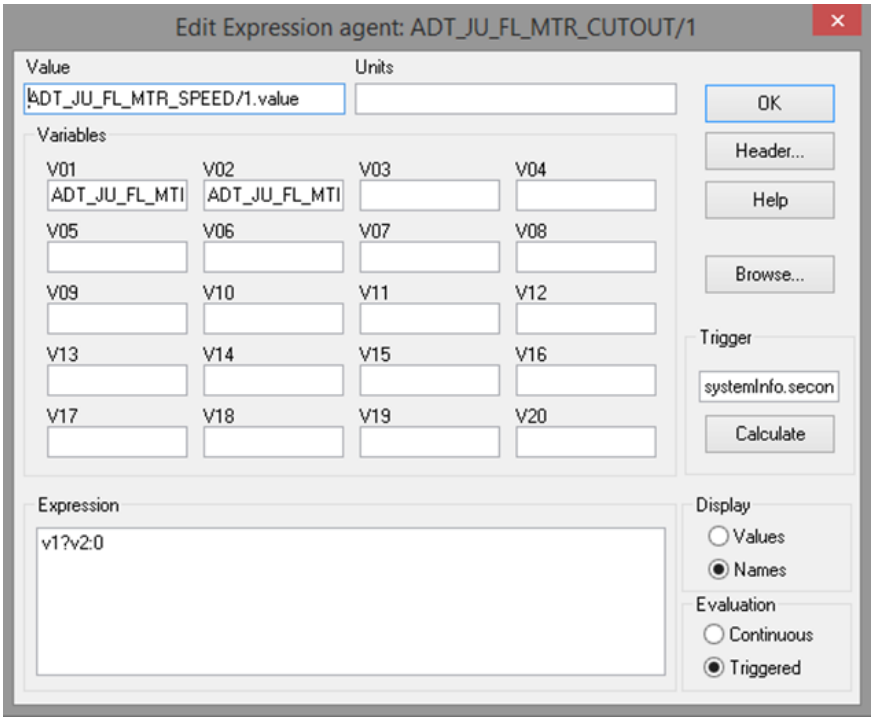


Figure 9 Typical application where an Expression Agent can be used

Each expression Agent can contain up to 20 input variables, each of which can be sourced from any other Agent attribute or directly from a front-end device. For complex calculations, expression Agents can be interconnected with the result of one expression becoming an input to the next.



The expression slot contains the actual mathematical ex3pression to be evaluated. To make the expression readable, any number of new lines or spaces are permitted. The expression syntax supported is that of a full ANSI-C expression (i.e., not a partial subset), including sub-expressions nested to an arbitrary depth, conditional operators, bit wise operators, logical operators, relational operators, unary operators, etc., as well as a comprehensive suite of mathematical functions.

The use of variable input names V1, V2, etc., are used in the expression so that it is generic. For example, by associating different *Agent.slot* combinations with the input variable definitions, an expression Agent can be reused in many different contexts without changing the actual expression. An expression can be configured to evaluate either continuously, or on receipt of a trigger pulse (the default). By connecting an expression's trigger input to another Agent slot that changes periodically it is possible to cause periodic evaluation of the expression.

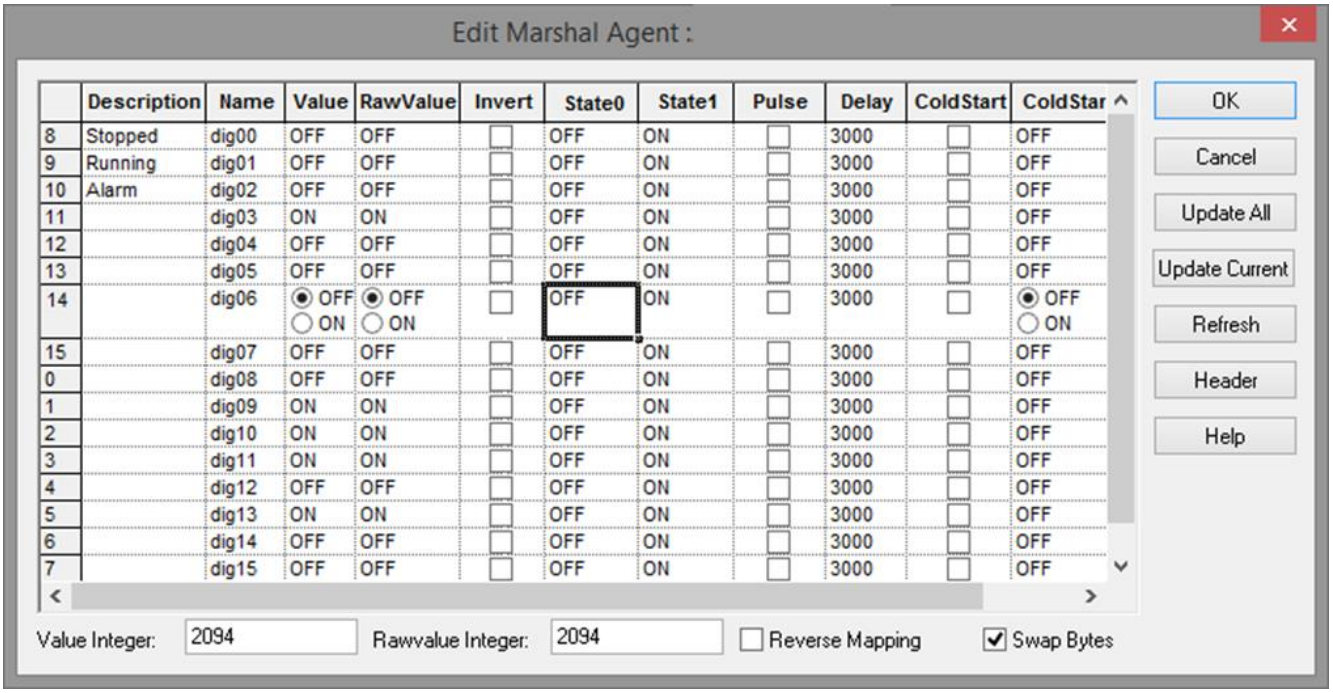
Alternatively, the trigger can be connected to some other input, possibly common to a number of other expressions, which may be used to constrain the group of expressions to evaluate, for example, only when data is valid.

4.2.5. Marshal Agent

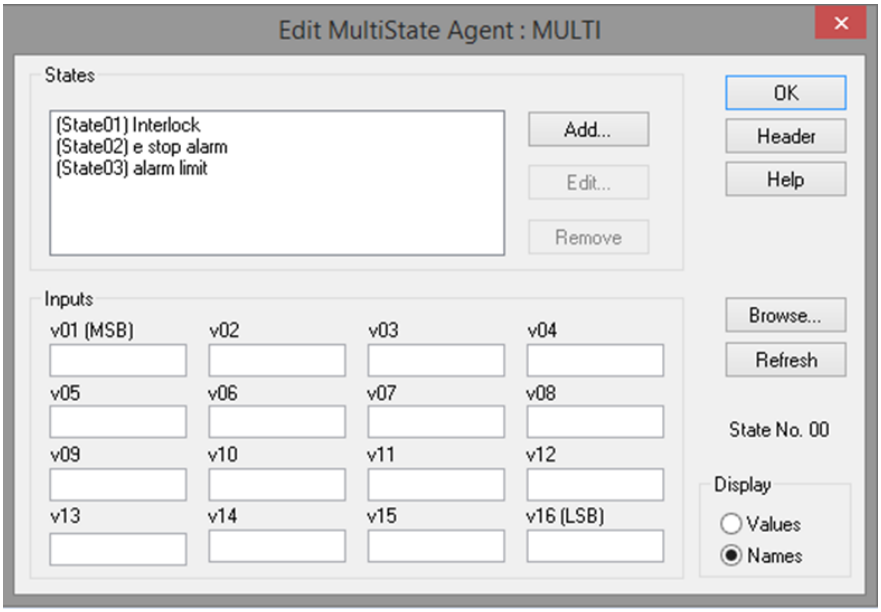
The Marshal Agent is designed to replace the popular marshalling function of the Multistate Agent, by gathering discreet binary data from its 16 bit rawvalue or value slot. By giving each of the individual bits the full functionality of a digital Agent the Marshal Agent exceeds the former capabilities of a marshalled mode Multistate Agent. These bits can then be alarmed, pulsed, named, and the 1 and 0 states of each bit can be named (on/off, open/closed, etc.).

Only the ON state is reflected in the status bit header slots of the Agent and therefore the alarming subsystem has been extended to allow the alarming of status bits that are ON or OFF. The rawvalue and value relationship can also be reversed, i.e., mirrored.

PLEASE NOTE that in the simpler HMI version of Adroit a single Marshal Agent consumes 16 Scan Points whereas in a full system a single Marshall consumes 1 Scan Point.



4.2.6. MultiState Agent



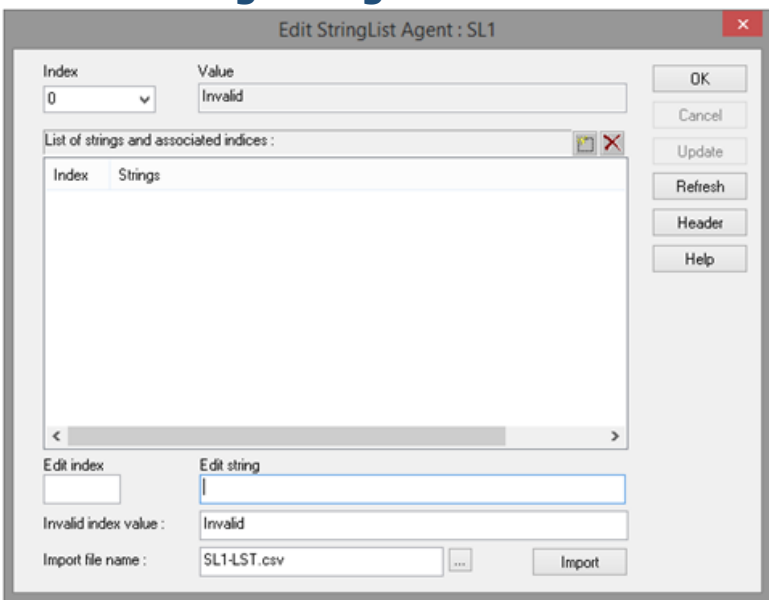
For installations that have a large number of drive or group controls, the MultiState Agent is provided to make configuration even easier and performance more efficient. This Agent can specify a drive's operating and control sequence by allowing a number of discrete states to be defined, made up of unique combinations of digital inputs and outputs, by means of a truth or logic table.

It can be used, for example, to implement a custom digital controller, or show the status of motors by means of changing background colour, text contents and colour as the drive group progresses through various state transitions. Prioritized alarming strategies can be achieved this way.

4.2.7. String Agent

The String Agent provides a repository for text. It can hold up to 79 characters.

4.2.8. StringList Agent



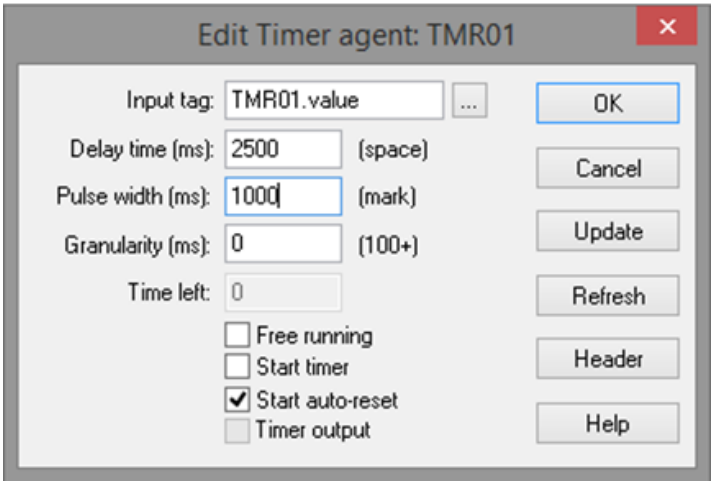
The StringList Agent provides a superset of functionality of the Frame Agent. It can store any number of character strings of variable length. These character strings are indexed and the current string being pointed to by the index is duplicated in the value field.

By configuring the Mulitstate Agent all the user needs to do is scan the Integer and display the value slot of the Agent to the user. No complex logic or putting text boxes on top of text boxes on a screen.

An example for using this Agent would be where the state engine of a plant would have an Integer value that represents the state;
0=stopped
1=Starting
2=Running
3=Stopping
4=Tripped
5=Maintenance Mode

4.2.9. Timer Agent

The Timer Agent is able to perform several commonly used timing operations, which previously could only be inefficiently accomplished by using expressions, recipes and scripts. This Agent also provides a Boolean value slot as an output, which is enabled (set to TRUE) and disabled (set to FALSE) by the Agent, depending upon its current mode of operation. This slot can therefore be used to enable or disable or schedule other Agents.



4.3. Advanced Agent Types

These provide the powerful SCADA and integration functionality which allows the basic Agents described above to be flexibly grouped, scripted and routed to external databases, as well as presenting related information to the user in event logs, etc. The most important advanced Agents are described below.

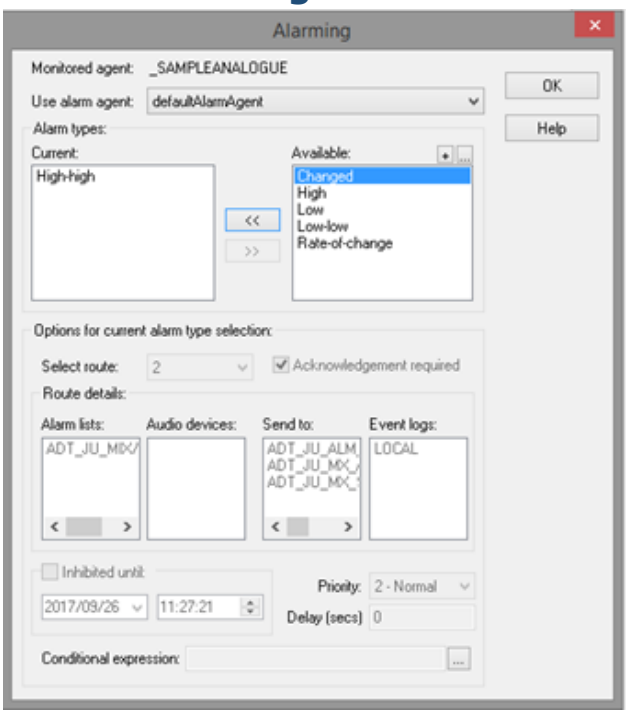
- The *AgentGroup, Alarm, AlarmList, AlarmManagement, ARec, Command, DBAccess. DBLog, DBQuery, EventLog, EventOutput, MultiMedia, Notify, Recipe, Scheduler, Script, Shift, Shift_Adv, Statistical Agent* types make up this group
- The *Statistical* Agent type performs statistical analysis of samples from continuous values.
- The *Alarm* and *AlarmList* Agent types perform the alarming of values and the display of these alarmed values. The alarming functionality of Adroit can fulfil even the most complex Alarm strategy and would be at this stage where users would start using and customising these Agents.

4.3.1. AgentGroup

Agents that are considered to have something in common can be grouped together by means of AgentGroup Agents. For instance, all Agents of type Analog can be grouped in an Agent group called ANALOGS, and all Agents belonging to the North Plant of a process plant would be grouped in the NORTH_PLANT Agent group. AgentGroups can, and usually do, have overlapping memberships.

By collecting Agents together in Agent groups, any changes made to certain AgentGroup status bits will affect the status bits of all Agents in that group. As an example, consider the situation when part of a plant (say the North Plant) is shut down for routine maintenance, and it is required to disable scanning of all Agents belonging to this plant section. By setting the Scan Inhibit bit for the NORTH_PLANT AgentGroup Agent, the Scan Inhibit bits would be set for all the Agents that belong to the North Plant.

4.3.2. Alarm Agent



In Adroit, incidents (alarms or events) are handled by *Alarm Agents*, which control the presentation and routing of the alarm events. Typically, each Agent Server is configured to contain an alarm Agent that routes the incidents to one or more *alarm lists*, output devices, audio devices and event logs.

The flexibility in configuring alarms allows the user total freedom in formulating an alarm strategy. Users may create as many alarm *types* as needed, which are the conditions or states of each Agent that you need to alarm. Any status bit that is set by an Agent can be used for alarming that Agent. In this way, the criteria for what constitutes an alarm condition can be left up to an Agent to decide. This is why alarming is Agent type-specific.

An incident is associated with a routing path, an alarm name and an acknowledgment option, as follows:

In the example above, an Analog Agent ANA001 has an alarm type Scan Inhibited (the user can define any alarm type name), which is activated when the scan inhibit bit (bit 8) in the status word of this Agent is set. This incident is routed to the following destinations:

- A Multimedia Agent which could display a video or play a sound file;

- To a local printer for a hard copy record;
- To an AlarmManagement Agent (AMA) which allows you to either keep a simple record and view of historical alarms or to license and use the Adroit Alarm Management and Analysis software to have efficient and meaningful incident reports to identify and remove nuisance alarms along with a host of useful analytics based on world-class alarm management standards; and
- To the event log on this computer.

Each alarm type can *report* the status of any number of applicable slots. Taking an Analog Agent as an example, the type HI can be configured to report the values contained in the *value*, *high alarm limit* and *units'* slots to the alarm list. The entry in the alarm list will then show the occurrence of the alarm event together with the actual value of the Agent, the engineering units and the associated high alarm limits. You can configure alarm types directly from the Alarming dialog and specify the **Priority** of each incident (**Current** alarm type), in addition to assigning an Agent-specific priority. So in the case of an Analog Agent the HighHigh incident can have a higher priority than the High incident, etc. Typically you prioritize your incidents to ensure that your operators ONLY receive IMPORTANT alarms since some alarms are critical, whereas others are more informational in nature.

You can also configure a **Delay** (secs) for each incident (**Current** alarm type). Typically this is added to delay an active incident from entering the alarm system to prevent intermittent alarms from flooding your alarm system or to ensure that this alarm is raised when the operator needs to respond to it.

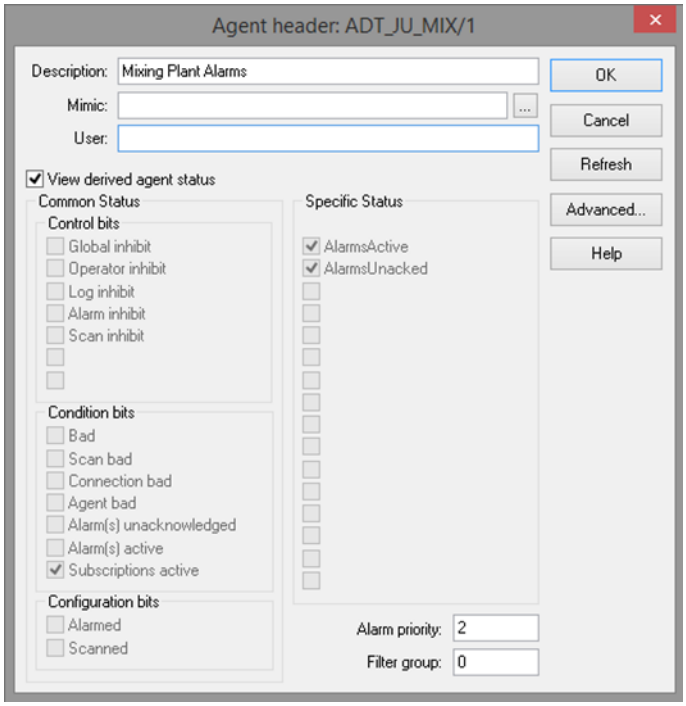
Do **Conditional expression** to ensure that each incident is unique and does not duplicate existing active incidents. In this way you can implement first-up alarming, which disregards the incidents that echo existing problems, so that your operators can focus on the real issues at hand.

It is also possible to do Alarm shelving, commonly used in areas where maintenance is being carried out and the user does not want false data making it's way into the Alarm Management Reports.

If necessary, you can temporarily suspend the alarming of an incident for a predefined period of time, using the **Inhibited until** option.

This option should be used when an alarm indicator may be malfunctioning or may need to be repaired or replaced or in situations when you need to disable ALL the alarms of a particular item of equipment and you cannot disable the alarming of an Alarm Agent.

4.3.3. AlarmList Agent

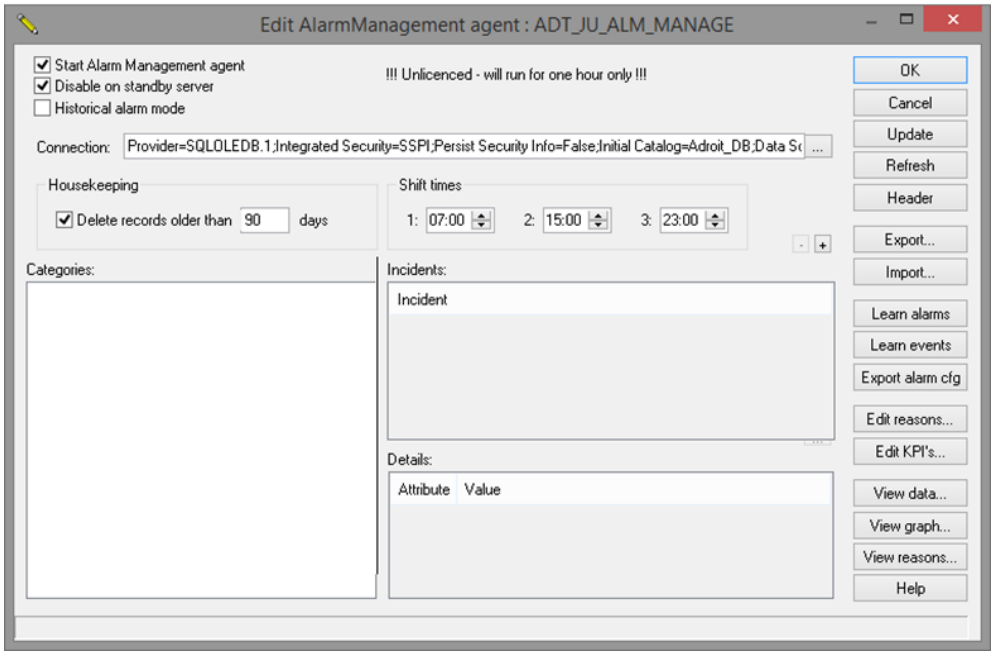


AlarmList Agents are the *middle ground* between Operator and server-side alarming capabilities in that they act as routing destinations for Alarm Agents and data sources for Alarm List Viewers, used on graphic forms.

4.3.4. AlarmManagement Agent

Alarm Management is an important aspect of process control required to counteract inefficient alarming.

Alarm management allows you to create an alarming system that alerts the operator to the most relevant alarm for the current incident. In this case fewer alarms are more effective as the human mind can only focus on a few things at a time.



The AlarmManagement Agent logs all incidents and their details, including their time of occurrence, their time of acknowledgement and the time at which they were cleared to a database. This repository of incidents then allows you to do two things:

- To create a repository of all historical alarms
- When used in this mode the Agent will log the relevant information surrounding an alarm to a database. The results can be then displayed, filtered within the Operator through a custom windows control developed for the purpose.

To create a repository of all historical alarms that contains much more around alarms given that you can categorise alarms, add related process values for post analysis and reporting on your alarm system. This is done through a set of standard reports that can be run in a Browser and assists in developing and

supporting a continuous improvement strategy around known ISA and the EEMUA standards. This is a licensed Agent and should you wish to use this product please contact your Sales engineer.

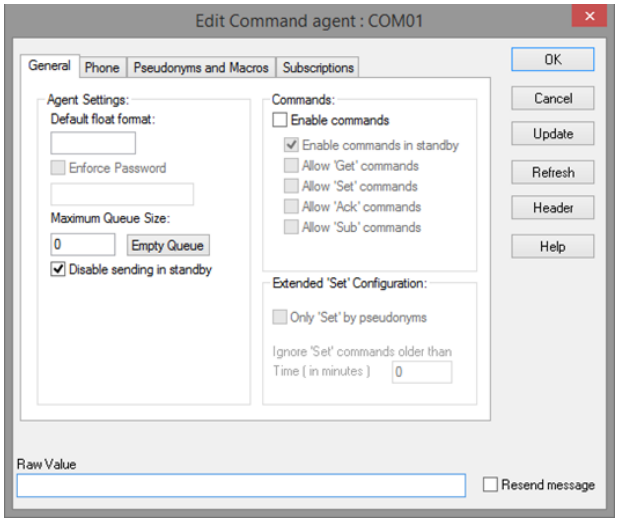
- Generate statistics from the data produced by the alarming system. This allows you to have efficient and meaningful incident reviews to identify and remove nuisance alarms by analysing the counts, frequencies, durations and acknowledgement times of alarms.
- Measure KPIs (Key Performance Indicators) to improve your alarming configuration.

An alphanumerically sorted category and incident tree making it easier to find the required incident and/or category; a KPI editor (click the **Edit KPI's…** button) to easily edit the values of the KPIs so they better reflect the goals of your own process.

An improved category tree that provides a separate **Incidents** list that enables the bulk-selection of incidents when manually adding and removing incidents to/from categories. The **Learn alarms** or **Learn events** buttons allow you to add all your un-configured incidents to the **_Uncategorized** category.

If you have purchased and licensed the Alarm Management and Analysis tools these plug directly onto the database for out of the box Alarm Management reports. If you haven't Adroit ships with a Historical Alarm Viewer to view all the historical alarms.

4.3.5. Command Agent



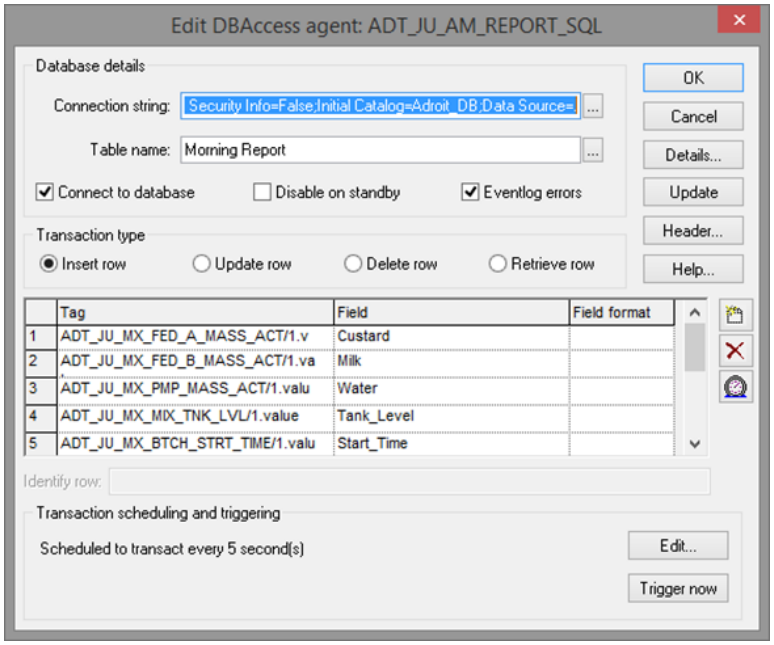
Command Agents can both send and receive SMS messages via a SMS GSM modem. SMS messages can be sent to a single recipient either manually or automatically, e.g., to remotely monitor the status of important process values. Sent SMS messages can also be monitored for auditing purposes and if they are not received and/or responded to within a specified period, can also be sent via another Command Agent to another recipient.

However, the name of this Agent is derived from its ability to receive SMS messages or textual commands from one or more callers that it can perform to remotely configure Adroit, for instance, by acknowledging alarms, setting process values, etc. In this way it is possible to remotely supervise and, if necessary, control your process.

Command Agents bypass the scanning subsystem by using a device of this new GSM_SMS driver directly. This means that instead of being restricted to only 79 characters, the full 160-character SMS message length can be utilized, both when sending and receiving SMS messages.

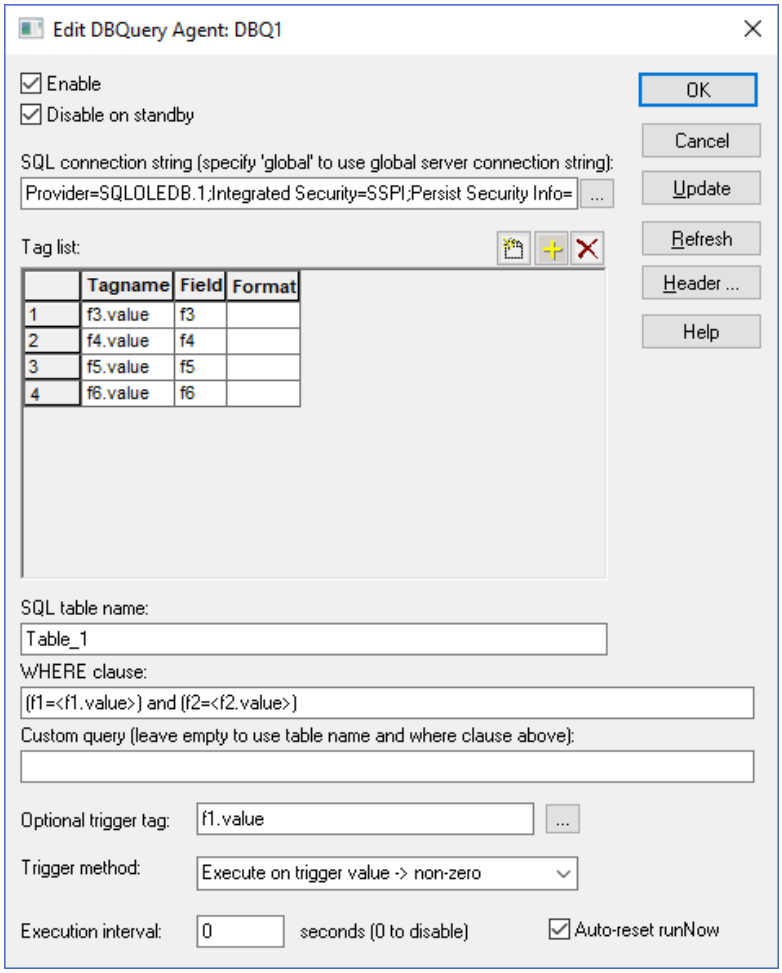
4.3.6. DBAccess Agent

DBAccess Agents provide a link between Adroit and various external databases such as SQL Server, MS Access, FoxPro and Oracle. To achieve this, the DBAccess Agent uses Microsoft's ADO technology (ActiveX Data Objects), which means that links can be established to any database product that has an ODBC driver or an OLE DB provider.



Each DBAccess Agent represents a transaction on a row, in a table, in a database. Append, update, delete and retrieve transactions are supported. These transactions can be done on demand, they can be scheduled and/or they can be triggered. Furthermore, extended SQL Querying has been added to identify the row upon which the transaction is being performed.

4.3.7. DBQuery Agent (ver 10.0.5)



The latest enhancement to this technology is the DBQuery Agent that allows the user to execute queries on a database that return a single record that would be available as the .value slot of the Agent.

The DBQuery agent has been created to supersede the older DBAccess agent technology and provides a more robust and improved methodology for executing OLE DB transactions asynchronously without causing agent locks and potentially freezing the agent server.

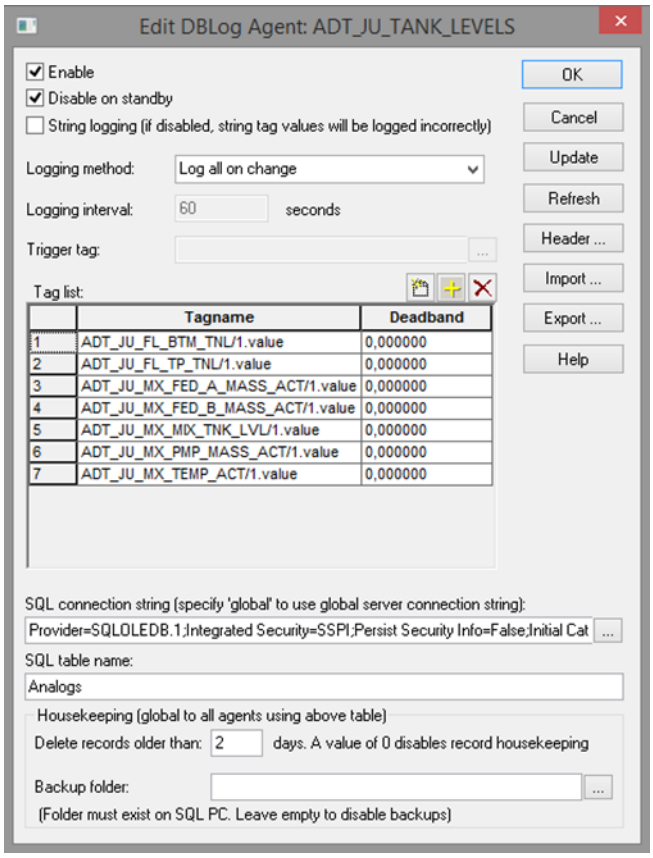
Where the DBLog agent is designed to easily perform INSERT transactions, the DBQuery agent is designed to easily perform SELECT transactions on a database - however the DBQuery agent can be configured to perform any SQL Query transaction on a database using the custom query option.
The agent can be configured to be run manually, triggered from a tag or periodically

4.3.8. DBLog Agent

This Agent is a bulk OLE DB logging Agent for typically logging Analog and/or Digital values to a specific table in a database for later retrieval. In other words, instead of logging each Analog and/or Digital value separately, you can simply specify a list of tags to log and either export/import this list from a file or browse in this list individually in the required DBLog Agent.

This Agent also provides:

- A **Logging method** list box to select the required methods for specifying when and how the listed tags are logged.
- A **String logging** checkbox (the BOOLEAN **strLog** slot), which when enabled logs the tag values as Strings and not numeric values; this also allows you to log other string slots (this will log the first 79 characters).
- A **Housekeeping** section to specify how long, in days, that you want to store your logged tags. This depends upon the storage capacity you have available for your database and how useful this data becomes to you the older it gets.
- A method of dealing with a problematic connection to the specified database (if the connection is lost) or when this Agent is logging a large number of tags frequently so that it becomes overworked.



4.3.9. EventLog Agent

Adroit event logging is handled by Eventlog Agents that use the standard Windows based Operating System functions for storage but supplement these facilities for retrieval and display. The number of logged events is limited only by disk space, or the user can limit the number by setting a fixed length file. With a fixed-length file older events are overwritten. Events can be directed to event log Agents anywhere on the network.

The scope of these events are extensive, including alarm events, control events and system events: *alarm events* are caused by Agents going into alarm, being acknowledged and cleared; *control events* are generated by any operator control action (e.g., opening or closing a valve, changing the set point of a process controller, etc.). The operator logging on or off, a start-up and shutdown of the control system, printer out of paper, communication failures or configuration activities (adding, deleting or modification of Agents), etc., are examples of *system events*.

This Agent can to log events to an OLE DB compatible database such as MS Access or MS SQL.

4.3.10. EventOutput Agent

The EventOutput Agent directs event log messages to output devices (printers, serial ports and parallel ports), and to SQL compliant databases. To cater for the various output devices, EventOutput Agents accept a variety of user configurable general fields that are used for the layout and format of the data, and the device specific information.

Output of data to the device will depend on the *triggering criteria* selected: on demand, line at a time as the event occurs, only after *x* lines, based on time criteria or as a combination of the above.

