

# CitectSCADA

Version 7.0

User Guide

July 2007

#### DISCLAIMER

Citect Pty. Ltd. makes no representations or warranties with respect to this manual and, to the maximum extent permitted by law, expressly limits its liability for breach of any warranty that may be implied to the replacement of this manual with another. Further, Citect Pty. Ltd reserves the right to revise this publication at any time without incurring an obligation to notify any person of the revision.

#### COPYRIGHT

© Copyright 2007 Citect Pty. Ltd. All rights reserved.

#### TRADEMARKS

Citect Pty. Ltd has made every effort to supply trademark information about company names, products and services mentioned in this manual.

Citect, CitectHMI, and CitectSCADA are registered trademarks of Citect Pty. Ltd.

IBM, IBM PC and IBM PC AT are registered trademarks of International Business Machines Corporation.

MS-DOS, Windows, Windows NT, Microsoft, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

DigiBoard, PC/Xi and Com/Xi are trademarks of Digi International Inc..

Novell, Netware and Netware Lite are are either registered trademarks or trademarks of Novell, Inc. in the United States and other countries..

dBASE is a trademark of dataBased Intelligence, Inc.

All other brands and products referenced in this document are acknowledged to be the trademarks or registered trademarks of their respective holders.

#### GENERAL NOTICE

Some product names used in this manual are used for identification purposes only and may be trademarks of their respective companies.

July 2007 edition for CitectSCADA Version 7.0

Manual Revision Version 7.0.

**Contact Citect today at [www.citect.com](http://www.citect.com)**

OCEANIA +61 2 9496 7300, NORTH AMERICA +1770 521 7511, LATIN AMERICA +1770 521 7511, AFRICA +27 11 699 6600,  
EUROPE +31 71 576 1550, MIDDLE EAST +31 71 576 1550, GREATER CHINA +86 21 6886 3799, NORTH ASIA +65 6866 3712,  
SOUTH EAST ASIA +65 6866 3712, INDIA +65 6866 3712.

# Contents

## Part I     Getting Started

### Chapter 1     Getting Technical Support

### Chapter 2     What's New in CitectSCADA v7.0

The Migration Tool .....	15
Clustering .....	15
Local Variables .....	16
Publish Alarm Property .....	16
Memory Mode .....	16
Client-side Online Changes .....	17
Publisher-Subscriber Model .....	17
Dual Network Support .....	17
Project-Based Network Configuration .....	18
Citect.ini Parameters in Version 7.0 .....	18
Cicode Functions in Version 7.0 .....	22
CtAPI Functions in Version 7.0 .....	29
Kernel Commands in Version 7.0 .....	29

### Chapter 3     Upgrading to CitectSCADA v7.0

Deprecated and Removed Functions .....	31
Upgrading Procedures .....	31

### Chapter 4     About CitectSCADA

### Chapter 5     CitectSCADA Tools

Configuration Tools .....	51
Runtime Tools .....	52
Drivers .....	53

### Chapter 6     Components of a project

Graphics components .....	56
Tags .....	57

Alarms .....	58
System components .....	59
Communications components .....	61
I/O Server components .....	61
Cicode / CitectVBA .....	62

## Chapter 7      Typical system scenarios

Standalone system .....	64
Distributed I/O system .....	64
Client-Server system .....	66
Redundant server system .....	67
Clustered control system .....	69
Reliable sub-control system .....	71
Load sharing system .....	73

## Part II      Using CitectSCADA

### Chapter 8      Planning a Project

The Physical Layout of a Plant .....	77
Operational Requirements .....	78
Project Design .....	80
Building Your Project .....	83
Deployment .....	86

### Chapter 9      Administering projects

Managing your projects .....	89
Archiving projects .....	97
Including projects .....	106
Working with the Project Editor .....	108
Using Find and Replace in a project .....	113

### Chapter 10      Configuring Your System

Running the Computer Setup Wizard .....	125
Project Configuration .....	126
Computer Role Configuration .....	126
Network Model .....	127
Internet Server Configuration .....	128
Alarm Configuration .....	128
Reports Configuration .....	129
Trends Configuration .....	130
CPU Configuration .....	130



---

	Events Configuration . . . . .	131
	Startup Functions Configuration . . . . .	132
	Cluster Connections Configuration . . . . .	133
	Time Configuration . . . . .	134
	Control Menu Security Configuration . . . . .	134
	Keyboard Security Configuration . . . . .	135
	Miscellaneous Security Configuration . . . . .	135
	General Options Setup . . . . .	135
	Finish . . . . .	136
<b>Chapter 11</b>	<b>Implementing Clustering</b>	
	Rules of Clustering . . . . .	137
	Cluster Definitions . . . . .	139
	Network Address Definitions . . . . .	140
	Alarm Server Definitions . . . . .	140
	Report Server Definitions . . . . .	141
	Trend Server Definitions . . . . .	142
	I/O Server Definitions . . . . .	142
	Assigning tags to a cluster at Runtime . . . . .	143
<b>Chapter 12</b>	<b>Building Redundancy into Your System</b>	
	I/O Server Redundancy . . . . .	145
	Data Path Redundancy . . . . .	150
	Alarms, Reports, and Trends Server Redundancy . . . . .	154
	Redundancy of Standalone Systems . . . . .	158
<b>Chapter 13</b>	<b>The Computer Setup Editor</b>	
	Getting Started with the Computer Setup Editor . . . . .	161
	About the citect.ini file . . . . .	161
	Before You Begin . . . . .	163
	Using the Computer Setup Editor Interface . . . . .	164
	Using Computer Setup Editor . . . . .	173
	Opening the configuration file . . . . .	173
	Maintaining Sections . . . . .	174
	Maintaining Parameters . . . . .	178
	Commenting the citect.ini file . . . . .	186
	Analyzing the citect.ini file . . . . .	189
	Using the Comparison Wizard . . . . .	190
	Saving changes to the citect.ini file . . . . .	194
<b>Chapter 14</b>	<b>Tagging Process Variables</b>	
	Tag Naming . . . . .	197

Configuring Local Variables .....	201
Configuring Variable Tags .....	203

## Chapter 15      Linking, Importing, and Exporting Tags

Linking tags .....	215
Importing tags .....	218
Exporting tags .....	224
Unity Support Matrixes .....	239

## Chapter 16      Securing CitectSCADA Projects

Overview .....	243
Securing a Top-level Project .....	244
Securing an Include Project .....	245
Making a Project Read-Only .....	247
Using CitectSCADA with Windows Security .....	250
Read-Only Privileges on Projects .....	251
Securing Runtime Computers .....	255

## Chapter 17      Defining and Drawing Graphics Pages

Creating a New Graphics Page .....	259
Using Page Templates .....	265
Using a Browse Sequence .....	267
Specifying a Startup Page .....	268
Sizing the Page .....	269
Defining Page Properties .....	270
Setting Default Page Settings .....	277
Understanding the Drawing Environment .....	279

## Chapter 18      Using Objects

Using groups .....	300
Reshaping objects .....	300
Using bitmaps .....	301
Importing graphics .....	301
Object Properties .....	301
Manipulating Objects .....	309

## Chapter 19      Understanding Object Types

Using Free Hand Line Objects .....	321
Using Straight Line Objects .....	323
Using Rectangle Objects .....	325
Using Ellipse Objects .....	328

Using Polygon Objects .....	334
Using Pipe Objects .....	338
Using Text Objects .....	340
Using Number Objects .....	349
Using Button Objects .....	350
Using Symbol Set Objects .....	352
Using Trend Objects .....	359
Using Cicode Objects .....	362
Using Pasted Symbol Objects .....	364
Using ActiveX Objects .....	367
Using Database Exchange Control Objects .....	372

## Chapter 20

### Defining Common Object Properties

3D Effects .....	373
Visibility .....	377
Movement .....	378
Scaling .....	386
Fill Color .....	393
Fill Level .....	404
Touch Commands .....	408
Keyboard Commands .....	411
Sliders .....	414
Access .....	419

## Chapter 21

### Defining Commands and Controls

Defining Commands and Controls .....	425
System Keyboard Commands .....	426
Keyboard Keys .....	428
Keyboards .....	429
Defining Key Sequences for Commands .....	430

## Chapter 22

### Configuring and Processing Alarms

Configured alarms .....	435
Using alarm delay .....	436
Using custom alarm filters .....	436
Alarm Categories .....	439
Digital Alarms .....	443
Multi-digital Alarms .....	447
Time-stamped Alarms .....	453
Analog Alarms .....	457
Advanced Alarms .....	463
Time-stamped Digital Alarms .....	466
Time-stamped Analog Alarms .....	470

	Formatting an Alarm Display .....	476
	Using Alarm Properties as Tags .....	482
	Handling Alarms at Runtime .....	485
	Using CitectSCADA Fonts .....	487
	Configuring Custom Color Fonts .....	490
<b>Chapter 23</b>	<b>Configuring Events</b>	
	Events Properties .....	493
	Running Events .....	494
<b>Chapter 24</b>	<b>Using Accumulators</b>	
<b>Chapter 25</b>	<b>Logging and Trending Data</b>	
	Trending Data .....	501
	Trend Graphs .....	508
	Printing Trend Data .....	510
	Exporting Trend Data .....	511
	Using Trend History Files .....	511
	Using Path Substitution .....	514
	Debugging Trending .....	515
<b>Chapter 26</b>	<b>Understanding Statistical Process Control</b>	
	Process Variation .....	517
	Statistical Control .....	518
	Process Capability .....	519
	XRS Control Charts .....	519
	Capability Charts .....	521
	Pareto Charts .....	521
	Using Statistical Process Control (SPC) with CitectSCADA .....	522
	SPC Tags .....	522
	SPC Control Charts .....	528
	SPC Alarms .....	530
	SPC Formulas and Constants .....	531
	Control Chart Line Constants .....	537
<b>Chapter 27</b>	<b>Reporting Information</b>	
	Configuring reports .....	539
	Running Reports .....	542
	Report Format File .....	544
	Handling Communication Errors in Reports .....	547

<b>Chapter 28</b>	<b>Using Security</b>	
	Maintaining User Records .....	551
	Defining User Privileges .....	554
	Defining Areas .....	556
<b>Chapter 29</b>	<b>Using Labels</b>	
<b>Chapter 30</b>	<b>Using Devices</b>	
	Using groups of devices .....	573
	Using devices to read data .....	573
	Configuring Devices .....	574
	Formatting Data in the Device .....	578
	Using Device History Files .....	583
	Using Command Fields .....	586
	About Print Management .....	586
<b>Chapter 31</b>	<b>Exchanging Data with Other Applications</b>	
	Using DDE (Dynamic Data Exchange) .....	589
	Network DDE .....	598
	DDE Shares .....	601
	Using DDE Trusted Shares .....	603
	Using the Citect Tags Excel macros .....	605
	Using External Databases .....	606
	Using Structured Query Language .....	607
	Using ODBC drivers .....	610
<b>Chapter 32</b>	<b>Using Genies and Super Genies</b>	
	Understanding Genies .....	624
	Using Super Genies .....	630
	Using Structured Tag Names with Genies and Super Genies .....	639
	Hiding Graphics Objects .....	641
<b>Chapter 33</b>	<b>Working with Multi-Language Projects</b>	
	Changing Languages .....	643
<b>Chapter 34</b>	<b>Using OPC Server DA2.0</b>	
	OPC Overview .....	649
	CitectSCADA OPC Server .....	650
	CitectSCADA OPC Server Installation .....	651
	Configuring Remote Access to the CitectSCADA OPC Server .....	652

Troubleshooting .....	682
-----------------------	-----

## Chapter 35

### Communicating with I/O Devices

Communication types .....	683
How CitectSCADA Communicates with I/O Devices .....	683
Performance Considerations .....	688
Serial Communications .....	695
Using Proprietary Boards .....	700
Serial Port Loop-Back Test .....	701
Setting Up Communications .....	703
Manually Configuring Communications .....	703
Wiring Cables .....	713
Common Serial Communication Standards .....	715
Using a Disk I/O Device .....	718
Citect Driver Accreditation .....	722
Advanced Driver Information .....	722
Scheduled Communications .....	728
Communicating with Diallable Remote I/O Devices .....	731

## Chapter 36

### Using the Communications Express Wizard

Express Communications Wizard - introduction .....	745
Express Communications Wizard - Server selection .....	746
Express Communications Wizard - Device selection .....	746
Express Communications Wizard - I/O device type .....	746
Express Communications Wizard - I/O device communications selection .....	746
Express Communications Wizard - TCP/IP address .....	747
Express Communications Wizard - I/O device address .....	747
Express Communications Wizard - I/O device connection schedule .....	747
Express Communications Wizard - Link to external database .....	749
Express Communications Wizard - Serial device .....	751
Express Communications Wizard - Summary .....	751

## Chapter 37

### Building Your CitectSCADA Project

Compiling the Project .....	753
Running the System .....	763
Running Your System Over the Internet .....	764
Using an Alternative INI File .....	768
Debugging the Runtime System .....	769
Server Redirection .....	770
Debugging I/O Devices and Protocols .....	771
Restarting the System Online .....	779
CitectSCADA Software Protection .....	783
Using the CitectSCADA Kernel .....	785

Gathering Runtime Information .....	812
-------------------------------------	-----

## Chapter 38      CitectSCADA Runtime Manager

Launching Runtime Manager .....	815
CitectSCADA Runtime Manager Interface .....	817
Hiding CitectSCADA Runtime Manager .....	817
Shutting Down CitectSCADA Runtime Manager .....	818
Monitoring Startup .....	818
Monitoring Runtime Processes .....	819
Starting a Process .....	819
Stopping a Process .....	819
Restarting a Process .....	820
Starting all Processes .....	820
Cancelling a Process .....	820
Viewing the Kernel .....	820
Troubleshooting .....	820

## Part III      Using the Web Client

### Chapter 39      CitectSCADA Web Client

Introduction .....	825
Getting Started .....	827
Choosing which Web Server platform to use - IIS or Tomcat? .....	829
Preparing a Project for Deployment .....	831
Configuring a deployment .....	835
Implementing Multiple Language Support .....	843
Web Client Upgrade Issues .....	847
Frequently Asked Questions .....	847

## Part IV      Using the CSV\_Include Project

### Chapter 40      Introducing CSV\_Include

Where to Find Information .....	855
---------------------------------	-----

### Chapter 41      Using Pages and Templates

Normal Page Template .....	858
Alarm Page Templates .....	858
Trend Page Templates .....	859
File Page Templates .....	863
Admin Tools Page Template .....	864

Common Toolbars .....	865
Navigation Toolbar .....	866
Alarms Toolbar .....	868

## Chapter 42

### Creating a New Project

Creating a Privileged User .....	869
Running the Computer Setup Wizard .....	870
Setting Up Instant Trending .....	870
Displaying a Project on Multiple Monitors .....	871
Implementing Audible Alarms .....	872
Creating Pages .....	872
Creating Custom Menus .....	875
Creating an Alarms Group .....	878
Creating a Trends Group .....	880
Using Environment Variables .....	881

### Frequently Asked Questions

Pages .....	883
Graphics .....	884
Runtime .....	884
Trends .....	885
Controls .....	886
Alarms .....	887
Miscellaneous .....	888



# Getting Started

This section contains information on CitectSCADA version 7 and describes the following:

[Getting Technical Support](#)

[Upgrading to CitectSCADA v7.0](#)

[About CitectSCADA](#)

[CitectSCADA Tools](#)

[Components of a project](#)

[Typical system scenarios](#)



# Chapter 1: Getting Technical Support

---

Citect provides various support options to help you get the most from this product.

- If you have questions about using CitectSCADA, consult the extensive online Help to answer your questions. You can use the **Contents** list to find the section you're interested in, enter an item into the **Index**, or enter an item using the **Search** tab.
- If you seek more technical information than is provided in the online Help, check the Citect [knowledge base](#).
- If you cannot find the information you need, you can obtain [technical support](#) and [Training](#). Consulting services are also available upon request.

See Also [Technical Support](#)  
[Contact information](#)

## Technical Support

Please note the following about technical support for your CitectSCADA product.