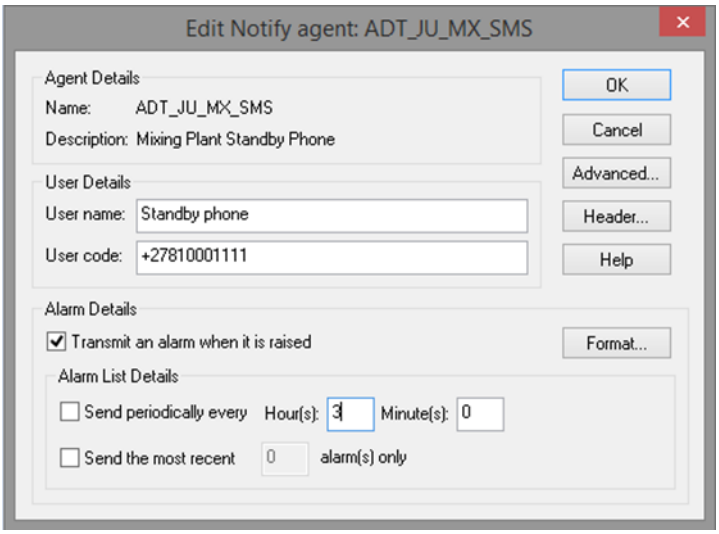
4.3.11. Multimedia Agent

A *Multimedia* Agent can be used to play .WAV, .MID and .AVI multimedia files to create special audio and/or video effects on a multimedia workstation.

4.3.12. Notify Agent

The Notify Agent allows alarms to be routed directly to a designated person, using:

- A pager service;
- A mobile phone using the Short Message Service (SMS); and
- An email address.



The alarm message to be transmitted can be configured to include up to 15 fields (Action, Agent, Alarm Type, Category, Computer, Date, Description, Event ID, Extra Data, Filter Group, Priority, Reported Data, Source, Sub-Source and Time), and custom messages can be sent on demand by the operator.

The Notify Agent can now transmit alarms by email to the required recipients by using the Email driver.

This is in addition to transmitting alarms as text messages via cellular phone Short Message Service (SMS) or a paging service directly to a designated person.

The Body and/or Subject line of these automatically sent email messages can contain text macros that are parsed to before the message is sent.

In addition to automatically sending alarms as email messages, the Notify Agent can also be configured to allow your operators to manually send email messages via an appropriately configured mimic or graphic form.

4.3.13. Recipe Agent

Inspired by S88 standard the Recipe Agent allows the SCADA to be an execution engine to execute Master and Control Recipe configurations.

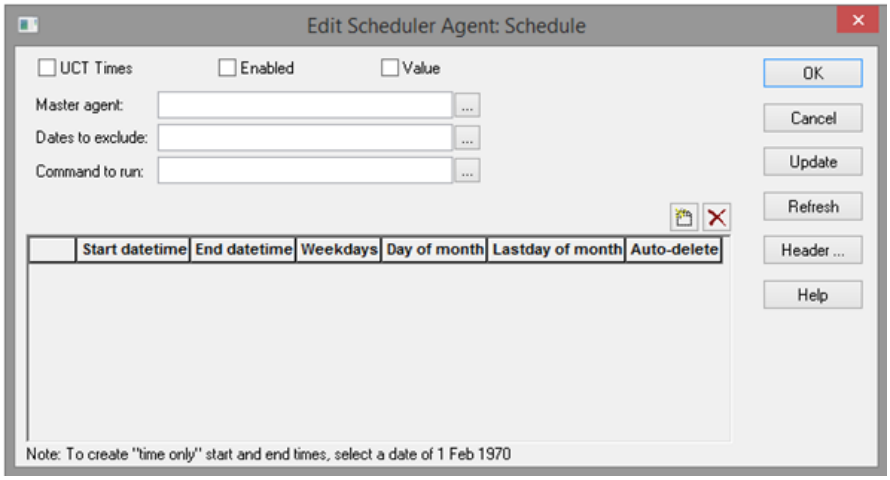
A recipe is a statement of the ingredients and procedures necessary for a batch control system to process its raw materials to make a desired product and by-products. The recipe Agent consists of a master recipe and one or more control recipes.

The master recipe contains a list of all ingredients and the operations to be performed on them, and the control recipes consist of quantity, timing and sequencing information. The Agent provides basic sequencing and batching and reporting capabilities and is designed to be used in conjunction with other Agent types such as Expression Agents.

4.3.14. Scheduler Agent

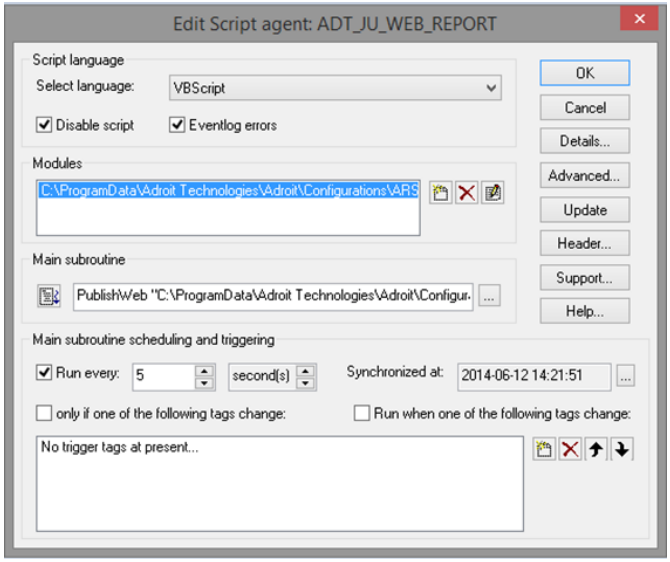
The Scheduler Agent can be used to define one or more periods of time during which its BOOLEAN output will be enabled. Alternatively, it is also possible to manually set the output.

This output can be used as an enabling/disabling tag for other Agent Server configurations; scanned to a front-end device, and/or an application can be run as soon as the output becomes enabled.



4.3.15. Script Agent

Adroit Script Agent provides a basic scripting language allowing users of Adroit to automate complex or repetitive tasks. Two scripting languages are supported – Visual Basic (VBScript) and Java (Jscript) – both allowing users to define their own subroutines and functions and pass data to other applications using OLE Automation.

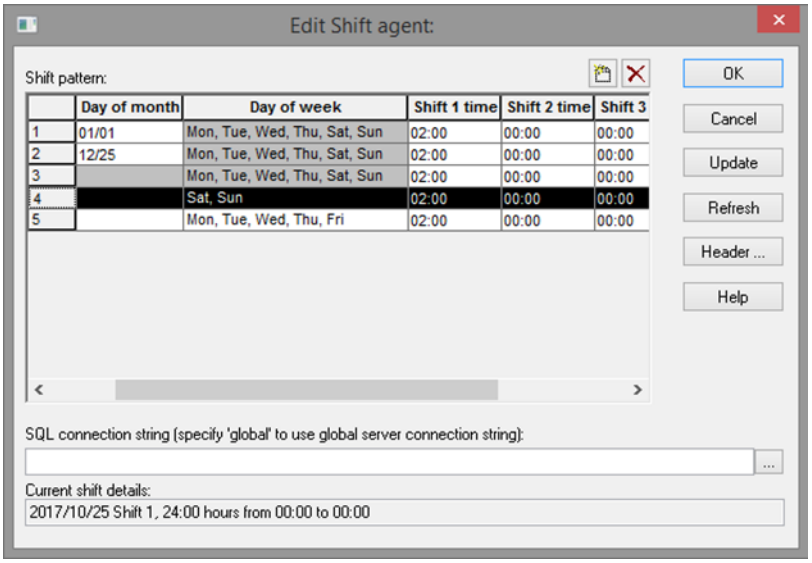


A script may be employed as an alternative to a recipe and can be used to perform simple scheduled tasks. The Script Agent can be executed on demand, by scheduling, triggering and by using a combination of triggering and scheduling.

A number of functions are supported to allow interaction with Agents within an Adroit system and include but are not limited to *GetTag, SetTag, and TriggerValue* allowing complex calculations to be carried out programmatically.

4.3.16. Shift Agent

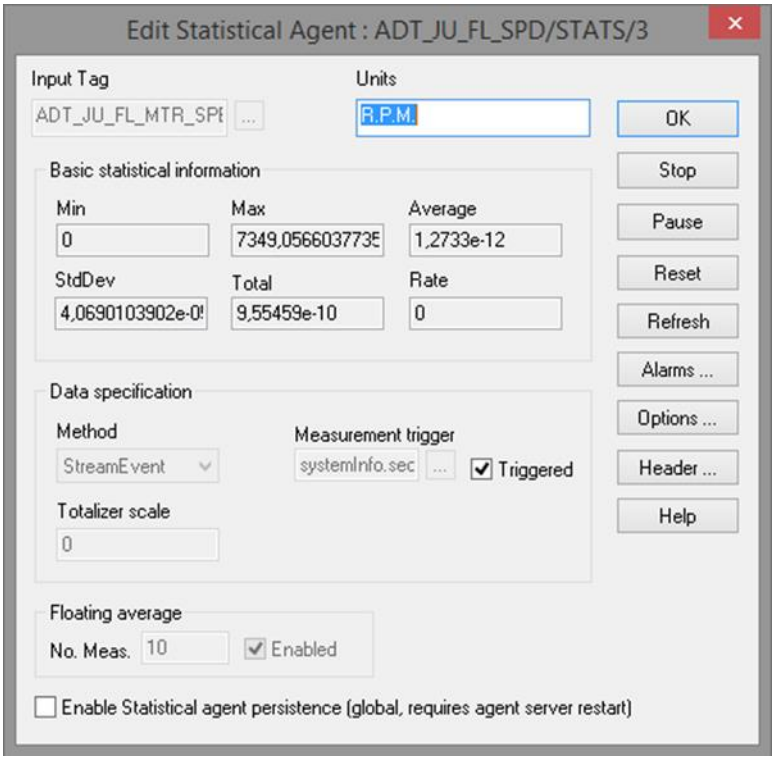
Various Agents in Adroit, such as OEE, require knowledge of the current shift number. Instead of configuring the shift pattern information into each instance of these Agents, you can instead configure these shift patterns globally, in a Shift Agent, for use by any interested Agents.



The **value** slot of the Shift Agent should also be scanned and output enabled to any PLCs that perform any shift-end functions, such as resetting shift count accumulators.

The Shift Agent continually monitors the current time and checks it against the configured shift pattern to determine the current shift number. If NO matching shift pattern is found, the shift value is set to 0.

4.3.17. Statistical Agent



A statistical Agent provides the basic functionality for a User to build a statistical process control strategy using its built-in data collection method and by linking the Agent to other Adroit Agents.

The method of data sampling offered by the statistical Agent is called streaming, whereby all the collected input values are placed in one sample for analysis. Each input value may be collected on an event basis. The statistical Agent calculates maximum, minimum, average, standard deviation, total and rate for all the values collected.

4.4. MIS/MES Agent Types

These Agents are the building blocks around getting more from your Adroit system. They are very specific and usually quite complex Agents used to do monitor, control and solve higher level shop-floor and general industrial challenges. The MIS/MES Agents are summarised here

The *Accumulator, OEE, PID and MaxDemand* Agent types make up this group.

4.4.1. Accumulator Agent

An Accumulator agent gets its name from the type of process value that it operates upon and NOT from what it actually does. Since an Accumulator agent requires as an input a value that accumulates over time in other words which simply gets bigger and bigger, which the Accumulator agent uses to calculate the difference in value (or delta) between successive events or over a specific time period (interval).

This agent calculates the total for a period based on a predefined trigger or time interval. When such an event occurs:

- the agent subtracts the previous interval input value from the current input value to determine the value differential for the current period, which represents for example the total production for the last hour or shift or day etc.
- the agent calculates the rate of change in value, which is this differential value for the current period divided by the interval of this period (value per hour).

In order to maintain a history of these periodic counts/totals, you need to connect this Accumulator agent to a MS SQL database. Each record specifies the start and end times of the period and its start and end values and the differential value for this period. If you want to provide the automatic management of this data, then a housekeeping period can be defined to ensure that only the most relevant (most recent) records are kept.

You can also configure alarm limits for both the (differential) value and rate of change in value for the current period, which will be used to monitor the current value and/or rate against these alarm limits: low-low, low, high and high-high. Any infringement of these alarm limits causes the agent to set its corresponding status bit, which can be alarmed.

4.4.2. OEE Agent and the new Enterprise OEE Agent

The Overall Equipment Efficiency (OEE)/KPI Agent can be used to measure how efficient a single piece of equipment or piece of plant is performing against the OEE standard. This OEE percentage takes into account all losses in downtime, speed and quality and can therefore be used for analysis and benchmarking.

Note: The new Enterprise OEE Agent is sold as part of the Enterprise OEE product.

Why use the OEE Agent?

The OEE Agent can:

- Either calculate the OEE for the monitored equipment and/or plant, targeting the three key OEE factors simultaneously which can affect significant improvements in production;
- Or measure a specific KPI (Key Performance Indicator, a quantifiable measurement that measures how well something is accomplishing its goals and objectives). In other words, individual OEE Agents can be applied to any component or part of a process for which a variable denoting a measurement of output is available.

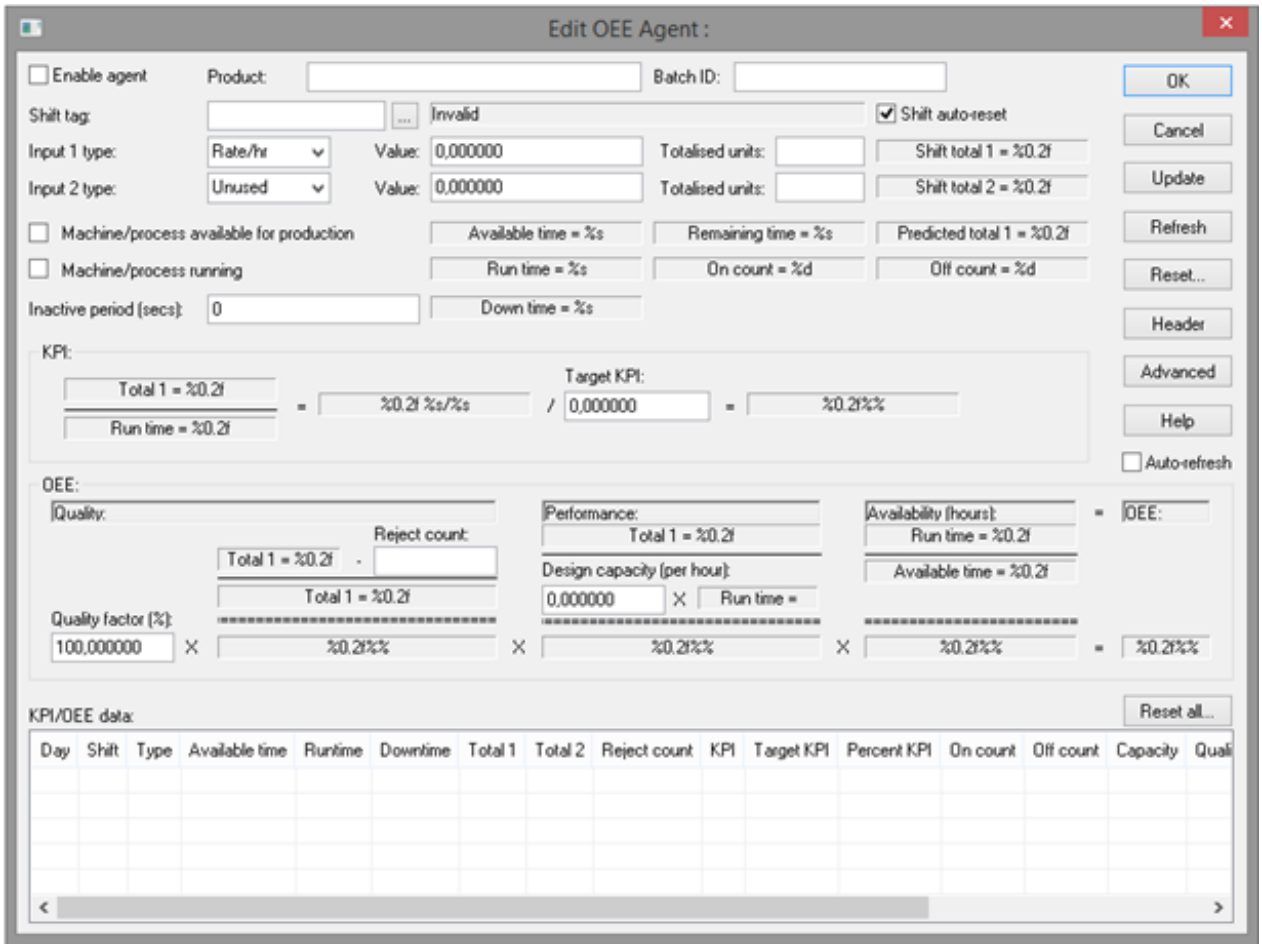
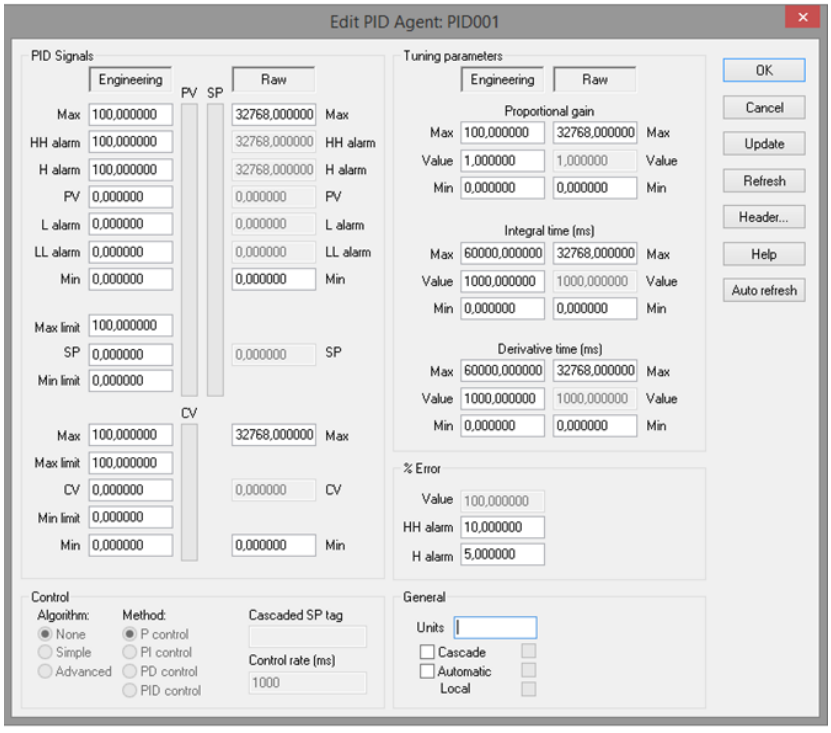


Figure 10 OEE Agent

4.4.3. PID Agent

The PID Agent is a faceplate container for the 25 signals and tuning parameters associated with an external PID controller Agent. This provides great savings in the overall number of Agents required in the configuration of a project that contains PID controllers. These values can then be scanned to the outside world, via their raw slots. The PID Agent also supports real-time catch-up.

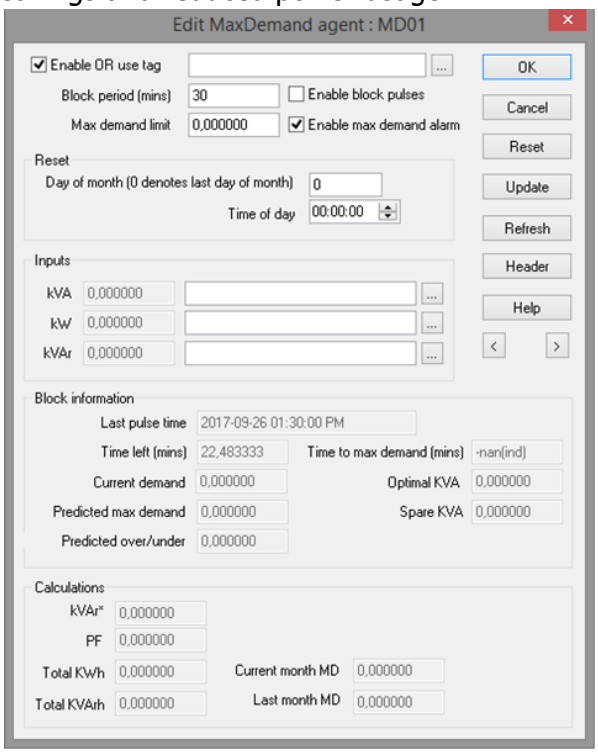
There are configurable high and low operation limits for the SP and CV slots, which will prevent these values from being set to potentially harmful values when they are being manually configured.



4.4.4. MaxDemand Agent

Maximum demand is an additional cost for most non-domestic power users. It is the method used by your electricity provider to control how much electricity you use, especially at peak times, when your

usage coincides with peak domestic usage. Monitoring your maximum demand can result in considerable savings and reduced power usage.



However, it is recommended to automate this monitoring by using a MaxDemand Agent, because one mistake can easily exceed your maximum demand limit for the current billing period.

The MaxDemand Agent predicts the maximum kVA demand for the current block period interval over which your electricity provider monitors your maximum demand, which is typically 30 minutes, so you can create alarms if the predicted consumption exceeds a predefined limit.

This enables you to control the electricity consumption during each block period and prevent your actual "maximum demand limit" from being exceeded.

Usually, control entails turning off machines which do not affect the main production process or which are not essential. While the primary aim of this Agent is to monitor your power usage, it can also automate the control of your electricity consumption. In other words you can use its "MaxDemand" status slot in a script or expression Agent to perform the necessary remedial actions.
The MaxDemand Agent can also calculate your instantaneous power factor, which also has the potential to save you money.

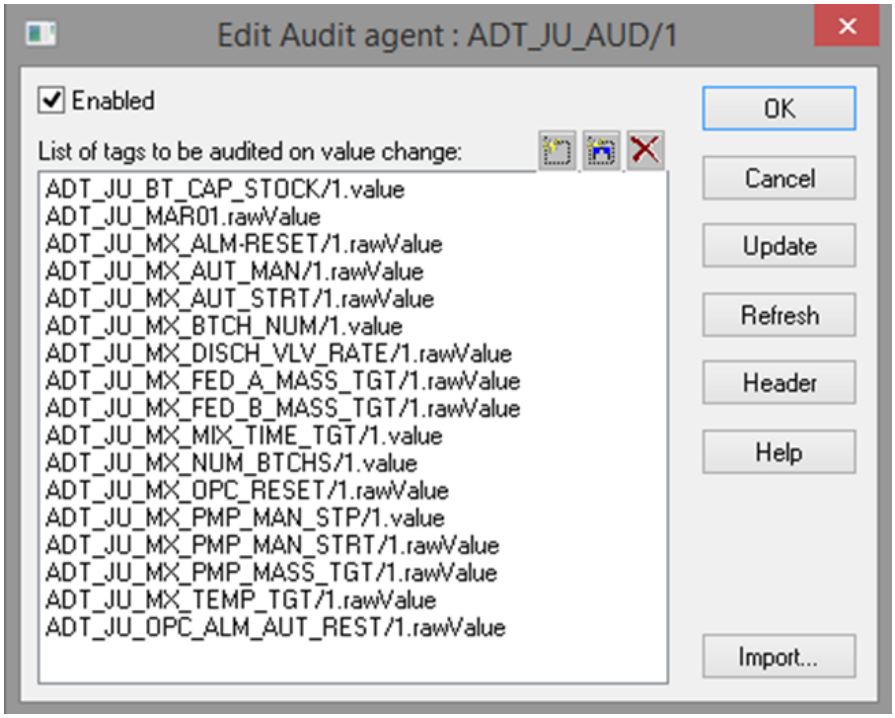
4.5. IT Agent Types

These Agents are the building blocks around integration and management of SCADA and IT and are summarised here and the most important IT Agents are described in more detail below.

- The *Audit, DumpConfig, ICMP, Perfmon and SNMPMgr Agent* types make up this group.

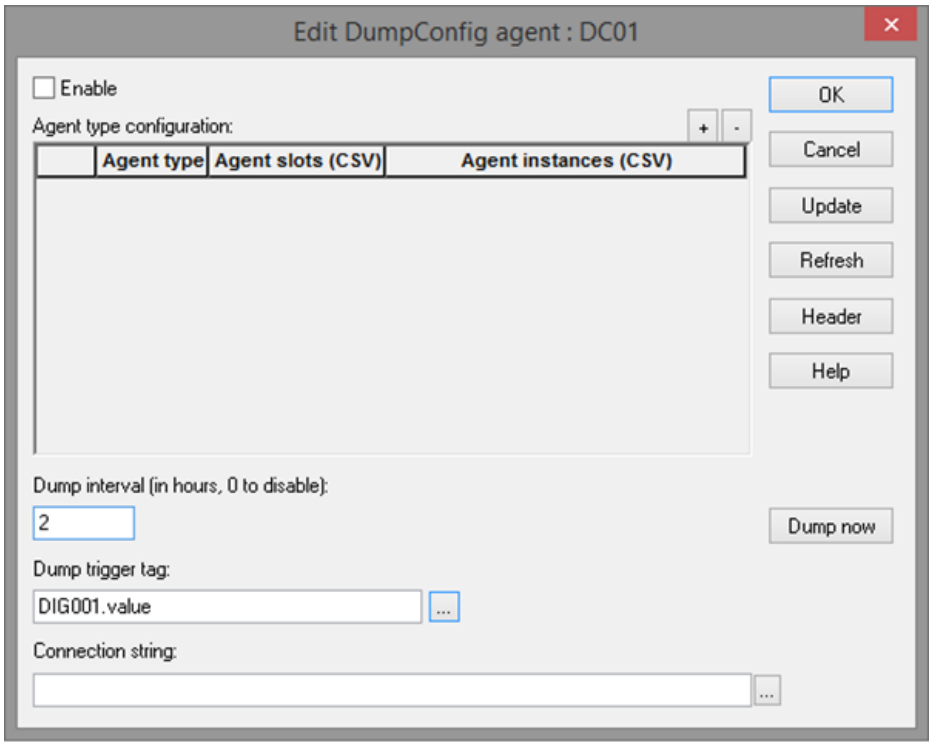
4.5.1. Audit Agent

This Agent supplements the auditing or logging of the Agent Server sessions and selected activities performed during these sessions to an OLE DB database. Use the Audit Agent to specify one or more tags that you want to monitor for ANY change in value. When a value changes for any reason, the Agent logs this change of value to the audit database. In other words, EVERY value change for EVERY specified tag is logged, regardless of whether Adroit or a PLC or something else causes this change in value.



Typically you will ONLY audit set points or alarm levels but not temperature or level values. In other words, do not specify process tags whose values that are required to change frequently as part of the normal operation, as these will simply flood your database with unnecessary records.

4.5.2. DumpConfig Agent

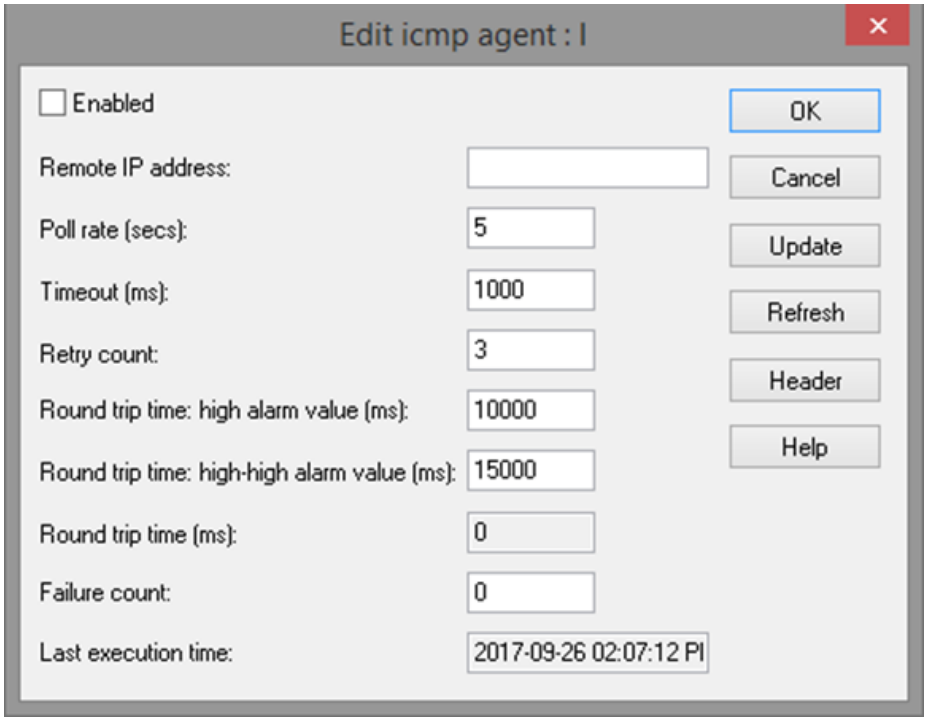


This Agent simply stores the configuration of specific Agents in an OLE DB database.

One example for its use is when you need to report on Agents in the context of their configuration at the time of the report, alternatively this could be used as an extensive method of auditing changes made to specific agents.

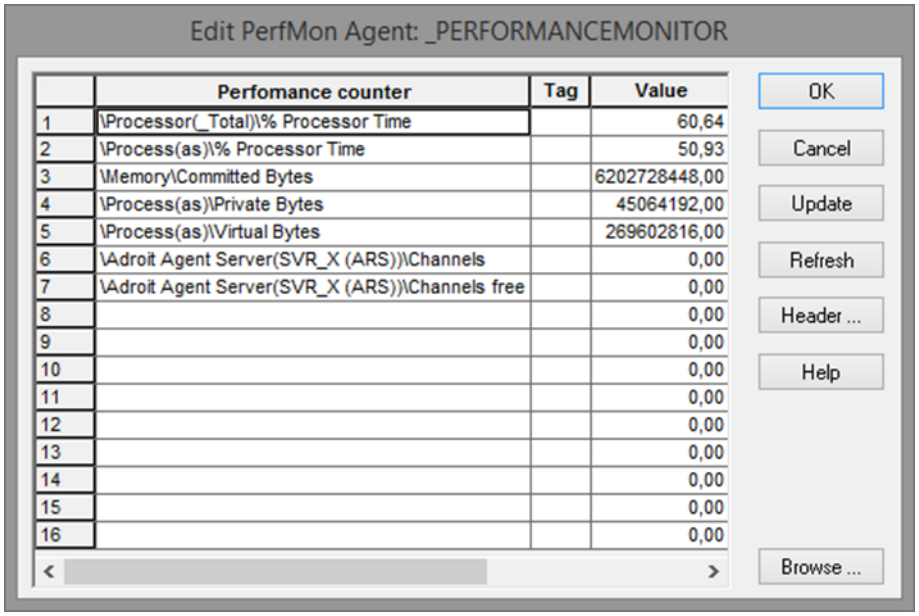
4.5.3. ICMP Agent

The ICMP or PING agent uses the Internet Control Message Protocol to determine the existence of a remote device on the network. Any IP network device is able to send, receive and process ICMP messages. These messages are only used for troubleshooting network problems and not to transfer data. The most common use of the ICMP protocol is by the PING diagnostic utility.



By continuously polling the specified remote IP address, this agent is therefore able to identify potential network problems as they occur by generating alarms when errors are obtained when attempting to contact the specified remote IP address and/or when the round trip time exceeds specified High and High-high times. This round trip time is the time that it takes to send an ICMP message to the remote IP address and to receive its reply.

4.5.4. PerfMon Agent



The PerfMon Agent is used to monitor Windows Performance counters, which belong to performance objects such as the Agent Server, Processor, Memory and others. These counters facilitate the monitoring of the computer's performance and assist in detecting performance bottlenecks that could adversely affect the functioning of Adroit.

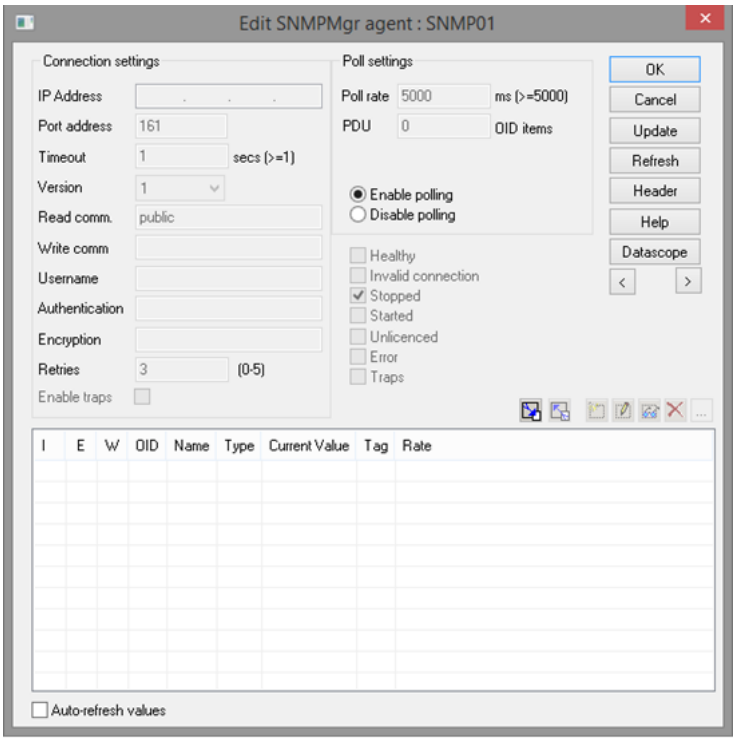
The values of these counters are updated every second and can also be assigned other Adroit tags, such as Analog Agents, which can then be appropriately alarmed. Up to 16 performance counters can be monitored per Agent.

4.5.5. SNMPMgr Agent

The SNMPMgr Agent (short for SNMP Manager) integrates network management using the Simple Network Management Protocol (SNMP) with the MAPS SCADA (Supervisory Control and Data Acquisition). SNMP has become the *de facto* standard for network management today and can be used to facilitate the management of diverse computer networks.

Examples of devices that can be monitored using SNMP include routers, switches, hubs, UPSs, printers, modems and even computers.
Each SNMPMgr Agent can access the data relating to the operation of a single device, in order to:

- *Monitor this device* by writing one or more of its values to Adroit tags. If necessary, you can retrieve values at different rates. This allows you to trend the data such as bandwidth, alarm the data such as when particular conditions arise like a device becoming unavailable, and log the data to analyse it for troubleshooting purposes.
- *Control this device* by using Adroit tags to set the values of this data. This allows you to change set-points within the managed device that control its operational functionality.
- *Receive SNMP traps (events) from this device* so that you can react to those that are important. This allows you to reduce the number of values that you need to monitor (poll) for each device. In other words, instead of monitoring every value on every device by using SNMP traps, a device can simply inform its SNMPMgr Agent of the important values without solicitation.



With the inclusion of this Agent, Adroit can manage a process network (in which process values are of concern), alongside a physical network (in which issues such as bandwidth and availability are of concern). Other features include the ability to import and export the list of selected values and the ability to select and configure the records in this list of selected values externally, such as from a mimic. A global SNMP configuration file allows each SNMPMgr Agent to store an unlimited number of values.

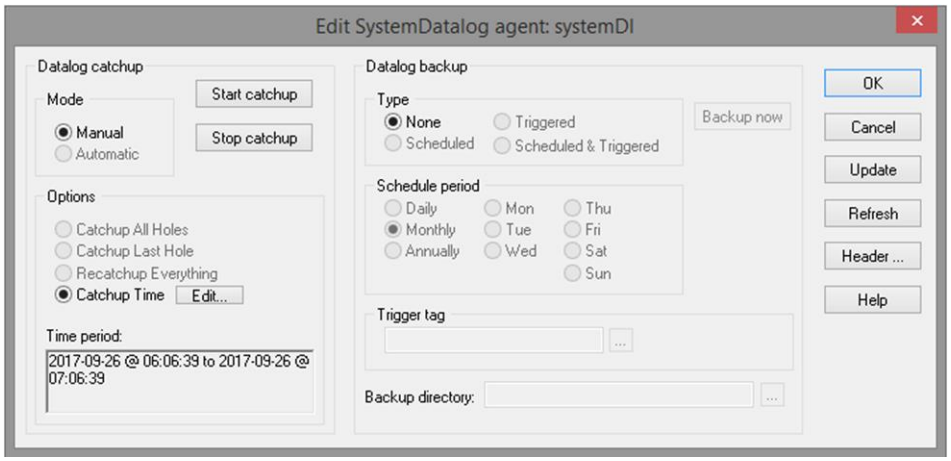
4.6. System Agent Types

System Agents are used internally by Adroit to perform certain SCADA and other specific functions. Instances of these Agents are created automatically and cannot be added manually by the user.

The System Agents are summarised here and the most important System Agents are described in more detail below.

- The *Alias, Beeper, DataLog, Device, Hasp, Proxy, Scan, SystemDataLog and SystemInformation* Agent types make up this group.

4.6.1. SystemDataLog Agent



Every Agent Server contains a single *SystemDataLog* instance called *systemDI*. This forms an integral part of the Adroit clustering subsystem that deals with the synchronization of all of the datalogs on the two nodes in an Adroit cluster.

This Agent is also responsible for providing the global administration of the **datalog backup** facility that enables the long-term storage of data logged data in multiple .CSV files.

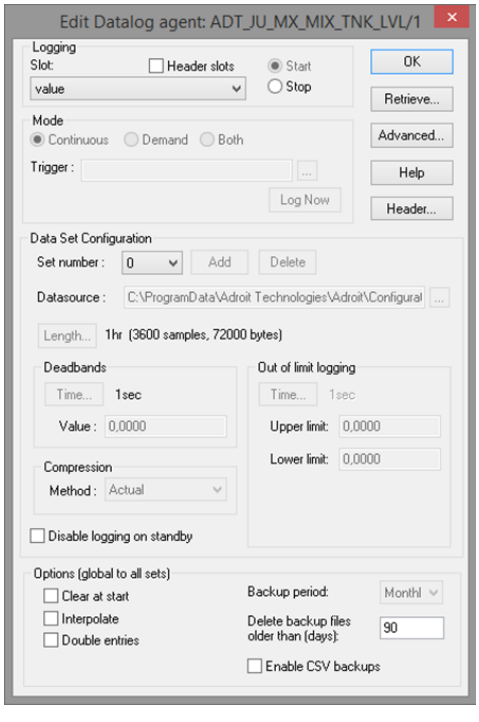
4.6.2. Alias Agent

Alias Agents are the fundamental *indirect scanning* mechanisms in Adroit. They act as a references or pointers to other Agents in the system. As a reference to the Agent it is aliasing, the Alias Agent has the same behaviour and state information but differs only in identity.

4.6.3. DataLog Agent

Adroit provides extensive historical logging facilities, either by using the proprietary data logging facility or by specifying the required OLE DB compliant database to log the data.

If this latter method is used, one can specify which table is used for logging this data.
Any attribute of any Agent can be configured to have its values historically logged, subject to the proviso that it is of type *Real, Integer* or *Boolean.*
Logging can take place either to native Adroit binary files or to any ADO datasource (SQL Server, Oracle, MS Access, etc.).
When logged, the values are time stamped and are made available via a DataLog Agent, to any client such as historical trends, reports, external spread sheets, etc.



Select the **Enable CSV backups** option to ensure that the logged data is backed up into multiple .CSV files for long-term storage. This logged data backup facility is globally managed by the SystemDataLog Agent.

Select the **Advanced...** button to configure ALL the global settings that affect the operation of the data logging subsystem.

Historical values are stored in a central file, with each DataLog Agent "owning" a defined portion of the file. The total amount of disk space required for data logging depends on the total number of values logged, as well as the rate at which and period for which each value is logged.
The following formula can be used to calculate disk file sizes:

		Formula
Index file size (bytes)	=	8 + (56 * number of logged values)
Raw buffer size (bytes)	=	20 * (Period/Rate) rounded up to next multiple of 8
Raw file size (bytes)	=	∑ Raw buffer sizes
Total disk space	=	Index file size + Raw file size

By default historical logging is an event-based function; in other words values are only logged when they change and only then if the amount of change exceeds the value deadband parameter.
When the value of a logged point does not exceed the previously logged value by more than the value deadband, the value is not logged.

Similarly, values that change more frequently than the user-defined time deadband will have a value logged that depends on the logging method selected: actual value, average value, minimum value or maximum value, as follows:

Value	Description
actual	The value logged is the value at the end of the time slice.
average	A time-weighted average is maintained of all the changes that occur and logged at the end of the time slice.
minimum	Only the lowest or minimum value is logged at the end of the time slice.
maximum	The highest or maximum value is logged.

The formula presented above calculates a raw log file size based on the value changing exactly at the logging rate (i.e., worst case scenario).

However, in a typical application, a compression factor of 3 to 5 or more is likely to be achieved. This means that the actual period for which data is retained may be 3 to 5 times longer than the specified logging period.

It is possible to override the default on-change-only logging behaviour and cause values to be logged periodically or according to some other application-relevant trigger criteria.

Historically logged values can be retrieved for viewing within Adroit or exported for further processing as a .CSV file. This file format is recognized by many commercial spread sheet and database packages.

Adroit also provides a means of merging values from a .CSV file into an existing historical database of Data Log Agents. The .CSV file may have been originally created by the Extract utility or exported directly.
Out of limit monitoring enables Datalog Agents to immediately log out of limit values (values that are not within their specified operating range), by ignoring the normal time deadband value and using a faster log rate specified by the out of limit time deadband instead.

Each Datalog Agent can create their own backup (.LGB) files, when their associated .LGD files wrap. Once created, the datalogging subsystem automatically retrieves the data from these .LGB files, if required by trends (charts) or other requests for historical data.

This feature can also extend the potential amount of logged data without creating larger log files, since larger log files have the side effect of causing the Agent Server to take longer to start up.

IMPORTANT: When backing up datalogs we recommend the creation and use of .LGB files, which unlike the legacy method of backing up datalogs via the SystemDataLog Agent, can have their stored historical data seamlessly retrieved for trending and reporting.

4.6.4. Device Agent

Device Agents, in conjunction with *protocol drivers*, coordinate scanning in Adroit and are responsible for creating, deleting and otherwise configuring *Scan* Agents which handle the actual scanning to and from front-end devices.

The number of device Agents in a system will depend solely on the number of different front-end devices. A system with one PLC controlling, say Boiler 1, would have only one Device Agent called BOILER1 (or any other name chosen by the user). On the other hand, a system with many different front-end devices, possibly of different types, will have one Device Agent per device.

Each scan Agent is in fact a separate, independently scheduled Windows thread which means that genuine, concurrent, parallel scanning will happen in Adroit whenever the characteristics and/or connection(s) to the front-end device(s) permit.

Due to Adroit's inherent flexibility, it is possible to scan in and out of any slot of any Agent. For example, a Data Log Agent can be started by scanning its *start* slot from an external device, thereby effectively achieving event-triggered logging; a new value can be uploaded to the high alarm limit of an Analog Agent, etc. Each slot to be scanned is associated with an address in an external device, a scan period in milliseconds, and a deadband. A value is only updated into the relevant slot when it changes by more than the deadband setting.

In addition, scanning is the means by which any two Agent slots are "linked" or "connected" together. The implication of this is that Agents can source their data from any other Agent as well as from a front-end device. This feature is particularly valuable when used in conjunction with Expression Agents (the Expression Agent is explained later).

4.6.5. Other System Agent Types

Other system Agent types include Beeper, Hasp, Proxy, Scan and SystemInformation. The *Beeper* Agent drives the built-in PC speaker and is typically used to annunciate alarms. A single *Hasp* instance called LICENCE holds details about how many scan points and users are currently licenced.

Proxy Agents are created automatically when an application requests information about an Agent that exists in a remote Agent Server. The proxy Agent assumes the role of the remote Agent in the local environment and is continuously updated with any changes that occur in the remote Agent. The function of updating the proxy is transparent to applications such as Operators as well as any 3rd party applications interacting locally or remotely with the server containing the proxy.

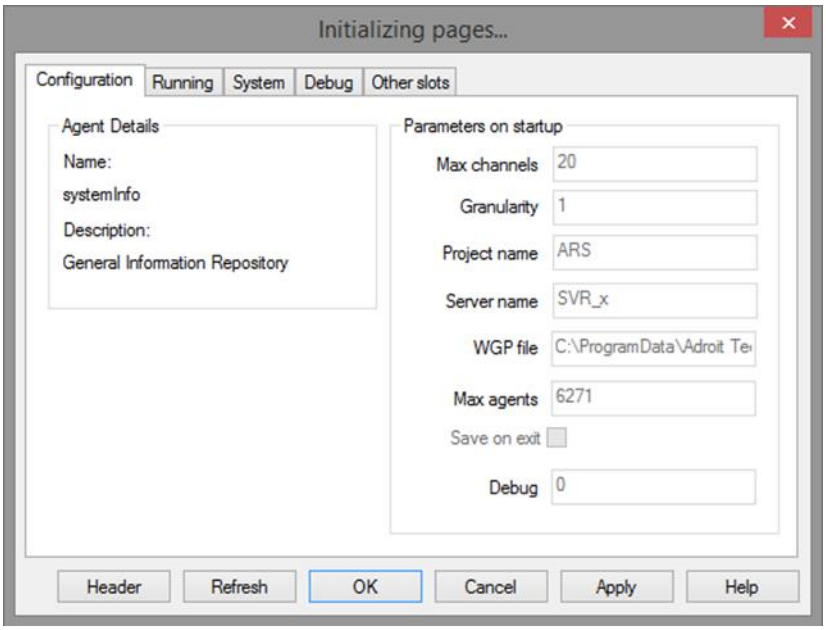


Figure 11 Systeminfo – Configuration

Scan Agents are created dynamically by the system as a function of the user-specified scan configuration and capabilities of the protocol driver mapped by the device creating the scan Agent.

There is also a single *SystemInformation* instance called *systemInfo* that contains a variety of system wide configuration, run-time, and diagnostic information. This Agent contains some very valuable information that can be used to present a more detailed view of what is going on within a Server. Again, as has been explained the information available are all contained with slots and available to be presented on a graphic form or used within a system.

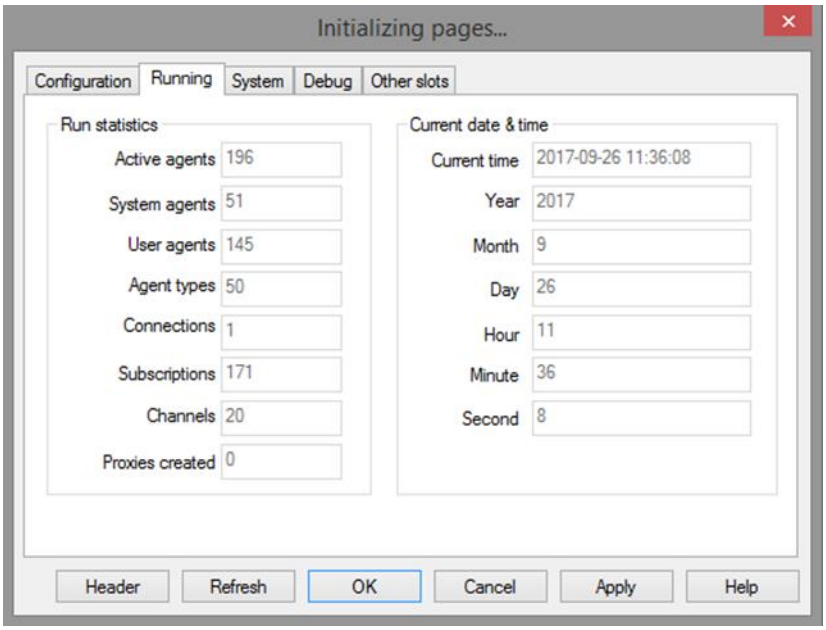


Figure 12 Systeminfo - Running

Examples of useful "Tags" would be

SystemInformation.SystemInfo.*Second* – is the local second of time on the PC and can be used to trigger events or used in scripts or calculations in Expressions

The same applies to *minute, hour, day, dayOfWeek, currentTime, month, year, saveNow, saveOnExit*

4.7. Custom Agent and UDT Types

4.7.1. User-defined Type

Certain new PLCs optimize their communication performance by consolidating multiple data values into a single User-Defined Type (UDT).

Adroit provides the UDT Agent Type Designer to support this new functionality. This utility creates agent types (templates) that duplicate the structure of any UDTs used by your PLCs, by adding properties (basic data types) to them and if necessary, to configure status slots for the purposes of alarming.

> ***NOTE**: This only supports UDTs that are composed of data types that can be represented by Boolean, Integer, Real, Datetime, String (80 characters or less) and VarString values. Array data types are NOT supported.*

Since each defined UDT is a separate agent type, you can add more than one of each UDT to your installation, as is necessary. While this process of defining UDT agent types is similar to the process of creating Custom Agents, they differ from Custom Agents, as follows:

1. UDT agent types can ONLY contain data and do not provide any scripting intelligence and
2. UDT agent types contain a "raw bytes" rawValue slot that communicates the ALL of its other SLOTS (the ENTIRE UDT) in ONE transaction. This accounts for the improved scanning performance of UDTs.

> ***NOTE:** Every instance (agent created) of an UDT Agent type will consume (number of slots-1) scan points. The other scan point will be consumed if and when its rawValue slot is scanned.*
>
> ***IMPORTANT:** UDT agent types must exist on all computers that require access to the UDT agents from the Configurator. Once you have created one or more UDT Agent types, you can import them to other computers.*

4.7.2. Custom Agent Type

Many plants make use of various complex devices that contain one or more value properties. Ordinarily, in order for users to represent these complex devices, multiple instances of Analog, Digital or other common Agent types would be used to represent a single device.

For example, your process may make use of a particular drive type; let's call it Drive Type XYZ. Multiple drives of this type exist in your process. This drive type has a couple of properties, for example Start, Stop, Running, Tripped, Speed and Temperature, requiring each of these values to be mapped to four digital and two Analog Agents respectively, therefore 10 of these drives would mean 60 Agents. Would it not be better to use a single Agent, each Agent representing a complete drive and containing the 6 items of information in 6 slots of the Agent, so that in this example only 10 Agents would need to be used instead of 60?

The Custom Agent has been designed to do just this. It is a generic Agent type used to enable the creation of user-defined Agent types that can be used to represent complex devices and processes. In addition to representing simple "container" Agents, which simply contain or encapsulate the values of a number of previous Agents in a single user-defined Agent, there is also the facility to provide VB Scripting intelligence to the Agent to drive (control) other derived slots of this custom Agent.

For example, we can add an extra slot to this Agent for defining a high temperature alarm value. The VB Script would monitor the actual temperature such that when the temperature exceeds this value a status bit is set for alarming purposes. We can also set a Trip status bit when the Tripped signal is switched on and we can also disable the drive from starting if the temperature is too high. This Custom Agent Designer utility facilitates the creation of these user-defined Agent types and can be launched from the Utilities list in the Diagnostics & Tools tab in the Designer.

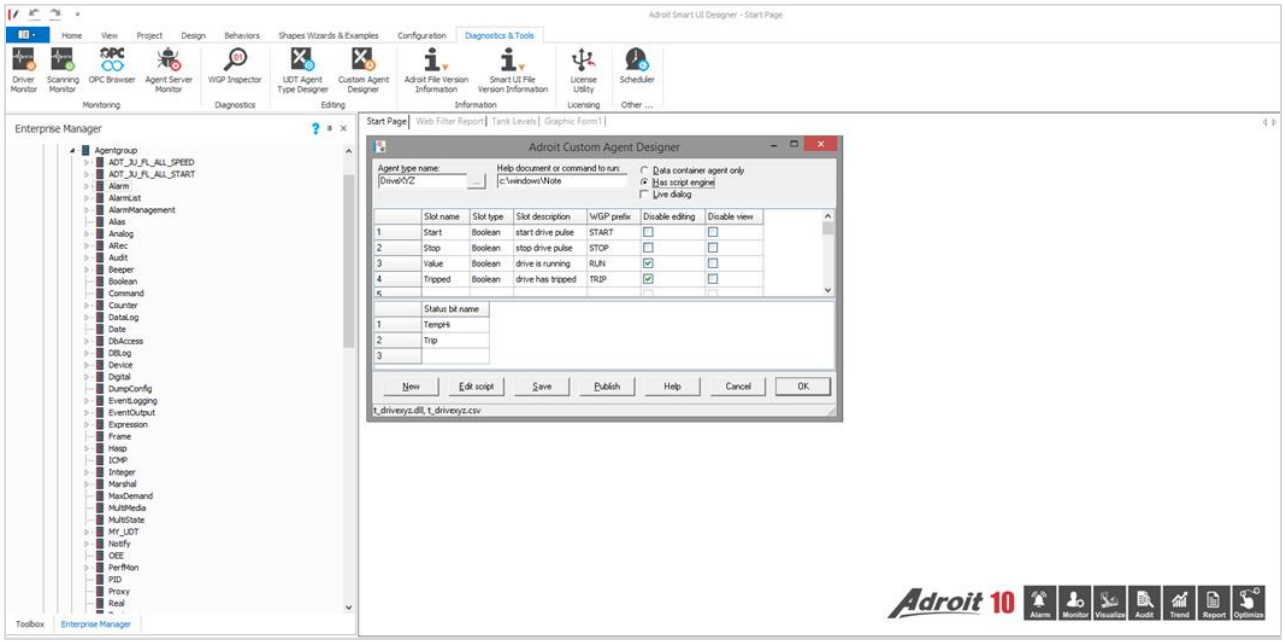


Figure 13 Diagnostics and Tools

5. Detailed Architecture – Client Side

Adroit consists of 3 main components, namely the SmartUI Server, SmartUI Designer and SmartUI Operator.

5.1. SmartUI Server

The SmartUI Server is the data portal which exposes data from a number of disparate datasources, both external and internal to it. The data sources included as standard are one Agent Server data source, multiple databases, web services multiple Windows event sources.

This Server brings all the data from the different sources into one portal where the user can monitor, control and manipulate and present the data.

In the Server you can add different datasources, configure profiles, security and create, edit and delete graphic projects.

5.1.1. Datasources

Due to the client-server model that is used, it is possible for one or more Designers to be connected to the same Server and to add, remove and/or modify its existing datasources provided they have the necessary security rights.

For this reason it is possible to update specific datasources to view the changes that have been made to it and to lock a datasource to be able to exclusively modify it.

When installing Adroit and Agent Server a local Eventlog datasource is created that is connected to the local Agent Server only. More datasources can be added, but additional licensing is required.

The SmartUI server also offers a free OLE DB datasource that can be used to connect to OLE DB enabled databases. There is no limit on the amount of OLE DB datasources that can be added.
More datasources available:

- **Simulation**: this datasource does not actually connect to outside data. However, it does provide a wide variety of different types of data. Each simulation datasource contains a number of simulation categories, whose data is either unchanging (static) or mathematically generated and therefore constantly varying. This simulated data is essentially a dependable source of data, which can be used when testing the animation of your graphic forms within the design phase.
- **Web Service**: this datasource connects to an XML Web service so that the functions contained within its Web object(s) can be used. A Web service is simply the provider of one or more Web objects which contain functions that can be accessed over the Internet and used remotely. Web services are able to seamlessly communicate between systems irrespective of their specific programming languages, hardware, and/or operating systems. For this reason XML Web services are being touted as the next revolutionary advancement of the Internet and may well become the fundamental structure that links together all computing devices.
- **Adroit Air**: Handles the configuration and management of the Mobile Air connections, configurations and licensing.
- **Smart Data Services, OPC and IoT Endpoint**:

  Smart Data Services is the collective name / term used for additional services (called micro-services) added to the Adroit Technologies' Technology Stack. Generally, the purpose of these micro-services is to allow for remote connections to local resources over TCP IP.
  Remote communication is secured with Security Tokens to initiate the connections and all subsequent communication on these connections are encrypted.

Some Micro-Services may require other Micro-Services to function, for example: OPC UA Server is dependent on the Adroit Micro-Service, though these inter-dependencies are not detected nor enforced.

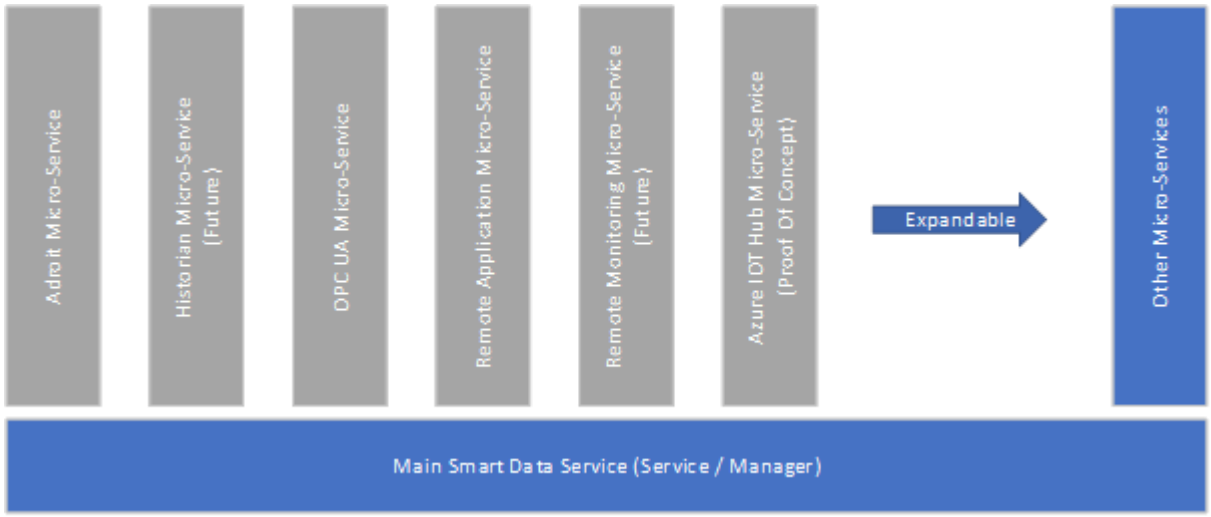


Figure 14 Smart Data Services

The Smart Data Services framework allows us to rapidly extend/expand additional networking functionality by adding fit-to-purpose micro-services, these micro-services can also use, as building blocks, the other micro-services already available, for example, both the OPC UA Server and Azure IOT Hub Services make use of the Adroit Micro-Service to do their communication to the Adroit Agent Server.

These services also benefit from the framework's uptime management, ensuring each service will run as required, without the need for explicitly code for these scenarios.

5.1.2. Management

This directory is where all the user settings and security configuration is done.

- **Profile**: Profiles allow the work environment of the Designer and Operator to be customized on a per user basis. In addition to merely customizing the display of these applications, profile settings can be used to limit the access users have within these applications too.
- **Security**: This manages security for the Server. By using the existing Windows security, this settings can be used to specify who should or shouldn't have access to it, its added datasources and their data elements and/or special functions.
- **System**: This setting provides a view into the Server and exposes information and functions pertaining to the Server itself and the Clients (Operators and/or Designers) that are connected to it. This information and the functions are mainly required for troubleshooting purposes.

5.1.3. Projects

Projects manage the creation, opening and storage of graphic forms and wizards, which can be organized through the creation of folders and sub-folders.
A project exists in two main states:

- The *design state*, which is used to add, import and configure or design the graphic forms within the SmartUI Designer.
- The *operational or run-time state*, which is used to view and interact with these graphic forms, typically within the SmartUI Operator although it is also possible to do this within the SmartUI Designer.

5.1.4. SmartUI Server Clustering

SmartUI Servers can be clustered to provide both load balancing and redundancy, in that any number of servers can belong to the same Cluster, in which case they are known as cluster partners. In other words all the servers within a Cluster are able to handle their combined workload evenly and provide continued operation in the event that one or more fail.

This combined workload is generated by their Operator clients that are made aware of this cluster. This workload is shared by the number of concurrent connections that can be made by these Operators to each of the cluster partners.

For instance, if one server has been licensed to have 10 client connections and it is placed in a cluster with a server that only has 5 connections, these two servers are only able to share the workload until the Server with 5 connections is fully utilized. Thereafter all new connections would be made to the remaining server with spare connections. Furthermore if the server with 10 licensed connections fails, only 5 Operators will be able to remain connected.

Therefore, in order to provide redundancy ensure that there are enough connections on any one server to satisfy all the Operator client connection needs, irrespective of how many servers are within the cluster. However, the greater the number of servers within the cluster the greater the assurance exists of a fail-safe operation.

Each server uses UDP broadcasts to inform its Clients of its availability and load. The load of each Server is determined by the number of Clients that are currently connected to it. Each cluster aware Operator maintains a list of servers for ALL the clusters that it connects to. The list is periodically updated by the Operator, which obtains the availability and load of these servers from their broadcasts. Cluster aware Operators automatically switch over to the server with the lightest load.

5.2. SmartUI Designer

The SmartUI Designer is the interface you will use to interact with and configure projects within the SmartUI Server.

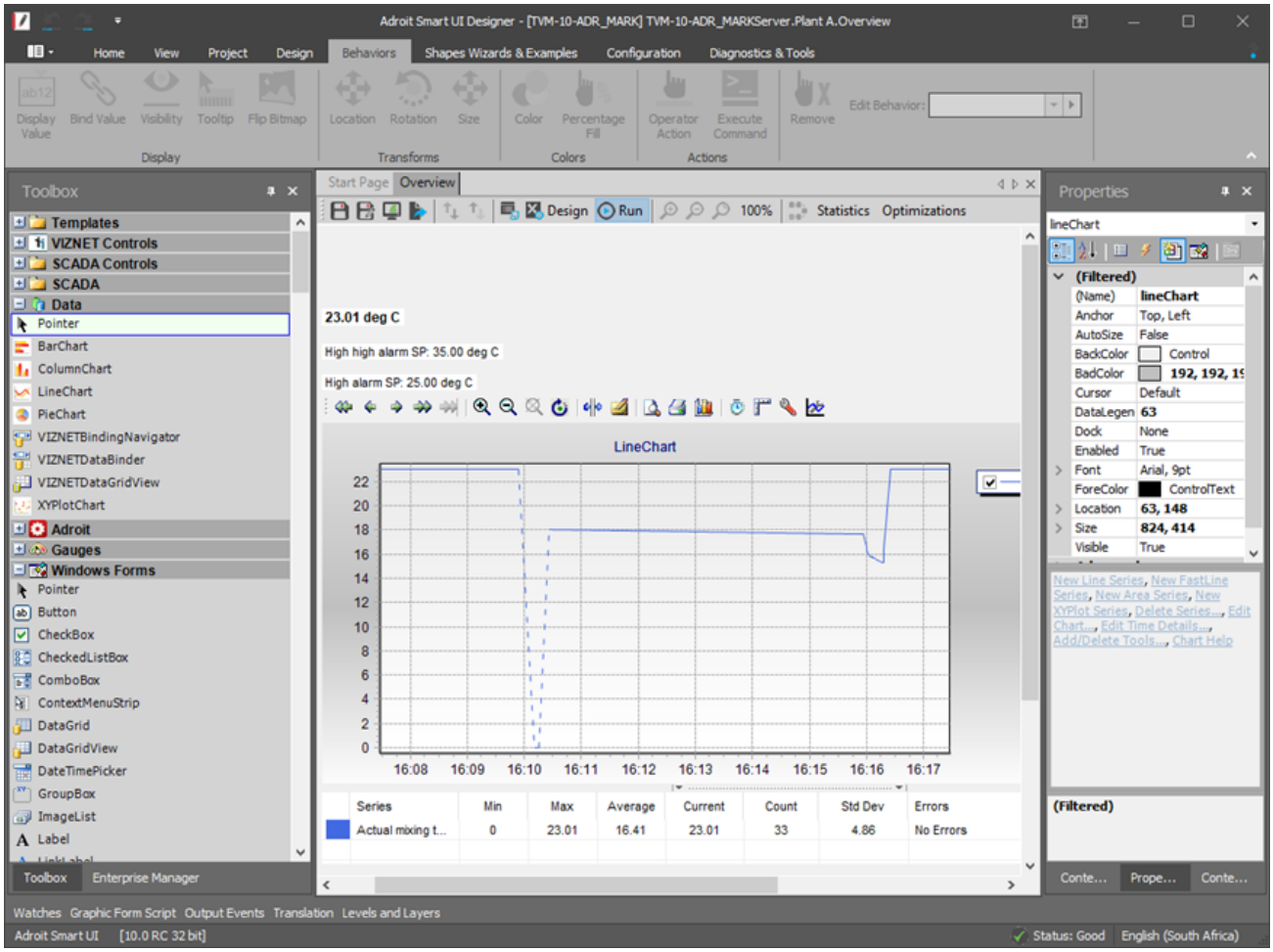


Figure 15 Designer – Detailed Display Example

The top part of the window is the quick access design menu. This menu is based on the very popular ribbon menu you see in Microsoft products.

5.2.1. The Home Menu:

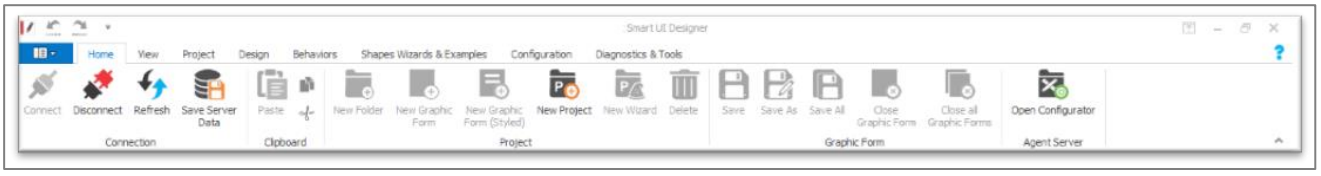


Figure 16 Home Menu

Connection	Description
Connect	This will connect the SmartUI Designer from the SmartUI Server.
Disconnect	This will disconnect the SmartUI Designer from the SmartUI Server.
Refresh	This refreshes the server view in the Enterprise Manager.
Save Server Data	All data in the SmartUI Server will be saved.

Clipboard	Description

Paste	When in design mode on the graphic form you can paste copied objects.
Copy	Copy any object on your graphic form.
Cut	Cut objects on your graphic form that you would like to paste
Project	**Description**
New Folder	This creates a new folder in your project.
New Graphic Form	This button creates a graphic form in the selected project or folder.
New Graphic Form (Styled)	This button creates a graphic form in the selected project or folder.
New Project	Creates a new project from this button.
New Wizard	In order to create a wizard object, it needs to be created on a wizard page.
Delete	Deletes the selected graphic form, wizard, folder or project.

Graphic Form	**Description**
Save	Save the graphic form.
Save As	Save the graphic form with another name.
Save All	Save all graphic forms as is.
Close Graphic Form	Close the current graphic form
Close All Graphic Forms	Close all graphic forms that are currently open.

Agent Server	**Description**
Open Configurator	This button will open the Configurator where users can create, log, alarm and scan the Agents within the Agent Server.

5.2.2. The Agent Configurator

The Agent Configurator is the engineering interface to Agent Server and has functions to find, add, edit, remove and otherwise configure Agents belonging to an Agent Server running either locally or on a remote machine somewhere on the network. Suitably authorized users therefore have access to all Agents in an Agent Server on the network, meaning that separate engineering stations are not required.

The Agent Configurator top-level dialog is shown below. It is from this dialog that users are led in a structured way to select the next aspect of configuration (scanning, logging, alarming, etc.).

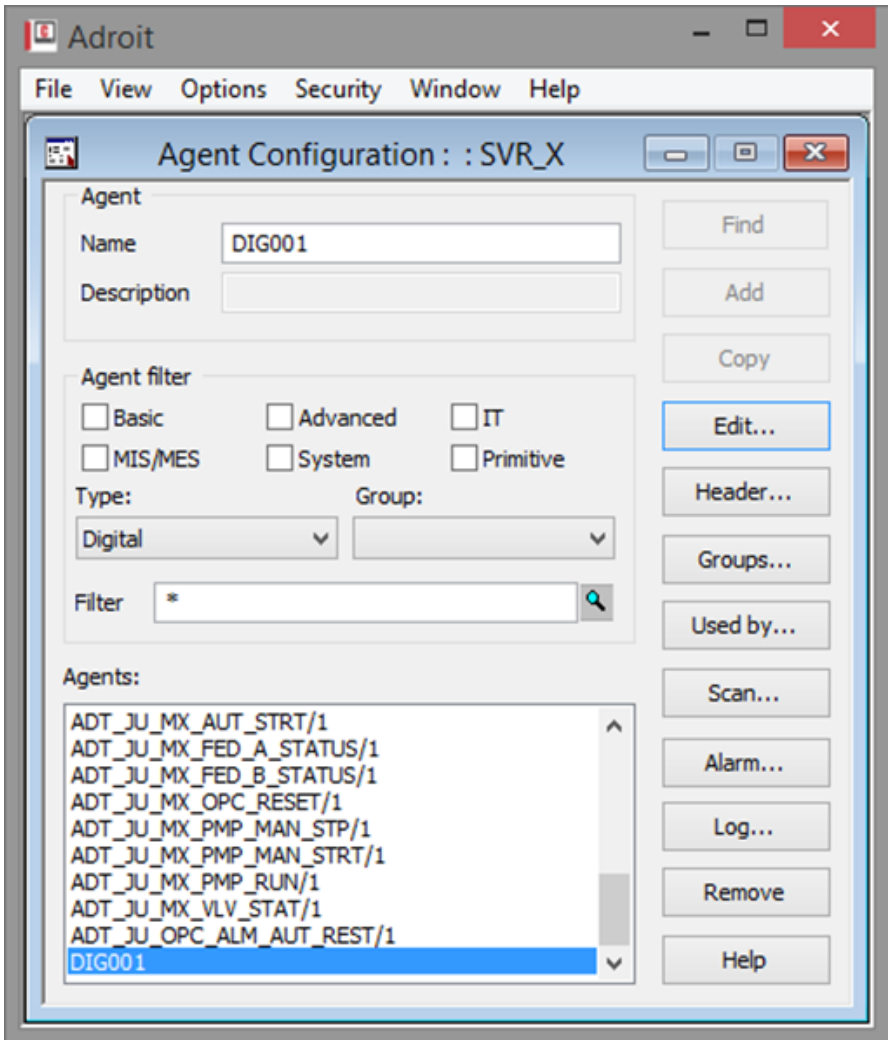


Figure 17 Configurator

5.2.3. The View Menu:

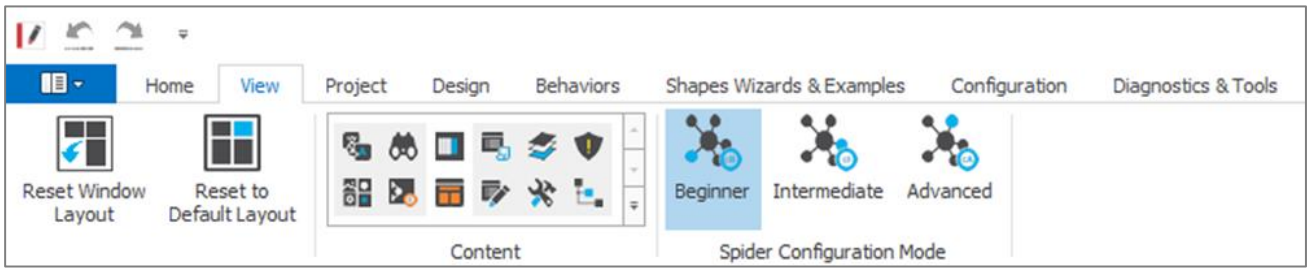


Figure 18 View Menu

Menu Section	Description
Reset Window Layout	This resets your windows layout only.
Reset to Default Layout	This button will reset your design workspace to the default layout settings.