### Citect Support

Citect Maintenance and Support Agreements are available to purchase to protect your CitectSCADA software investment. To purchase a Maintenance and Support Agreement, the price of which is determined by the list price of your CitectSCADA system, you must contact your local Citect representative.

Citect offers a range of Support services; for further information visit <http://www.citect.com/support> or ask your Citect representative. Upon receipt of your order for a Maintenance and Support Agreement, you will receive a site number and a list of all dongle serial numbers, which you must quote when you contact Citect Support. A site or dongle serial number enables the Citect Customer Support team to provide quality service product by logging and tracking your Customer Service Requests (CSRs).

### Before Calling Customer Support

Fill out the online CitectSCADA & CitectHMI Support Request form, found at the Citect web site <http://www.citect.com>.

If you are unsure of which contact details to use, email: [support@citect.com](mailto:support@citect.com).

**Training** Various training facilities for CitectSCADA are also available. Contact your local CitectSCADA distributor for more information.

---

See Also [Contact information](#)

Contact information

**For contact information in your region, consult the support web site at:**

**<http://www.citect.com/support/contact-support.html>**

## Chapter 2: What's New in CitectSCADA v7.0

---

CitectSCADA v7.0 incorporates the following new features:

- [The Migration Tool](#)
- [Clustering](#)
- [Local Variables](#)
- [Publish Alarm Property](#)
- [Memory Mode](#)
- [Client-side Online Changes](#)
- [Publisher-Subscriber Model](#)
- [Dual Network Support](#)
- [Project-Based Network Configuration](#)

For changes to:

- Citect.ini parameters, see [Citect.ini Parameters in Version 7.0](#).
- Cicode functions, see [Cicode Functions in Version 7.0](#).
- CtAPI functions, see [CtAPI Functions in Version 7.0](#).
- Kernel commands, see [Kernel Commands in Version 7.0](#).

For details on how to configure an existing project to run in version 7.0, refer to [Upgrading to CitectSCADA v7.0](#).

### The Migration Tool

The Migration Tool is a separate application which should be manually run after the automatic upgrade has been executed, and initiated by you after you have prepared the project for final migration. This tool will accommodate the critical changes in project functionality which are incorporated in version 7.0.

See Also [Migration Tool](#)

### Clustering

Clustering allows you to group different sets of the runtime components within a single project, allowing multiple independent systems to be monitored and controlled.

There are countless variations in how a CitectSCADA clustered system can be configured. The most appropriate configuration will depend on the requirements for the solution to be deployed and the environment in which it is being deployed. For more information see [Typical system scenarios](#).

See Also [Included projects](#)  
[Implementing Clustering](#)

## Local Variables

Local variables allow you to store data in memory when you start your runtime system. They are created each time the system starts, and therefore do not retain their values when you shut down.

Local variables are useful when you need each process to have a separate copy of the data. Each process has its own copy of each local variable configured in the project, and the values in a local variable are available only to the process that wrote them.

See Also [Configuring Local Variables](#)

## Publish Alarm Property

Alarm devices were defined as devices with their Protocol field set to "Alarm". The function of these devices are now configured on an Alarm Server by setting the "Publish Alarm Properties" property to True.

See Also [Alarm Server Definitions](#)

## Memory Mode

I/O devices can now be configured to run in memory mode:

### ■ Memory Mode

An I/O device running in memory mode is created in memory and its values stored in memory at runtime.

Devices using memory mode are not connected to any hardware, and write their values to a cache. Memory mode is useful when you are configuring a system for the first time, as you can design and test your system before connecting a physical I/O device.

**Note:** Memory mode replaces Memory I/O devices, which are no longer supported. Devices configured in CitectSCADA as Memory I/O devices will be converted to local variables during the upgrade to version 7.0

See Also [Using Memory Mode](#)

---

## Client-side Online Changes

The following live changes can now be made to the project without restarting clients:

- IO Devices (restart the IO server)
- Tags (restart the IO server)
- Alarms (restart the Alarm server)
- Trends (restart the Trend server)
- Reports (restart the Report server)
- Accumulators (restart the Report server)

Clients only require that graphics, code and communications configurations are deployed to them. Other configuration information is deployed to the appropriate server.

Project changes are still deployed as before

- Manually
- Run/Backup Run/Copy
- FTP (IDC)
- HTTP (Web Client)

## Publisher-Subscriber Model

CitectSCADA now uses a Publisher-Subscriber data acquisition model. Client computers subscribe to configured tags and receive notification when the tag values change. Cicode functions can also be triggered by the change of a tag, removing the need to poll, and improving the efficiency of the CitectSCADA system.

See Also

[TagSubscribe](#)  
[TagUnsubscribe](#)

## Dual Network Support

Previous CitectSCADA versions have been able to support redundant networks via NetBIOS. From version 7.0 users can only use TCP/IP for network configuration and can specify multiple IP addresses for each server, providing native support for network redundancy.

## Project-Based Network Configuration

In version 7.0, the project topology is embedded in the project, and network configuration can be performed from within the Project Editor. Servers and their IP addresses are set up in the **Network Addresses** dialog in the Project Editor.

This means that physical computers in the system can easily be changed. As long as the IP address or computer name of the new machine is the same as the one being replaced, the new computer will be able to immediately take the same role.

## Citect.ini Parameters in Version 7.0

The following sections detail the changes made to Citect.ini parameters in CitectSCADA v7.0:

- [New Parameters](#)
- [Obsolete Parameters](#)

### New Parameters

The following parameters are new in version 7.0:

#### Alarm Parameters:

<a href="#">[Alarm.ClusterName.ServerName]Clusters</a>	Sets which clusters this Alarm Server process connects to at startup
<a href="#">[Alarm.ClusterName.ServerName]CPU</a>	Sets the CPU that the Alarm Server process is assigned to
<a href="#">[Alarm.ClusterName.ServerName]Events</a>	The list of events that this CitectSCADA Alarms Server process enables
<a href="#">[Alarm.ClusterName.ServerName]ShutdownCode</a>	Determines the Cicode function to run when CitectSCADA Alarm Server process shuts down
<a href="#">[Alarm.ClusterName.ServerName]StartupCode</a>	Determines the Cicode function to run when CitectSCADA Alarm Server process starts up

#### Backup Parameters:

<a href="#">[Backup]SaveiniFiles</a>	Determines whether the "Save ini files" checkbox is checked by default during Backup.
--------------------------------------	---

#### Client Parameters:

<a href="#">[Client]Clusters</a>	Selects which clusters the client is connected to at startup.
<a href="#">[Client]ComputerRole</a>	Specifies the role of the computer
<a href="#">[Client]Events</a>	Sets the events to be enabled on the Display Client.
<a href="#">[Client]ForceClient</a>	Starts only the client process, and connects to a configured Citect servers using a network connection.
<a href="#">[Client]FullLicense</a>	Specifies that a display client will use a full license



[\[Client\]ShutdownCode](#)

Determines the Cicode function to run when CitectSCADA DisplayClient component shuts down.

[\[Client\]StartupCode](#)

Determines the Cicode function to run when CitectSCADA DisplayClient component starts up.

[\[Client\]WaitForConnectAtStartup](#)

Specifies that connection to a server will wait until it can establish a connection to the server processes before starting up.

#### **CtCicode Parameters:**

[\[CtCicode\]FastFormat](#)

Controls whether fast formatting is used in the Cicode Editor

#### **Dial Parameters:**

[\[Dial\]MissedScheduleTolerance](#)

Specified how many consecutive scheduled dial attempts can be missed before the cache becomes stale

#### **General Parameters:**

[\[General\]Multiprocess](#)

Determines whether CitectSCADA runs as a multi-process or single-process application.

#### **IOServer Parameters:**

[\[IOServer.ClusterName.ServerName\]Clusters](#)

Sets the clusters that the IO Server process will connect to at startup

[\[IOServer.ClusterName.ServerName\]CPU](#)

Sets the CPU that the IO Server process is assigned to

[\[IOServer.ClusterName.ServerName\]Events](#)

The list of events that this CitectSCADA IO Server process enables

[\[IOServer.ClusterName.ServerName\]ShutdownCode](#)

Determines the Cicode function to run when CitectSCADA IO Server process shuts down

[\[IOServer.ClusterName.ServerName\]StartupCode](#)

Determines the Cicode function to run when CitectSCADA IO Server process starts up

#### **Report Parameters:**

[\[Report.ClusterName.ServerName\]Clusters](#)

Sets the clusters this Reports Server process will connect to at startup

[\[Report.ClusterName.ServerName\]CPU](#)

Sets the CPU that this Report Server process is assigned to

[\[Report.ClusterName.ServerName\]Events](#)

The list of events that this CitectSCADA Report Server process enables

[\[Report.ClusterName.ServerName\]ShutdownCode](#)

Determines the Cicode function to run when this CitectSCADA Reports Server process shuts down

[\[Report.ClusterName.ServerName\]StartupCode](#)

Determines the Cicode function to run when this CitectSCADA Reports Server process starts up

**Trend Parameters:**

<a href="#">[Trend.ClusterName.ServerName]Clusters</a>	Sets the clusters this Trends Server process will connect to at startup
<a href="#">[Trend.ClusterName.ServerName]CPU</a>	Sets the CPU that the Trend Server process is assigned to
<a href="#">[Trend.ClusterName.ServerName]Events</a>	The list of events that this CitectSCADA Trend Server process enables
<a href="#">[Trend.ClusterName.ServerName]ShutdownCode</a>	Determines the Cicode function to run when CitectSCADA Trend Server process shuts down
<a href="#">[Trend.ClusterName.ServerName]StartupCode</a>	Determines the Cicode function to run when CitectSCADA Trend Server process starts up

**Obsolete Parameters**

The following parameters are no longer supported in version 7.0:

**Alarm Parameters:**

[Alarm]CPU	Sets the CPU that the Alarm Server component is assigned to
[Alarm]Primary	Determines if this Alarms Server is the Primary Alarms Server
[Alarm]Process	Sets the CitectSCADA process the Alarm Server component is assigned to
[Alarm]Server	Determines whether this computer is an Alarms Server

**Client Parameters:**

[Client]Display	Sets the CitectSCADA computer as a Display Client
[Client]Manager	Sets the CitectSCADA computer as a Manager Client
[Client]Primary	The name of the primary CitectSCADA server
[Client]Process	Sets the CitectSCADA process the Display Client component is assigned to
[Client]Standby	The name of the standby CitectSCADA server

**Code Parameters:**

[Code]AlarmShutdown	Determines the Cicode function to run when CitectSCADA Alarm server component shuts down
[Code]AlarmStartup	Determines the Cicode function to run when CitectSCADA Alarm server component starts up
[Code]AutoReRead	Controls whether the ReRead() function is automatically called
[Code]IOServerShutdown	Determines the Cicode function to run when CitectSCADA I/O server component shuts down
[Code]IOServerStartup	Determines the Cicode function to run when CitectSCADA I/O server component starts up
[Code]ReportShutdown	Determines the Cicode function to run when CitectSCADA Report server component shuts down
[Code]ReportStartup	Determines the Cicode function to run when CitectSCADA Report server component starts up
[Code]Shutdown	Determines the Cicode function to run when CitectSCADA Display client component shuts down

[Code]Startup	Determines the Cicode function to run when CitectSCADA Display client component starts up
[Code]TrendShutdown	Determines the Cicode function to run when CitectSCADA Trend server component shuts down
[Code]TrendStartup	Determines the Cicode function to run when CitectSCADA Trend server component starts up

#### DNS Parameters:

[DNS]<Server name>	Determines the IP address (or fully qualified host name) of the primary I/O Server
--------------------	--

#### Event Parameters:

[Event]Alarm	The classes of events to be enabled by the Alarm server
[Event]IOServer	The classes of events to be enabled by the I/O server
[Event]Name	The classes of events to be enabled
[Event]Report	The classes of events to be enabled by the Report server
[Event]Trend	The classes of events to be enabled by the Trend server

#### General Parameters:

[General]BadOptimise	Determines whether certain strings are replaced with labels on compile
[General]CitectRunningCheck	checks if a project is currently running on the local machine when a compile is triggered

#### IOServer Parameters:

[IOServer]BlockWrites	Determines whether CitectSCADA will try to block optimize writes to I/O devices
[IOServer]CPU	Sets the CPU that the IO Server component is assigned to
[IOServer]Name	The name of the default I/O server
[IOServer]Process	Sets the CitectSCADA process the I/O Server component is assigned to
[IOServer]SaveBackup	This parameter has been superseded by SaveNetwork
[IOServer]Server	Determines whether this computer is an I/O Server

#### LAN Parameters:

[LAN]Bridge	Determines the bridge level
[LAN]CancelOnClose	For users with Novell NetBIOS emulator problems
[LAN]Disable	Enables/disables CitectSCADA from the LAN
[LAN]GroupName	Determines whether CitectSCADA uses the group name 'CITECT STATION50' or the computer name specified by the [Lan]Node parameter.
[LAN]KillPiggyBackAck	Controls whether CitectSCADA will try to optimize network protocols which support piggy back ack
[LAN]LanA	Defines the protocol stack that CitectSCADA uses for NetBIOS communication
[LAN]NetBIOS	Enables/disables NetBIOS
[LAN]NetTrace	Determines whether the NetBIOS window is enabled on startup

[LAN]NetTraceBuff	Sets the number of trace buffers.
[LAN]NetTraceErr	Enables the error mode of the NetBIOS window
[LAN]NetTraceLog	Enables the logging mode of the NetBIOS window
[LAN]Poll	Puts CitectSCADA LAN communications into polled mode
[LAN]RemoteTimeOut	The timeout period for remote I/O device write requests from a Display Client to the I/O Server
[LAN]Retry	The number of times to retry establishing communications after a timeout - before an error message is generated
[LAN]SendTimeOut	The timeout to send a network packet across the network
[LAN]SesRecBuf	The number of receive NetBIOS Control Blocks (NCBs) that CitectSCADA uses for all sessions
[LAN]SesSendBuf	The number of send NetBIOS control blocks that CitectSCADA uses for all sessions
[LAN]TimeOut	The timeout to send a network packet across the network

#### Proxi Parameters:

[Proxi]<I/O Server name>	Defines a list of proxi server associations
--------------------------	---

#### Report Parameters:

[Report]Primary	Determines if this Reports Server is the Primary Reports Server
[Report]Process	Sets the CitectSCADA process the Report Server component is assigned to
[Report]Server	Determines whether this computer is a Reports Server

#### Server Parameters:

[Server]Name	The name of the CitectSCADA server
--------------	------------------------------------

#### Trend Parameters:

[Trend]BlockByIODevice	Ensures that I/O problems causing gaps for a trend tag do not cause gaps for all trend tags
[Trend]CPU	Sets the CPU that the Trend Server component is assigned to
[Trend]Process	Sets the CitectSCADA process the Trend Server component is assigned to
[Trend]Redundancy	Enables/disables trend redundancy action
[Trend]Server	Determines whether this computer is a Trends Server
[Trend]StaggerRequestSubgroups	Reduces network traffic by spacing out trend sample requests

## Cicode Functions in Version 7.0

The following sections detail the changes made to Cicode functions in CitectSCADA v7.0:

- [New Functions](#)
- [Obsolete Functions](#)

## New Functions

### ■ Modified Functions

The following functions are new in version 7.0:

#### **Miscellaneous Functions**

<a href="#"><u>AccControl</u></a>	Controls accumulators e.g. motor run hours.
<a href="#"><u>AccumBrowseClose</u></a>	Closes an accumulator browse session.
<a href="#"><u>AccumBrowseFirst</u></a>	Gets the oldest accumulator entry.
<a href="#"><u>AccumBrowseGetField</u></a>	Gets the field indicated by the cursor position in the browse session.
<a href="#"><u>AccumBrowseNext</u></a>	Gets the next accumulator entry in the browse session.
<a href="#"><u>AccumBrowseNumRecords</u></a>	Returns the number of records in the current browse session.
<a href="#"><u>AccumBrowseOpen</u></a>	Opens an accumulator browse session.
<a href="#"><u>AccumBrowsePrev</u></a>	Gets the previous accumulator entry in the browse session.
<a href="#"><u>ProcessIsClient</u></a>	Determines if the currently executing process contains a Client component
<a href="#"><u>ProcessIsServer</u></a>	Determines if the currently executing process contains a particular server component.
<a href="#"><u>ServiceGetList</u></a>	Gets information about services running in the component calling this function.