Content Section	Description
Translation	This button will hide or show the translation properties window at the bottom of the screen
Watches	This will hide or show the Watches window at the bottom of the screen.
Progress	This will hide or show the Progress window at the bottom of the screen.
Launch Script Editor	
Levels and Layers	This will hide or show the levels and layers window at the bottom of the screen.
Output Events	
Contents	The button will hide or show the contents window on the right side of the screen.
Enterprise Manager	This will hide or show the Enterprise Manager.
Properties	This will hide or show the properties window on the right of the screen.
Toolbox	This will hide or show the toolbox window on the left of the screen.
Context View	

Spider Configuration Mode Section	Description
Beginner	
Intermediate	
Advanced	

5.2.4. The Project Menu:

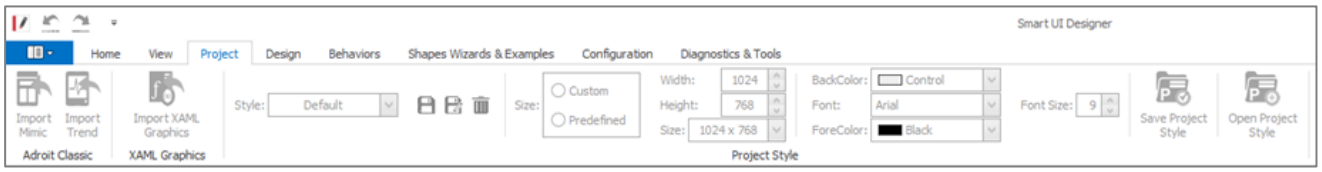


Figure 19 Project Menu

Adroit Classic	
Import Mimic	This will import any Classic UI mimic onto the graphic form.
Import Trend	This will import any Classic UI trend onto the graphic form.

XAML Classic	
Import XAML	This allows you to import any graphics that is in XAML format.

Project Style	
Style	When creating a project, you can choose to associate either a predefined or custom style to the project, which creates a hidden graphic form, called !Style, which configures the following common graphic form display properties to save you design time and ensure a consistent look and feel.
Size	This allows you to set the physical dimensions, specified in pixels: either select a Predefined width and height, which specify common monitor resolutions or specify your own Custom dimensions.
Width	This allows you to set the width.
Height	This allows you to set the height.
Size	This allows you to set the size.
BackColor	The BackColor of the graphic form. Note: Depending on the controls that you add to this graphic form, they may inherit the specified BackColor.
Font	The default Font of the graphic form and any controls that you add to this graphic form.
ForeColor	The ForeColor of the graphic form.
Font Size	The Font Size of the graphic form.
Save Project Style	After editing this project style graphic form, click the Save Project Style item to save these project style changes. Note: You can only edit the style of a single project at a time.
Open Project Style	Open Project Style opens the project style graphic form (!Style), so that you can add form components to this project style graphic form, such as a banner (header) or footer or logo or any other element or graphic that all your styled graphic forms have in common and/or configure their default properties.

5.2.5. The Design Menu:

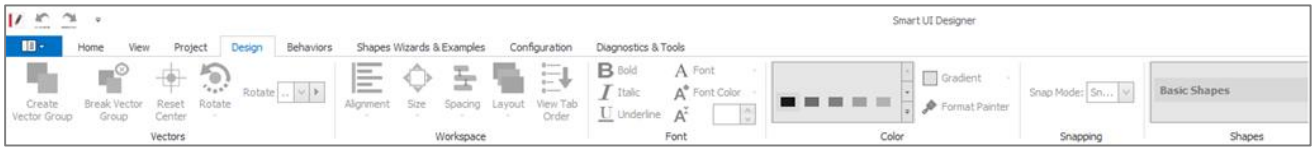


Figure 20 Design Menu

Vectors	
Create Vector Group	This creates a group of objects into one group.
Break Vector Group	Press this button to break the one group into separate objects.
Reset Centre	This will reset the centre of the object.
Rotate	This rotates the object.
Rotate	This rotates the object by the selected degrees.

Workspace	
Alignment	Use this to align selected items.
Size	Use this to size all selected items to the same size.
Spacing	Use this to space selected items equally.
Layout	Select an item to be in front or at the back of another object.
View Tab Order	

Font	Specify your font settings here.

Colour	Choose a colour to be applied to an object.

Snapping	Snap Mode.

Shapes	Choose a basic shape.

5.2.6. The Behaviors Menu consists of:

Figure 21 Behaviors Menu

Display	
Display Value	This behaviour allows you to display values from Agents in the Agent Server.
Bind Value	This behavior can drive a data element (value) to and/or from the value of a property of a control or vector.
Visibility	This is a simple true/false visibility behaviour to show an object.
Tooltip	This behaviour brings up a window with info on it for an action.
Flip Bitmap	Define an index list of images and when manipulating the index, the image assigned to the index number will display.

Transforms	
Location	This behaviour allows to dynamically move the object linearly in a defined X-Y direction by means of monitored value.
Rotation	This behaviour dynamically rotates the object by means of a monitored value.
Size	This behaviour dynamically resizes the object linearly by means of a monitored value.

Colors	
Color	This behaviour allows assigning colours to the two states of a digital (Boolean) value; blink a specific colour property or the VISIBILITY property either continuously or triggered by a value.
Percentage Fill	This behaviour dynamically fill the object by means of a monitored value.

Actions	
Operator Action	This behaviour typically provides methods that allow end-users to change a value at runtime.
Execute Command	This behaviour perform commands such as launching an application or opening, closing or printing a graphic form or logging a user on or off and/or switching users.

Remove	This removes selected behaviours on objects.

Edit Behaviour	This edits behaviours on objects

5.2.7. Shapes Wizards & Examples Menus

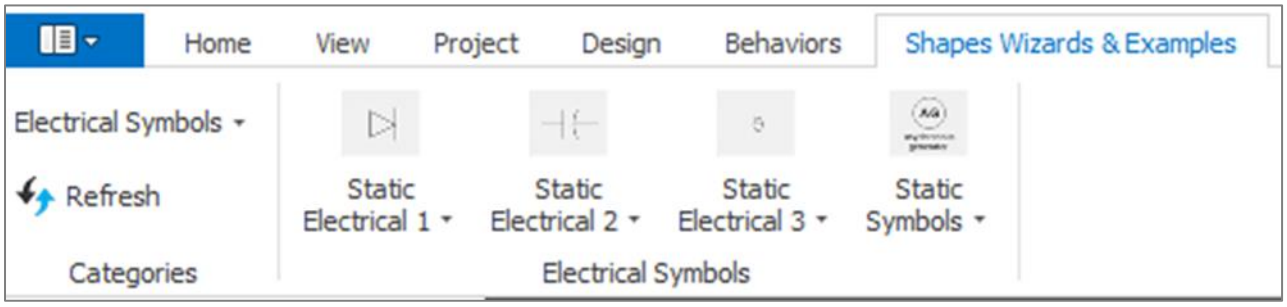


Figure 22 Shapes Wizards & Examples Menu – Electrical Symbols

The Shapes Wizards & Examples menu consists of electrical symbols.

Figure 23 Shapes Wizards & Examples Menu – Hardware

The Shapes Wizards & Examples menu consists of hardware symbols.

Figure 24 Shapes Wizards & Examples Menu – Process Symbols

The Shapes Wizards & Examples menu consists of process symbols.

Figure 25 Shapes Wizards & Examples Menu – Situational Awareness Symbols

The Shapes Wizards & Examples menu consists of situational awareness symbols.

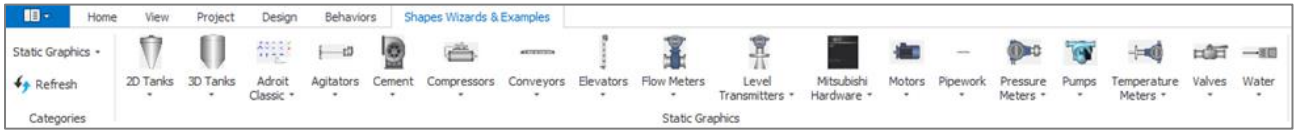


Figure 26 Shapes Wizards & Examples Menu – Static Graphics Symbols

The Shapes Wizards & Examples menu consists of static graphics symbols.

Figure 27 Shapes Wizards & Examples Menu – Templates

The Shapes Wizards & Examples menu consists of templates symbols.

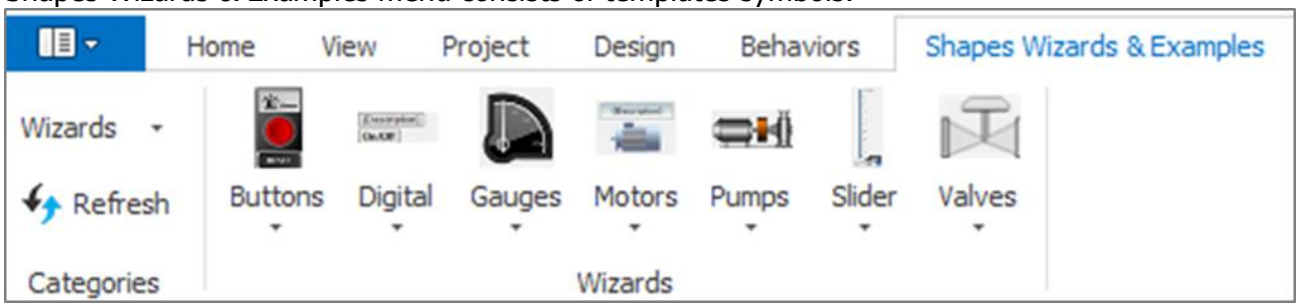


Figure 28 Shapes Wizards & Examples Menu – Wizards

The Shapes Wizards & Examples menu consists of wizard's symbols.

5.2.8. Enterprise manager & Toolbox Menus

On the left is a window with 2 tabs. The default one is the *Enterprise Manager* which is the portal into the SmartUI Server. The second is the *Toolbox* which is the tools that are used when doing the design work.

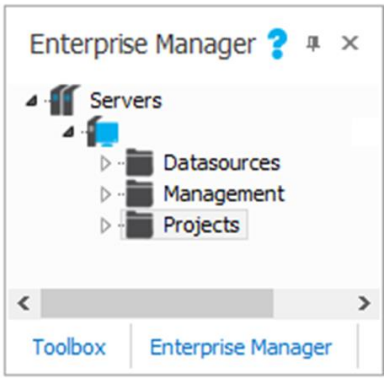


Figure 29 SmartUI Designer Left Tabs

In the right window are three tabs, one is for the Contents, Properties and the other is Context View. Each object that is created consists of properties which can be changed and manipulated via behaviours, spiders or scripting.

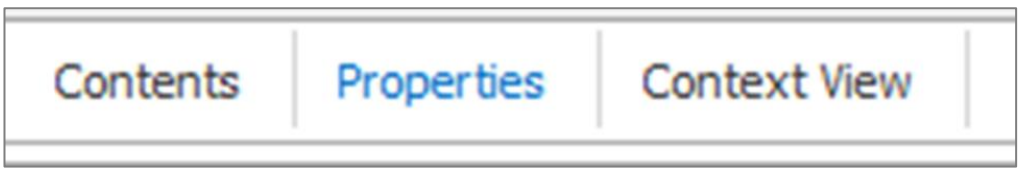
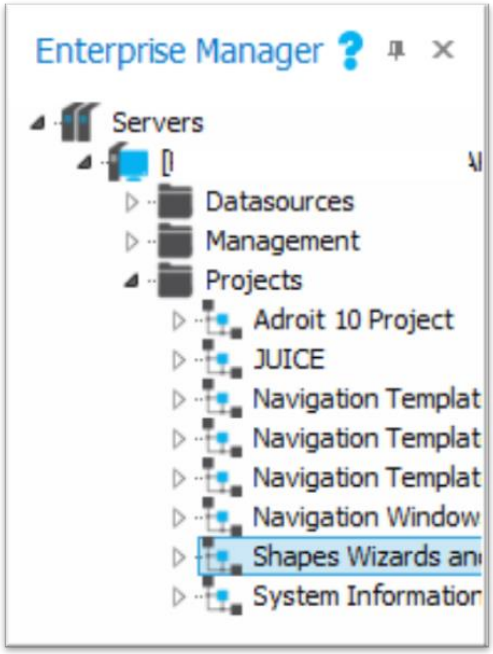


Figure 30 SmartUI Designer Right Tabs

5.2.9. Graphics Library



The SmartUI designer has a new updated equipment library that clients can use to design a project. This library can be found in Projects → Shapes Wizards and Examples. This library consists of Graphic Form examples, ISO symbols library, Samples (Spider examples), Static Graphics which is the different equipment that behaviours can be applied on, and Wizards. You can right-click and preview each folder to see what is in the folder. This way you can quickly look through the entire library for equipment. If you want to use specific equipment that you see, simply drag the equipment to the Favourites window on the right of your screen.

5.2.10. Navigation Templates

The SmartUI also includes navigation templates that users can choose from to use in their projects.

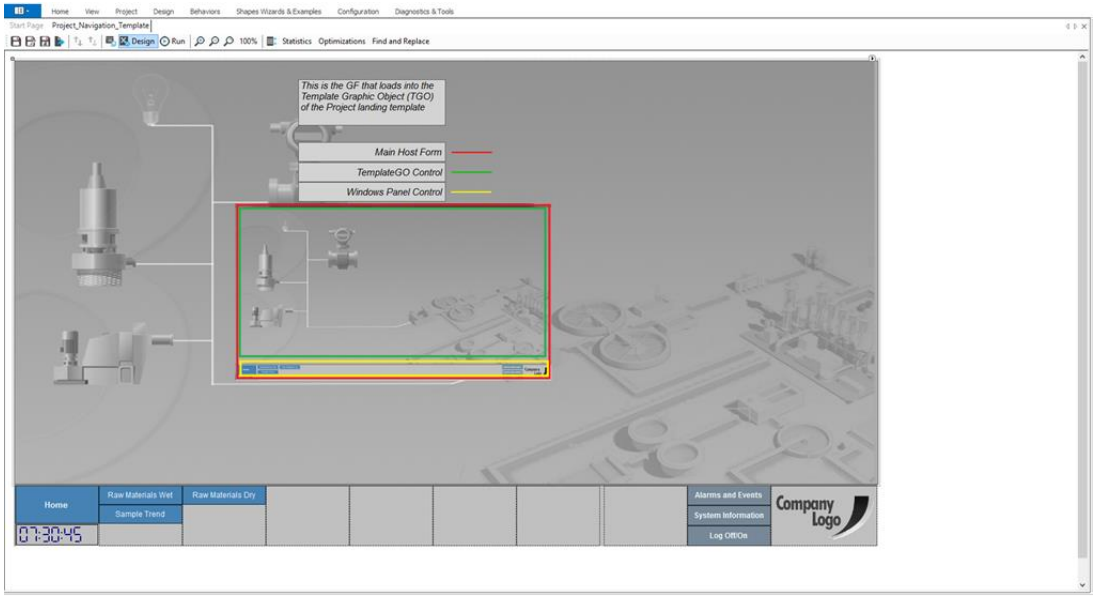


Figure 31 Project Navigation Template

5.2.11. Toolbox

The Toolbox is a set of tools you will use to create the project graphics, trending, alarming and database interacting.

Template

The template directory consists of:

- **The TemplateGO control**: This control allows an existing graphic form to be displayed within another graphic form, or even itself, if necessary.

In addition to simply acting like a picture frame, the contained graphic form can expose one or more properties that can be separately configured, if required. In this way, the graphic form contained within a TemplateGO control can become a configurable form component.

Apart from allowing you to connect the same graphic form to different data, TemplateGO controls allow you to perform modular or object orientated design, by creating an extensive library of graphic forms to simplify the process of creating graphic forms.

It is possible to specify whether the graphic form that is contained within the TemplateGO control will display any changes that are made to its original after it has been contained or not.

This control can also copy the controls and/or graphic elements and their spiders of the referenced graphic form to the new graphic form, thereby allowing you to quickly construct (clone) graphic forms from preconfigured graphic forms.

- **The TemplateGONavToolbar control**: This is a specialized TemplateGO control that provides built-in functionality to help you create a graphic form navigational system or toolbar containing references to one or more graphic forms and can also build navigation from your project hierarchy.

Data

The data directory consists of:

- **BarChart**: Each charted series is either a series of horizontal bars representing values or a series of points that are usually connected to form lines.
- **ColumnChart**: Each charted series is either a series of vertical columns representing values or a series of points that are usually connected to form lines.
- **LineChart**: Each charted series is a series of points that are usually connected to form lines.
- **PieChart**: Each charted series is a series of slices or segments to form a complete pie or circle. Typically you only use one series per chart.
- **VIZNETBindingNavigator**: The VIZNETBindingNavigator control provides a toolbar of commands that allows a user to interact with the data from a table or view of a DataBinder.
- **VIZNETDataBinder**: This is a view to select which tables and their fields to display and then automatically generates and adds the necessary controls and components to the graphic form to view and/or modify the records within each table.
- **VIZNETDataGridView**: This is just a simple view into the table of the database.
- **XYPlotChart**: Each charted series is a series of points that are plotted from two sets of values, one providing the X or horizontal coordinate and the other providing the Y or vertical coordinate of each point.

Adroit

In this directory there are Alarm and Event viewers that are similar to those in the Classic Operator.

- **AlarmViewer**: This tool has specifically been designed for the datasource to display Agent Server configured alarms. Operators can view and acknowledge alarms locally and globally. Security can be applied as to which user can acknowledge alarms.
- **AdroitHistoricalAlarmViewer**: The AdroitHistoricalAlarmViwer can connect to the AlarmManagement Database to display the list of historical alarms logged by using the Adroit Alarm Management Agent. These alarms/events can be filtered.
- **EventViewer**: The Event viewer can connect to the log computer's events or to a remote computer's events. These events can be filtered.

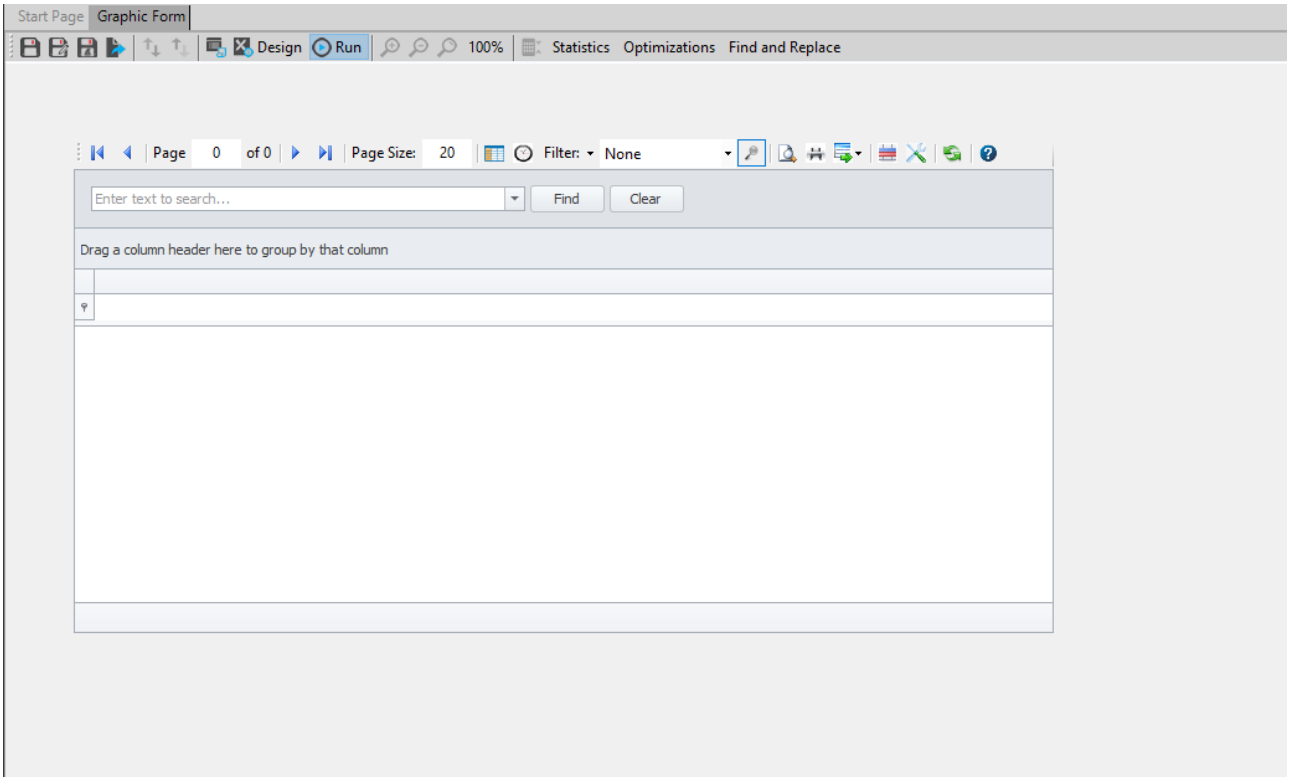


Figure 32 AdroitHistoricalAlarmViewer

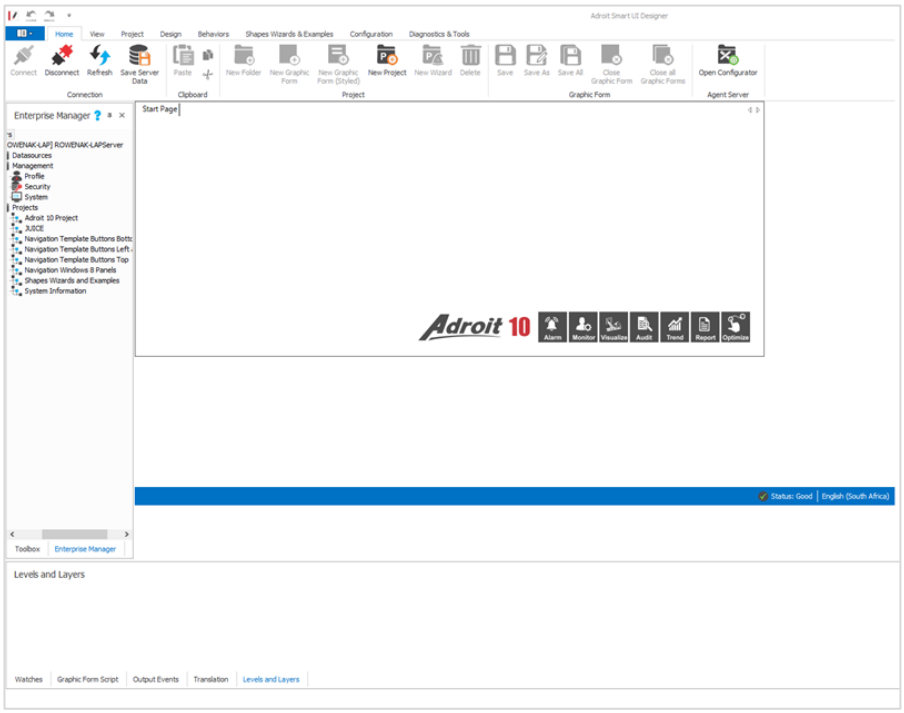
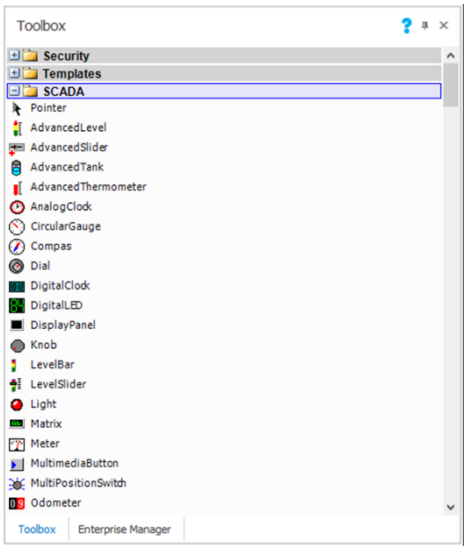


Figure 33 Event Viewer

SCADA



In this directory there is a list of SCADA tools users can use to engineer their solution. These tools are really self-explanatory.

Gauges

This is a new directory with different new gauges to use. Each gauge has a set of themes that can be changed from the properties window of the gauge.

Window Forms

This contains the standard .NET Windows Forms controls usually associated with Windows dialogs.

Custom Built Windows Controls (.NET)

The design of SmartUI allows for the use of other Windows controls, which can be imported into the Toolbox for use within graphic forms. Therefore existing libraries of Windows controls do not necessarily need to be discarded and completely re-engineered for use in SmartUI. This provides you with the flexibility of building complex or simple controls, to further customise the design and look and feel of your graphic forms.

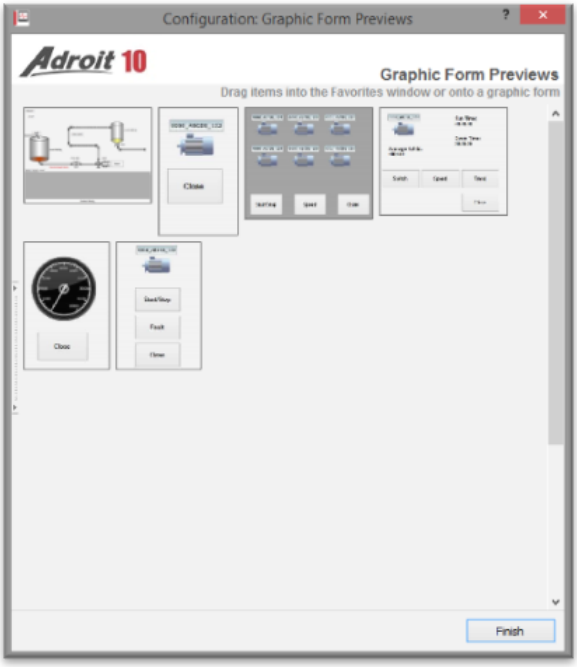
Custom Built Window and ActiveX Controls

The design of SmartUI allows for the use of ActiveX controls, which normally cannot be used in .NET, to be imported into the Toolbox for use within graphic forms. Therefore existing libraries of ActiveX controls do not necessarily need to be discarded and completely re-engineered for use in SmartUI.

5.2.12. Wizards

Wizards allow you to reuse the same graphical object and its assigned behaviours (if any), over and over again with different values (data elements) without having to recreate this object from scratch every time. In other words, a wizard is a reusable design-time object, much like a Windows control. Wizards also have the facility to update their instances that have been added to graphic forms to their current configuration, when they are modified after having been used.

A Wizard is a pre-drawn and/or preconfigured skeleton of graphical objects and their assigned behaviours (if any), which has placeholders (substitutions) for data elements which can simplify and speed up the task of creating graphic forms. But, instead of directly specifying the values that drive these behaviours, these are replaced with value-placeholders (or substitutions).

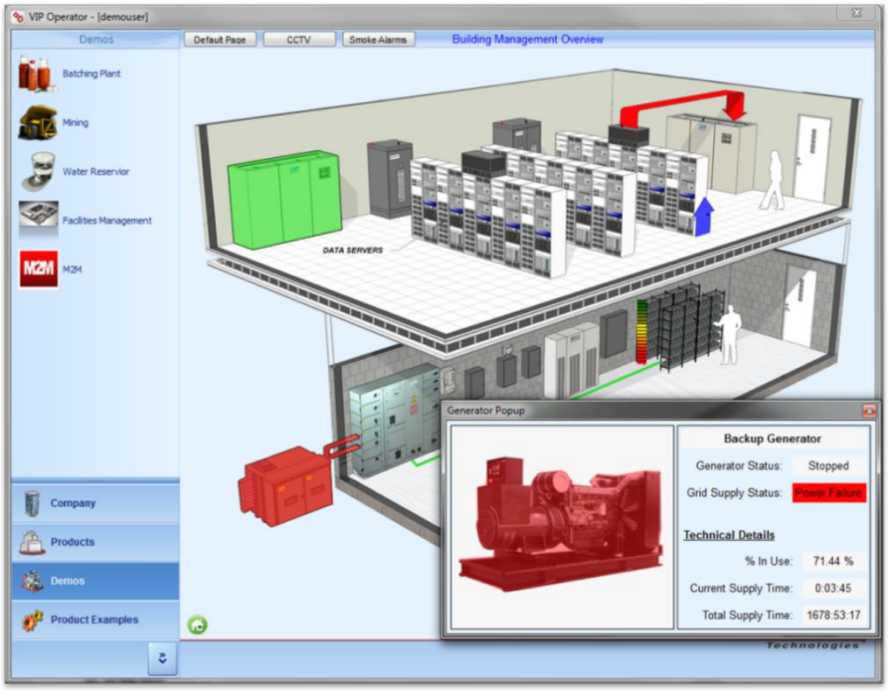


Each time a wizard is inserted into a graphic form, a dialog is displayed whereby values must be specified for all the substitutions that occur within this wizard. For instance, instead of redrawing a pump or a motor each time one is required, a wizard can be created and simply inserted wherever a pump or motor is needed at which time you link the required data elements (values) to its predefined substitutions.

5.2.13. Templates

Templates are reusable graphic forms that can be used at run-time. A template should be used if you would normally require a large number of identically configured documents to represent similar items.

The fundamental difference between a graphic form and a template is that a template has aliases and is therefore designed to be displayed within a Template Graphic Object control that displays a graphic form and connects data elements to any aliases of this graphic form control.



For instance, consider the situation where a plant uses 1000s of motors, each of which require the same process data to be reported and/or controlling actions to be performed. In this case a single template mimic could be created instead of creating separate mimics for each motor.

5.3. SmartUI Designer Advanced Functionality

5.3.1. Spiders

Spiders and their links (silks) make up a visual programming language, which allows elements to be transferred and/or manipulated in the Designer. This configuration occurs within what is known as a spider workspace or engine, typically by means of drag and drop. Spiders are primarily used in SmartUI to bind (connect) data to properties that are exposed by graphic forms and their form components.

This configuration occurs within the Spider Configuration window for the actively displayed graphic form, as follows:

- *Automatically* occurs when data is purely being connected to properties by dragging the required data element onto the necessary graphic form or its form component directly.
- *Manually* generally only occurs when this data needs to be manipulated or modified before it is linked to the required property. However, by using spiders in this way it is possible to provide a rich, interactive environment for the end-user of a graphic form without necessarily needing to write any scripting at all. This is provided, in part, by the ability of spiders to be operated or triggered by means of events or actions either performed by another spider or by a user on a control that is attached to the graphic form. In this way users are able to perform custom and/or complicated manipulation of data elements, literally at the touch of a button or a click of a checkbox, etc.
- *Bulk*, in addition to manually configuring spiders and silks, perform bulk configuration by exporting the current spiders and/or silks configured within a spider workspace to a Microsoft Excel XML file, performing the additional configuration and re-importing this configuration.

The spider workspace is known as a spider engine as it can either be stopped or running. Any spiders and silks that have been added to it are only live or able to be triggered and perform their operations when the spider engine is running.

For graphic forms, this only occurs when viewing graphic forms in the Operator or when opening the graphic form in running or preview mode in the Designer. The spider workspace provides a visual or drag and drop interface to create the required spiders, silks and template sets. Each spider workspace provides a toolbar and a right-click menu that provides tools to assist in the adding, arrangement and configuration of these items, as illustrated by the following screenshot:

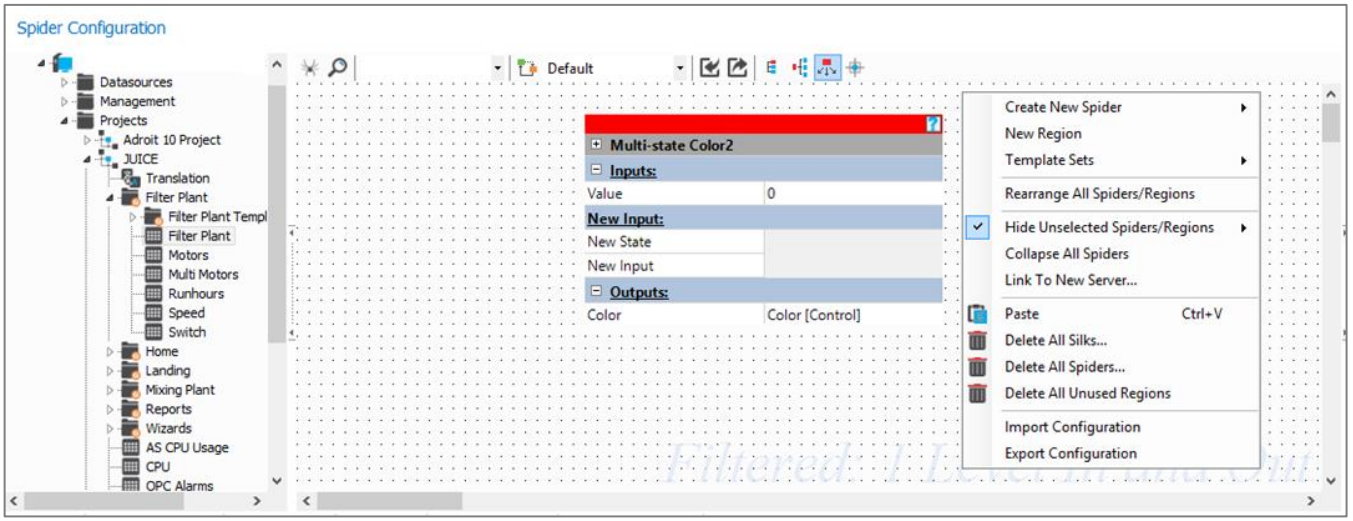


Figure 34 Spider Workspace

As this screenshot indicates, the spider workspace:

- Provides a number of indicators that visually display the status of common spider and silk properties; and
- Allows spider regions to be added to organize (de-clutter) the spider workspace, by providing logical groupings of spiders which can be expanded or collapsed to display or hide these spiders. Typically you create a spider region when one or more spiders have something in common.

Once added, each spider has a configuration view that can be expanded or collapsed and which allows its inputs and/or outputs to be configured. Each spider has a number of inputs and/or outputs, determined by the function that it performs.

Once triggered, a spider will communicate and/or manipulate data from its inputs to its outputs in order to perform its function. In addition to be being triggered by means of an event, spiders can be triggered whenever one of their input values change or by using a spider whose sole function is to trigger other spiders, in the same spider workspace, in a customisable order.

Data elements are transferred between spiders by means of silks, which are their interconnecting links. Silks can be grouped together into what are known as template sets. As only one template set can be active at any one time, this provides an easy way to vary the data displayed by a graphic form. Any spiders that are connected to silks that are not associated with the active template set will not be used, so the functionality provided by each template can vary. By way of example, consider the case where a graphic form is constructed to represent a pump which displays its status, its flow rate, its pressure and flow by linking this data to the necessary properties by means of silks.

Now consider the case where a battery of similar pumps is required, each requiring this same information to be displayed. Instead of redrawing this pump and replicating it on multiple graphic forms and then attaching the correct data to each of these pumps individually, template sets can be used. In this case the silks required to supply each individual pump with its required data could be created to this single pump graphic form and then associated with its own template set. Then when the spider workspace is

running, the active template set can be dynamically changed, as required, to display the correct information for each pump.

5.3.2. Available Spiders

An extensive list of spiders is provided that are able to perform a wide range of functions and operations. Although all of these are generally applicable spiders, it is possible to create spiders that are datasource-specific spiders which perform customised data-specific operations and/or processing.

> ***NOTE**: Data spiders are a special "hidden" subset of spiders, which are currently added automatically to the spider workspace. These are responsible for managing the transfer of data to and from the spider workspace.*

This list of spiders is the same list that is displayed when manually adding a spider to a spider workspace that has been organised into the following categories to make it easier to select the required spider:

> ***NOTE**: You will notice that the icons of the spiders have a different background colour for each of the spider categories to visually identify which category they belong to and give an indication as to their function/purpose.*

Advanced

These spiders can be used to perform the following advanced functionality or configuration:

Icon	Name	Description
	**Levels and Layers**	To change the visible level of a graphic form at runtime.
	**Performance Counter**	To add any number of performance counters so that you are able to monitor key indicators of the performance of your computer.
	**Script Engine**	To create script objects that interacts with the Client and if necessary, with the Server, so that they can perform advanced customised operations. Both input and output variables can be provided for this script object and if necessary, whose values can be provided by or linked to other spiders, properties or data elements. These script objects are run whenever these spiders are triggered. For further details of script objects and scripting, see the Scripting section. For instance, a script can be used to open a graphic form.
	**Special Function**	To use any special function provided by any of the configured datasources. In addition to the datasource-specific special functions, this spider can be used to perform queries and stored procedures that have been added to OLE DB datasources or to execute functions exposed by an XML Web Service.
	**String Format**	To format one or more values, if necessary, and display them as a single formatted string. Values of data (data elements) can be formatted to customise their appearance before they are displayed, since the default formatting of a value is often not expressive or informative enough to be of much use. This is particularly useful when customisable numeric and date and time formats are required.
	**Variable Data Element**	To dynamically change the names of data elements and if necessary, to configure their values. A typical example of usage would be for a redundant server system or process where only the datasource name changes between the common setups. In this case, this spider can be used for all the relevant data elements, instead of creating an entirely redundant set of spiders.

Animation

The following spiders can be used to animate graphic forms:

Icon	Name	Description
	**Blinking**	To start and stop a colour alternating between two predefined colours at a specified interval and to display another predefined colour when this blinking is stopped. This spider can be used to configure colour properties of graphic forms and their form components.
	**Multi-state Colour**	To define colours that provide a visual indication of the current value of either a String, Double or Integer value. This is achieved through the creation of more than one "state" where each state has a particular colour that is associated with either a specific string, a range of values or a logical expression that compare one or more input values. This spider can be used to configure colour properties of graphic forms and their form components.

Application

The following spiders can be used to launch or interact with other applications:

Icon	Name	Description
	**Application**	To remotely interact with a graphic form within the Operator, such as the ability to open or close a graphic form when this spider is triggered. For instance, this spider can be used to open or close a graphic form from an event with a click of a button.
	**Data Entry**	To enter a data value by displaying a dialog for this purpose at runtime. It is possible for this change in value to be confirmed and/or logged to an event log, if required.
	**Execute Command**	To run an application (.EXE), command file (.CMD) or a batch file (.BAT) and, if necessary, specify command-line parameters. For instance it is possible to open a specific document within Notepad, or some other application.

Databasing

The following spiders perform database-specific tasks:

Icon	Name	Description
	**DataBinder Trigger**	To display or to save changes made to the data of a table in a DataBinder. For more details about DataBinders see the OLE DB datasource description above.
	**Query**	To perform queries and execute statements provided by any OLE DB datasource.

Extended Type

The following spiders can be used to configure or expose properties:

Icon	Name	Description
	**Colour**	To select a colour using a colour edit box or define a colour by individually manipulating it's A, R, G and B constituents. Where the R, G, B colour constituents specify the relative intensity of red, green and blue respectively in the resultant colour. The A constituent, however, specifies the required transparency of this colour, which is only supported by specific colour properties, e.g., for the BackColour property of a button. For instance, by driving one or more of these colour constituents remotely, it is possible to create a constantly varying colour which can be used to configure a colour property of a graphic form or one of its form components.
	**Data Row Splitter**	To filter out a row (field) from dataset data and expose the values of its columns as outputs of this spider. In other words, this spider is able to split a data row into its column values. For instance, once the individual column values of a row have been split into separate outputs, these outputs can be displayed by form components or can be used to set their properties, etc.
	**Extended Data Element**	To either display the properties of an existing data element or to create a data element from any input. For instance, this spider can be used to obtain the name of a data element.
	**Font**	To select a new font using the standard font dialog or to modify an existing font by independently adjusting certain of its attributes. The font attributes that can be individually manipulated are Regular, Bold, Italic, Font Size, Underline and Strikethrough. For instance, by driving one or more of these font attributes remotely, it is possible to create a constantly varying font which can be used to configure a font property of a graphic form or one of its form components.
	**Hidden Property**	To expose and/or manipulate properties of an object such as a form component which are generally hidden. These properties will either be displayed as inputs if they are writable, and/or outputs if they are readable. For instance, to be able to set the SelectedIndex property of a ComboBox control.
	**Mapping**	To map (connect) inputs to outputs so that each input has an associated and identically named output. This spider is intended to act as a connection interface for reusable graphics to which the data required for the display and animation of the graphic can be connected. In other words, this spider eliminates the need of painstakingly connecting the required data to each of the various properties and other spider inputs every time this graphic is reused. This configuration is performed once for all, by simply creating the necessary mapping pairs and connecting their outputs to these properties and other spider input. Then, when the graphic is reused, the data is simply connected to the inputs of this Mapping spider which correspond to its already connected outputs.
	**Multi-state**	To create more than one "state" for a specific String, Double or Integer value so that when this value falls within a defined state the spider returns the value associated with this state.
	**Tooltip**	

Logic

The following spiders can be used to construct simple or complex logic expressions to perform various tasks when TRUE by triggering one or more spiders, as an alternative to writing script to perform the same functionality.

Icon	Name	Description
	**AND**	To create a logical expression and, if necessary, to AND its result with another Boolean input.
	**IF**	To either create a single logical expression or, if necessary, to create the initial logical expression that will be operated upon by the other logical spiders.
	**Logic Trigger**	To trigger one or more spiders in sequence when its specified logical expression input becomes TRUE after having been FALSE.
	**NOT**	To NOT (inverse) its specified logical expression input.
	**OR**	To create a logical expression and, if necessary, to OR its result with another Boolean input.

Mathematical

The following spiders can be used to perform mathematical calculations:

Icon	Name	Description
	**Constant**	To create user-defined constants and/or use the existing constants to set the values of other spiders' inputs.
	**Expression**	To implement user-defined calculations this can make use of variables. For instance, such calculations can be used for totalising or accumulating values, implementing advanced control strategies, creating sophisticated information displays and reports, etc.
	**Multiply and Divide**	To multiply and/or divide one or more values to produce a result and if necessary, to add one or more offset values too.
	**Operator Action**	To perform various operations on a numerical or Boolean input. It is possible for this change in value to be confirmed and/or logged to an event log, if required. For instance, this can be used to toggle Booleans or set Booleans to start and stop a pump for example.
	**Scaling**	To convert or scale a value in a certain range to a desired range. For instance, while the incoming value may be specified in the range from 0 to 50, this spider can be used to scale the value between 0 and 100 instead.
	**Summation**	To add and/or subtract one or more values to produce a result, this totals the amounts to be added and subtracted.

Simulation

The following spider can be used to produce simulated data:

Icon	Name	Description
	**Simulation**	To create simulated data and configure how this simulated data is generated. For instance, this can be used when the Designer is not connected to any datasource to test the animation of graphic forms.

Spider Workspace

The following spiders can be used to perform spider-specific functionality:

Icon	Name	Description
	**Template Set**	To change the active template set at runtime.
	**Trigger**	To customize the triggering of existing spiders, by triggering one or more spiders in sequence, whenever this spider is triggered.

Statistical

The following spiders can be used to perform statistical operations:

Icon	Name	Description
	**Counter**	To record the number of On and Off events that occur, for a Boolean value, and the individual and total time of these events.
	**Statistical Functions**	To collect data from an analog (floating-point) value and to calculate statistical information (maximum, minimum, mean, standard deviation and total) from this collected data.

XML

The following spider can be used to operate on or produce data in XML format:

Icon	Name	Description
	**Extended XML**	To expose XML properties defined within an XML document as outputs so they can be used individually, as required. A common usage for this spider is to expose the required values from the otherwise complex XML output, produced by XML Web Services.

5.3.3. Scripting

SmartUI has been specifically designed with the aim of providing simple non-scripting methods to perform advanced functionality. This aim is primarily achieved through the use of spiders and silks, a visual programming language which provides a simple yet powerful alternative to scripting. For instance, by using spiders and silks it is possible to easily create graphic forms with a rich, interactive Operator without using scripting at all. For details, see the Spiders section.

However, scripting is provided for the more advanced user and can easily be used in tandem with these other non-scripting methods. This is in accordance with the general philosophy of openness with which SmartUI has been written.

Since SmartUI has been created using the .NET framework, scripts can only be written in the following .NET scripting languages: Microsoft Visual Basic.NET or Microsoft C# (CSharp).

The default scripting language can be set by the profile associated with the currently logged in user, if necessary. For further details on profiles, see the section on User Management.

Before describing scripting any further, it is necessary to understand a bit about the scripting architecture used within SmartUI and its benefits.

In SmartUI each script is seamlessly encapsulated in a "script object", for this reason these terms can be used synonymously. The purpose or need for the script object is simply to provide a self-contained environment within which the script can be run. Therefore it does not matter whether the script is run from the Server or a Client (Designer or Operator), it will still carry out its intended purpose.

For instance, one function of the script object is to manage the list of external assemblies (DLLs) that are referenced by a script so that their properties and methods (functions), etc., can be used within the script.

Scripting has been tightly woven into the very fabric of SmartUI and as so can perform the following functions:

- **Graphic form scripting**: These scripts can be used to programmatically animate or enliven graphic forms by creating event handlers for events exposed by the graphic form and its added components. In this way it is possible to provide interaction between components on the graphic form, such as drag and drop and mouse handling capabilities, etc. In this case, a script is associated with each graphic form that is executed on the Client whenever and wherever the graphic form is operated. This script can typically be accessed using the Graphic Form Script window that allows one to easily switch between the script and its graphic form.
- **Client scripting**: These scripts can be used to customise or enhance the usage of the Designer or Operator by opening graphic forms, etc. In this case, a Script Engine spider can be used to contain the required script which is executed on the Client whenever the spider is triggered. In these scripts input and output variables can be used whose values can be provided by or linked to other spiders, properties or data elements.
- **Server scripting**: These scripts are typically used to perform manipulation of data upon the Server itself. In this case, a Server Scripts datasource can be used to store the required script which is executed on the Server. These scripts can either be executed manually or by using a special function provided by this datasource or they can be performed remotely, in which case no client interaction is required at all.

Due to the self-contained nature of each script provided by its script object, there is no need to separate the Client and Server scripting, which can be combined within a single script if necessary. Furthermore, running a script from the Server or a Client should not affect its speed of execution, since this speed is usually dependent on the script itself, the utilization of external references and correct programming techniques and other system-dependent factors such as CPU load, etc.

For these reasons, there are a number of benefits associated with adding a script object (script) to a Server Scripts datasource, such as:

- Each script object that is stored can be used and executed within a Script Engine spider.
- Each script object that is stored is immediately accessible by all the Clients connected to its Server.
- Each script object becomes a data element and can therefore be individually secured, if necessary. Unlike a script object created within a Script Engine spider for instance which cannot be secured directly and instead relies upon the security applied to the graphic form with which this spider is associated. For further details, see the section on Security.

***NOTE**: It is possible to create scripts that use profile-specific variables, whose values can be stored for later retrieval within the profiles, so that scripts can be customised per profile.*

Scripts are typically written and debugged within the Script Editor; however users can also use Visual Studio 2005 (and later) if this is already installed on their computer. The default script editing environment is set by the profile associated with the currently logged in user.

In the case of graphic form scripting, the Graphic Form Script window is displayed by default which provides the same basic editing environment as is found in the Script Editor and allows one to easily switch between the script and its graphic form. If necessary, these scripts can be edited within the Script Editor to take advantage of the other editing functionality that it provides.

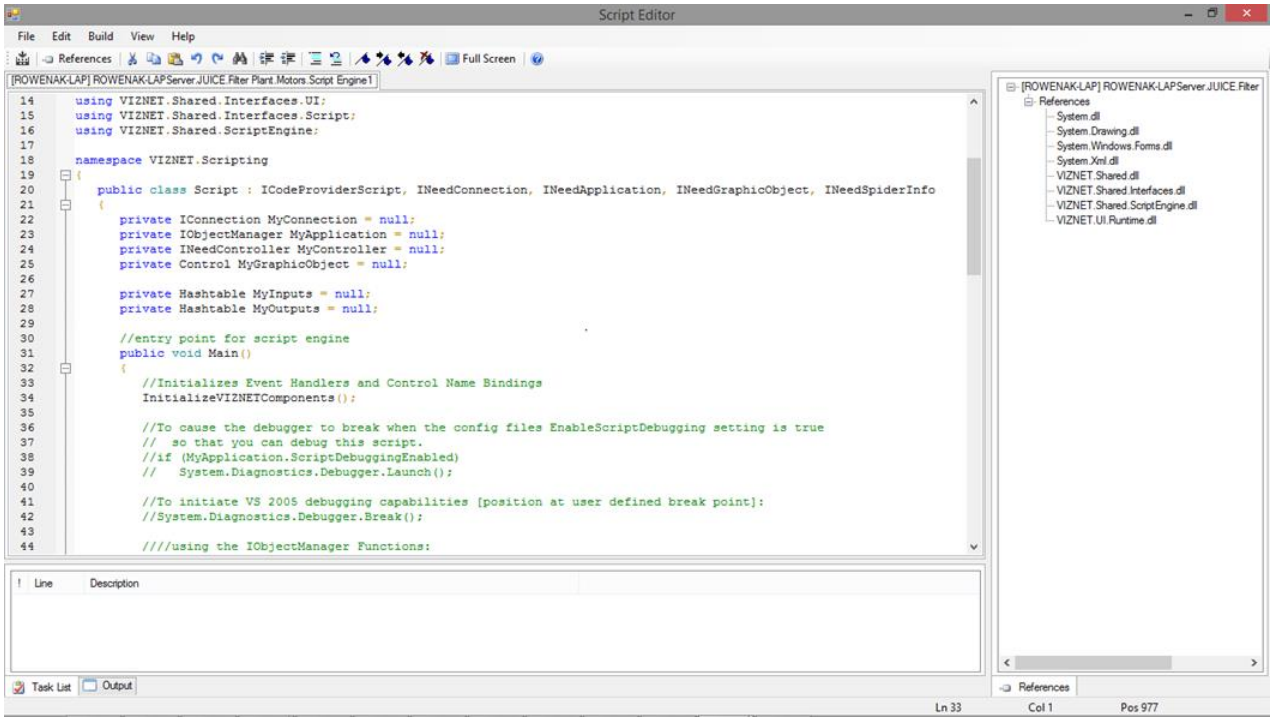


Figure 35 Script Editor

The Script Editor provides standard programming functionality such as bookmarks, commenting lines of text, indenting lines of text, compiling the script, compiling and running the script, etc. Access these commands from the toolbar, the menus or by right-clicking in the Editor Workspace window. The script editor can provide IntelliSense, a powerful feature that can dramatically reduce the time required to produce your scripts, within the Editor Workspace window.

For instance, when coding you do not need to leave the Editor Workspace window in order to perform searches on language elements, since IntelliSense provides the following features:

- Valid member variables or functions can be selected from drop-down lists for any class, struct, union or namespace which is defined in any of the references added to your script and within your script itself. The required member can then be inserted directly into your code.
- Pop-up windows can display the following information for methods and functions – their namespace, the names and types of their parameters and their return type. The required parameter list can be selected for overloaded functions.
- Immediate feedback can be provided whenever misplaced brackets or open-ended segments of code are detected.

Since the required INeed interfaces are dependent upon whether a script interacts with either the Server or the Client or both of them, all the INeed interfaces are declared within in each script and only once the script has been complied, are the required INeed interfaces used when the script is run.

For this reason, when a script is initially created default source code (text) is automatically added to it, which provides standard functions and the declarations of all these INeed interfaces. This default source code is provided in the default scripting language which is set by the profile associated with the currently logged in user. Both the scripting language and this source code can be changed to the other scripting language, if required.

In addition to the functionality provided by these INeed interfaces and the other standard functions, the script is to provide additional functionality through adding references to external assemblies (.DLL files), whose properties, methods, etc., are made available to this script and can be implemented within it. The script object then manages these references for its script.

5.3.4. Object and Context Model

Adroit offers users the ability to minimise their engineering, maximise the quality of the solution when the Object Model is used. From the Designer, under the Datasource window it is possible to create your own predefined and tested templates and to then engineer projects from these objects that will in turn replicate the complete SCADA configuration including tag names, scanning, alarming and trending.

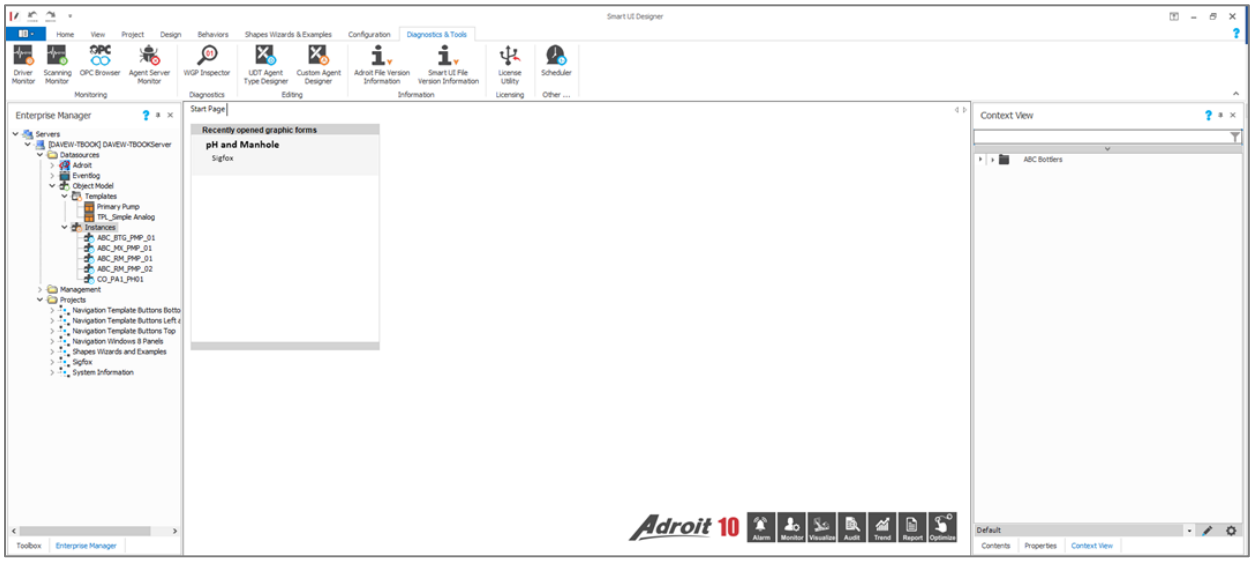
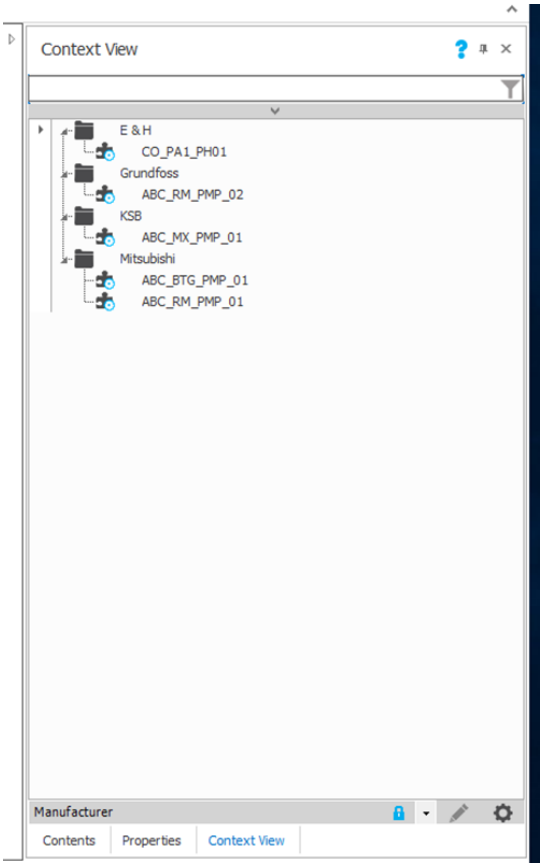


Figure 36 Designer – Object Model

The approach is fairly simple and the process users would follow would be:

- Create a Wizard and populate it completely with tags configured for all the Scanning, Logging and Alarming required for such an object.
- Right-Click and create a Template from this configured Wizard. This Template would also have the possible graphical options available for users to use at the time of deployment. In addition where the user has defined their own Context (such as Manufacturer, Equipment Type and Description, Power Rating) this will be carried through to all instances and be available for allowing Context based sorting and searches throughout a project to allow easy management of the configuration.



- Once created then it is possible to create as many instances as you need for a project.
    - This can be done in a hierarchy under the "instances" tree view
    - Another way is to add a physical hierarchy in the Context view window that appears on the right hand side of the Designer. Once the view is created you could simply drag the created instances into the view, or it is possible to drag or right-click and add an instance within the Context View. If users have designed the tree correctly all the Agents that will be created when deploying the instance would be built using a unique tag naming convention as defined by the user.

Building the graphics would simply require the user to define to graphic forms in his Projects directory and then to drag the instances onto the forms and link them all together with the required static graphics, add the navigation and deploy to a site.

5.4. SmartUI Operator

The SmartUI Operator display the Enterprise Manager with the projects listed for the user to select form. A graphic form can be set as default loading page for the default user or according to user profiles.

The SmartUI Operator can be run from an operator application install or from a web interface. In order run the web interface, SmartUI Web Services needs to be installed on Microsoft Internet Information Services (IIS).

The SmartUI ClickOnce Operator can run on most popular internet browsers. For Internet Explorer you need to use *ClickOnce*.

The SmartUI ClickOnce Operator is part of the SmartUI Zero Footprint Deployment (ZFD) strategy and allows a user to operate and interact with SmartUI project(s) using a fully-fledged SmartUI Operator. By using Internet Explorer, a user can simply browse to a specific URL to download the necessary files and then run the SmartUI Operator normally, as a separate application.

By using Microsoft's ClickOnce deployment mechanism, this means of installing the Operator provides the following additional benefits:

- No administrator privileges are required for this installation; and
- ClickOnce supports caching which improves the load time of the Operator after the initial download.

The ClickOnce Operator is installed per user, not per machine and is sandboxed or isolated from other running programs for added security.

For the other internet browsers you will use the *Launcher* application. One of the advantages offered by the .NET architecture is the ability to easily manage distributed applications. Within SmartUI we make this easy by using our Launcher application.

Although the Launcher does not perform any client or server functionality itself, it can install the Operator component remotely (either over a LAN or the Internet) and/or ensure that this component remains up to date with the latest files, if required.

In order to use the Launcher a "distribution" computer needs to be configured first. This "distribution" computer can be any computer that has SmartUI Distribution service installed and running, although it is typically a Server computer.

The SmartUI Distribution service simply identifies a "distribution" folder on its computer. The Launcher then connects to this preconfigured distribution computer over a LAN or the Internet and synchronises the contents of its installation folder with the specified "distribution" folder.

In other words, once the Launcher has connected to the specified distribution computer the files and sub-folders within its installation folder are compared to those in the specified distribution folder.

Only the file(s) and folder(s) which are newer on this remote computer or which do not exist on the local computer will be displayed for downloading. Any executables that are downloaded from the distribution computer can then be run or launched from this application if necessary, hence the name of this application.

Therefore the files placed in the user-configurable distribution folder on the distribution computer will be the files that are distributed whenever a Launcher is connected to it. Typically this folder contains the files required for the Operator.

As mentioned previously, Internet connections are only possible when SmartUI Web Service is installed on a web server, to provide a gateway between the Client and Server computers. This web server can be installed securely within a DMZ (de-militarized zone) environment, if necessary.

Enhancement: Instead of your users needing to configure the necessary details required to connect to your Distribution computer, you can use the "Setting up the Launcher" page of the Config Editor to preconfigure these for your Launcher installation.

5.5. Secure Mobile Gateway

This secure technology is offered as a service and gives the user of the SCADA the ability to view the Adroit SCADA graphics on any device, from anywhere. Built with Secure Access at it's core it creates a secure socket connection and renders the graphics through a web based application creating an "airgap" between the user and the SCADA.

5.6. Security

Since the SmartUI Server is a data portal that can aggregate data from numerous sources, it is possible that data with varying degrees of sensitivity may be made available to the SAME Server. For this reason it is essential that the data is appropriately secured.

As the architecture allows the SmartUI Server to be accessed by both Designers primarily used by engineers, and Operators primarily used by end-users, it is necessary to ensure that different users have different levels of access to this data.

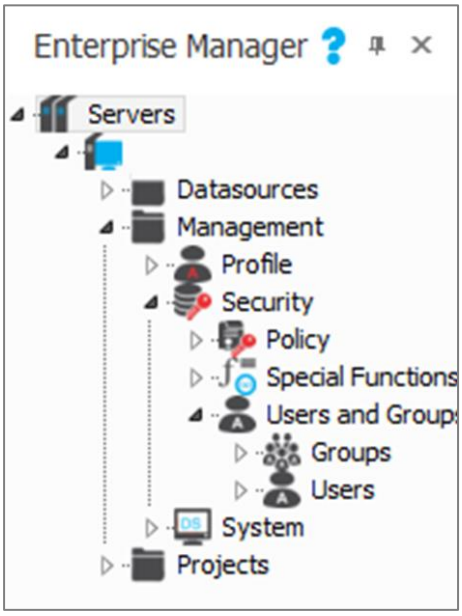
> ***NOTE**: SmartUI "locks" down everything and that one of the functions of the designer of the system is to grant the necessary access to the required users. This approach to security follows very much along the lines of Windows security, as implemented by Microsoft in Windows XP SP2.*

Security in SmartUI is user-centric and operates on top of the traditional Windows operating system security, using a configurable sub-set of the existing users and groups on each Server computer known as the "allowed" users and groups.

For instance, when a Client (Designer and Operator) is launched, it is necessary to login to a Server and only a user that is ether an allowed user or a member of an allowed group on this Server is able to log into it.

In addition to controlling the access of these "allowed" users and groups to the Server, its configured datasources and their data, it is possible to control or limit the access that these users have within each Client (Designer and Operator).

The following diagram illustrates how security in SmartUI operates on top of the traditional Windows operating system security by means of the allowed users and groups, which is a selected set of the existing Windows users and groups on the SmartUI Server machine.



Security should typically be implemented during the deployment of SmartUI so that the initial allowed users and groups can be configured on each Server. Then the access of these allowed users and groups should be specified within each Server and Client.

However, security configuration should continue throughout the life of the project to secure new datasources by limiting the access of new users and modifying the access levels of existing users as required, etc.

5.6.1. Server Security

Server security must be configured per Server in the Enterprise Manager by using its Security component within the Management category.

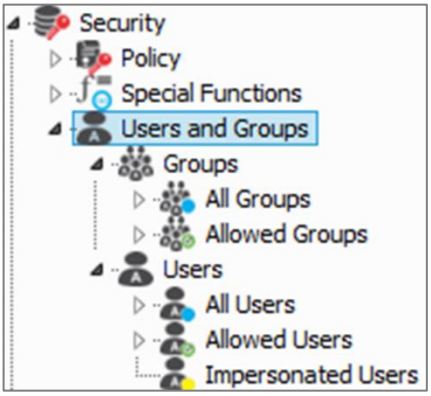
Most importantly, this Security component configures the allowed users and groups which provides the first or foundational level of ALL user security and management within SmartUI, not only for Server security but also Client security too and is therefore mandatory.

Furthermore this Security component configures the first level of purely Server-specific security in the form of Policies, which can be used to provide a general (large-scale) level of control to safeguard the data that is made available to the Server.

The second level of Server-specific security can be broadly termed Datasource-specific security, which can be used to provide a precise (finely tuned) level of control to safeguard the data that is made available to the Server. This is therefore configured per datasource and its exposed data elements and special functions. For details of the structure of data within SmartUI, see the section on *Data Architecture*.

5.6.2. Allowed Users and Groups

As previously discussed, security in SmartUI is user-centric and operates on top of the traditional Windows operating system security. This is made possible by using a configurable sub-set of the existing users and groups on each Server computer known as the "allowed" users and groups.



> ***NOTE**: SmartUI uses the Active Directory to expose the available Users and Groups from which the person designing the SmartUI application can then choose his allowed SmartUI users.*

These allowed users and groups therefore provide the first or foundational level of ALL security within SmartUI, hence their creation is mandatory for every SmartUI Server. In other words, unless a user is either an allowed user or a member of an allowed group, he/she

will be unable to login to the Server and use SmartUI. Before a SmartUI client (Designer or Operator) can be used, it is necessary to login to a Server first.

The allowed users and groups are the means by which all other user security and management in SmartUI is applied. They are typically added and removed via the Designer in the Enterprise Manager by using the Users and Groups list of the Security component that is within the Management category of each Server.

However, it is possible to give certain users and groups total or global security access to each Server, which essentially become "master" users and groups that:

- Permanently become allowed users and/or groups that cannot be removed from the lists of the allowed users and groups maintained by this Server so that these users will never be able to lock themselves out of the Server; and
- Are always able to login to and have unrestricted access to this Server, its datasources and their data.

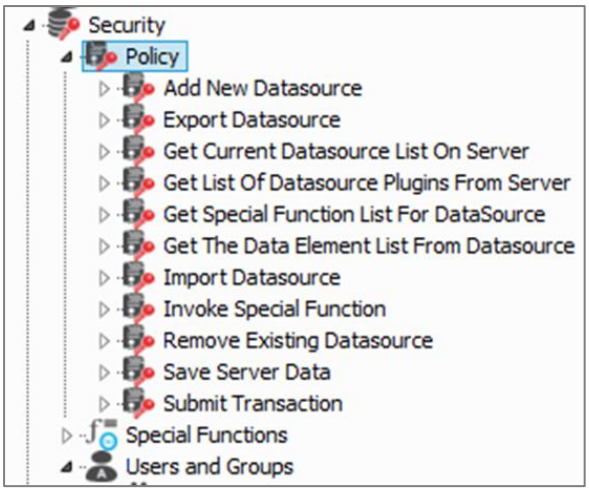
These global security users and groups should be used to administer the security of the Server by adding the other allowed users and groups and assigning the necessary access rights to each of them. By default, only the Administrators group and the default login user is assigned global security access to each Server.

In addition to adding existing users and groups to the allowed users and groups for a Server, it is possible to create users that impersonate or masquerade as an allowed user.

For instance, when users are not part of the Windows security or where you do not want to expose a domain username and password when someone logs in to a Server over the Internet. The impersonated user is then able to login to a SmartUI client (Designer or Operator) and merely assumes the user security and management settings of the allowed user that is being impersonated.

5.6.3. Policies

Policies can be used to provide a general (large-scale) level of control to safeguard the data that is made available to the Server. They can only be implemented after allowed groups and/or users have been added to this Server.



Policies are essentially a configurable set of rules that govern the access to a Server and its available datasource plugins (the types of data that the Server is able to connect to), as follows:

- Each policy represents a specific service or activity that can be performed by the Server, such as the ability to retrieve a list of its available datasources.
- Each policy can then be associated with one or more allowed users and/or groups with each policy.
- Any allowed user and/or group not associated with a policy is unable to perform the service that it represents.

In other words, whenever a Server receives a request for a service it first determines whether the currently logged in user is part of the allowed users or groups that are able to perform this request, and then denies or grants access based on this result. By default, the Users group and any group(s) and/or

user(s) that have been assigned global security access to this Server is automatically associated with every policy.

Certain policies represent datasource plugin specific services such as the ability to add or remove a new datasource. In this case, the policy contains a list of the available datasource plugins so that the required allowed users and/or groups need to be associated with the applicable datasource plugin. Once a datasource plugin has been secured using a policy, this security will be applied to all of its existing and future datasources that are configured.

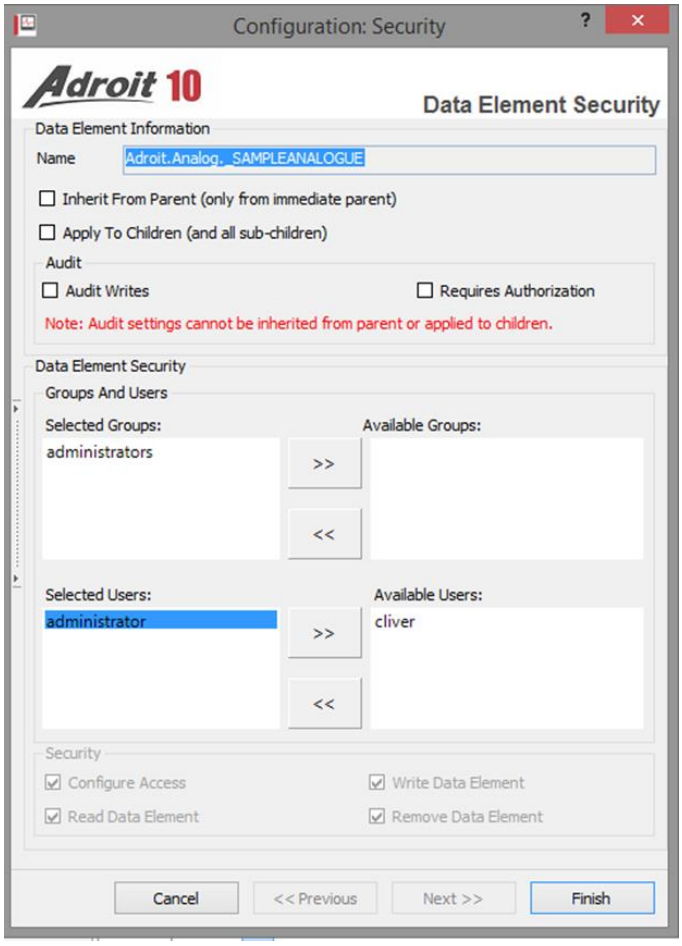
***NOTE**: In order to secure a specific datasource and/or its data, datasource-specific security is required.*

Allowed users and/or groups are added to and removed from policies via the Designer in the Enterprise Manager by using the Policy list of the Security component that is within the Management category of each Server.

5.6.4. Datasource-specific Security

Datasource-specific security is a general classification that encompasses all the security measures that can be used to provide a precise (finely tuned) level of control to safeguard the data that is made available to the Server. This too, can only be implemented after allowed groups and/or users have been added to this Server.

As its name implies, this essentially involves securing the specific datasources that have been added to a Server and their configured exposed data elements and/or special functions.



However, since datasources are themselves data elements, datasource-specific security simply consists of the following two security methods:

Data Element Security

Since every datasource exposes its data in the form of data elements, it is therefore essential that the access to these data elements be sufficiently controlled to provide the necessary security for your data.

Each data element (and datasource) can be assigned security permissions that determine which users are able to configure its security settings and which users are able to view, set and/or remove it.

Data element security is applied via the Designer in the Enterprise Manager by right-clicking the required datasource and/or data element and selecting the Security item.

***NOTE**: This item is only available if the currently logged in user has the necessary security permissions to configure the security settings of this datasource and/or data element.*

The following strategy is recommended for implementing data element security:

- Specify the extent of the access that allowed users and/or groups are given to the available datasources so that only the necessary datasources are available for them to perform their required duties.
- Specify the extent of the access which allowed users and/or groups are given to the specific data elements within these datasources so that only the necessary data is available for them to perform their required duties.