#### **Alarm Functions:**

<a href="#"><u>AlarmDsplLast</u></a>	Displays the most recent, unacknowledged alarms.
<a href="#"><u>AlmSummaryAck</u></a>	Acknowledges the alarm at the current cursor position in an active data browse session.
<a href="#"><u>AlmSummaryClear</u></a>	Clears the alarm at the current cursor position in an active data browse session.
<a href="#"><u>AlmSummaryClose</u></a>	Closes an alarm summary browse session.
<a href="#"><u>AlmSummaryCommit</u></a>	Commits the alarm summary record to the alarm summary device.
<a href="#"><u>AlmSummaryDelete</u></a>	Deletes alarm summary entries from the browse session.
<a href="#"><u>AlmSummaryDeleteAll</u></a>	Deletes all alarm summary entries from the browse session.
<a href="#"><u>AlmSummaryDisable</u></a>	Disables the alarm at the current cursor position in an active data browse session.
<a href="#"><u>AlmSummaryEnable</u></a>	Enables the alarm at the current cursor position in an active data browse session.
<a href="#"><u>AlmSummaryFirst</u></a>	Gets the oldest alarm summary entry.
<a href="#"><u>AlmSummaryGetField</u></a>	Gets the field indicated by the cursor position in the browse session.
<a href="#"><u>AlmSummaryNext</u></a>	Gets the next alarm summary entry in the browse session.
<a href="#"><u>AlmSummaryOpen</u></a>	Opens an alarm summary browse session.
<a href="#"><u>AlmSummaryPrev</u></a>	Gets the previous alarm summary entry in the browse session.
<a href="#"><u>AlmSummarySetFieldValue</u></a>	Sets the value of the field indicated by the cursor position in the browse session.
<a href="#"><u>AlmTagsAck</u></a>	Acknowledges the alarm tag at the current cursor position in an active data browse session.

<a href="#"><u>AlmTagsClear</u></a>	Clears the alarm tag at the current cursor position in an active data browse session.
<a href="#"><u>AlmTagsDisable</u></a>	Disables the alarm tag at the current cursor position in an active data browse session.
<a href="#"><u>AlmTagsEnable</u></a>	Enables the alarm tag at the current cursor position in an active data browse session.
<a href="#"><u>AlmTagsFirst</u></a>	Gets the oldest alarm tags entry.
<a href="#"><u>AlmTagsGetField</u></a>	Gets the field indicated by the cursor position in the browse session.
<a href="#"><u>AlmTagsNext</u></a>	Gets the next alarm tags entry in the browse session.
<a href="#"><u>AlmTagsNumRecords</u></a>	Returns the number of records in the current browse session.
<a href="#"><u>AlmTagsOpen</u></a>	Opens an alarm tags browse session.
<a href="#"><u>AlmTagsPrev</u></a>	Gets the previous alarm tags entry in the browse session.

### Super Genie Functions

<a href="#"><u>AssGetProperty</u></a>	Gets association information about the current Super Genie from the datasource
<a href="#"><u>AssGetScale</u></a>	Gets scale information about the associations of the current Super Genie from the datasource
<a href="#"><u>AssInfoEx</u></a>	Replaces the AssInfo function and supports online changes.

### Cluster Functions

<a href="#"><u>ClusterActivate</u></a>	Allows the user to activate an inactive cluster.
<a href="#"><u>ClusterDeactivate</u></a>	Allows the user to deactivate an active cluster.
<a href="#"><u>ClusterFirst</u></a>	Allows the user to retrieve the first configured cluster in the project.
<a href="#"><u>ClusterIsActive</u></a>	Allows the user to determine if a cluster is active.
<a href="#"><u>ClusterNext</u></a>	Allows the user to retrieve the next configured cluster in the project.
<a href="#"><u>ClusterServerTypes</u></a>	Allows the user to determine which servers are defined for a given cluster.
<a href="#"><u>ClusterStatus</u></a>	Allows the user to determine the connection status from the client to a server on a cluster.
<a href="#"><u>ClusterSwapActive</u></a>	Allows the user to deactivate an active cluster at the same time as activating a deactivate cluster.

### I/O Device Functions

<a href="#"><u>SubscriptionAddCallback</u></a>	Adds a callback function to a tag subscription.
<a href="#"><u>SubscriptionGetAttribute</u></a>	Reads an attribute value of a tag subscription.
<a href="#"><u>SubscriptionRemoveCallback</u></a>	Removes a callback function from a tag subscription
<a href="#"><u>TagGetProperty</u></a>	Gets a property for a variable tag from the datasource.
<a href="#"><u>TagGetScale</u></a>	Gets the value of a tag at a specified scale from the datasource
<a href="#"><u>TagSubscribe</u></a>	Subscribes a tag for periodic monitoring and event handling.
<a href="#"><u>TagUnsubscribe</u></a>	Unsubscribes a tag for periodic monitoring and event handling.

### Tag Functions

<a href="#"><u>TagInfoEx</u></a>	Replaces the TagInfo function and supports online changes.
----------------------------------	--

[TagWriteEventQue](#)

Opens the tag write event queue.

### Task Functions

[TaskCluster](#)

Gets the name of the cluster context in which the current task is executing

### Trend Functions

[TrnBrowseClose](#)

Closes a trend browse session.

[TrnBrowseFirst](#)

Gets the oldest trend entry.

[TrnBrowseGetField](#)

Gets the field indicated by the cursor position in the browse session.

[TrnBrowseNext](#)

Gets the next trend entry in the browse session.

[TrnBrowseNumRecords](#)

Returns the number of records in the current browse session.

[TrnBrowseOpen](#)

Opens a trend browse session.

[TrnBrowsePrev](#)

Gets the previous trend entry in the browse session.

[TrnGetCluster](#)

Gets the name of the cluster the trend graph is associated with.

[TrnGetPenComment](#)

Gets the comment of a trend pen.

### Report Functions

[RepGetCluster](#)

Retrieves the name of the cluster the report is running on.

## Obsolete Functions

The following functions are not supported from version 7.0. If your project uses any of these functions, an error will be raised during compilation.

### Cluster Functions

[ClusterGetName](#)

Returns the names of the primary and standby cluster servers.

[ClusterSetName](#)

Connects to a specific cluster server.

### Display Functions

[DspCol](#)

Displays a colour at an AN.

## Task Functions

### [ReRead](#)

Causes CitectSCADA to re-read the I/O Device data associated with the current Cicode task.

Tags are now subscribed at the start of a function and updated tag values are sent to the subscribing function. Tag subscriptions are made at the update rate of:

- the graphics page if called from a page
  - the default subscription rate as determined by [\[Code\]TimeData](#) if called from Cicode (default 250ms)
  - the update rate requested of a task created using [TaskNewEx](#)
  - the update rate requested of a subscription created using [TagSubscribe](#)
- You should ensure that the subscription update rate matches the requirements of your function.

After removing ReRead from looping code you may need to extend the period of the [Sleep](#) function. This is to replace the pause ReRead created while it read all the tag values.

## Modified Functions

The following functions have been modified in version 7.0:

### Alarm Functions

#### [AlarmAck](#)

Acknowledges alarms.

#### [AlarmAckRec](#)

Acknowledges alarms by record number

#### [AlarmActive](#)

Determines if any alarms are active in the user's area.

#### [AlarmClear](#)

Clears acknowledged, inactive alarms from the active alarm list.

#### [AlarmClearRec](#)

Clear an alarm by its record number

#### [AlarmDelete](#)

Deletes alarm summary entries.

#### [AlarmDisable](#)

Disables alarms.

#### [AlarmDisableRec](#)

Disables alarms by record number

#### [AlarmDsp](#)

Displays alarms.

#### [AlarmDsplLast](#)

Displays the most recent, unacknowledged alarms.

#### [AlarmEnable](#)

Enables alarms.

#### [AlarmEnableRec](#)

Enables alarms by record number

#### [AlarmFirstTagRec](#)

Searches for the first occurrence of an alarm tag, name, and description

#### [AlarmGetDelayRec](#)

Gets the delay setting for an alarm via the alarm record number

#### [AlarmGetFieldRec](#)

Gets alarm field data from the alarm record number

#### [AlarmGetThresholdRec](#)

Gets the thresholds of analog alarms by the alarm record number

#### [AlarmNextTagRec](#)

Searches for the next occurrence of an alarm tag, name, and description.

#### [AlarmNotifyVarChange](#)

Activates a time-stamped digital or time-stamped analog alarm

#### [AlarmSumAppend](#)

Appends a new blank record to the alarm summary.

#### [AlarmSumCommit](#)

Commits the alarm summary record to the alarm summary device.

#### [AlarmSumDelete](#)

Deletes alarm summary entries.

#### [AlarmSumFind](#)

Finds an alarm summary index for an alarm record and alarm on time.



<a href="#">AlarmSumFirst</a>	Gets the oldest alarm summary entry.
<a href="#">AlarmSumGet</a>	Gets field information from an alarm summary entry.
<a href="#">AlarmSumLast</a>	Gets the most recent alarm summary entry.
<a href="#">AlarmSumNext</a>	Gets the next alarm summary entry.
<a href="#">AlarmSumPrev</a>	Gets the previous alarm summary entry.
<a href="#">AlarmSumSet</a>	Sets field information in an alarm summary entry.
<a href="#">AlarmSumSplit</a>	Duplicates an alarm summary entry.
<a href="#">AlarmSumType</a>	Retrieves a value that indicates a specified alarm's type.

### I/O Device Functions

<a href="#">DriverInfo</a>	Provides information about the driver for a particular I/O Device.
<a href="#">IODeviceControl</a>	Provides control of individual I/O Devices.
<a href="#">IODeviceInfo</a>	Gets information on an I/O Device.

### Miscellaneous Functions

<a href="#">AccControl</a>	Controls accumulators e.g. motor run hours.
<a href="#">ServerInfo</a>	Gets client and server information.
<a href="#">ServerInfoEx</a>	Gets client and server information from a specified process in a multiprocessor environment.
<a href="#">Shutdown</a>	Ends CitectSCADA's operation.

### Report Functions

<a href="#">RepGetControl</a>	Gets report control information.
<a href="#">Report</a>	Runs a report.
<a href="#">RepSetControl</a>	Sets report control information.

### SPC Functions

<a href="#">SPCAlarms</a>	Returns the status of the specified SPC alarm.
<a href="#">SPCProcessXRSGet</a>	Gets the process mean, range and standard deviation overrides.
<a href="#">SPCProcessXRSSet</a>	Sets the process mean, range and standard deviation overrides.
<a href="#">SPCSpecLimitGet</a>	Gets the specification limits (USL and LSL) for the specified tag.
<a href="#">SPCSpecLimitSet</a>	Sets the specification limits (USL and LSL) for the specified tag.
<a href="#">SPCSubgroupSizeGet</a>	Gets the size of a subgroup for the specified SPC tag.
<a href="#">SPCSubgroupSizeSet</a>	Sets the subgroup size for the specified SPC tag.

### Super Genie Functions

<a href="#">Ass</a>	Associates a variable tag with a Super Genie.
<a href="#">AssPage</a>	Associates up to eight variable tags with a Super Genie and displays the Super Genie in the current window.
<a href="#">AssPopUp</a>	Associates up to eight variable tags with a Super Genie and displays the Super Genie in a popup window.

[AssTag](#)

Associates a variable tag with the current Super Genie. The association will be created for the current Super Genie only, and will only come into effect after you re-display the Super Genie.

[AssVarTags](#)

Associates up to eight variable tags with a Super Genie. This association is only made for the next Super Genie you display (either in the current window or in a new window). You can use this function repeatedly to associate more than 8 variable tags to a Super Genie.

[AssWin](#)

Associates up to eight variable tags with a Super Genie, and displays the Super Genie in a new window.

**Tag Functions**[TagGetProperty](#)

This function reads a property of a variable tag from the datasource

[TagGetScale](#)

Gets the value of a tag at a specified scale from the datasource

[TagRamp](#)

This function will increment a Tag by the amount defined by iPercentInc

[TagRead](#)

Reads a variable from the I/O device

[TagScaleStr](#)

Gets the value of a tag at a specified scale

[TagWrite](#)

Writes to an I/O device variable by specifying the variable tag.

**Task Functions**[MsgOpen](#)

Opens a message session with a CitectSCADA server or client.

**Trend Functions**[TrendDspCursorTag](#)

Displays the tag name of the current pen.

[TrnAddHistory](#)

Restores an old history file to the trend system.

[TrnDelHistory](#)

Deletes an old history file from the trend system.

[TrnEventSetTable](#)

Sets trend data from a table, for a specified trend tag.

[TrnEventSetTableMS](#)

Sets event trend data and time data (including milliseconds) for a specified trend tag.

[TrnFlush](#)

Flushes the trend to disk.

[TrnGetDefScale](#)

Gets the default engineering zero and full scales of a trend tag.

[TrnGetPen](#)

Gets the trend tag of a pen.

[TrnGetTable](#)

Stores trend data in an array.

[TrnInfo](#)

Gets the configured values of a trend tag.

[TrnNew](#)

Creates a new trend at run time.

[TrnSelect](#)

Sets up a page for a trend.

**Window Functions**[WinCopy](#)

Copies the active window to the Windows clipboard.

[WinFile](#)

Writes the active window to a file.

[WinPrint](#)

Prints the active window.

## CtAPI Functions in Version 7.0

The following section details the changes made to CtAPI functions in CitectSCADA Version 7.0:

### ■ [Obsolete Functions](#)

#### Obsolete Functions

Previously available "Point" related functions are now no longer available, and if used will return an error indicating that they are not supported. In order to obtain the same result as was previously invoked by those function, replace them with the Tag based equivalent using the appropriate Tag arguments and conditions.

The "point" functions that are no longer available are listed below along with their replacement functions:

Function	Replacement
ctPointGetProperty	ctTagGetProperty
ctPointNew	ctListNew or ctTagWrite and ctTagRead
ctPointRead	ctTagRead or ctListRead with ctListData
ctPointWrite	ctTagWrite or ctListWrite

If you are using the point functions on single tags, you should use the ctTagRead, ctTagWrite functions instead. If you are building up multiple tags into one point, you should use ctListNew and add tags to the list through ctListAdd. Then use ctListWrite, ctListRead and ctListData to write and read from the tags.

The following functions are not relevant to tag based operations. They are obsolete and there is no replacement function.

- ctPointBitShift
- ctPointClose
- ctPointCopy
- ctPointDataSize
- ctPointToStr
- ctStrToPoint
- ctTagToPoint

## Kernel Commands in Version 7.0

The following Kernel commands are now obsolete in CitectSCADA Version 7.0:

- Kernel Alarm

- Kernel Trend
- Kernel Report
- Kernel IOServer
- Kernel Client
- Probe
- NetBIOS
- PageNetstat

# Chapter 3: Upgrading to CitectSCADA v7.0

---

To upgrade your project to run in CitectSCADA v7.0 you will need to:

- **Upgrade CTAPI Applications**

Ensure that CTAPI applications are upgraded before upgrading and running any CitectSCADA V7.x projects.

- **Configure I/O Devices**

Before upgrading, ensure that all I/O devices are configured as required to run in the project.

- **Define Clusters**

Clusters can now be defined. The project must be configured to use at least one cluster.

- **Configure Network Addresses**

The network addresses and ports of the computers to be used as servers are now defined in the project.

- **Configure Servers**

The Alarm, Report, Trend, and I/O servers are now defined in the project.

- **Configure Tags to use Clustering**

Alarms, reports, trends, SPC tags, and accumulators can now be configured to run in a specific cluster.

**Note:** If you are running CitectSCADA version 5.5, ensure that you upgrade your projects to version 6.x before upgrading to version 7.0.