In this way the currently logged in user should only be able to see the datasources and their data elements that are necessary to perform his/her function. Furthermore, whenever any data element is accessed, this Server will always determine whether the currently logged in user is able to perform the requested data element operation and accept or reject the request accordingly.

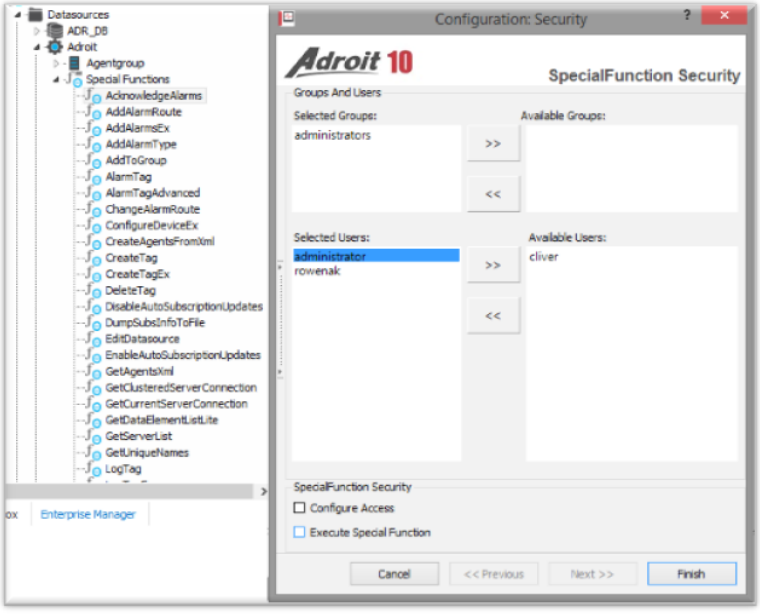
By default, only the group(s) and/or user(s) that have been assigned global security access to this Server are automatically assigned full access to all datasources and their data elements. Data elements need to be secured separately for EACH datasource that is created.

However, data element security does not need to be applied manually as it is also possible to automatically:

- apply the security configuration applied to a datasource to all of its data elements (both existing and future); and
- Inherit the security configuration applied to a datasource to simply configuring the security for a data element.

Special Functions Security

In addition to making its data available as data elements, a datasource may only provide special functions. By securing the provided special functions it is possible to secure this specific functionality provided by the datasource itself, which is especially evident when the special function is the only means provided for performing a specific action.



Although special functions are provided by datasources, their security is entirely unaffected by the security settings applied to their datasources. Each special function can be assigned security permissions that determine which users are able to configure its security settings and which users are able to use it.

Special function security is applied via the Designer in the Enterprise Manager by right-clicking the required special function and selecting the Security item.

> ***NOTE**: This item is only available if the currently logged in user has the necessary security permissions to configure the security settings of this special function.*

The following strategy is recommended for controlling access to special functions:

- Specify which allowed users and/or groups are able to use special functions at all, by associating them with the *Invoke Special Function* policy.
- Specify which special functions these allowed users and/or groups are able to use for each datasource so that they will be able to perform their required duties.

In this way, whenever a special function is used either specifically or by the datasource itself, the required Server first determines whether the currently logged in user is part of the allowed users or groups that are able to use special functions. If this is so, it will then determine whether the user is actually allowed to use the special function associated with this specific datasource and accept or reject the request accordingly.

By default, only the group(s) and/or user(s) that have been assigned global security access to this Server are automatically assigned full access to all special functions. Special functions need to be secured separately for EACH datasource that is created.

5.6.5. Client Security

Security is normally implemented by the Server to secure the data exposed by it. However, the Client (Designer and Operator) can implement its own security. Although these security measures are distinct from Server security, they can only be implemented after the necessary allowed groups and/or users have been added to the Server. Client security is usually configured by means of user profiles to control the access a user has within the Designer and Operator applications. In addition to customising the display of these applications, profile settings can be used to limit the access users have within these applications too, as follows:

- to specify which menus and even which menu items can be accessed within the Designer; and
- To specify the permitted user interaction within the Operator.

Profiles can also specify whether you want to automatically logoff its assigned users from the Client (Operator or Designer) after a specific period of inactivity. This is an added security feature that is often required by particular industries. For instance, this is required to comply with CFR (The Code of Federal Regulations), the codification of the general and permanent rules published in the Federal Register by the executive departments and agencies of the Federal Government 21 (Food and Drugs) requirements.

It is also possible when designing graphic forms, to specify which users have access to which properties exposed by the controls that have been added to the graphic form. In other words, it is possible to provide alternate sets of properties, from the default or standard properties displayed for controls, for one or more of the allowed users and/or groups.

6. Redundant Agent Servers (Active Clustering)

One of the most important and vital attributes of a world-class SCADA HMI software product should be high availability. If a vital hardware, firmware, or software component should fail, then mechanisms need to be in place to ensure that plant visibility and control is restored in a bumpless and timely fashion.

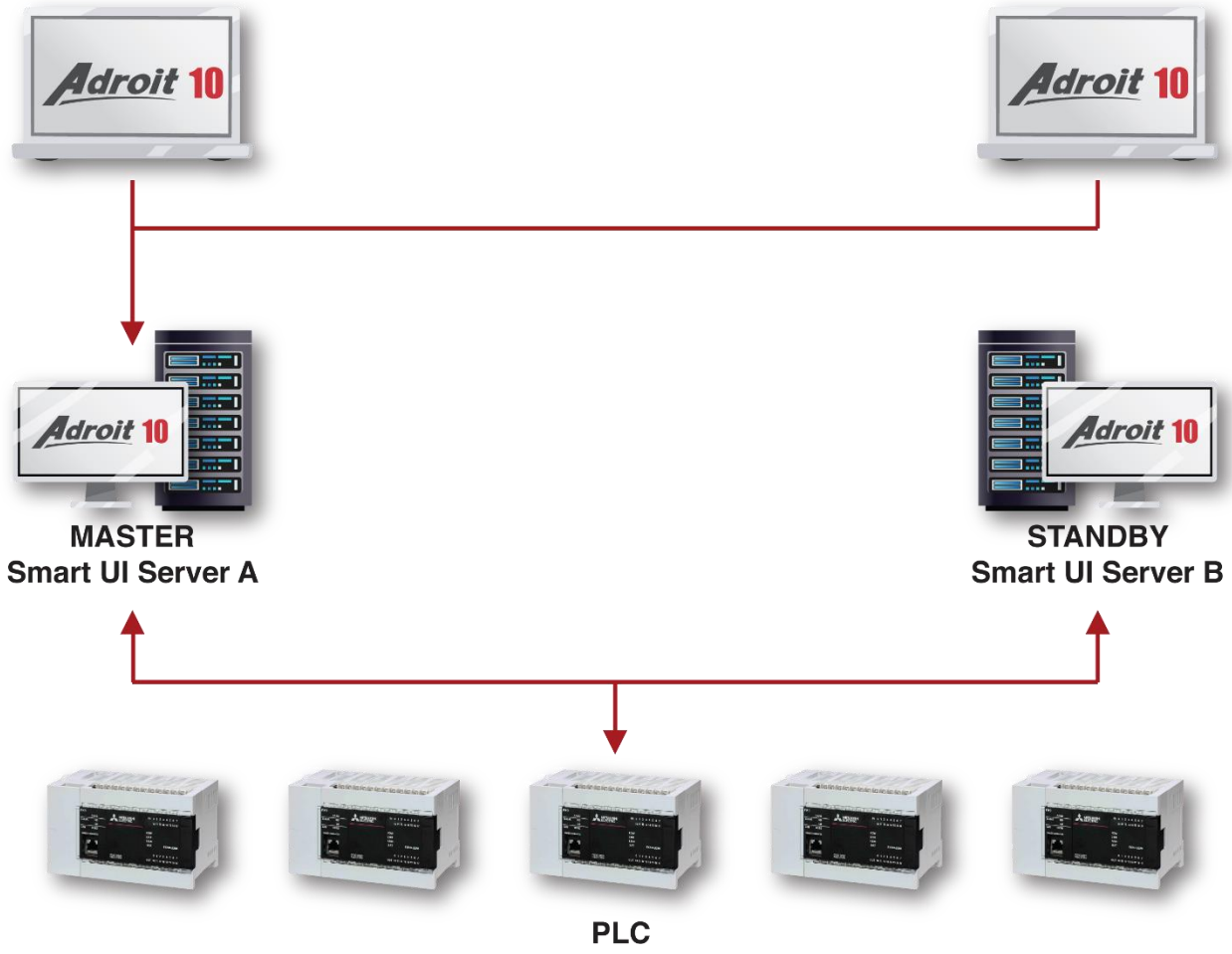


Figure 37 Active Clustering

Bumplessness means that after failure and subsequent recovery, any control and monitoring displays must be in the same state as they were immediately before the failure. Timeliness implies that recovery should take place in no more than five seconds.

6.1. Active Clustering - Foundation

An Active Cluster is a mechanism for providing client and server redundancy enabling high availability of the system. Typically, redundancy can be applied to all the levels within the plant. There are three general components in Adroit which are redundant aware, namely the clients, the servers and the server I/O.

In Adroit the Agent Server can be setup to be a Cluster Aware Server. Setup is simple and done at the datasource connection setup through the Designer. A cluster is a set of two servers running a replicated database (WGP file) on the same project. A cluster server can be networked with other normal (non-cluster) servers.

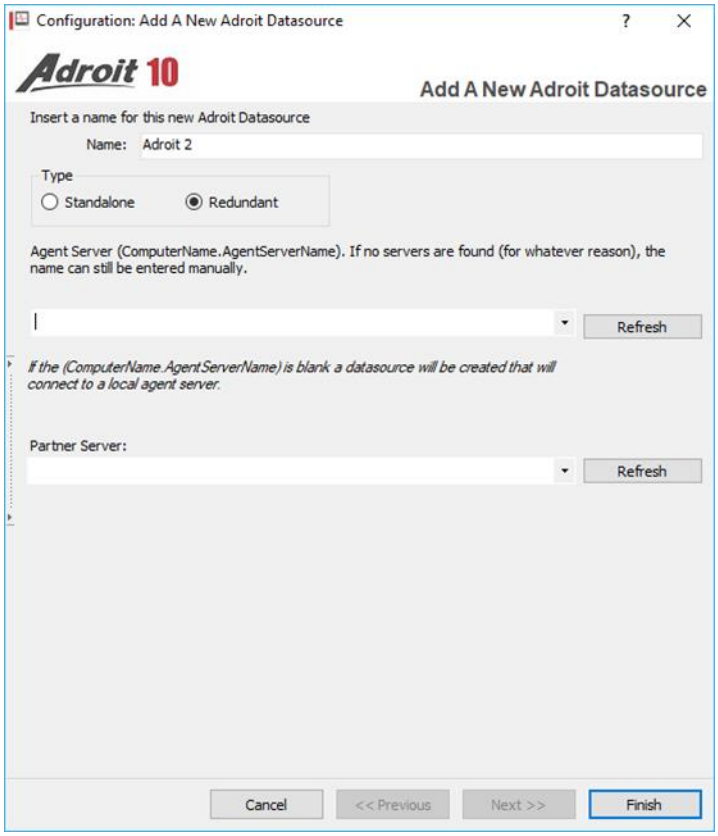


Figure 38 Add a New Adroit Datasource

A cluster server is either in the mode of master or standby server and can either be in the state of healthy or unhealthy. The general idea of clustering is that the master server services all the I/O and client tasks, while the standby server replicates the master server. If the master server or machine fails, the standby server takes over as master and the UIs and scanning tasks take appropriate action. There are four key mechanisms in Active Clusters, namely *Catch-up*, *Real-time Catch-up, Replication* and *Switchover*.

6.1.1. Catch-up

When a new server is added to the cluster, it must become a mirror image of the active member of the cluster. The new server must restore its historical data (historical catch-up) that it lost during the down time. A new Agent called the SystemDatalog Agent facilitates this historical synchronisation or Datalog catch-up.

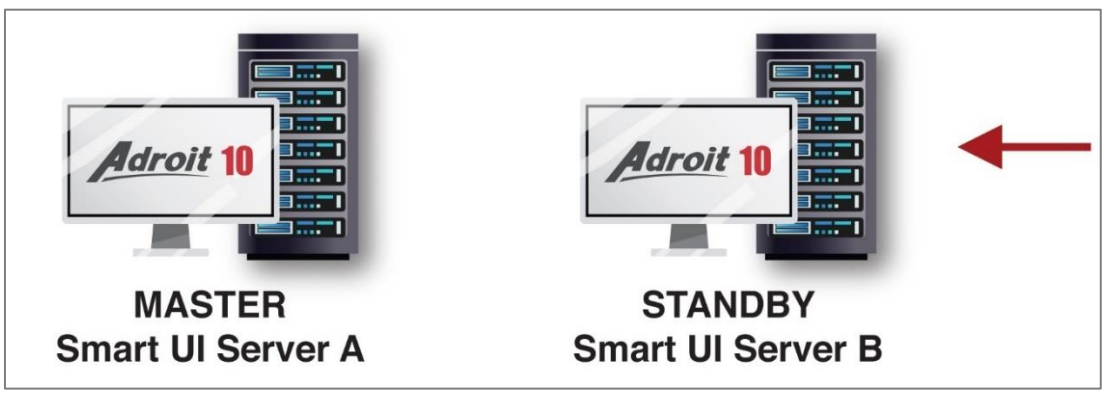


Figure 39 Catch-up

6.1.2. Real-time Catch-up

Certain Agent types on a cluster server require synchronisation of internal data structures with their master partner. This process is called real-time catch-up. When a cluster partner starts up and its Master server already exists, this synchronisation takes place while the standby server is loading up.

***NOTE**: Real-time catch-up is in no way associated with shadowing and datalog catch-up.*

The following Agent types support real-time catch-up: Analog, Counter, Expression, MaxDemand, Scheduler and Statistical.

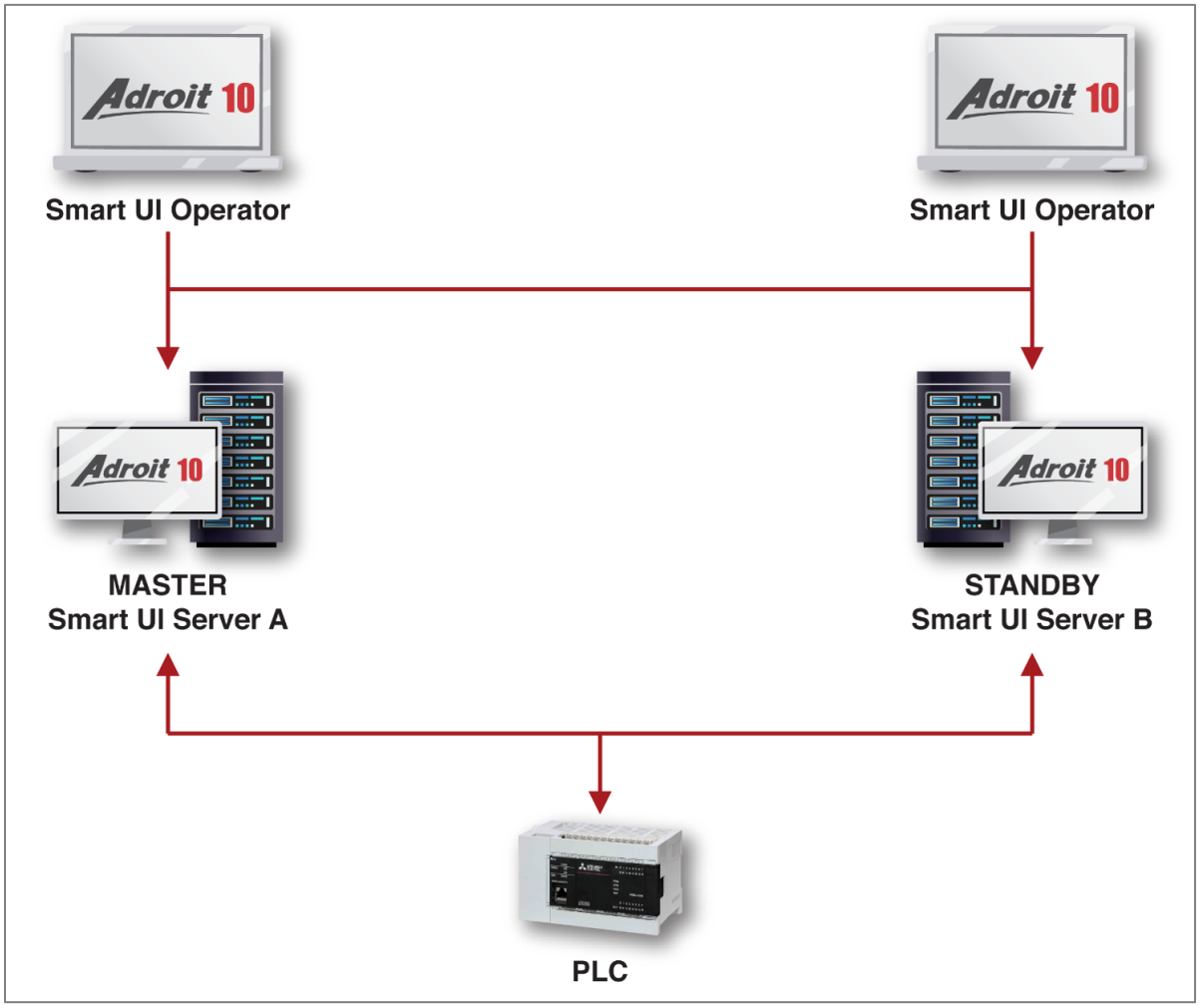


Figure 40 Real-time Catch-up

6.1.3. Replication (Shadowing)

Catch-up deals with the requirement for a new server being added to the cluster to be a mirror image at the current point in time. But, what about later on when changes occur as a result of external influences on the cluster?

External changes affecting the cluster members can occur from a *client* or from *server scan inputs*. Client changes are typically a result of changing a set point from within a UI. All these sources of change are *replicated* to the other server in the cluster in order to maintain the mirror image state of all the servers.

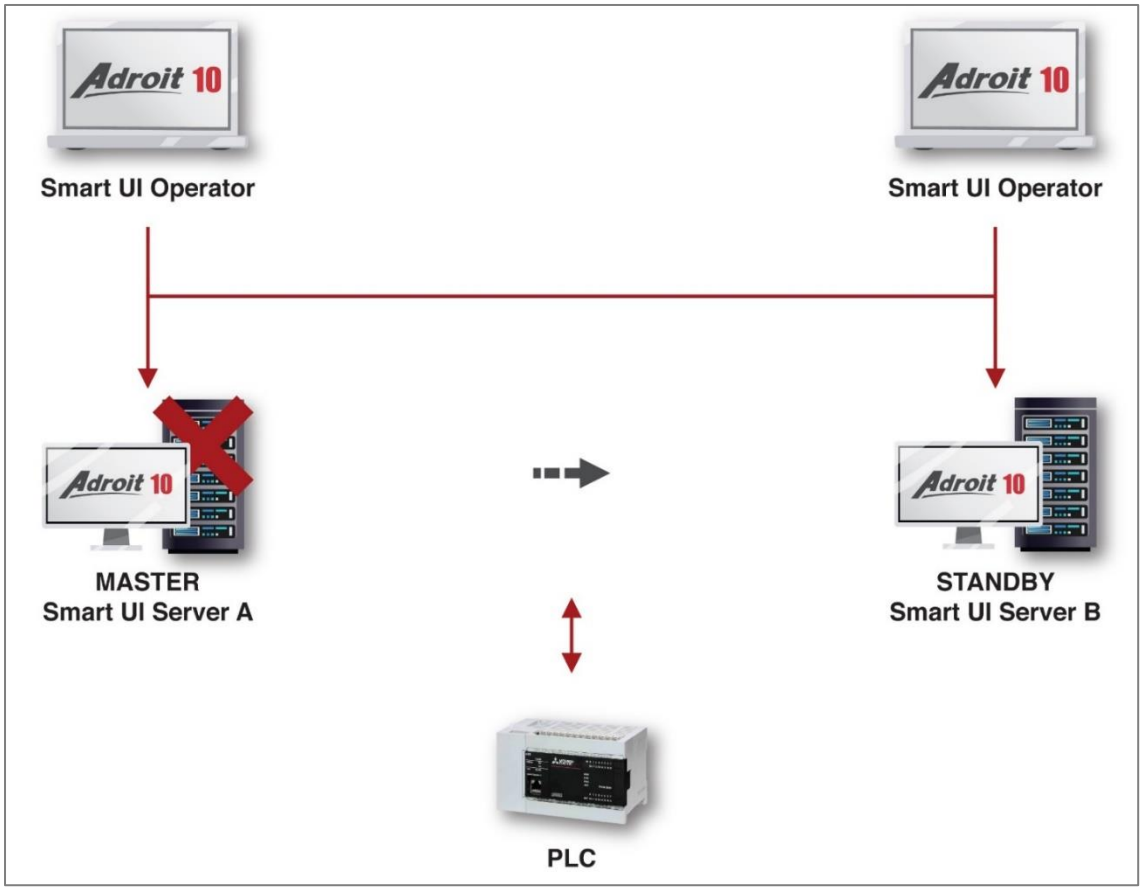


Figure 41 Replication

6.1.4. Switchover

When one server, particularly the master server, is removed from the cluster, the remaining *server* must become the master server. In addition, the *clients* need to take appropriate action if they were connected to the missing server.

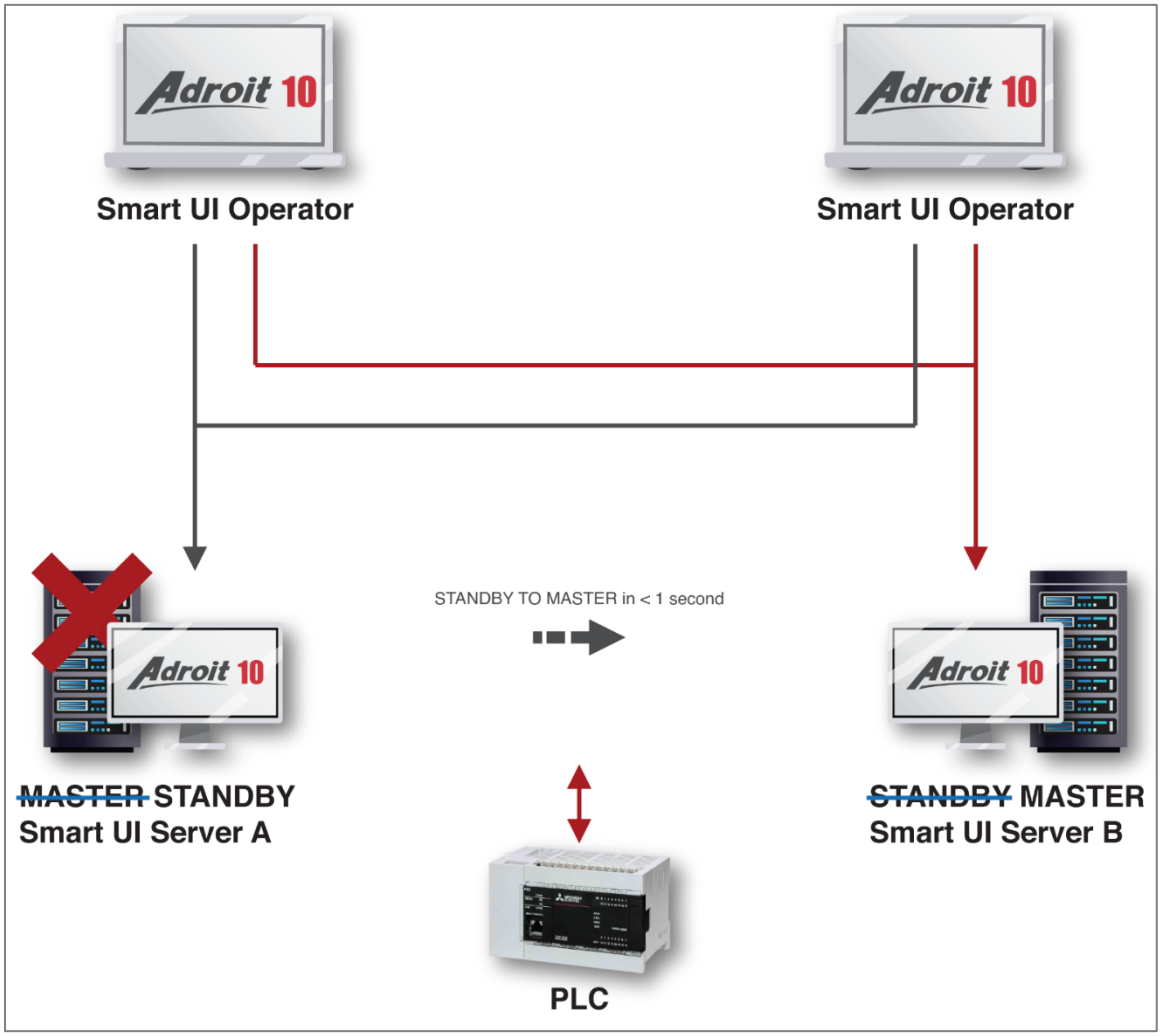


Figure 42 Switchover

6.2. Client to Server Connection

The Operator connection to all datasources is managed through the SmartUI Server. The basic premise of the Adroit UI connection to an Agent Server is that there is only one such connection. In addition, this connection is fixed to a particular server at start-up of the UI. Clearly, a static UI connection to a server is insufficient for clustering. The UI connection must be capable of dynamically roaming to any server in the cluster or automatically tracking the master server in the cluster.

When the machine or server to which the UI is connected fails, or the connection to that machine fails, the UI connection could automatically switch over to another server in order to provide high availability of services to the UI.

In Adroit, this is achieved for each of the UIs by listing all the possible Active Cluster Servers in the setup program. It does not mean that the UI is concurrently connected to more than one server, rather the premise of one UI connection is maintained but the connection can migrate to another server when that connection/server/machine fails, or by manually setting the connection at run time.

6.2.1. Basic principles of Dynamic UI connections

The following arbitrary scenarios of two servers and three operators illustrate some of these concepts. The servers and UIs are named Server A and B and Operator A, B and C respectively by their left to right ordering in the diagrams below.

Scenario 1: Semi-automatic UI switchover, i.e., all servers have manual master mode assignment and all UIs configured as automatic switchover UI clients. Servers A and B are running with Server A designated as the master server, while B is designated as the standby server. The actual configuration details in Adroit will be discussed later.

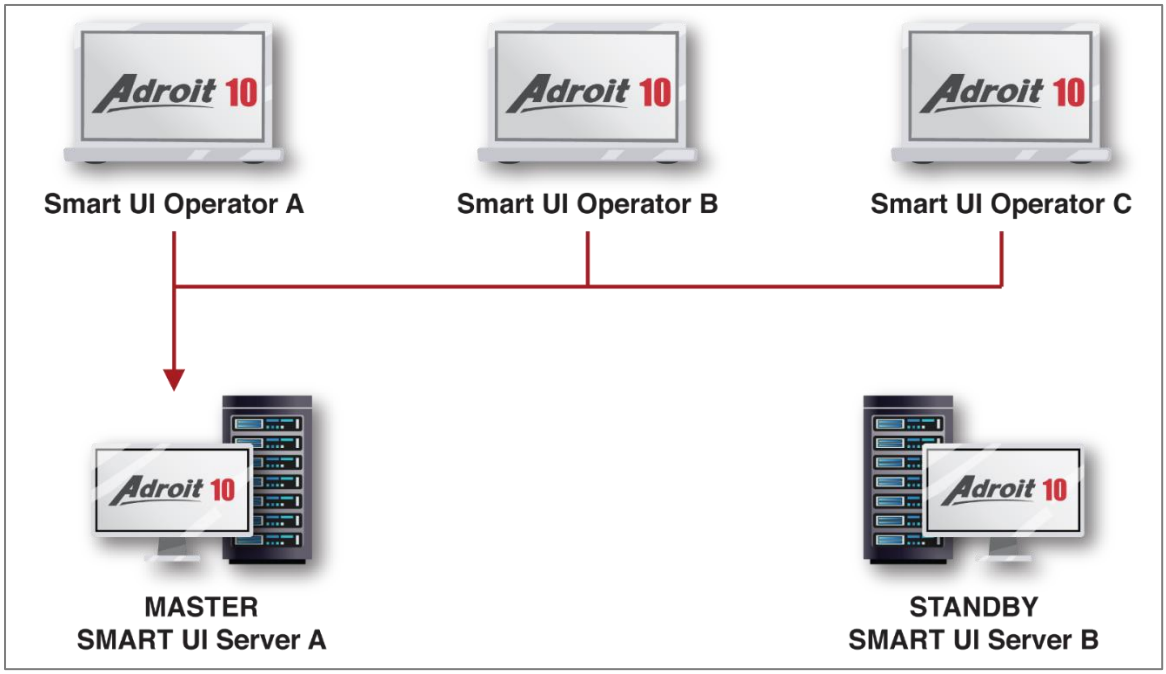


Figure 43 Semi-automatic UI switchover

UI C manually changes connection to Server B using a menu option and then marks A as standby, and B to be master.

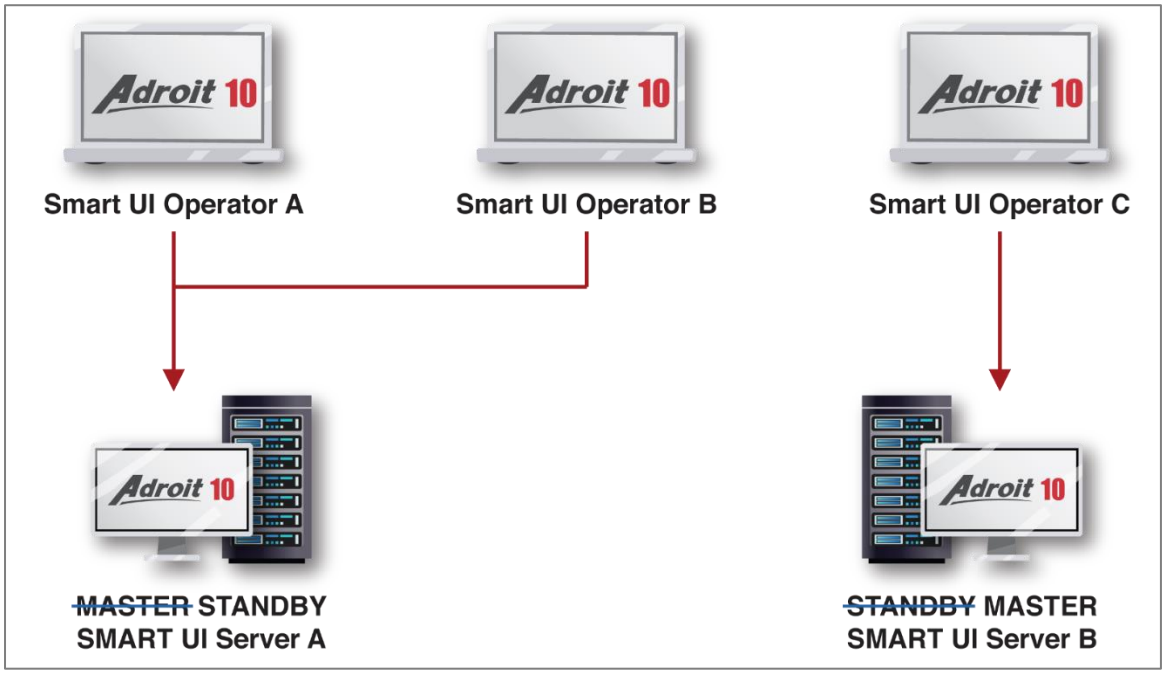


Figure 44 Semi-automatic UI switchover

Now all the UIs switch connection to Server B automatically because they have been configured as automatic switchover UI operators. The UIs know that Server A was master and now is in standby mode and that B is the new master, and that they must switch to Server B.

Server A is now a standby server and can, for instance be taken down for maintenance. Alternatively, Server B may have been a repaired machine, which is now manually set to take over as master after it has re-joined the cluster.

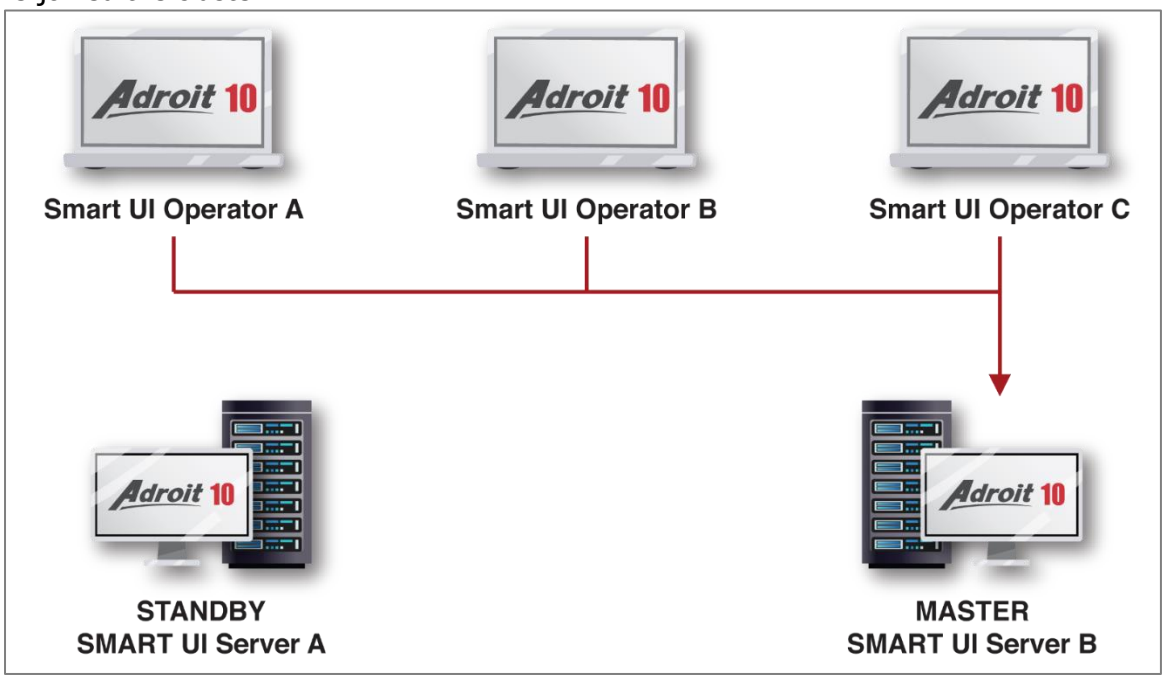


Figure 45 Semi-automatic UI switchover

Scenario 2: Continuing from Scenario 1 with the master server machine falling over. Master Server B falls over while all UIs are connected to it.

All operator UIs will only switch connection when a new master is made known. At this stage they lose connection to Server B and do not reconnect to another server because there is no master server in their list of server connections.

UI Operator C manually changes its connection to Server A using a menu option and sets Server A to be master, with server B immediately becoming the standby. Server A becomes the master through manual intervention, and only now do all UIs switch connection to Server A automatically, because they have been configured for the switchover option.

Scenario 3: Fully automated switchover, i.e., all servers has automatic master mode assignment and all UIs configured as automatic switchover UI operators.

When the master Agent Server goes down, the remaining server will automatically become the new master server. This is purely a server task and requires no UI intervention. Automatic master mode assignment can be used either to facilitate faster switchover when system failures occur, or during routine system maintenance.

If the master server falls over or an event (user configured) within the Agent Server (based on tags) occurs, thereby making the current master server unhealthy. The servers must now decide amongst themselves who is set to master. The UI connections will automatically track the master server when the servers decide who is master. Generally, there is always one master server in a cluster unless the network connection between the two servers is broken. This will cause both Agent Servers switching to master mode until the connection is restored, to ensure that the master server is always available.

If a user manually designates a new master server, the existing master server will switch to standby mode to honour the user's decision.

If a user manually designates a current master server to become a standby server, the server will only switch into standby mode if there is a connected partner server who is currently in standby mode. Then the existing master server will switch to standby mode as its partner changes into master mode.

6.3. Configuring Redundant Servers

Configure each of the servers as a server of type "Cluster Server", then specify the Active Cluster specific configuration options. This is done using the Adroit Configuration Editor application – Agent Server Setup.

6.3.1. Server Side Configuration

Client Runtime Transaction Shadowing

All transactions initiated by a client on a particular server may require shadowing to all the servers in the cluster in order to complete data replication.

For instance, if from within a picture, a UI changes the value of an unscanned tag in a cluster server, the remaining cluster servers will not receive this new value in order to update their replicate of the tag. Therefore, it may be necessary for a cluster server to replicate these operations to its partners in the cluster.

This constitutes runtime transaction shadowing and can be configured in the setup program in the Agent Server tab, server type options dialog. This is on by default and it should only be disabled after careful consideration of the consequences.

Server I/O Options

- **Shadow scanned data from master to standby**: The default mode of operation where only the master node communicates with the front-end devices and all I/O changes are shadowed to the standby.
- **Parallel scanning**: The alternative to automatically shadowing scanned data where both the nodes communicate with the front-end devices and I/O reads and/or writes can be disabled on the standby server.

Server Master Mode Assignment

Configure the desired master mode assignment strategy that is either manual or automatic. If an automatic assignment strategy is selected, then optionally configure the post start-up period after which the server will attempt to set it as master.

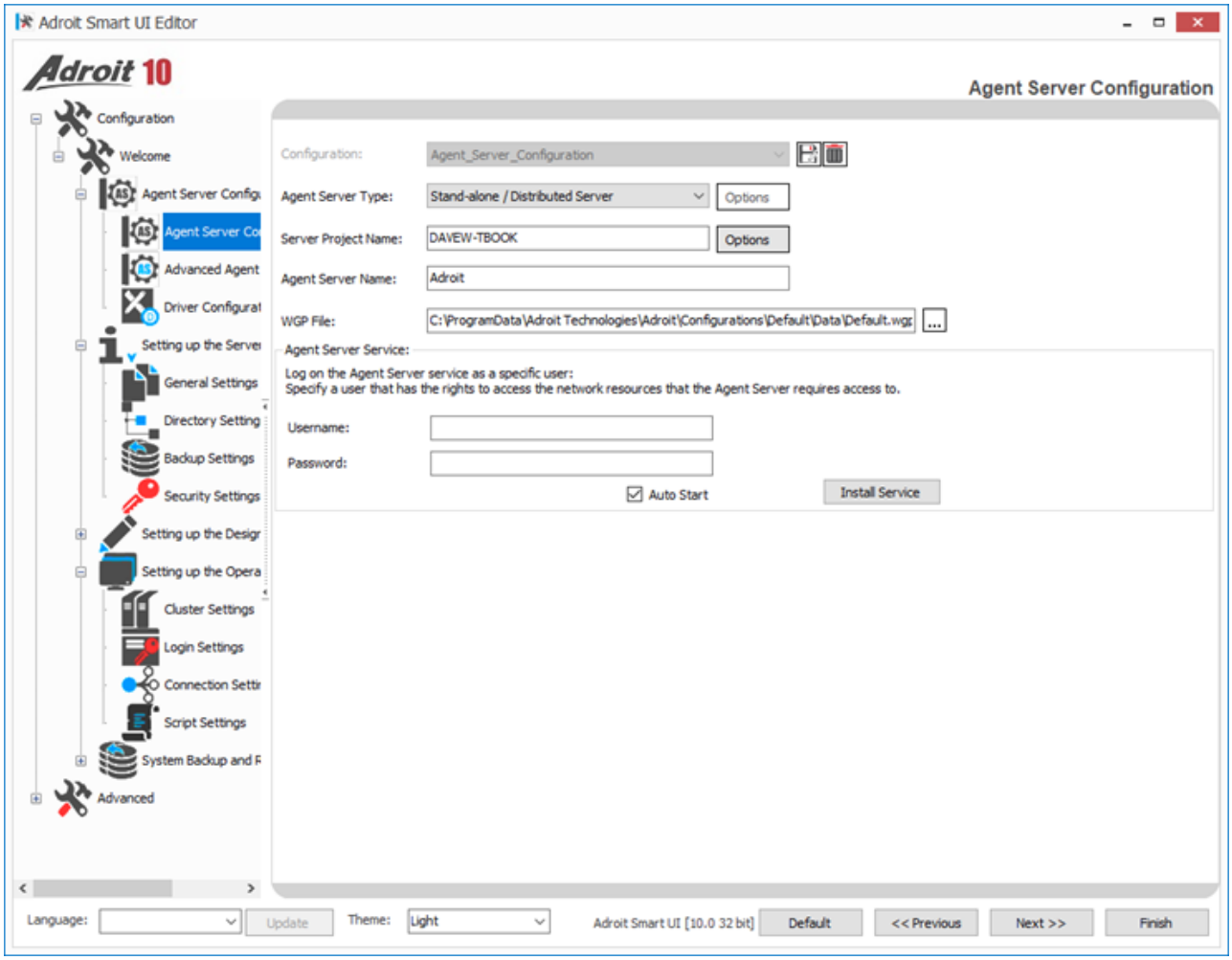


Figure 46 SmartUI Configuration Editor

6.3.2. Client Side Configuration

In the case of Active Clusters, the following configuration options are required:
Open the Configuration editor, either as an application or launch it through from the Configuration tab in the Designer. Making an Operator Cluster Aware is a done in the Operator Configuration Set-Up and is as simple as checking a box.

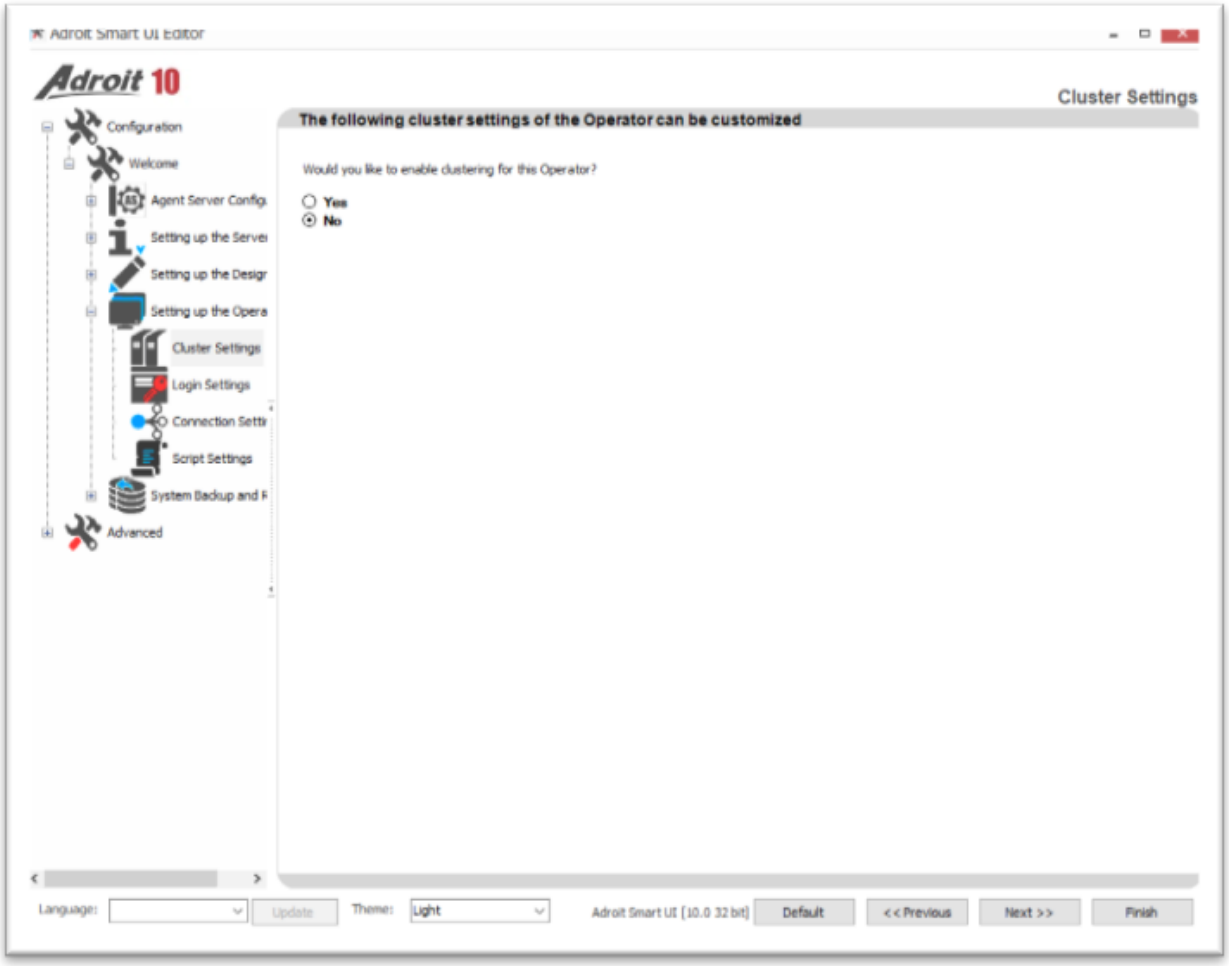


Figure 47 SmartUI Editor – Cluster Settings

7. Application Protocol Interface – Gateway to Performance

Adroit is an *Open Automation Technology.* At the heart of the server is an application protocol interface encompassing Agent types and instances with their associated properties, attributes, or slots depending on your preferred terminology.

Furthermore, this application protocol interface Adroit is completely open – allowing access to *any* property of *any* object from *any* application, written in *any* language, and from *any*where that has sufficient security privilege and access to a server with at least one available network connection. This extraordinary degree of openness is achieved by virtue of the fact that the Adroit application protocol interface object model is exposed to other applications via a COM/DCOM interface.

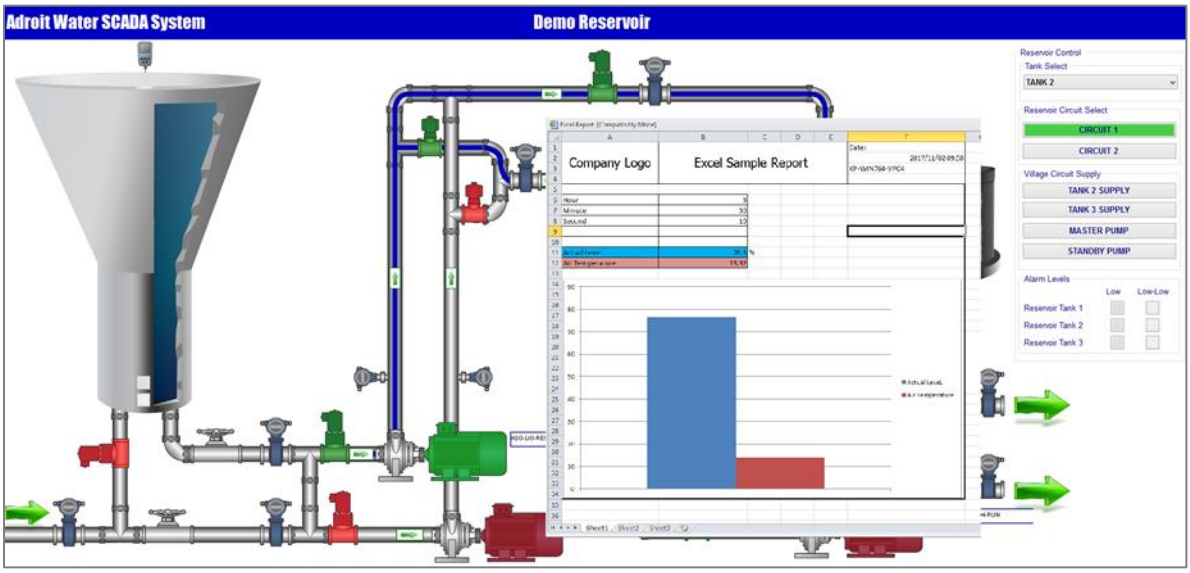


Figure 48 Application Protocol Interface

Specifically, on installation servers register an ActiveX Automation Server interface and corresponding type library that can be accessed from any COM/DCOM aware application such as industry-standard spreadsheet and database packages. For example, a remote user running MS Excel can request a list of tags from Adroit and then use these tags to obtain a snapshot of data and/or a real-time update. On installation, a range of samples illustrating use of standard spreadsheet and database packages are available for users to explore, understand and customise for themselves.

7.1.1. Bulk Engineering Tools - EXCEL

The standard installation includes a very powerful EXCEL plug-in that allows a live connection to an Agent Server and allows users to bulk engineer Agents on the fly in real-time. This saves an enormous amount of engineering time. All the Agents, Scanning, Alarming can be freely configured and imported to your project Agent Server.

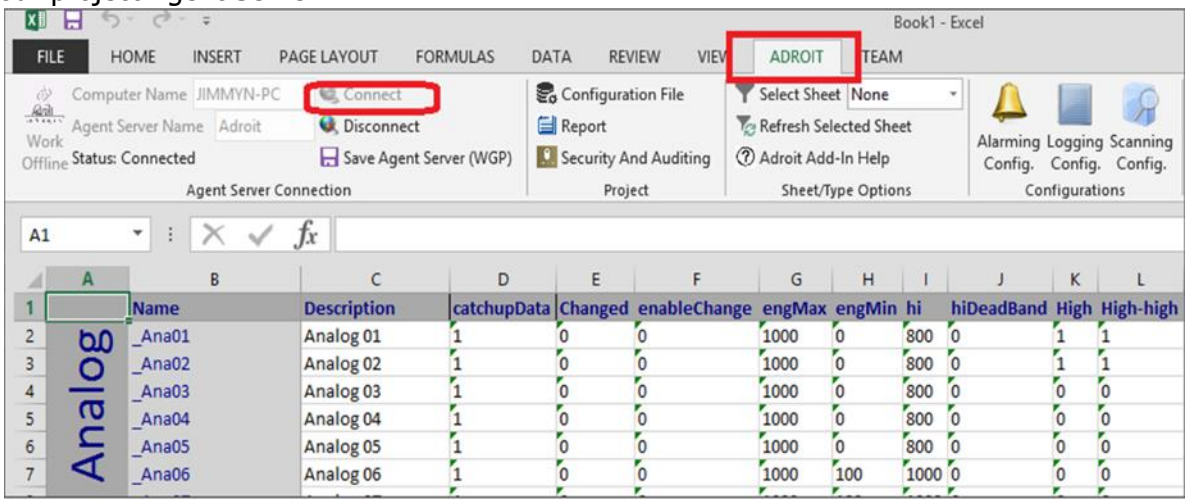


Figure 49 Bulk Engineering Tools - EXCEL

7.1.2. Open Interfaces

The ActiveX Automation Server Interface is a simple yet powerful API designed to provide full access to Agent Server from 3rd party applications running on different development environments such as Microsoft's Visual C++, Visual Basic, MS Excel, MS Access and Delphi, etc.

OLE DB connectivity means that Adroit can connect to any OLE DB provider, unlike ODBC that was only created to provide access to relational databases. It is ideally suited to writing data inside the Agent Server periodically to a database, such as at the end of a batch job, for reporting reasons. As its transactions can be done on demand, they can be scheduled and/or they can be triggered.

The *Script Agent* provides a way for the user to produce process reports by using a Visual Basic or Java script. This script can then be executed on demand, they can be scheduled and/or they can be triggered. The Script Agent is able to produce reports in the following formats:

- A text file (*.TXT);
- A comma-delimited file (*.CSV);
- An Excel spreadsheet (*.XLS);
- As an e-mail sent to a designated person; and
- A web page (*.HTM) or (*.HTML).

Other types of reports compiled from data within Adroit can either be exported to 3rd party packages for formatting and printing, such as the polar coordinate plot using Microsoft Excel, below, or can be created directly using 3rd party reporting applications such as Adroit Report Suite.

7.2. Management Information and Reporting

Within Adroit, reporting falls into the categories of Report Suite, Event Reports and Process Reports.

7.2.1. Report Suite

The Adroit Report Suite is a value-add information system based on specific Adroit agent data logged to an OLE DB database. Information is presented in the form of web based reporting through the Microsoft SQL Server Reporting Services platform.

If you have the ability to develop your own reports, then using the DBAccess Agent or DBLog you will be able to build your own web-based Reporting solution. Alternatively, we encourage you to look at the Adroit SCADA Intelligence product which offers the latest business intelligence platform for reporting and analysis.

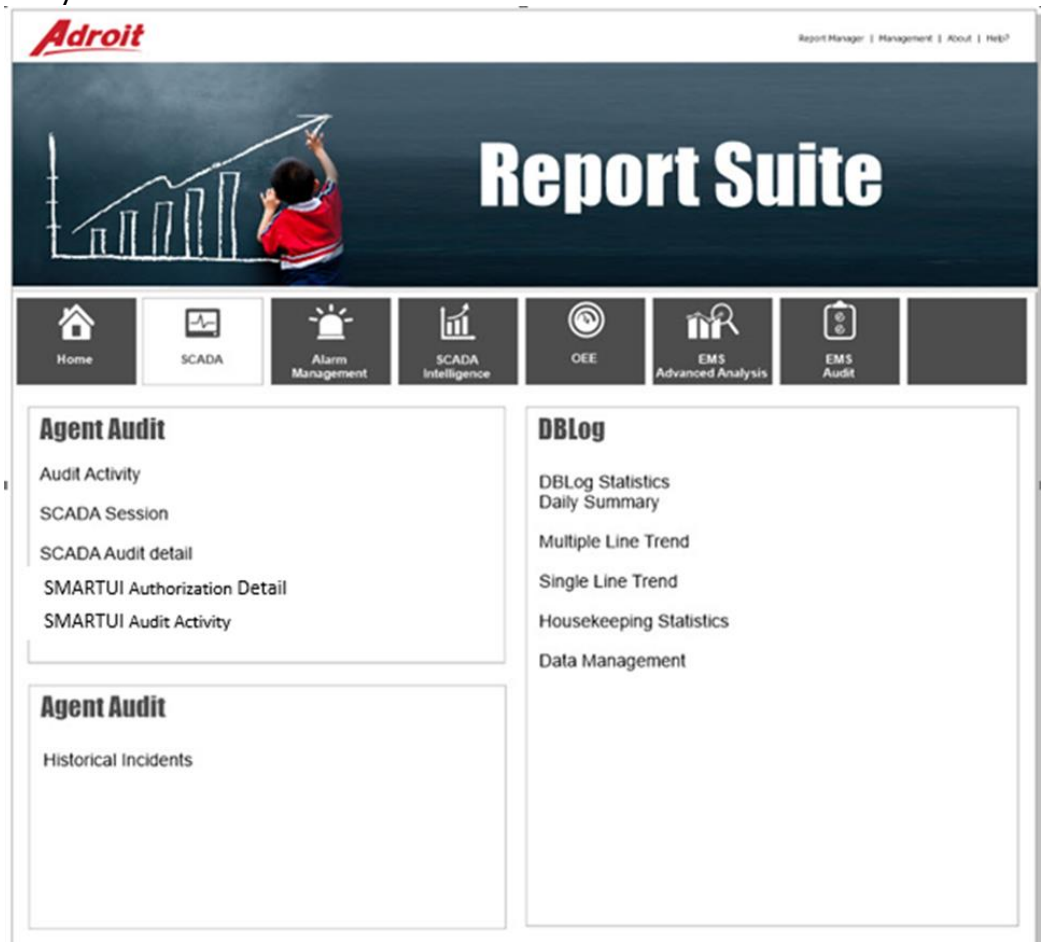


Figure 50 Report Suite

7.2.2. Report Pack

Built-in Microsoft Reporting Services the reports offer a basic insight to your system and process as well as offering you a free web-based view into historical trends. These are managed reports that are included in the standard installation and categorized as follows: Auditing, Datalogging and General.

Auditing Reports

Auditing provides the tools necessary to view the activities of the Agent server and SmartUI server with reports such Adroit Audit Activity and SmartUI Audit Activity. It is also a report set to measure the duration of the session of the Agent server with a report such as the Agent Server Session.

This version of the Adroit Report Pack also introduce us to a report that shows changes that required authorization on a SmartUI data element (agent.slot). This report is built on the same methodology of Title 21 CFR Part 11.

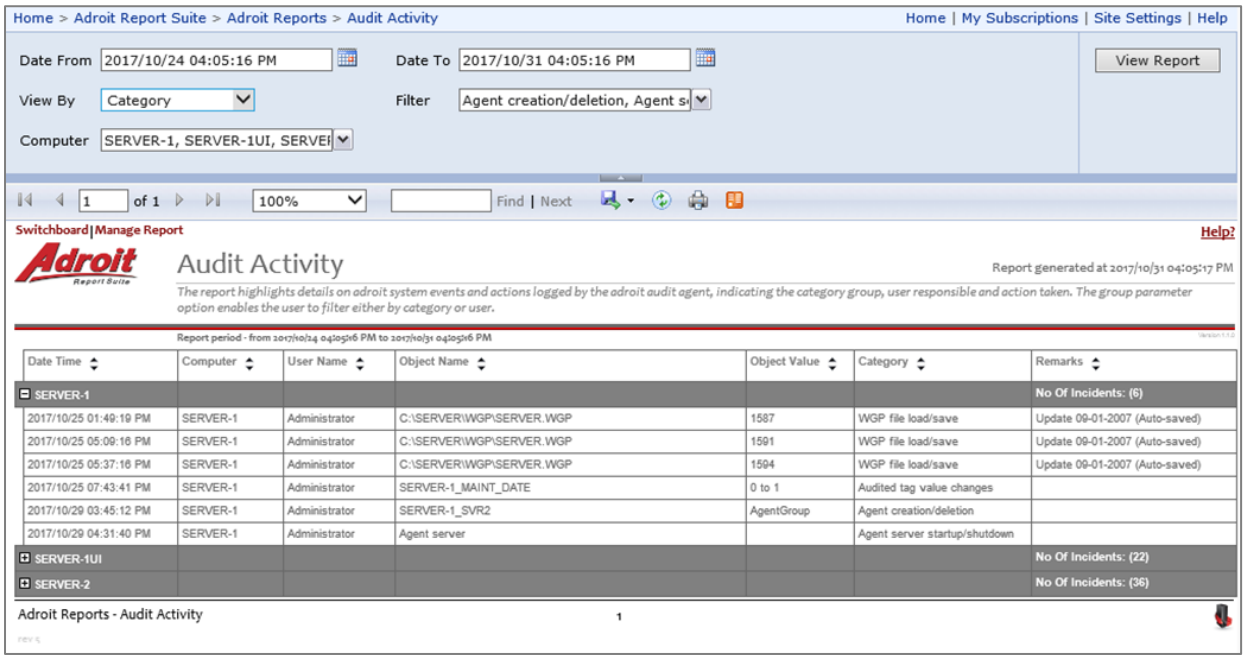


Figure 51 Audit Activity Report

Data Logging Reports

Data logging is extended through a standard DBLog agent that can be configured to log data points for any agent to an OLEDB compliant database. The report set combines summary tables with statistics and line charts to plot data trends. All of which combines through drill down and drill through functionality provided.

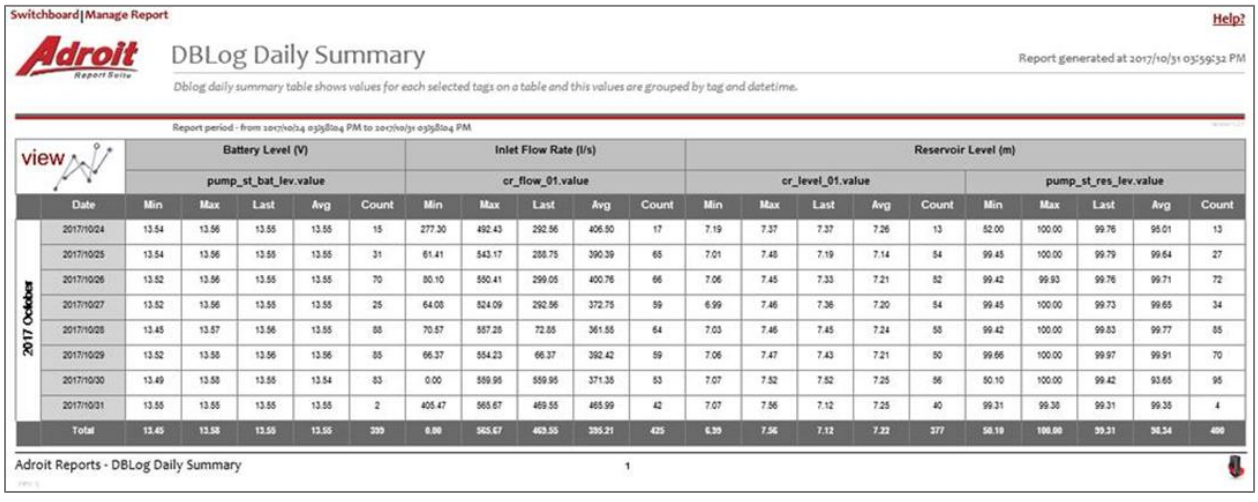


Figure 52 DBLog Daily Summary Report

Figure 53 Multiple Line Trend Report

General Reports

General refers to ad hoc reports based around various server or agent functionality. A New DumpConfig (Export) agent was launched with Adroit that provides functionality to export specific agent types with current slot values as configured. A configuration export report is available to use as a configuration datasheet or snapshot view on agent status. Similarly, a Historical alarm report is based on a limited alarm management dataset for basic information.

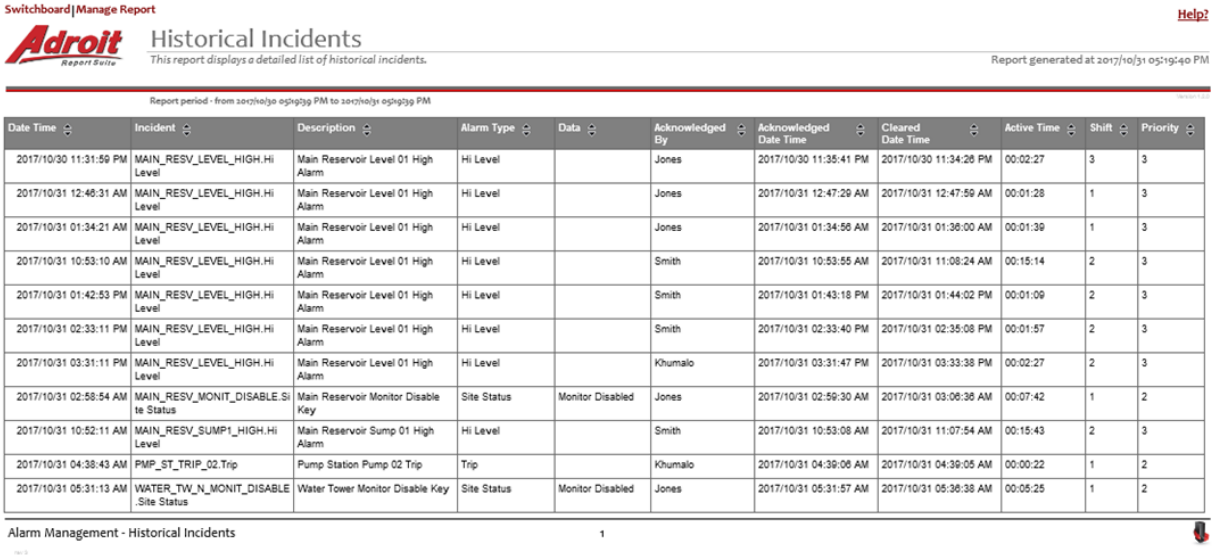


Figure 54 Historical Incidents Report

7.2.3. Event Reports

These are managed by the *event output Agent(s)* that have configurable event printing options. With these options events are printed a line at a time as the event occurs, or only after *x* events have occurred (where *x* is any configurable value between 2 to a full page), on a time interval basis or absolute time basis. Events may be printed on a local serial or parallel printer, or output to some networked device. Alternatively, events can also be written to any SQL compliant database.

7.2.4. Process Reports

These are printed based on either time/date, process event or on demand. The format, content and printing of the various report types is user configurable. The *DbAccess Agent* provides a link between Adroit and various external databases such as Access, SQL Server, FoxPro and Oracle by using OLE DB technology that is supported by Microsoft.

OLE DB connectivity means that Adroit can connect to any OLE DB provider, unlike ODBC that was only created to provide access to relational databases. It is ideally suited to writing data inside the Agent Server periodically to a database, such as at the end of a batch job, for reporting reasons. As its transactions can be done on demand, they can be scheduled and/or they can be triggered.

The *Script Agent* provides a way for the user to produce process reports by using a Visual Basic or Java script. This script can then be executed on demand, they can be scheduled and/or they can be triggered. The Script Agent is able to produce reports in the following formats:

- A text file (*.TXT);
- A comma-delimited file (*.CSV);
- An Excel spreadsheet (*.XLS);
- As an e-mail sent to a designated person; and
- A web page (*.HTM) or (*.HTML).

Other types of reports compiled from data within Adroit can either be exported to 3rd party packages for formatting and printing, such as the polar coordinate plot using Microsoft Excel, below, or can be created directly using 3rd party reporting applications such as Adroit Report Suite.

8. Performance Anywhere Client

The standard installation includes a single free Performance Anywhere client. It allows users to build and deploy HTML5 mobile and web-friendly interfaces to the Adroit.

Install the **Performance Anywhere Client** on your web server. The Performance Anywhere solution makes use of HTML5 interface, which makes it unattached to an O/S (unlike the 2 options above which can only run on a fully-fledged Windows O/S) and is perfect for mobile devices. The Operator can be launched from within any browser that supports HTML5 and by just specifying the web URL.
The Performance Anywhere client has "widgets" or graphic objects that include;

- Alarm Lists for Adroit
- Historical Trends
- Bar Charts
- Analogue Displays
- Buttons and Sliders for interacting with Agent Server

A familiar environment makes building reports very easy. Create a page, add a widget and bind the value to a data element using the browser, save and run.

Figure 55 Performance Anywhere Client

9. Utilities

Utilities has been added to provide a central location from which the installed utilities can be launched, instead of installing a host of separate shortcuts for each individual utility.

This shortcut is also installed on the Desktop. This application also describes their supported command line options and by right clicking the utility you can specify which command line options you want to use when launching the utility.

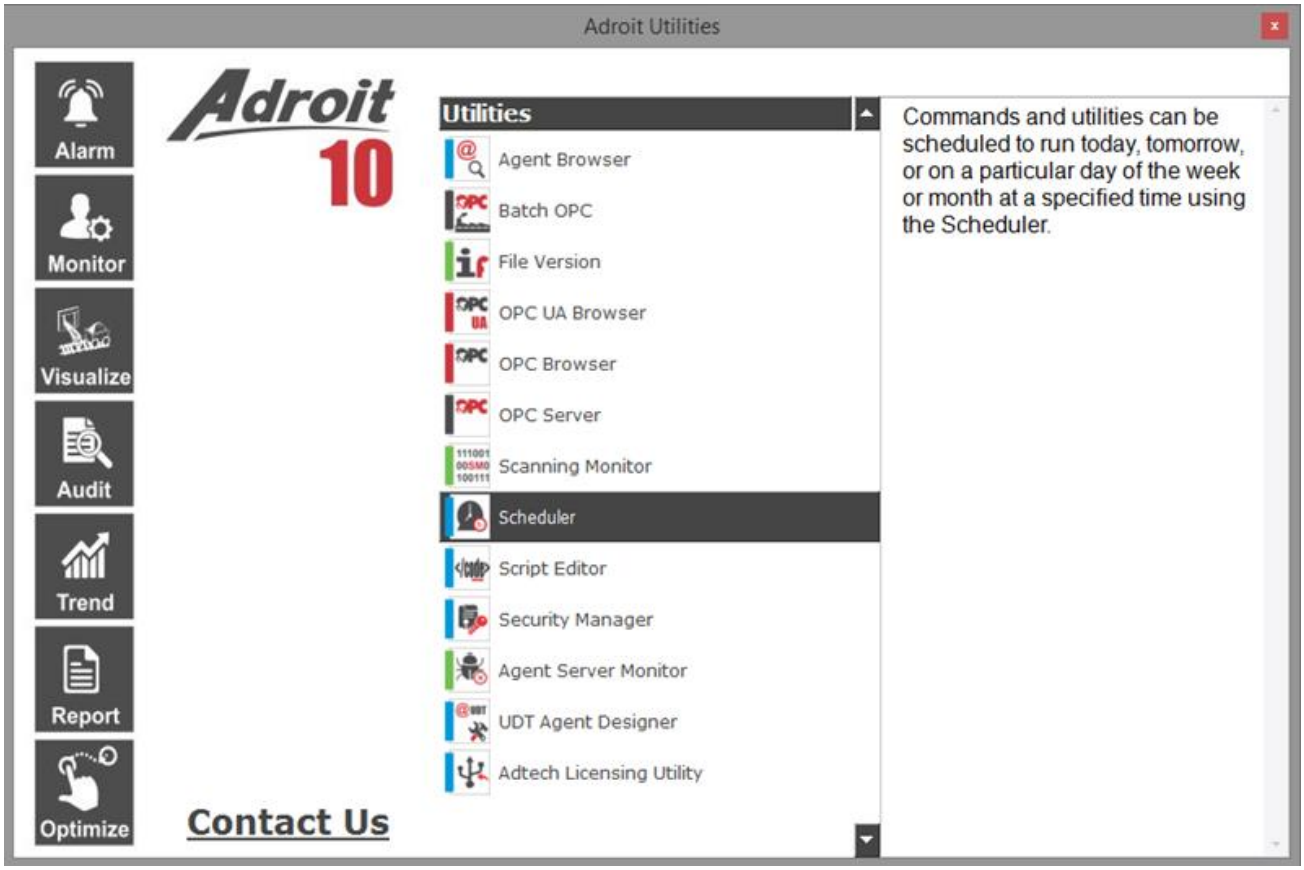


Figure 56 Utilities

9.1. Agent Browser

The Agent Browser is a utility that allows one to browse the configured tags within an Agent Server to be viewed and selected if required for configuration purposes. This also displays status slot names and tag (Agent.slot) values and, if the necessary command line option is provided, these tag values can be modified.