See Also [Upgrading Procedures](#)  
[Deprecated and Removed Functions](#)

## Deprecated and Removed Functions

Some Cicode functions have been deprecated, removed, or modified. For details, refer to [What's New in CitectSCADA v7.0](#).

## Upgrading Procedures

To upgrade an existing project to version 7.0, you should perform each of the following procedures:

- [Upgrade CTAPI Applications](#)
- [Configure I/O Devices](#)
- [Run the Citect Installer](#)
- [Launch CitectSCADA](#)
- [Define Clusters](#)
- [Configure Network Addresses](#)
- [Configure Servers](#)
- [Configure Tags to Use Clustering](#)
- [Compile the Project](#)
- [Run Computer Setup Wizard](#)

## Upgrade CTAPI Applications

To ensure that CitectSCADA v7.0 can communicate with CTAPI applications in your system (including CitectSCADA Reports and Ampla), ensure that these products have been upgraded to their latest versions before running any upgraded CitectSCADA v7.0 projects.

Similarly, if you are using any custom CTAPI applications, upgrade CitectSCADA on the computers where these applications are installed before upgrading any other CitectSCADA computers.

See Also [Configure I/O Devices](#)  
[CtAPI Functions](#)

## Configure I/O Devices

The upgrade process ensures that the functionality of your project is upgraded to version 7.0. To facilitate this, parts of your project configuration may change during the upgrade. It is therefore important to ensure that a project is configured the way you want it to run before you upgrade.

In particular, memory I/O devices defined in your project (specifying MEMORY as the port) will be configured in the upgraded project as local variables, since memory I/O devices are no longer supported. Local variables offer the same functionality as memory I/O devices without the need for further configuration.

However, this also means if you have I/O devices temporarily defined as memory I/O devices for testing or simulation, they will be incorrectly configured as local variables by the upgrade process. Ensure that you configure these devices as you require them to run before upgrading to version 7.0.

To ensure that you configure all of your I/O devices correctly, first read the information in [Configuring Local Variables](#) and [Using Memory Mode](#). This provides details about local variables and the other I/O device options that replace memory I/O devices, allowing you to select and configure your project appropriately.

You should also configure your Alarm servers to use the Publish Alarm Properties property on the Alarm Server, since Alarm devices with their Protocol property set to "Alarm" are no longer supported and will be removed by the Migration Tool.

The reconfiguration will take place when you run the Migration Tool. For detailed information on this tool, refer to [Migration Tool](#).

See Also [Run the Citect Installer](#)  
[Configuring Local Variables](#)  
[Using Memory Mode](#)

## Run the Citect Installer

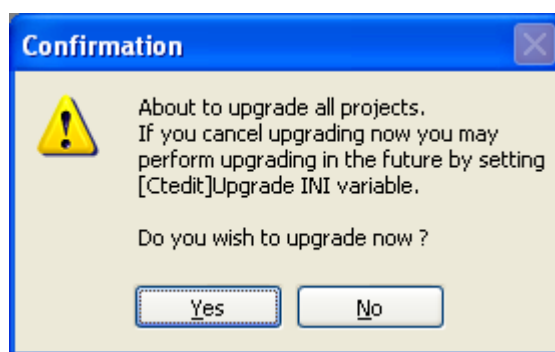
To begin the upgrade, run the CitectSCADA v7.0 installer. The installer will lead you through a number of steps until the installation is complete.

**Note:** Ensure that you uninstall version 6.x before installing version 7.0, as CitectSCADA does not support different versions running side-by-side. Additionally, to use the version 7.0 Example and CSV\_Example projects, it is recommended that you delete the existing Example and CSV\_Example projects using Citect Explorer before starting the installation.

## Launch CitectSCADA

An automatic upgrade of your CitectSCADA projects will occur when you initially launch CitectSCADA.

- 1 To launch CitectSCADA, click **Start | All Programs | Citect | CitectSCADA 7 | CitectSCADA Explorer 7**. The following message will display:



- 2 Click **Yes** to confirm the upgrade.

See Also [Define Clusters](#)  
[Migration Tool](#)

## Migration Tool

An automatic update is carried out when you initially launch CitectSCADA v7.0. This update is a passive action which updates the database field definition for any database that has been changed between the two versions and copies new files that are required in version 7.0. Prior to the automatic upgrade

proceeding you are given the option of cancelling the upgrade. The upgrade can be invoked at a later time by setting the Update parameter to True in the Citect.ini file.

After the automatic update has completed you should then prepare your projects prior to running the CitectSCADA Migration Tool.

The Migration Tool is a separate application which should be manually run after the automatic upgrade has been executed, and initiated by you after you have prepared the project for final migration. This tool will accommodate the critical changes in project functionality that are incorporated in version 7.0.

It is important that you prepare your existing projects for a successful upgrade using this tool.

Some of the features introduced in version 7.0 of CitectSCADA require changes in the project data from CitectSCADA version 6.1

The following topics describe the changes that will be made by the Migration Tool:

[Memory Devices](#) - Devices with their port value set to "Memory"

[Alarm Devices](#) - Devices with their Protocol property set to "Alarm"

See Also [Using the Migration Tool](#)

#### Memory Devices

In previous versions of CitectSCADA an I/O device could be defined as a memory device by setting the port value to "Memory". This was generally done for one of the following purposes:

- To provide for future devices that were not currently connected to the system, but their points needed to be configured at this stage of project.
- For virtual devices where there was no corresponding physical I/O device and you needed a data storage with all the I/O variables functionalities (such as alarms).
- To act as a variable which was local to the process be used in place of Cicode global variables.

You can still use I/O devices for future or virtual devices in version 7.0, but you should manually set the Port parameter to a value other than Memory, and set the Memory property of the device to True to indicate that it is an off line in-memory device before running the Migration Tool.

You need to review your project to identify which memory I/O devices are local variable holders and which ones need to be changed to non-memory so that the Migration tool does not convert their variables.



The Migration Tool will set any I/O device's port which is identified as a Memory device to the new Local Variable, and the original device record will be deleted.

See Also [Configure I/O Devices](#)  
[Alarm Devices](#)  
[Converting Memory Variables](#)

#### Alarm Devices

In previous versions of CitectSCADA Alarm devices were defined as devices with their Protocol property set to "Alarm". In version 7.0 the function of configuring such a device is now replaced by setting the Publish Alarm Properties property to True on the Alarm Server.

All Alarm devices with their Protocol property set to "Alarm" will be deleted from I/O devices table by the Migration Tool. Before running the Migration Tool you should identify these devices and set the Publish Alarm Properties property to True on the Alarm Server.

See Also [Alarm Server Definitions](#)

The Migration tool can delete all the memory and alarm device records. If you want to convert the variables and delete the devices at a later time, deselect the "Remove obsolete Memory and Alarm Devices" option

See Also [Converting Memory Variables](#)

#### Converting Memory Variables

A memory variable is a variable with its I/O device Port property set to either "Memory" or "MEM\_PLC".

If there are multiple I/O devices with the same name, possibly on different I/O servers, then the device would not be considered as a memory device regardless of its port value. In other words the Migration tool will not process the variables for memory devices with duplicate names.

See Also [Inserting new local variables](#)  
[Deleting Variable Tags](#)

#### Inserting new local variables

When the Migration Tool runs, a local variable record will be inserted for each identified memory variable, and the variable data will be copied into the new local variable.

Local variables have fewer fields than variables; the following table shows the mapping from variable to local variable when copying their data.

Variable Tag parameter or constant value	Local variable parameter
Variable Tag name	Name

Variable Tag parameter or constant value	Local variable parameter
Data Type	Date Type
(Empty)	Array Size
Eng. Zero Scale	Zero Scale
Eng. Full Scale	Full Scale
Comment	Comment

With the exception of the Array Size, which has been introduced in version 7.0 exclusively for local variables, all fields receive their values from the same or similar field.

See Also [Deleting Variable Tags](#)

#### Deleting Variable Tags

Once the Migration Tool has created the local variable records it will insert all those variable tag records that have been converted in the previous step, and delete the original variable tag.

If an error occurs during the insertion of the local variables, the deletion of the variable tags will not be performed. If this occurs it is possible to have two records with same name and data, one in the local variable (the newly inserted record) and one in the variable tags (the original record that has not been deleted). You need to delete either of the variables manually, or restore the backed up project after removing the cause of the error then run the Migration Tool again.

See Also [Default Scale](#)  
[Deleting Obsolete I/O Devices](#)

#### Default Scale

The Scale properties in both variable tags and local variables are optional. If a Scale value is not specified the default value is indicated by a parameter in the Citect.ini file. The parameter name is "DefaultSliderScale" under the [General] section in the Citect.ini file. The default values for Scale is 0-32000, unless the default slider scale is true in which case the default value depends on the type e.g. Integer, String etcetera.

The Migration tool will read this parameter and if it is not set, or set to false, then it will explicitly set any empty Scale property to a value in to the range of 0 to 32000. This will be done even if either of the Zero Scale or Full Scale parameters has a value, in which case the empty Scale parameter will receive the default value.

If the DefaultSliderScale in the Citect.ini file set to True, the Scale parameters will not be populated with a default value if they are empty, rather they will be interpreted at runtime.

#### Deleting Obsolete I/O Devices

Deleting obsolete I/O devices is an optional step in the Migration Tool and will be performed after all the memory variables are converted. If the delete option is chosen, then all obsolete Memory devices and Alarm devices will be deleted as the final step of the Migration Tool operation.

See Also [Included Projects](#)

#### Included Projects

Each project may contain multiple included projects. Additionally any included project may contain its own included project so creating a cascading project.

The Migration Tool needs to process the original project and all included projects in a single step. The reason for this is that variables can be defined in one project that refer to I/O devices defined in another included project.

The Migration Tool performs this procedure sequentially on the "master" project then each included project.

In the case where two master projects share the same project as an included project, it is important that you do not select the "Remove obsolete Memory and Alarm devices" check box when you process a project that contains shared included projects. This is because the removal is performed at the conclusion of the migration process on each master and included projects sequentially. This could cause the deletion of an I/O device in the first master project which is referenced by a tag in a shared included project which is processed in a later step.

If two separate "master" projects contain the same included project, run the Migration Tool on each "master" project without selecting to delete obsolete devices.

To remove obsolete devices it is recommended that once the Migration Tool has completed successfully (without the check box being selected), you should run it a second time with the check box selected. This will safely remove the devices since all tag conversions were completed in the first pass of the Migration Tool.

#### Using the Migration Tool

**Note:** Before you use the Migration Tool is strongly recommended that you familiarize yourself with the process that it performs, and the preparatory steps that you need to carry out with your existing projects as described under [Migration Tool](#).

To run the Migration Tool:

- 1 Backup the project/s that you intend to migrate.
- 2 From the Citect Explorer or Citect Editor menu bar select Tools, Migration Tool to display the Migration Tool dialog.

- 3 Either accept the project displayed in the edit box, or browse for the project that you wish to upgrade. If the project contains included projects, only select the “master” project. Any includes will be processed automatically.
- 4 Select the Remove obsolete Memory and Alarm devices check box if you wish to delete these devices after successful migration.

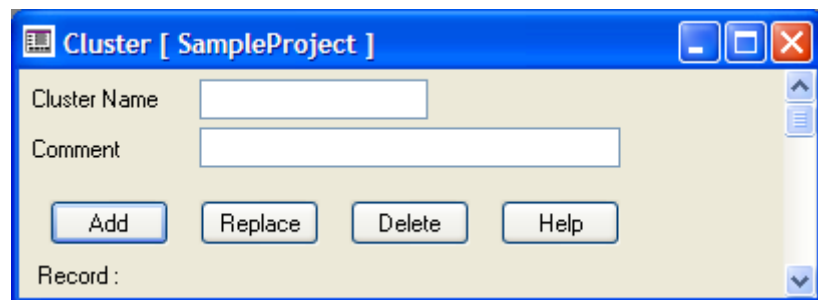
**Note:** Do not select this check box if the project contains any included projects which are shared with more than one master project when you run the tool for the first time on such projects. Run the tool a second time using this option if the migration is successful after it is run the first time if you want to delete the devices.

- 5 Click Migrate to begin the migration process, or click Close to exit without performing the migration.
- 6 The migration process will begin and display a progress dialog indicating the stage of the conversion and the name of the project being migrated. If you wish to cancel the migration at this point click the Abort button. Please note that aborting a migration will stop the migration process, and any changes already completed will not be rolled back. You will have to restore your project from the backup created in the first step.
- 7 When the migration process is concluded a confirmation dialog box will display indicating the number of variables converted and the number of I/O devices deleted (if device deletion was selected at the start of migration)
- 8 Click the Close button to close the dialog.

## Define Clusters

Even if you do not intend to use clusters in your project, you must define at least one. All tags and servers will default to run in the defined cluster.

- 1 In the Project Editor, select **Servers** | **Clusters**. The Cluster dialog box displays:



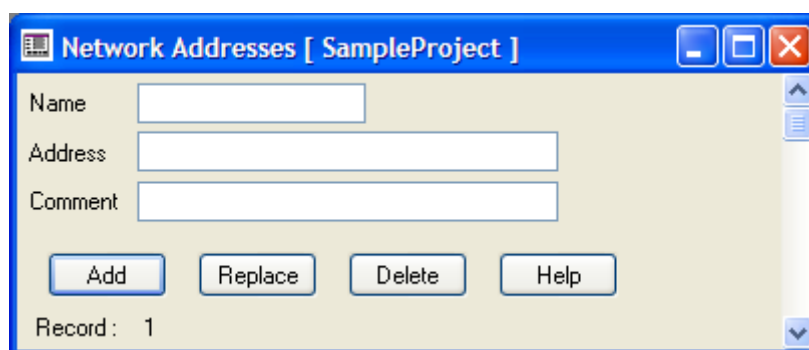
- 2 In the **Cluster Name** field, enter the name of the cluster (maximum of 16 characters). The name must be unique to the project and must not contain spaces.
- 3 In the **Comment** field, enter any useful comment. This property is optional and is not used at runtime.
- 4 Click **Add**.

See Also [Configure Network Addresses](#)

## Configure Network Addresses

In the Project Editor, configure the network address of each machine to be used as a server in the project.

- 1 In the Project Editor, select **Servers | Network Addresses**. The Network Addresses dialog box displays:



- 2 In the **Name** field, enter a name for the network address being configured (maximum of 16 characters). The name must be unique to the project and must not contain spaces.
- 3 In the **Address** field, enter the IP address or computer name of the machine being configured.
- 4 In the **Comment** field, enter any useful comment. This property is optional and is not used at runtime.
- 5 Click **Add**.

See Also [Configure Servers](#)

## Configure Servers

All Primary and Standby Alarm, Report, Trend, and I/O servers are now defined using the Project Editor. This involves specifying a network address and a cluster for each server.

### Default Ports

Each server has a unique default port assigned to it. This default port may only be used with that type of server. Attempting to use a default port on another type of server will result in a compilation error of:

"Invalid port number (2073-2082,20222,21) are reserved"

The following table lists the default port numbers and their associated server type.

Default Port	Server Type	Server Role
21	FTP Server	Page downloads for IDC
2073	CTAPI	CTAPI Communications
2074	Client	Cicode Debugging
2075	Report Server	Report Server communications
2076	Alarm Server	Alarm Server communications
2077	Trend Server	Trend Server communications
2078	IO Server	Legacy IO Communications
2079	IDC	Internet Display Server/Client communications
2080	Alarm Server	Alarm Properties Connector
2081	Time Server	Time Server communications
2082	IO Server	Publish Subscribe IO Server Communications
20222	ODBC	ODBC Server

For details on configuring each type of server refer to:

- [Alarm Server Definitions](#)
- [Report Server Definitions](#)
- [Trend Server Definitions](#)
- [I/O Server Definitions](#)

## Configure Tags to Use Clustering

If you have defined multiple clusters, you can configure alarms, reports, trends, SPC tags, and accumulators to run on particular clusters. Using the Project Editor, you can now specify the appropriate cluster for each tag.