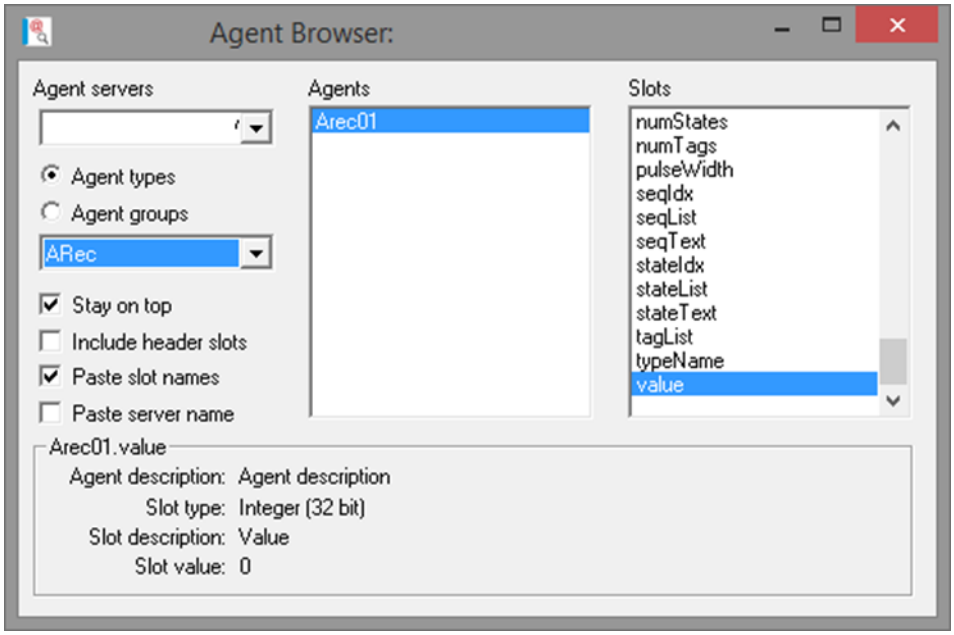


Figure 57 Agent Browser

9.2. File Version

The File Version tool allows you to view the version of your files for your Adroit installation.

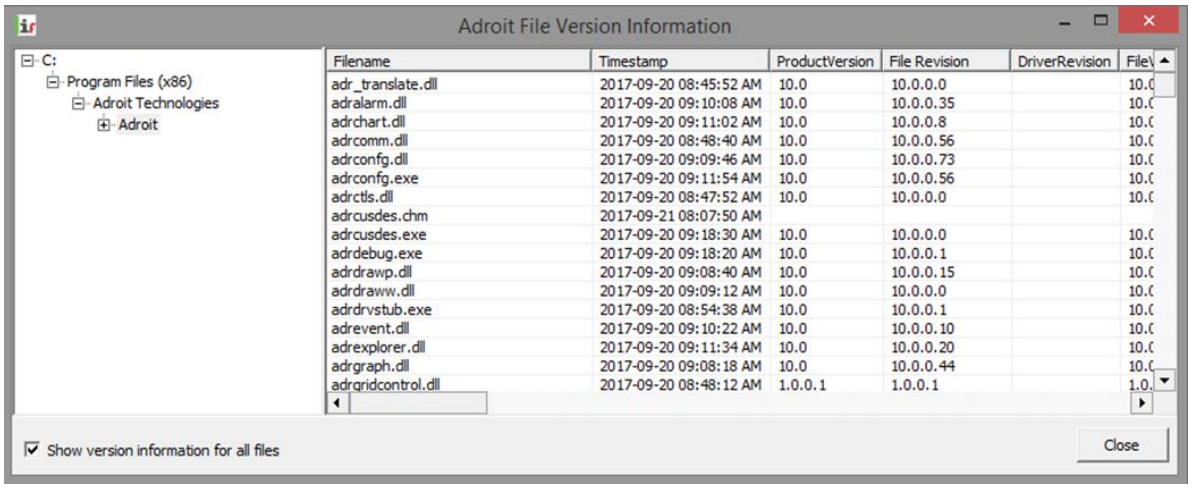


Figure 58 File Version

9.3. OPC Technology

9.3.1. Adroit as an OPC Client

The OPC Client driver opens up the product to hundreds of third-party OPC Servers available on the market.

9.3.2. Adroit as an OPC Server

The OPC Server gives an OPC Client application access to all of the tags as I/O points. Initially the server must be run manually for configuration purposes, but thereafter whenever a client application connects it will launch itself automatically. When the OPC Server is started, it is immediately added to the tray in the bottom right of the screen, as the following icon. Double-click this icon to open the OPC Server application.

When typing in an OPC item name, the "dot" notation used in OPC has been extended in that an OPC item can be referred to firstly by its node, a node is basically the same as an Agent Server, but can also transparently provide redundancy (especially for clustering) by specifying a backup Agent Server, this backup server then provides the values if the preferred server is not available. The node, which is configured by default, has the local machine's network name and refers to the local Agent Server.

The OPC Server can run on any machine from which it will connect to the Agent Server specified by the currently selected node. To avoid any DCOM setup considerations when creating remote OPC connections, it is recommended to run the OPC server on the same computer as the client application.

This application automatically monitors the Agent Server connection and informs OPC clients of any disconnections by displaying all tags in red in the OPC Server item-address browser. When the connection is re-established with the server, all items are displayed in green, where appropriate.

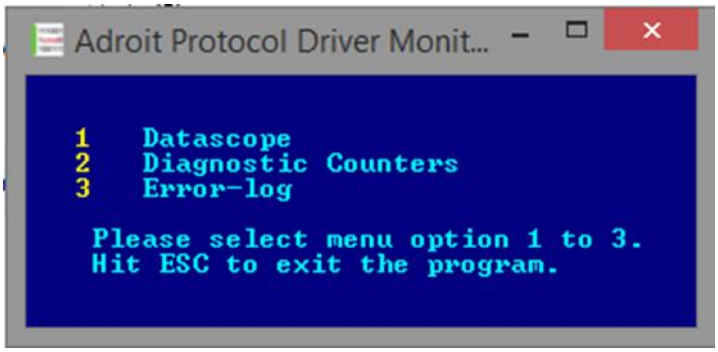
9.3.3. OPC UA Client

The latest version of Adroit is also an OPC UA client.

9.4. Protocol Driver Monitor

Adroit includes a Protocol Driver monitor program that allows the user to monitor any of the Adroit protocol drivers. This program allows the user to:

- Monitor the communications between Adroit and the external device;
- Display statistics for the comms line; and
- View any errors generated by the external device.



9.5. Scanning Monitor

The Scanning Monitor utility is designed to provide information on the scanning (communications) configuration for a particular Agent Server to and from its configured front-end devices, such as PLCs, RTUs, etc. The provided information can then be used to manually fine-tune the scanning subsystem to achieve optimal performance.

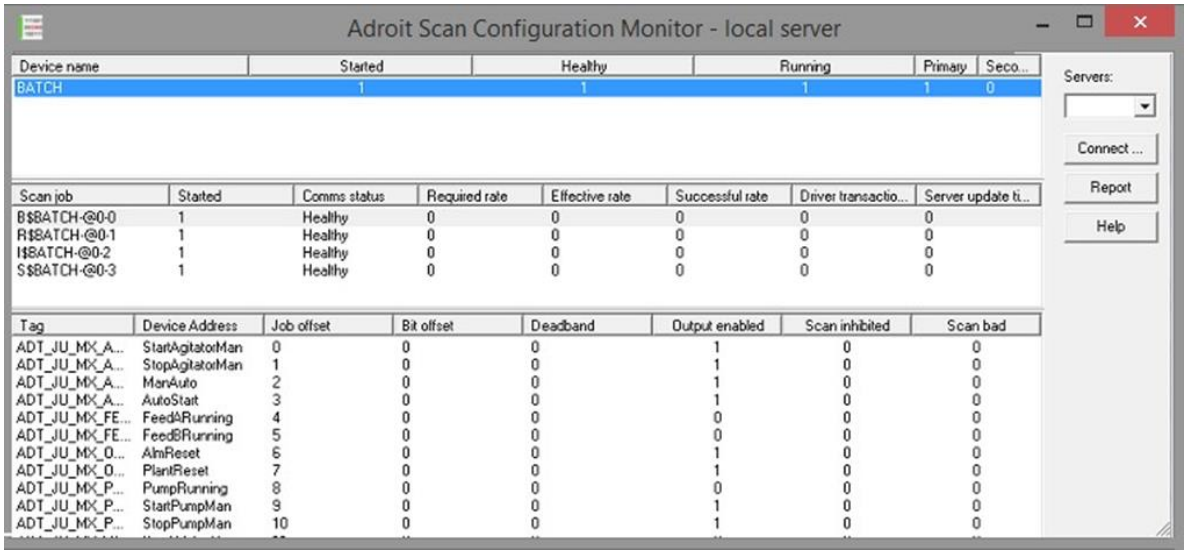


Figure 59 Scanning Monitor

9.6. Scheduler

Commands and utilities can be scheduled to run today, tomorrow, or on a particular day of the week or month at a specified time using the Scheduler. Typical examples of its use would be printing a report, archiving a file, starting a backup procedure, extracting logged data or doing a Putslot operation.

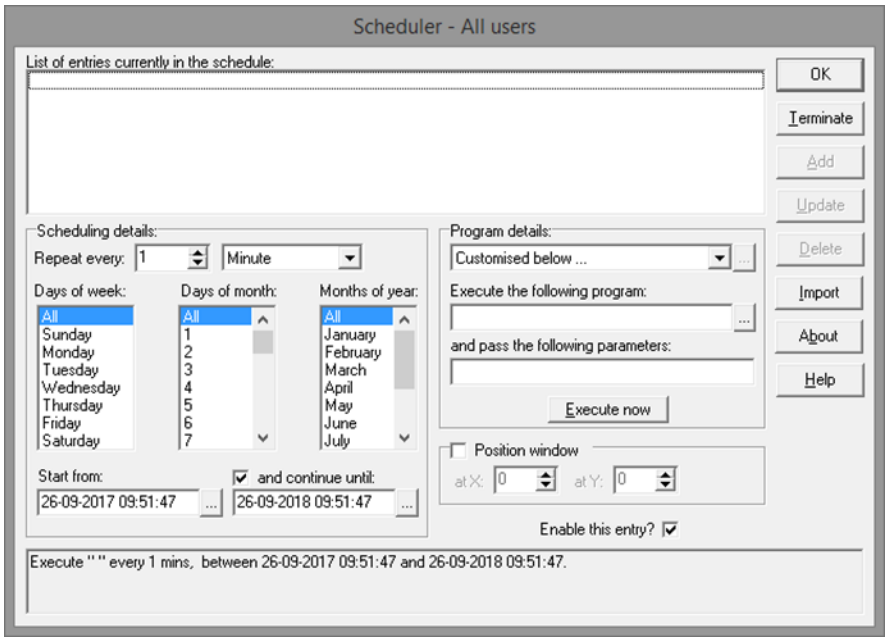
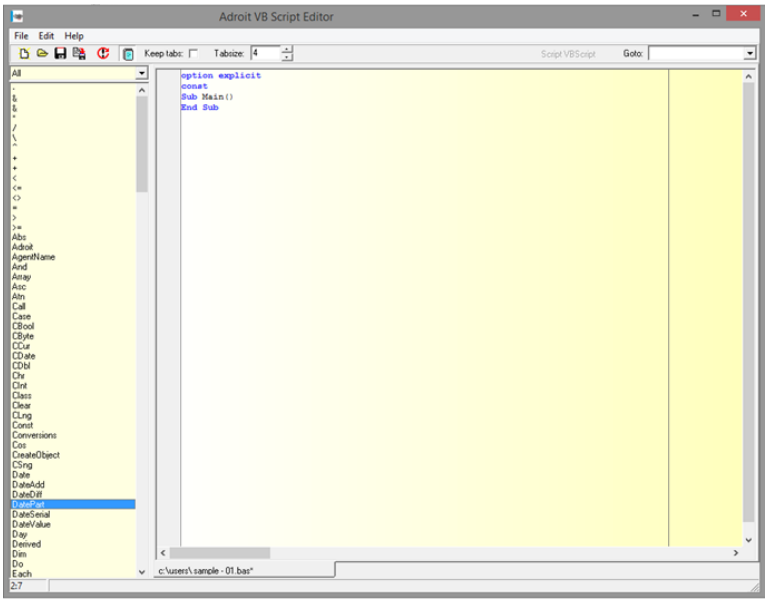


Figure 60 Scheduler – All users

9.7. Script Editor

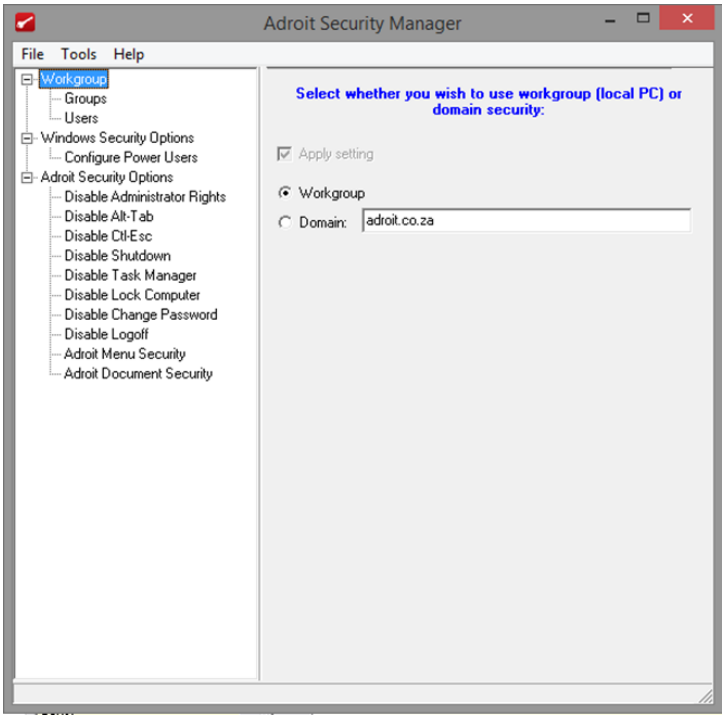


This Script Editor is usually launched from the Script Agent and the derived custom Agents edit dialogs so that their VB Script files can be edited. This editor is also launched by the Custom Agent Designer when adding VB Script functionality to a derived custom Agent.

It allows the concurrent editing of the following script file types: Custom Agent VB Script files (*.VBS), Script Agent VB Script files (*.BAS) and Custom Agent Java Script files (*.JAV).

9.8. Security Manager

The Security Manager provides a single utility that contains all the Windows and security settings you require to fully secure your installation. This utility also allows you to save the security settings that you configure so that these can be applied to other computers, as needed.



The left hand pane of this utility provides the following three expandable/collapsible branches:

***NOTE**: These branches are displayed in the order in which you will typically configure these security settings.*

- **Workgroup/Domain**: specify the required Windows user groups and/or users for your installation. If necessary, the typical (default) groups and users can be created for you.
- **Windows Security Options**: these settings apply to the Windows operating system.
- **Security Options**: these settings specifically apply to the UI application.

You can also use the Security Manager to export and import configured users and groups.

9.9. Agent Server Monitor

The Agent Server Monitor allows advanced users to diagnose or troubleshoot the Agent Server and its client connections.

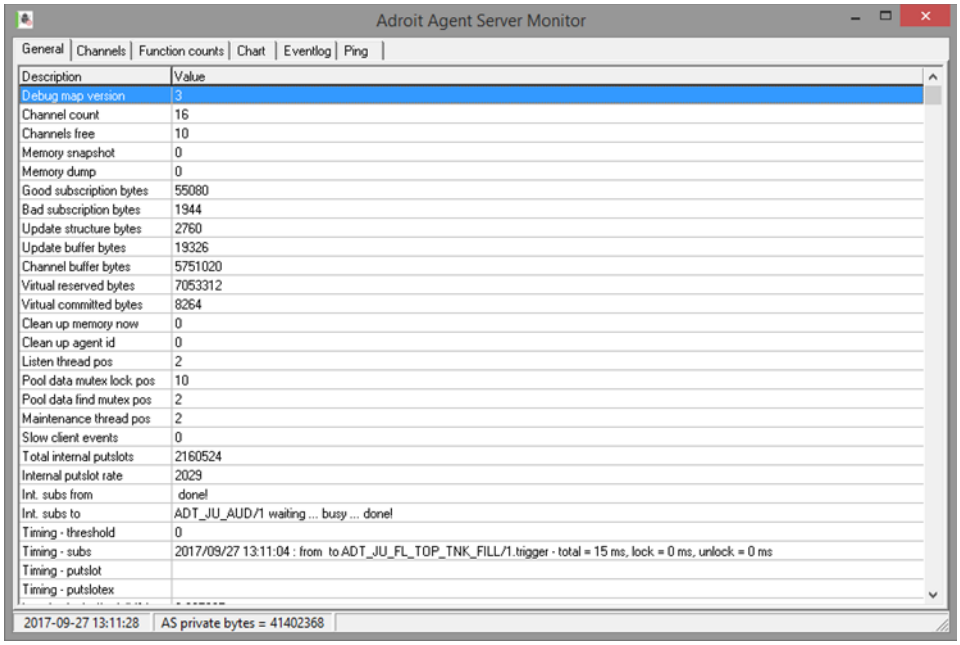


Figure 61 Agent Server Monitor

9.10. Tag Monitor

The Tag Monitor utility is designed to display the current values of one or more tag values from a particular Agent Server without having to specifically draw a mimic to do so. These values are initially displayed in tabular format but can also be monitored in a trend format.

9.11. Watches Window

The Watches window is designed to display the current values of one or more tag values from a particular Agent Server.

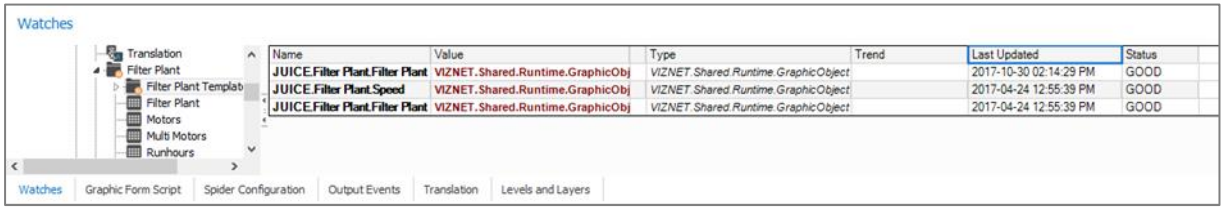


Figure 62 Watches Window

9.12. UDT Agent Type Designer

Currently new PLCs such as from Allen Bradley, Beckhoff and LonWorks optimize their communication performance by consolidating multiple data values into a single User-Defined Type (UDT).

Adroit provides the UDT Agent Type Designer to support this new functionality. This utility creates Agent types (templates) that duplicate the structure of your UDTs in your PLCs, by adding properties (basic data types) to them.

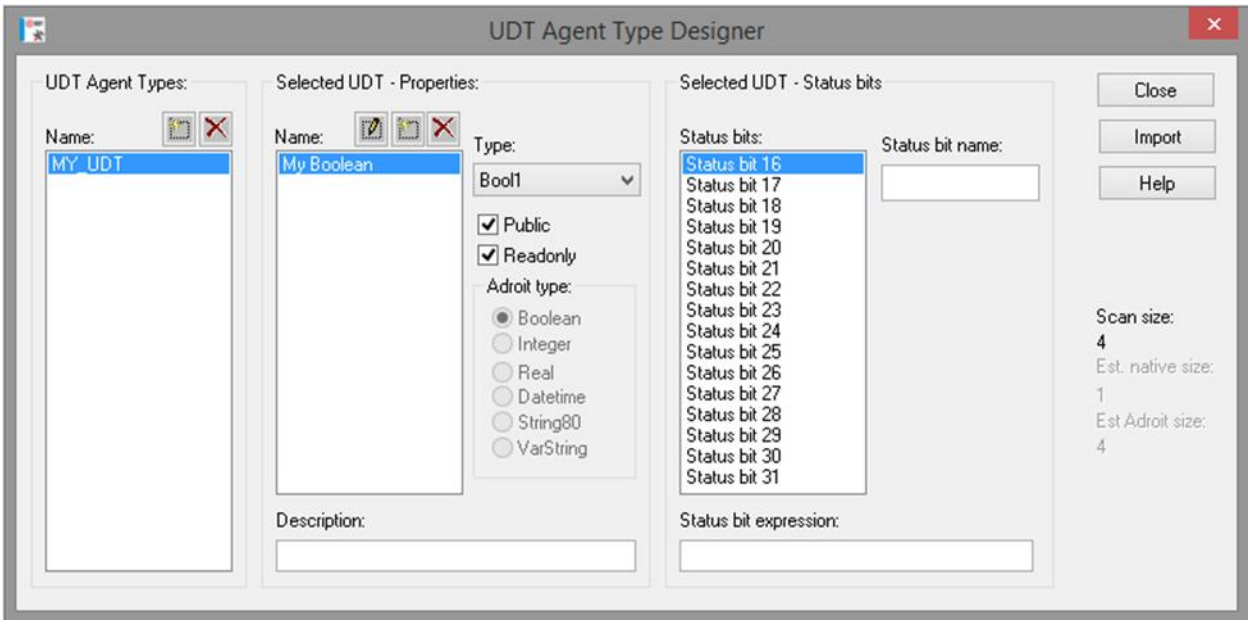


Figure 63 UDT Agent Type Designer

Since each defined UDT is a separate Agent type you can add more than one of each UDT to your installation, as is necessary. While this process of defining UDT Agent types is similar to the process of creating custom Agents, they differ from Custom Agents as follows:

- UDT Agent types can ONLY contain data and do not provide any scripting intelligence; and
- UDT Agent types contain a "raw bytes" rawValue slot that communicates to ALL of its other SLOTS (the ENTIRE UDT) in ONE transaction. This accounts for the improved scanning performance of UDTs.

9.13. Adtech Licensing Utility

The Adtech Licensing Utility allows users to get, view and/or update the product license which can be licensed via a HASP hardware key or a software license.

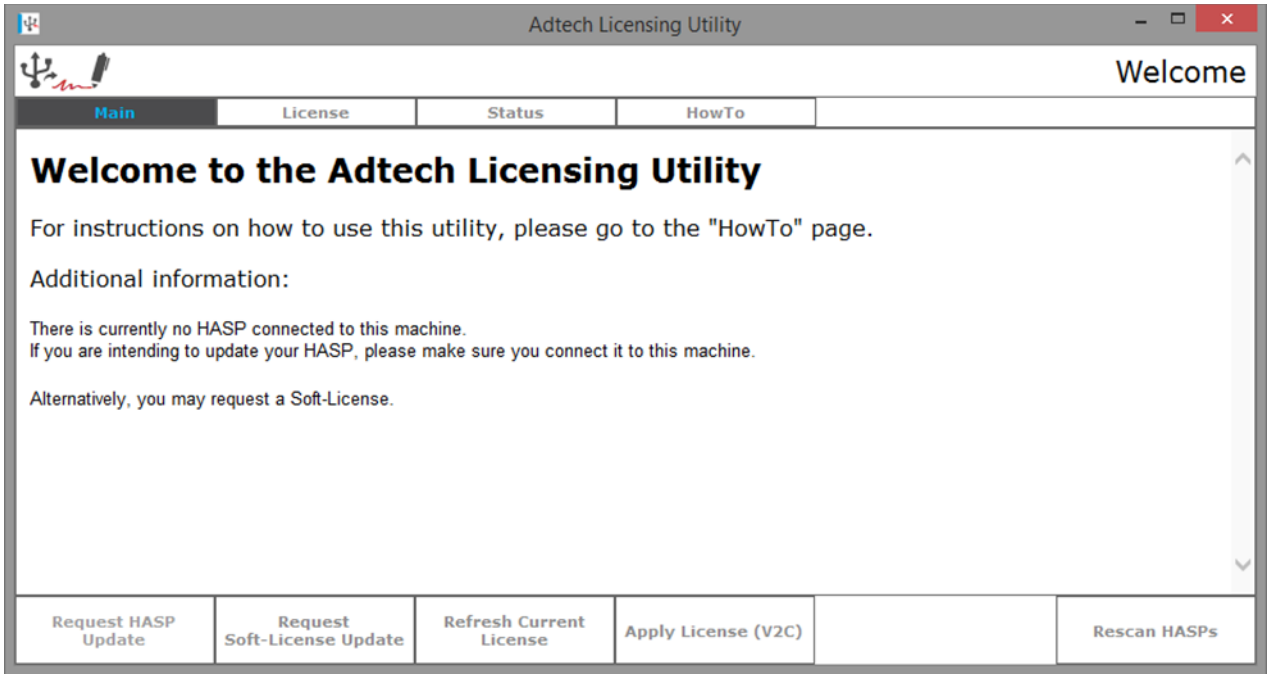


Figure 64 Adtech Licensing Utility

9.14. WANLink

WANLink allows two or more Agent Servers to communicate over a slow or Wide Area Network. The following diagram and explanation describes how WANLink works and how it is to be configured:

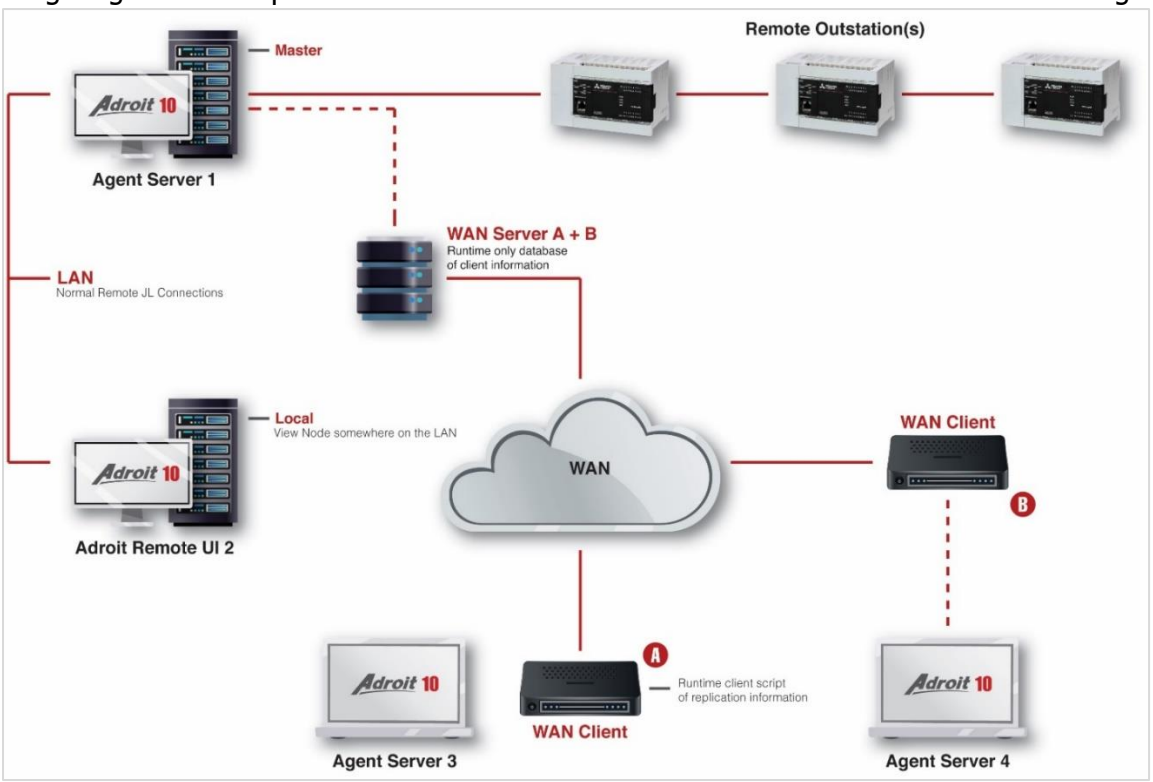


Figure 65 WANLink

WHERE:

- **Machine 1** is the "primary" Agent Server (AS1) which obtains its I/O from its remote outstations. WanServer is also running on this machine, which is connected to AS1, no other configuration is required.
- **Machine 2** is a Remote node which is able to view and control data on AS1 directly because it IS on the same LAN.

NOTE: *Since this machine does not have its own Agent Server it directly accesses the primary machine's Agent Server (AS1).*

- **Machines 3 and 4** are the "remote Agent Servers", these Agent Servers (AS3) and (AS4), may or may not have local data derived from other I/O sources. WanClient needs to be running on each of these machines.

Each WanClient needs to provide the following configuration:

- The communication settings of each WanServer that is required to communicate with (AS1); and
- The communication settings for the groups of tags that are to be communicated to each WanServer:
  - Select a **Normal** group, for links that require a one-way communication, so that the remote Agent Servers (AS3), (AS4) are only able to view the value of a "linked" tag and any changes made to this tag will not be made to its associated tag on the primary Agent Server (AS1).
  - Select a **Control** group, for links that require a two-way communication, so that the remote Agent Servers (AS3), (AS4) are able to modify the value of a "linked" tag and its associated tag on the primary Agent Server (AS1) will reflect this change.

NOTE: *With this method of linking a remote Agent Server (**Machines 3** and **4**) can operate as if it was a remote node (**Machine 2**) of the primary Agent Server it is being connected to (**Machine 1**).*

  - Which local tags in these remote Agent Servers (AS3), (AS4) need to be "linked" to tags that reside in the primary Agent Server (AS1) and the communication settings to its WanServer connected to the primary Agent Server (AS1).

NOTE: *It is not necessary to link to all the tags in the primary Agent Server (AS1), only link those tags that are required.*

NOTE: *Historical data cannot be transferred over Wanlink, only current runtime values will be sent and retrieved.*

In other words, Wanlink allows remote Agent Servers to behave as if they were normal remote nodes on a Local Area Network (LAN), because data can be read from the Agent Server and also written back. But the transmission and reception of this data behaves in an indirect manner, and not directly as a normal remote node would since speed is the most important limiting factor when communicating across a WAN.

10. MAPS Process Suite

10.1. Overview

MAPS stands for Mitsubishi MAPS Process Suite, an integrated software solution that combines the separate development tools of Mitsubishi Electric, Adroit Technologies and EPLAN.

- Mitsubishi Electric manufactures PLCs and Drives for the automation industry.
- Adroit Technologies develop the Adroit and associated software.
- EPLAN develop engineering design tools.

Adroit Technologies is focused around offering an integrated PLC/SCADA programming and management tool that works seamlessly with the Mitsubishi Q-series PLC hardware and the tried and tested Agent Server and through the Mitsubishi GX Works.

MAPS delivers value along the entire life-cycle and value chain of any automation solution, from the initial process design, the engineering phases by providing a more structured approach to projects, which enforces the use of re-usable objects and standards and structure.

MAPS, however, goes one step further to address the shortcomings of current offerings and solutions around the commissioning, handover and operations phase of an automation project. By using the EPLAN integration to ensure that all drawings, documentation and PLC and SCADA changes made are kept synchronized and up to date. This prevents the gradual degradation of the delivered solution and associated documentation as a customer maintains and develops the process system along the way and, thereby, ensures that the plant's integrity and efficiency remains at a high level long after the System Integrators have left site.
The Enterprise Manager is the central integrated engineering solution for your documentation, PLC programming and SCADA engineering - giving you a project life-cycle management solution.

10.2. MAPS Projects

We need to explain why and how MAPS provides a more structured approach to projects, enforcing the use of re-usable objects and standards and structure - so that you can better understand some of the benefits provided by MAPS.

Firstly, it needs to be said that MAPS is not a magical tool that does all the thinking and work for you. In fact, using MAPS forces you to think and to plan ahead of time – since the key to a successful automation project is PLANNING. So you should do the following before creating a project:

- Create a comprehensive schedule of all the electrical and instrumentation equipment required in the plant, which are organized according to the following extended physical model of the hierarchical structure specified by the ISA S88 and S95 standards:

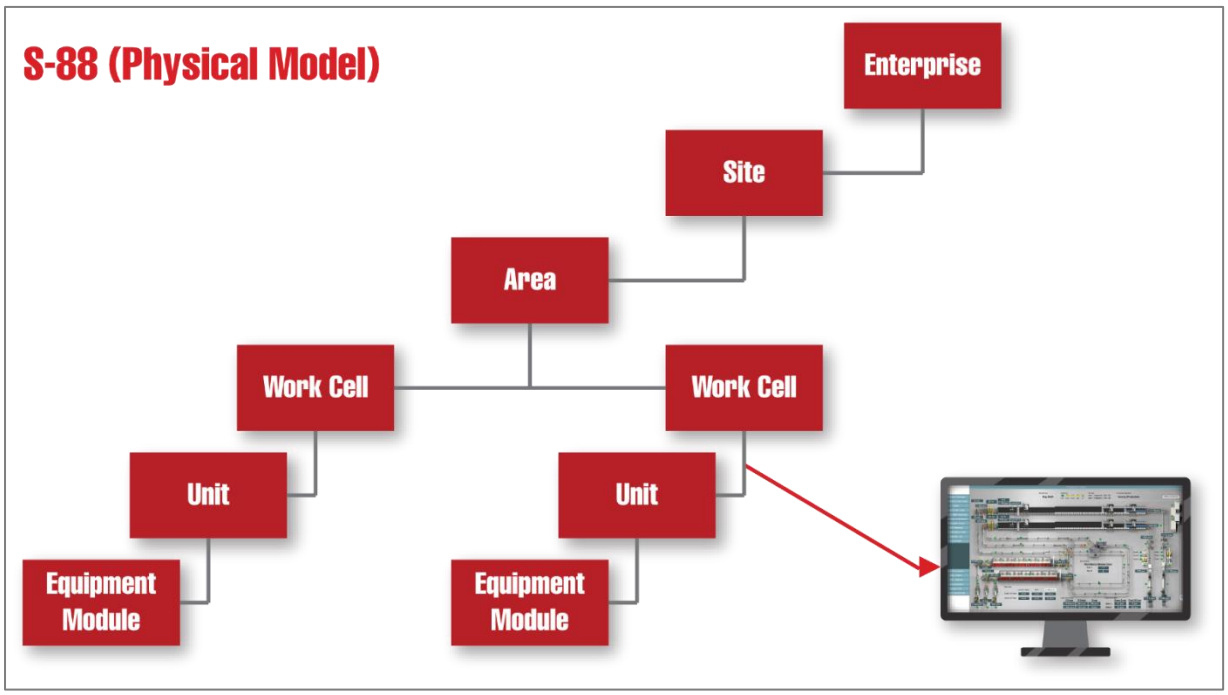


Figure 66 S88/S95 Extended Physical Model

- The **Enterprise** level is a collection of plants.
- The **Site** level is the plant (project). So you will have one project per site. For example, XYZ Foods.
- The **Area** level is the defined Plant Area/s of the plant (project) - Agent Server. For example, Batching Plant.
- The **Work Cell** level is the PLC/s of each defined plant area. For example, BATCHING PLC.
- The **Unit** level is the collection/s of equipment items within each PLC.
  Typically these relate to the plant area-specific operations, such as Filling, Packing, Flocculation or Batching. However, you can ALSO create a unit for each item of complex equipment i.e. equipment that consist of more than one Electrical or Instrumentation item, such as conveyors.
- The **Equipment Module** level consists of the defined items of equipment that are classified into either the Electrical or Instrumentation categories for each of the defined units.

***NOTE**: These hierarchical categories ALL have a 1-to-many relationship with their components i.e. a Site can contain MANY Areas, an Area can contain MANY Process Cells (PLCs) etc.*

- Specify and group the IO cards used in each PLC along with the virtual memory addressing to ensure efficient scanning (communication) and allocation by MAPS.

Secondly, templates are the means by which MAPS enforces the use of re-usable objects, since you need to assign a template to each **Equipment Module** or item of electrical and instrumentation equipment of your automation solution.

Each template associates a SCADA graphic (including an intelligent faceplate for operator control) and a PLC function block to an item of equipment. The template also creates the required SCADA tags for its various signals and their PLC variables and links these together.

***NOTE**: The available Templates are described in detail in the **Template Reference** section below.*

Thirdly, to take advantage of the complete automation solution life-cycle management provided by MAPS, your design should start with EPLAN Electric P8 application to perform the detailed Electrical design. Then use the generated Excel Exported workbook and import it to create your project. This will ensure that all your drawings, documentation, PLC and SCADA changes made are kept synchronized and up to date.

***NOTE**: You need to purchase the EPLAN Electric P8 and EPLAN API Runtime License in order to integrate the use of EPLAN into MAPS.*