- [Configure Alarm Tags to use Clustering](#)
- [Configure Report Tags to use Clustering](#)
- [Configure Trend Tags to use Clustering](#)
- [Configure SPC Tags to use Clustering](#)
- [Configure Accumulators to use Clustering](#)

### Configure Alarm Tags to use Clustering

In the Project Editor, select **Alarms**, and then the type of Alarm you are configuring. The dialog box for the chosen alarm type displays:

The screenshot shows the 'Advanced Alarms [ SampleProject ]' dialog box. It features a title bar with standard window controls. The main area contains the following fields and controls:

- Alarm Tag:** A text input field.
- Cluster Name:** A dropdown menu.
- Alarm Name:** A text input field.
- Alarm Desc:** A text input field.
- Expression:** A text input field with a dropdown arrow on the right.
- Category:** A text input field, a 'Help' button, and a dropdown menu.
- Delay:** A text input field with a dropdown arrow.
- Comment:** A text input field.

At the bottom of the dialog, there are four buttons: 'Add', 'Replace', 'Delete', and 'Help'. Below these buttons is a 'Record :' label followed by a dropdown arrow.

For each alarm, in the **Cluster Name** field, select the name of the cluster that will run the alarm. If there is only one cluster defined in the project, you can leave this field blank. The alarms will default to the defined cluster.

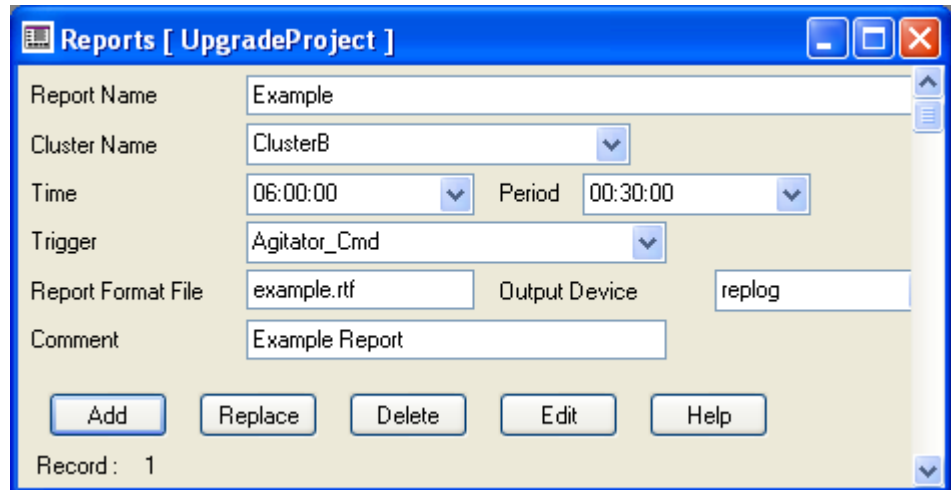
If the project has multiple clusters, and you do not select a cluster name in this dialog, then CitectSCADA considers the alarm to run on all defined clusters.

Click **Replace** to save the changes.

See Also [Configure Report Tags to use Clustering](#)

Configure Report Tags to use Clustering

In the Project Editor, select **System | Reports**. The Reports dialog box will display:



For each report, in the **Cluster Name** field, select the cluster that will run the report. If there is only one cluster defined in the project, you can leave this field blank. The reports will default to the defined cluster.

If the project has multiple clusters, and you do not select a cluster in this dialog, then CitectSCADA considers the report to run on all defined clusters.

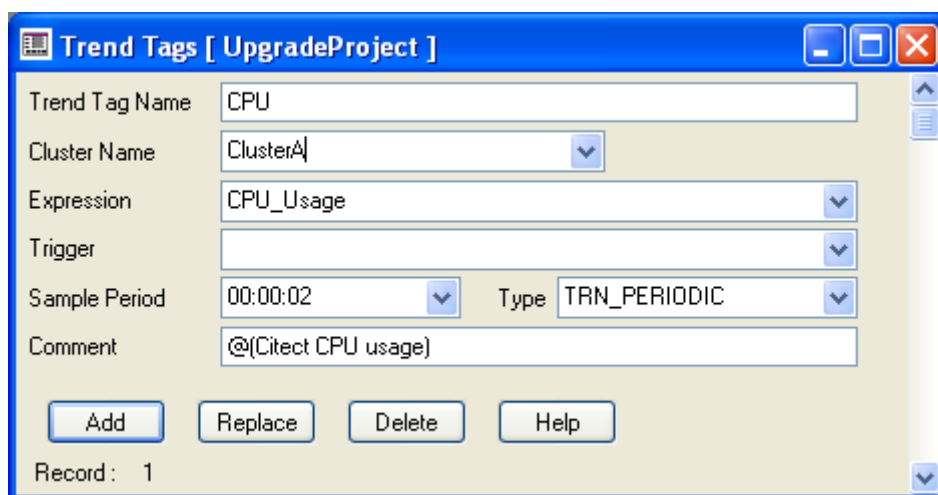
Click **Replace** to save the changes.

See Also [Configure Trend Tags to use Clustering](#)



Configure Trend Tags to use Clustering

In the Project Editor, select **Tags | Trend Tags**. The Trend Tags dialog box will display:



The screenshot shows the 'Trend Tags [ UpgradeProject ]' dialog box. It contains the following fields and controls:

- Trend Tag Name:** A text box containing 'CPU'.
- Cluster Name:** A dropdown menu with 'ClusterA' selected.
- Expression:** A text box containing 'CPU\_Usage'.
- Trigger:** A dropdown menu.
- Sample Period:** A dropdown menu with '00:00:02' selected.
- Type:** A dropdown menu with 'TRN\_PERIODIC' selected.
- Comment:** A text box containing '@(Citect CPU usage)'.
- Buttons:** 'Add', 'Replace', 'Delete', and 'Help'.
- Record:** A label showing 'Record : 1'.

For each trend, in the **Cluster Name** field, select the cluster that will run the trend. If there is only one cluster defined in the project, you can leave this field blank. The trends will default to the defined cluster.

If the project has multiple clusters, and you do not select a cluster in this dialog, CitectSCADA considers the trend to run on all defined clusters.

Click **Replace** to save the changes.

See Also [Configure SPC Tags to use Clustering](#)

Configure SPC Tags to use Clustering

In the Project Editor, select **Tags** | **SPC Tags**. The SPC Tags dialog box will display:

The SPC Tags dialog box is titled "SPC Tags [ UpgradeProject ]". It contains the following fields and controls:

- SPC Tag Name:** Text box containing "Feed\_SPC".
- Cluster Name:** Dropdown menu showing "ClusterA".
- Expression:** Text box containing "800 + Rand(50)".
- Trigger:** Dropdown menu.
- Sample Period:** Text box containing "2".
- Type:** Dropdown menu showing "TRN\_PERIODIC".
- Cp and Cpk Calculations:**
  - Lower Spec Limit:** Text box containing "790".
  - Upper Spec Limit:** Text box containing "860".
- Comment:** Text box containing "Feed input to conveyor cv103".
- Buttons:** "Add", "Replace", "Delete", and "Help".
- Record:** Label "Record : 1".

For each SPC tag, in the **Cluster Name** field, select the cluster that will run the SPC tag. If there is only one cluster defined in the project, you can leave this field blank. The SPC tags will default to the defined cluster.

If the project has multiple clusters, and you do not select a cluster in this dialog, CitectSCADA considers the tag to run on all defined clusters.

Click **Replace** to save the changes.

See Also [Configure Accumulators to use Clustering](#)

Configure Accumulators to use Clustering

In the Project Editor, select **System** | **Accumulators**. The Accumulators dialog box will display:

**Accumulators [ UpgradeProject ]**

Name: PUMP\_A\_ACC

Cluster Name: ClusterA

Trigger: PUMP\_A\_CMD

Run Time: PUMP\_A\_ART

No. of Starts: PUMP\_A\_AS

Totaliser Inc: 10

Totaliser: PUMP\_A\_TOT

Comment: Pump\_A Accumulator

Add Replace Delete Help

Record : 1

For each accumulator, in the **Cluster Name** field, select the cluster that will run the accumulator. If the project has only one cluster defined, you can leave this field blank. The accumulator will default to the defined cluster.

If the project has multiple clusters, and you do not select a cluster in this dialog, CitectSCADA considers the accumulator to run on all defined clusters.

Click **Replace** to save the changes.

See Also [Compile the Project](#)

## Compile the Project

Once you have configured your project, compile it and ensure that there are no errors.

At this stage you may want to reconfigure some of your Cicode to support online changes. Particularly the [AssInfo](#) and [TagInfo](#) functions that will be deprecated in future versions of the software. In most cases they can be replaced with the functions [AssInfoEx](#) and [TagInfoEx](#).

## Run Computer Setup Wizard

Run the Computer Setup Wizard for each computer running the project. At each stage of the Wizard, configure the appropriate settings for that computer.

See Also [Running the Computer Setup Wizard](#)

## Troubleshooting

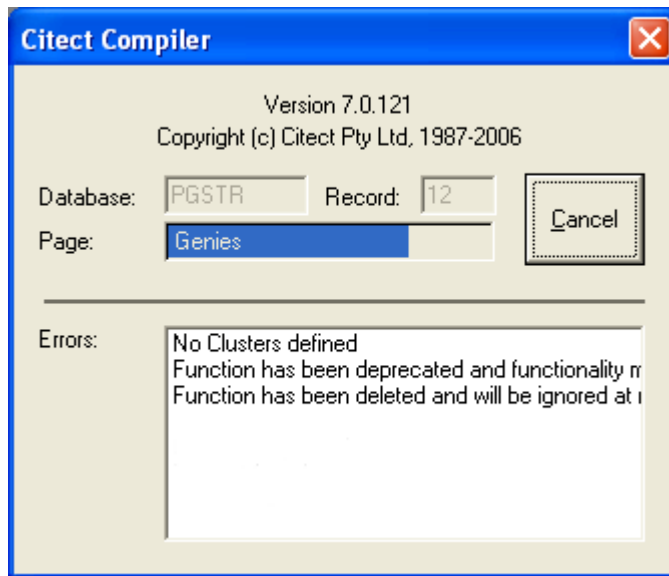
The following issues should be noted when upgrading to CitectSCADA v7.0:

- [Compiler Errors](#)

### ■ [Upgrading a Project that uses Distributed Servers](#)

#### Compiler Errors

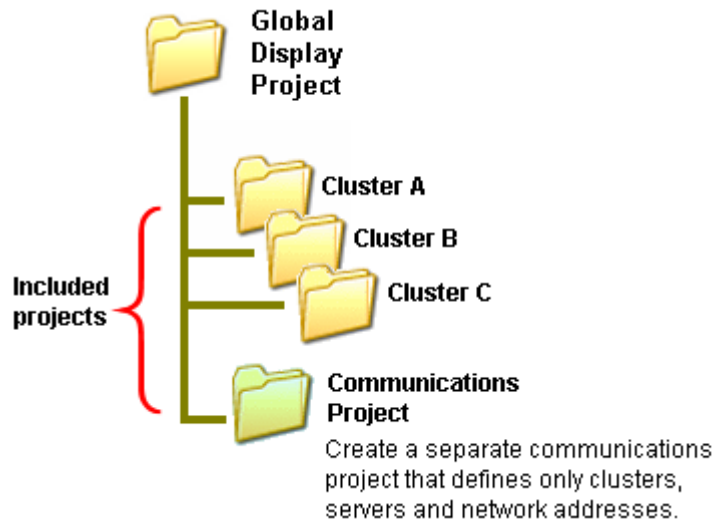
Before you configure your project to run in version 7.0, compiling the project will generate a number of compiler errors. These may include warnings about deprecated and deleted functions, as well as the error 'No Clusters defined'. This error will be resolved once you define a cluster in the project.



See Also [Troubleshooting](#)

#### Upgrading a Project that uses Distributed Servers

If you have implemented clustering in Version 6.x using Distributed Servers, a Global Include Project, and Cluster Projects, you should configure your project to use clustering in version 7.0.



The existing structure of the Global Display Project and Cluster Projects can remain the same (that is, the Global Display Project includes each of the Cluster Projects).

The following points describe a recommended project structure for clustering in version 7.0:

- Include a separate communications project in the Global Display Project. In this project, it is recommended that you define only the Network Addresses, Clusters, and Servers.

**Note:** Defining a separate communications project means that when the Global Display Project compiles, it has the communications information without needing to load all of the data from the Cluster Projects.

- In each Cluster Project, specify the appropriate cluster for alarms, trends, reports, SPC tags, and accumulators.
- You may need to modify the buttons and pages in the Global Display Project, particularly if they are using Cluster functions which have been modified or deprecated.

See Also [Deprecated and Removed Functions](#)



# Chapter 4: About CitectSCADA

---

CitectSCADA is a Supervisory Control and Data Acquisition (SCADA) solution that is used to manage and monitor processes in manufacturing, primary production, utilities delivery and facilities management.

The graphics, controls, configuration data and programming associated with a CitectSCADA installation is configured and implemented through projects. A project acts as a digital representation of your production facility that is deployed in tandem with your plant infrastructure, allowing the entire system to be monitored and controlled in real-time.

## Configuring a CitectSCADA project

Initially you use CitectSCADA's configuration environment to identify and address the devices and datasources included in a project by tagging the variables associated with each.

You can then use templates to design graphics pages that reference these tags, creating an interface your staff can use to view and control the system.

With these graphics pages, you can:

- use animations to display the operating status and performance of a plant
- provide operators with centralized or local control of production equipment using keyboard commands and graphical tools
- develop a multi-layered security system that controls user access according to functional groups or geographical areas
- implement historical and millisecond trending of tag data in a graphical format.

A powerful scripting language is also included to enable customized, programmable functionality.

## Deploying CitectSCADA

The project is then deployed across a client-server network architecture. The servers are used to manage communication with plant equipment and collate production data, while the clients provide the interface for operators and managers to assess and interact with the system.

This architecture allows the flexibility to adapt CitectSCADA to any production scenario, with support for full scalability, server clustering, and system redundancy.

### Running a project

When a project is eventually compiled and implemented in runtime, your production staff can visually monitor the system, initiate production processes and respond to alarm conditions.

Historical and trend data can also be collated and distributed to assess issues such as production performance, efficiency, and maintenance requirements.



# Chapter 5: CitectSCADA Tools

---

CitectSCADA's architecture can be divided into three distinct areas of functionality:

- configuration
- runtime
- drivers

Configuration involves all the tasks required to prepare and build a project, while runtime is the implementation of a project in a live production environment.

Drivers enable communication with devices via a number of communication protocols. The driver defines the specific project settings required for CitectSCADA to communicate with a particular device.

When considering the tools included with CitectSCADA, it is easiest to look at their roles in either configuration or runtime.

See Also [Configuration Tools](#)  
[Runtime Tools](#)  
[Drivers](#)

## Configuration Tools

The following tools enable you to configure a project and its components, and set up computers to use CitectSCADA:



Citect Explorer

The application used to create and manage your CitectSCADA projects. Citect Explorer displays a list of all projects, and provides direct access to the components of each. You can use Explorer to rename, back up, restore or delete a project.

See [Administering projects](#)



Citect Project Editor

The application used to create and manage the configuration information for your project, including tags, alarms, system components, and communications components

See [Components of a project](#)



Citect Graphics Builder

The application used to design, create, and edit the graphics components of a project, including templates, graphics objects, symbols, genies, and Super Genies

See [Defining and Drawing Graphics Pages](#)



Computer Setup Editor

A utility for editing configuration files and generating reports to compare and analyze files

See [Computer Setup Editor](#)



Computer Setup Wizard

A wizard that allows you to customize a computer's setup and define its role and function

See [Running the Computer Setup Wizard](#)

## Runtime Tools

The following tools enable you to run, monitor, and control projects during runtime..



CitectSCADA Web Client

Displays a live CitectSCADA project within a web browser

See [CitectSCADA Web Client](#)



Internet Display Client

A computer used to run a CitectSCADA project over the Internet from a remote location.

See [Running Your System Over the Internet](#)



Process Analyst

An Active X control that allows you to compare and analyze historical and real-time trend and alarm data during runtime.

See [Configuring the Process Analyst](#)



CitectSCADA Runtime Manager

An application used to manage and control the CPU configuration of the project, and the running state of each component

See [Launching Runtime Manager](#)

---

## Drivers

CitectSCADA can communicate with an array of I/O devices, including PLCs (Programmable Logic Controllers), loop controllers, and distributed control systems (DCS).

The I/O devices may be local (directly connected to a CitectSCADA I/O Server) or remote (connected to CitectSCADA via an intermediate communications means like a phone line).

Drivers enable communication with devices via a number of communication protocols (including Ethernet, TCP/IP, and Serial). The driver defines the specific project settings required for CitectSCADA to communicate with a particular device. This includes information about:

- Boards
- Ports
- Devices
- Tag addressing

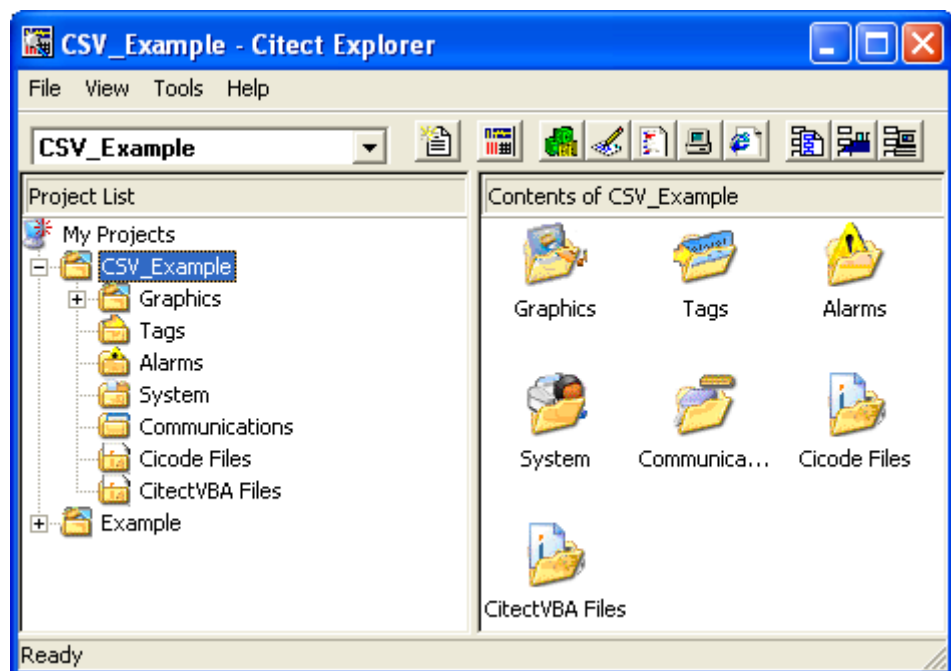


## Chapter 6: Components of a project

The components you can incorporate in a CitectSCADA project are logically divided across the following categories:

- [Graphics components](#)
- [Tags](#)
- [Alarms](#)
- [System components](#)
- [Communications components](#)
- [I/O Server components](#)
- [Cicode / CitectVBA](#)

These categories are represented in Citect Explorer through the set of folders associated with each project.



As you build a CitectSCADA project, the components you include are listed in the relevant project folder. Selecting an item from one of these folders launches the selected component in the tool required to edit its properties.

See Also [Graphics components](#)  
[Tags](#)  
[Alarms](#)  
[System components](#)  
[Communications components](#)  
[I/O Server components](#)  
[Cicode / CitectVBA](#)

## Graphics components

The graphical components of a CitectSCADA project represent the content used to create the screens presented on display clients. They include:



Pages

The basis for the screen layouts

See [Defining and Drawing Graphics Pages](#)



Templates

A collection of page layouts used to standardize display screens

See [Using Page Templates](#)



Symbols

Graphics objects stored in a library for reuse

See [Using symbols](#)



Genies

Objects that group multiple graphical and functional elements for easy duplication

See [Understanding Genies](#)

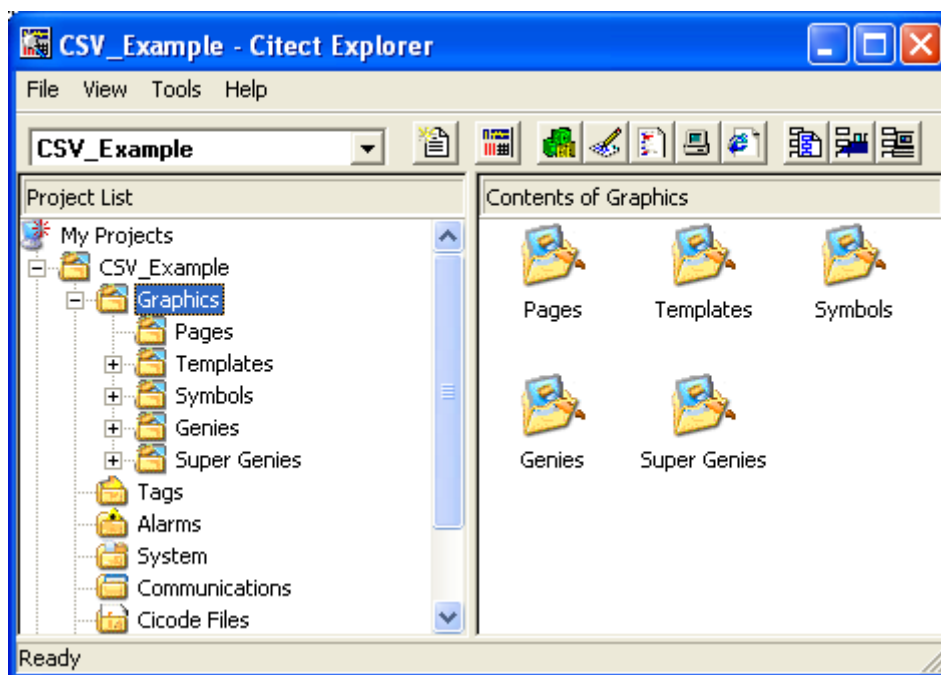


Super Genies

Genies that can be passed device specific information at runtime.

See [Using Super Genies](#)

As you create project pages in Graphics Builder, the included components are added to the relevant subdirectory in the current project's Graphics folder.



## Tags

Tags are used to identify the end points in the infrastructure you are using CitectSCADA to monitor and control. The name you give to a tag becomes a label for a register address, allowing it to be intuitively applied across graphics pages and in alarm notifications.

Three tag types are included in a project's Tags folder in Citect Explorer:



Variable tags

used to label register addresses

See [Tagging Process Variables](#)



Trend tags

used to label tags for data trending

See [Trending Data](#)

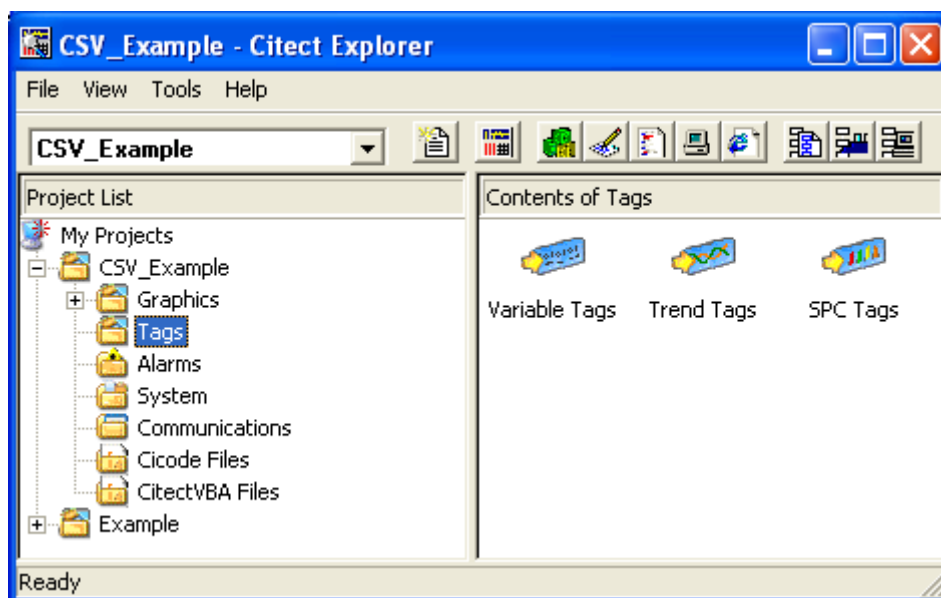


SPC tags

used to label tags according to Statistical Process Control principles

See [SPC Tags](#)

Selecting one of these tag types in Citect Explorer calls up the associated configuration dialog in Project Editor.



## Alarms

Alarms are used to identify conditions in a CitectSCADA system that require attention. CitectSCADA supports seven different alarm types:



Digital



See [Digital Alarms](#)



Analog

See [Analog Alarms](#)






	Time-stamped See <a href="#">Time-stamped Alarms</a>
	Advanced See <a href="#">Advanced Alarms</a>
	Multi-Digital See <a href="#">Multi-digital Alarms</a>
	Time-stamped Digital See <a href="#">Time-stamped Digital Alarms</a>
	Time-stamped Analog See <a href="#">Time-stamped Analog Alarms</a>

You can also use alarm categories within your CitectSCADA project to help identify and manage alarms.

## System components

The system components of a CitectSCADA project allow you to customize, manage, and track your runtime system. They include:

	Keyboard key	A meaningful name assigned to a keyboard key See <a href="#">Keyboard Keys</a>
	Keyboard commands	Key sequences with associated instructions See <a href="#">System Keyboard Commands</a>
	Reports	Customized presentation of runtime data and special conditions See <a href="#">Reporting Information</a>



## Events

Commands that execute in response to specific runtime triggers, such as a Cicode expression or variable tag. When the event trigger is true, the command will execute

See [Configuring Events](#)



## Accumulators

Variable tags tracking continuous runtime data. Data can be monitored and displayed by animating or trending the variable tags

See [Using Accumulators](#)



## Devices

Components that can transfer high-level data to other components such as RTF files, ASCII files, and printers

See [Configuring Devices](#)



## Users

User profiles to restrict and grant access to the runtime system

See [Maintaining User Records](#)



## Groups

Groups of system areas used to simplify the management of user profiles

See [Defining Areas](#)



## Label

System-wide substitutions for commonly used commands and expressions

See [Using Labels](#)



## Fonts

Fonts for displaying alarms and objects

See [Using CitectSCADA Fonts](#)



## Parameters

Built-in operating settings for fine tuning the runtime system

See [Citect.ini File Parameters](#)



#### Included projects

Predefined projects with database records automatically included in user projects

See [Including projects](#)

## Communications components

The communications components of a CitectSCADA project are the configured representation of the communications hardware in your system. They include:



#### Board

Hardware enabling various types of communication with I/O devices

See [Boards Properties](#)



#### Port

The physical connection between the board and the I/O device

See [Ports Properties](#)



#### Modem

Hardware used to connect CitectSCADA to a diallable remote I/O device.

See [Modems Properties](#)



#### I/O devices

An item of equipment that communicates with plant-floor control or monitoring equipment

See [I/O Devices Properties](#)



#### I/O device address

The unique address of an I/O device CitectSCADA is communicating with

## I/O Server components

The server components of a CitectSCADA project are the configured representation of the server computers in your system. They include:



#### Cluster

Logical groups of servers running across several physical machines

See [Implementing Clustering](#)



#### Network addresses

The IP addresses or machine names of the primary and standby servers

See [Network Address Definitions](#)



#### Alarms servers

Servers that monitor alarms and display them on the appropriate display client(s)



#### Reports servers

Servers that control the processing of reports



#### Trend servers

Servers that control the accumulation and logging of trend information



#### I/O servers

Dedicated communications servers that exchange data between I/O devices and CitectSCADA display clients

## Cicode / CitectVBA

CitectSCADA offers two programming languages with which you can control and manipulate CitectSCADA components:



#### Cicode

A structured programming language designed for use in CitectSCADA to monitor and control equipment

See [Introducing Cicode](#)



#### CitectVBA

A Visual Basic for Applications (VBA) and VBScript-compatible Basic scripting language

See [Introducing CitectVBA](#)

# Chapter 7: Typical system scenarios

---

The scenarios described in this chapter demonstrate how CitectSCADA can be used to support typical processes found in primary production, utilities delivery, and manufacturing.

In reality, a CitectSCADA project will incorporate a combination of the scenarios described here, with a high degree of customization and scalability. However, these examples have been simplified to demonstrate how CitectSCADA can be configured and deployed to meet the specific requirements of a production system.

- **Standalone system**

All the components of a CitectSCADA system run on a single computer. See [Standalone system](#).

- **Distributed I/O system**

CitectSCADA is used to monitor and manage distributed devices that are each connected to remote I/O servers. See [Distributed I/O system](#).

- **Redundant server system**

One or more of the servers associated with a CitectSCADA system are duplicated and defined as primary and standby units, allowing the system to keep running in the event of a hardware failure. See [Redundant server system](#).

- **Client-server system**

The servers and display clients associated with a CitectSCADA system are independently distributed across a number of computers on a network, offering greater accessibility and performance benefits. See [Client-Server system](#).

- **Reliable sub-control system**

Remote or geographically separate sections of a production system have fully operational sub-systems in place that are monitored and controlled locally. In the event of a shift change or system failure, this arrangement is supported by the ability to take control of one sub-system using the display clients of another. See [Reliable sub-control system](#).

- **Cluster controlled system**

A production system is broken into discrete areas being monitored by operators within each area. However, there is also a level of control that supervises all areas of the system. See [Clustered control system](#).

- **Load sharing system**

The CitectSCADA system splits the load of an otherwise stressed system across multiple machines, better utilizing the available infrastructure. See [Load sharing system](#).

See Also [Cluster Connections Configuration](#)

## Standalone system

A standalone installation of CitectSCADA runs all the server and client components of a system on a single computer. These include:

- I/O server
- Alarm server
- Trend server
- Reports server
- Display client

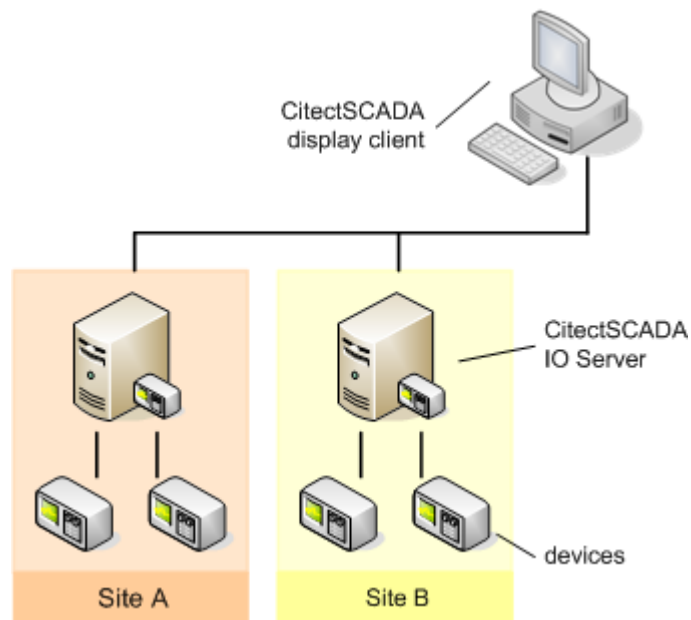
This allows CitectSCADA to be run as a small, self contained system.

**Note:** You can run the server and client components of a standalone system as a single-process or multi-process system. It is recommended that a single- process setup only be used as a short term solution for your control system, or to run demonstrations and test projects.

## Distributed I/O system

This scenario demonstrates a method of connecting CitectSCADA to a number of devices that are distributed across several sites over a wide geographical area.

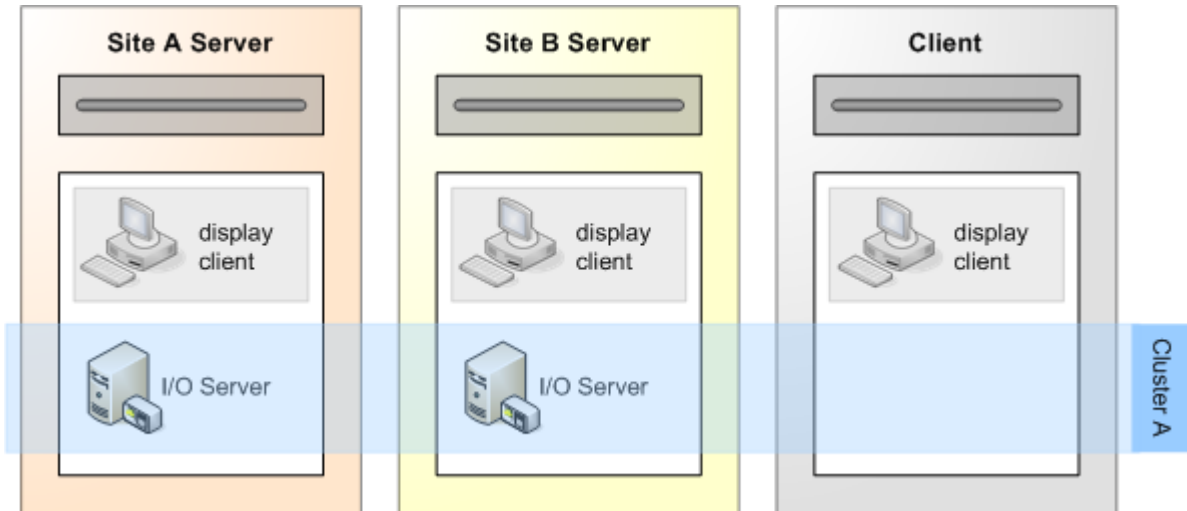
Instead of attempting to connect devices directly via a remote connection, an I/O Server is placed at each site, enabling communication to be managed within the CitectSCADA system.



This model is also useful in plants that contain devices with a serial port or limited communications capabilities. By placing I/O servers on the factory floor to interface with these devices, you can contain the communications limitation and improve overall performance.

Despite the geographical distribution of I/O servers across many sites, this type of system can be configured as a single cluster system, as a cluster is able to support many I/O servers.

The diagram below demonstrates how to approach the deployment of this type of system across the server machines using a single cluster.



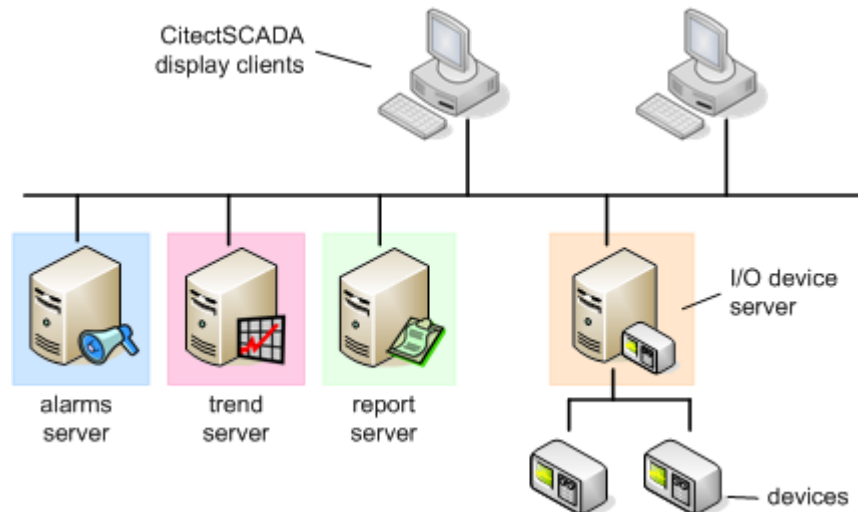
A second cluster will only become necessary if your project requirements call for more than one redundant pair of alarms, trends or reports servers.

### Client-Server system

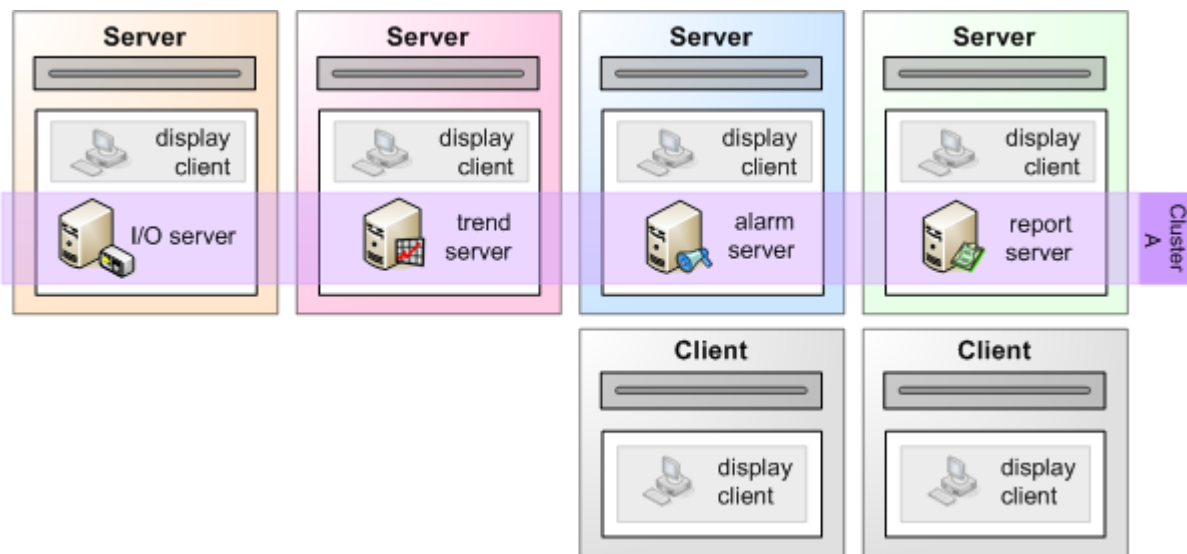
CitectSCADA's client-server architecture allows the components of a system to be distributed across a number of computers on a LAN, creating a system that offers geographical flexibility and performance benefits.

Each component is simply identified within the CitectSCADA project by an address, allowing the location and hardware requirements for each to be considered independently.





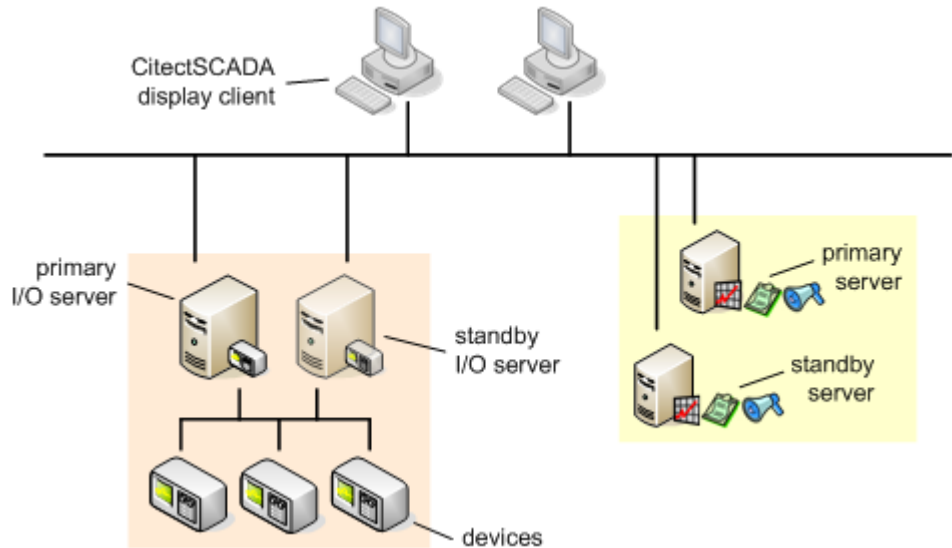
The diagram below demonstrates how this example can still be configured within a single cluster.



Note that each server also acts as a display client across the system architecture.

## Redundant server system

The ability to define primary and standby servers within a CitectSCADA project allows hardware redundancy to be built into your system infrastructure. This avoids the creation of a single point of failure within a system that requires continuous operation and/or secure data collection.

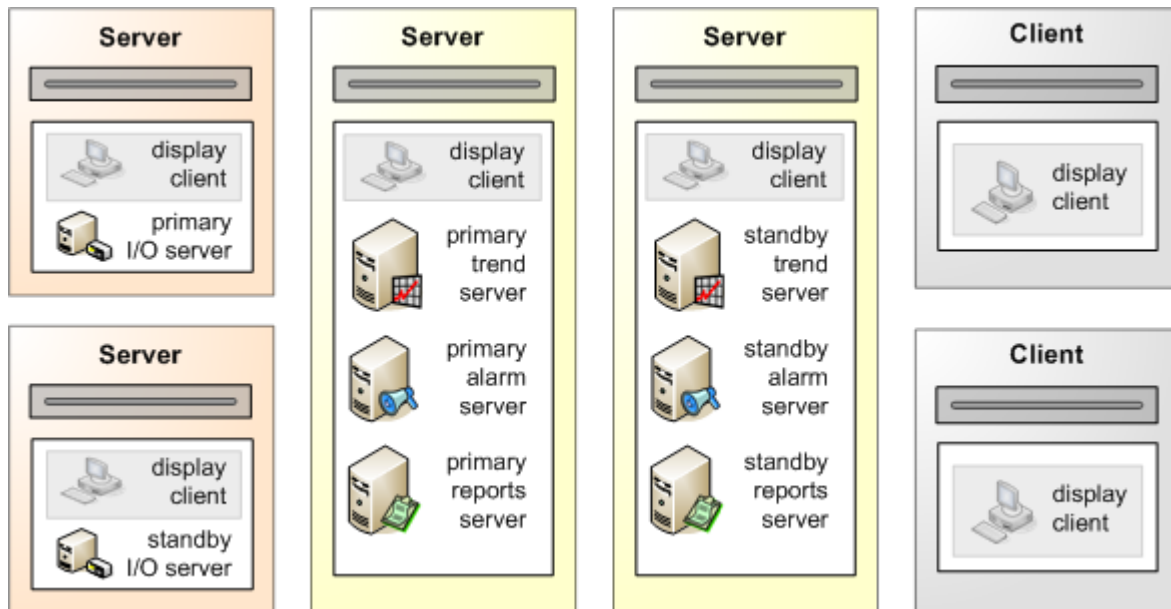


In the case of I/O server redundancy, a standby server is maintained in parallel to the primary server. If a hardware error occurs, the standby server can assume control of device communication without any interruption to the system. You can also use redundant I/O servers to split the processing load.

Alarm, report and trend servers can also be implemented as redundant servers. This ensures clients continue to have access to data from a standby server in the case of a primary server failure. CitectSCADA maintains identical data on both servers.

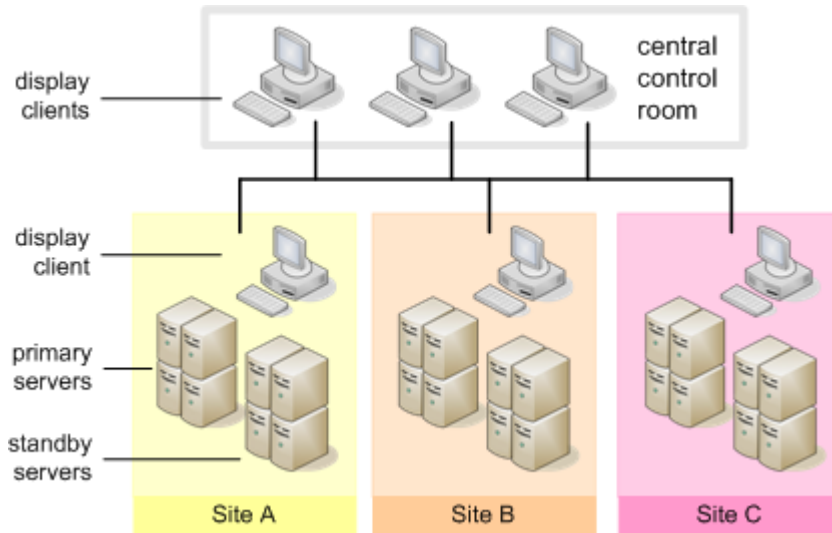
In the diagram below, the primary and standby IO servers are deployed independently, while the alarms, trends and reports servers are run as separate

processes on common primary and standby computers. In this case, the entire system can be configured as a single cluster.



### Clustered control system

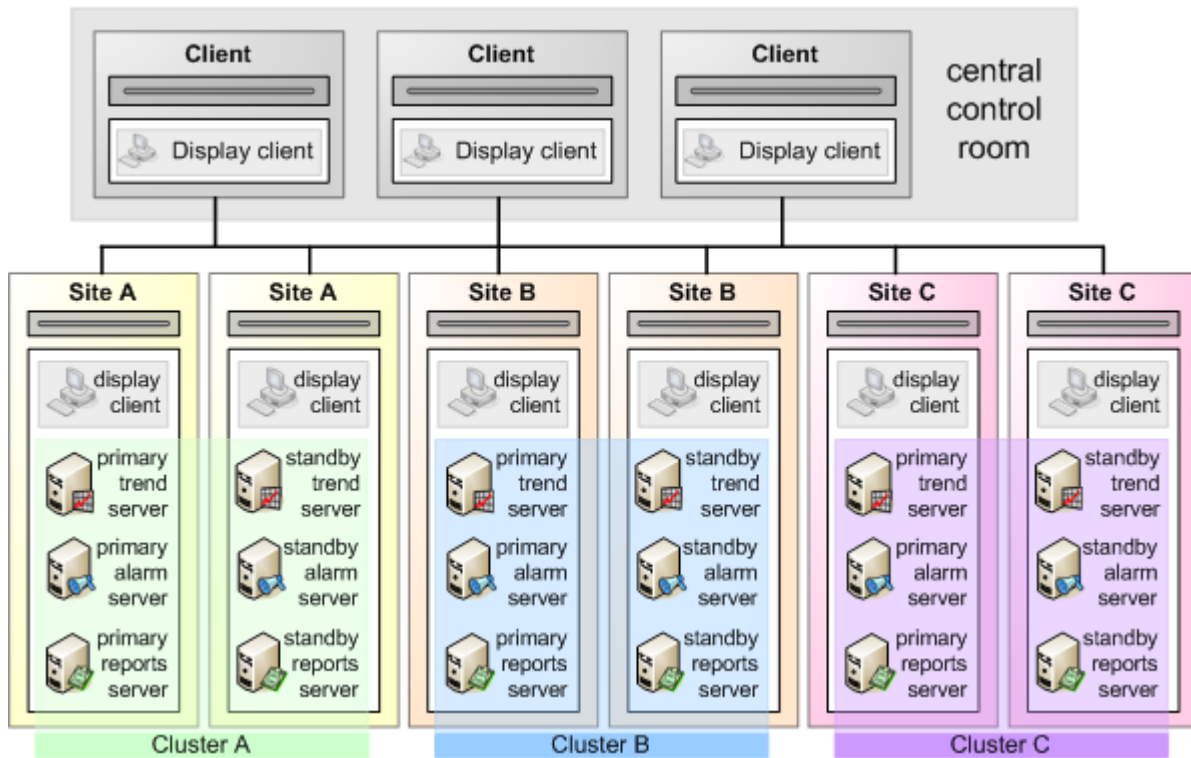
In this scenario, the system is broken into discrete sites being controlled by local operators, and supported by local redundant servers. At the same time, there is a level of management that requires all sites across the system to be monitored simultaneously from a central control room.



Each site is represented in the project with a separate cluster, grouping its primary and standby servers. Clients at each site are only interested in the local cluster, whereas clients at the central control room are able to view all clusters.

The deployment of a control room scenario is fairly straightforward, as each site can be addressed independently within its own cluster. The control room itself only needs display clients.

The deployment of servers could be mapped out as follows:



CitectSCADA's support for dynamic clustering means each site can be monitored and controlled from the central control room if required. For example, if an operator at a particular site only works during regular business hours, then the monitoring can be switched to the central control room after hours.

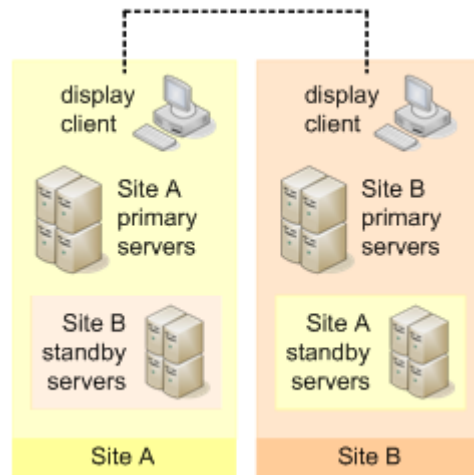
## Reliable sub-control system

In this scenario, a project represents a number of locally operated sites each containing its own set of servers and clients. For example, a number of pumping stations across a water distribution system, or multiple production lines in a manufacturing facility. However, there is a requirement for monitoring to continue in the event of system failure at one of the sites.

This is achieved by distributing the primary and standby servers across the different sites, or by placing all the standby servers in a central location.

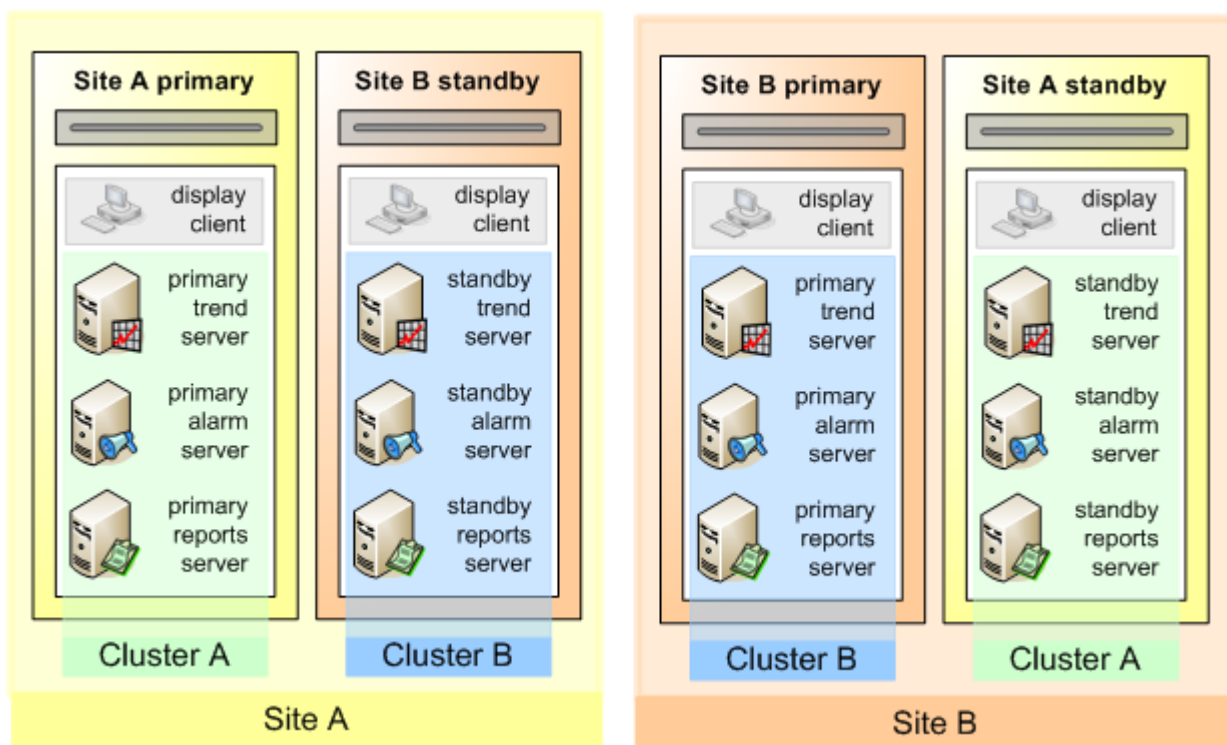
Clustering is used to define the role of the different servers at each site, all of which can be viewed in a common project running on every display client. This

means Site A can be monitored from Site B, and vice versa, if a system failure occurs at one of the sites.



The example above would require the creation of two clusters, so that the project can include two sets of primary and standby servers. The clusters represent the

redundant pairs of servers, and would be deployed across the two sites as follows:



The clusters offer the benefit of keeping a logical structure to the project during configuration, despite the unusual distribution of redundant server pairs.

## Load sharing system

Load sharing of system components across different computers and CPUs means the work load of a potentially stressed system can be split across multiple machines, better utilizing the available infrastructure.

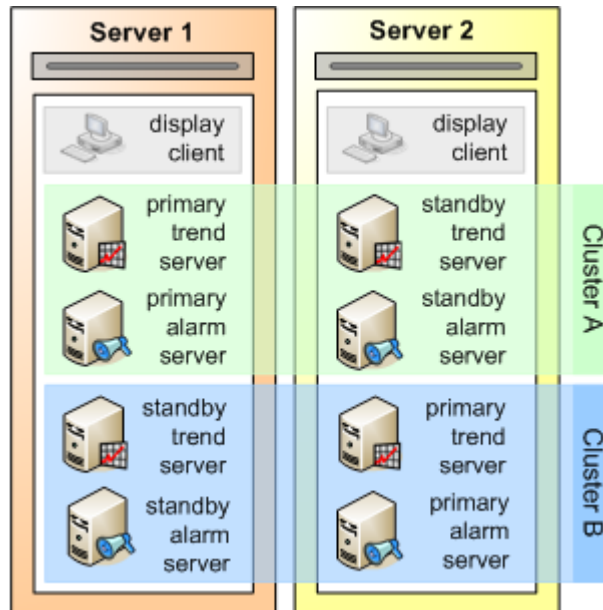
For example, managing alarms can draw heavily on a CPU's performance, while trending data can use a lot of disk space. By assigning your trend and alarm servers to different processes on a shared computer, an alarm server can be used as a standby trends server, making practical use of idle disk space.

This approach can be used to improve network performance, data access times, and general system stability.

If you introduce clustering, you have the flexibility to run multiple servers of the same type on a single computer. As long as a client has access to all the clusters

configured in a project, it doesn't matter if a set of servers is distributed across a number of clusters.

In the diagram below, two servers have been configured to act as standby units for each other, supporting two sets of redundant trend and alarm servers.



Both machines have an even balance of trend and alarm servers, ensuring effective use of the CPU and disk space. By distributing the servers across two clusters, the servers are also able to act as redundant units to each other. This has reduced the required number of computers from a maximum of eight down to just two.



# Using CitectSCADA

This section contains information on using CitectSCADA and describes the following:

- [The Physical Layout of a Plant](#)
- [Administering projects](#)
- [Securing CitectSCADA Projects](#)
- [Tagging Process Variables](#)
- [Defining and Drawing Graphics Pages](#)
- [Using Objects](#)
- [Understanding Object Types](#)
- [Defining Common Object Properties](#)
- [Defining Commands and Controls](#)
- [Configuring and Processing Alarms](#)
- [Configuring Events](#)
- [Using Accumulators](#)
- [Logging and Trending Data](#)
- [Understanding Statistical Process Control](#)
- [Reporting Information](#)
- [Using Security](#)
- [Using Labels](#)
- [Using Devices](#)
- [Exchanging Data with Other Applications](#)
- [Using Genies and Super Genies](#)
- [Working with Multi-Language Projects](#)
- [Using OPC Server DA2.0](#)
- [Communicating with I/O Devices](#)
- [Using the Communications Express Wizard](#)
- [Building Your CitectSCADA Project](#)



# Chapter 8: Planning a Project

---

This chapter describes the planning phase of a CitectSCADA system.

A planned approach to the design and configuration of your CitectSCADA system allows you to make optimal use of the product's features and performance capabilities, while ensuring all the requirements of your production facility are met. It also helps avoid unnecessary rework during the configuration of your CitectSCADA project.

It is important to consider the following when planning a system:

- 1 [The Physical Layout of a Plant](#)
- 2 [Operational Requirements](#)
- 3 [Project Design](#)
- 4 [Building Your Project](#)
- 5 [Deployment](#)

## The Physical Layout of a Plant

You need to consider the physical layout of the plant where you would like to implement CitectSCADA.

This information will help determine the architecture of your CitectSCADA project, and many of its operational requirements. It also allows you to assess the available equipment, to determine if any additions or modifications are required.

The following items should be considered when examining the physical layout of a plant:

### Geography

The physical layout of your facility, including whether the plant is spread across multiple geographical locations or specific areas of functionality, such as a number of production lines running in parallel.

### Machinery

The equipment (machines, physical connections, and devices) in your plant that will be monitored and controlled by your CitectSCADA system.

### Existing computer hardware

The computers that currently exist within your facility to support CitectSCADA's client-server architecture. The specifications and limitations of

the existing equipment will have to be considered to determine if the operational requirements of your system can be supported.

#### Network configuration

The current configuration of the network that will support your CitectSCADA system and its communication with plant equipment. This will include the protocols used, the performance capabilities of the system, and security.

See Also [Operational Requirements](#)

## Operational Requirements

By developing a set of operational requirements for your CitectSCADA project, you'll define a comprehensive list of needs and objectives that your system must support to effectively monitor and control production.

The things you should consider to determine the operational requirements include:

- Architecture
- Security
- Reliability
- Monitoring
- Data Collection

### Architecture

#### Production Processes

The operating processes within your production facility need to be considered to determine how they can be logically represented and supported within your CitectSCADA project. If the processes are dependent on each other, you also need to consider how the interaction between them will be managed, particularly if unexpected circumstances occur.

### Security

#### User access

You need to consider who will be using the CitectSCADA system and which parts of a project they will need to have access to. If security is a concern, you can restrict access on an individual basis, or through the creation of groups of users (e.g. operators, supervisors, managers).

#### [Maintaining User Records](#)

#### Areas of control

Depending on how your plant is structured, you may need to define areas within your project based on functional areas (e.g. receivals, despatch, and processing) or geographical areas. Areas of control work in tandem with CitectSCADA's user-based security to effectively manage control of the production processes.

### Defining Areas

#### Reliability

The nature of your production processes will determine the importance of system reliability. Consider issues such as:

- The need for uninterrupted operation
- the impact and cost of down-time
- the need to collect and protect system data
- the severity of alarm conditions.

This will help determine if your project should include redundancy, the type of redundancy required, and the most appropriate way to implement it.

For more information, refer to [Redundancy](#).

#### Monitoring

System monitoring is a key function of a SCADA system and needs to be considered in terms of the required interaction between personnel and production processes.

You need to consider if the delivery of data is time-critical. For example, alarm conditions should be presented in real time, trend data may be delivered with a slight delay, while maintenance data can be accumulated and viewed as required.

The system may also need to be monitored at different levels, from machinery operators to control room personnel managing plant-wide processes. For each level of monitoring, consider the data that needs to be presented, and the specific fault conditions that need to be flagged.

#### Data collection

Consider the kind of data you need to collect from the production process, and how it will be used. Depending on your requirements, CitectSCADA can collect:

- Production data
- Purchasing requirements
- Batch processing statistics
- Equipment status and performance data
- Maintenance scheduling information
- Process performance data
- Dynamic visual analysis data

An assessment of the likely amount of accumulated data should also be carefully considered, as it will significantly impact on your computer hardware and network performance requirements.

For more information, refer to [Logging and Trending Data](#).

See Also [Project Design](#)

## Project Design

Once you have developed a clear set of operational requirements, you need to plan how to design your project to best meet these requirements. When designing your project, you should consider the following issues:

### Naming Standards

By adopting naming standards, you can configure project components with meaningful names that convey useful information, such as the location or type of component. The standard that you use depends on the type of information that will be useful to system operators. A naming standard helps promote consistency throughout the project, making it easier to quickly identify components, and reducing duplication and user training. Naming standards may be useful for devices, variable tags, reports, graphics objects, and pages.

### Page Templates

Page templates are predefined page layouts that you can use to build the display screens (graphics pages) for your project. Templates allow you to create new pages quickly, and ensure that your runtime system has a consistent look and feel. They can incorporate standard navigational and support tools that are common to every page. CitectSCADA includes a number of standard templates, and you can also design new templates to suit the requirements of your system.

See Also [Using Page Templates](#)

### Genies and Super Genies

Genies are like object templates that you can place on graphics pages to help simplify the configuration of many similar devices. They group functional and graphical elements, and present device information using configurable string substitutions as placeholders for specific tags or expressions. The information is then presented during runtime.

Super Genies are genies that can be passed device-specific information at runtime. Super Genies are useful for dynamic controls, like a popup switch that can be used to control many devices.

See Also [Using Genies and Super Genies](#)

### Clustering

Clustering allows you to group independent sets of CitectSCADA's server components within a single project, allowing multiple systems to be monitored and controlled simultaneously.

The most appropriate configuration will depend on the **requirements** for the solution to be deployed and the **environment** in which it is being deployed.

Some typical clustering configurations include:

- [Standalone system](#)
- [Distributed I/O system](#)

- [Client-Server system](#)
- [Redundant server system](#)
- [Clustered control system](#)
- [Reliable sub-control system](#)
- [Load sharing system](#)

CitectSCADA's implementation of clustering allows for the flexible deployment of graphics pages that can access data from different clusters dynamically. A page can be allocated a cluster context when it is called, and any elements on that page will be assigned the same cluster, unless they have a cluster explicitly specified. See [About cluster context](#).

See Also [Typical system scenarios](#)  
[Rules of Clustering](#)

#### About cluster context

Many CitectSCADA items require a cluster to be specified in order for them to work correctly. This ClusterName can either be explicitly specified when the item is called or displayed, or an item can use the default cluster context of the calling process or page. If your system has only one cluster, then no cluster name needs to be specified. See [Cluster context rules](#).

Cluster context, therefore, is the default cluster used for tag resolution and execution of expressions and cicode functions. In the case of tags, a cluster is explicitly supplied by prefixing the tag name with the cluster name. For built-in Cicode functions ClusterName is usually an optional parameter.

Server processes (Alarm, Trend, Report) have their default cluster context set to their own cluster, so that, for example, Alarm definitions that contain variable tags without clusters explicitly supplied will attempt to resolve those tags to their own cluster. Cicode tasks started from server code will inherit the servers cluster unless one is explicitly given.

Graphics pages have no cluster context by default. A page's cluster can be supplied statically in the page appearance properties, can be inherited from a previous context, or can be applied dynamically using an optional parameter in the Cicode function that displayed the page. Any Cicode tasks started from a page will inherit the cluster context of that page.

See Also [Cluster context rules](#)

#### Cluster context rules

- 1 Single cluster systems always use that cluster and do not require the use of cluster prefixes.

- 2 Variable Tags referenced in an Alarm, Report or Trend Server context are implicitly resolved to that cluster unless explicitly stated otherwise using a cluster prefix such as *ClusterName.TagName*.
- 3 Cicode called in an Alarm, Report or Trend Server context runs with the server's cluster context as default.
- 4 Server to Server connections (for example, SPC Alarms) are within the cluster by default.
- 5 Client pages can have a cluster context associated with them.
- 6 Client pages can be configured with a cluster context or inherit the cluster from the previous (or parent) page.
- 7 Client pages can have the configured cluster context overridden dynamically at runtime.
- 8 Client pages have no default cluster and do not inherit context in multi-cluster systems by default.
- 9 The [TaskNew](#) Cicode function inherits the caller's (either page or task) cluster context unless explicitly overridden.

See Also [About cluster context](#)

## Included projects

If you have a large production environment, you can simplify the configuration and management of your CitectSCADA system by designing your project as a collection of smaller "included" projects.

Included projects can operate independently, however, they share resources and merge seamlessly during runtime. This means you can create and test projects representing functional or physical sections of a plant, and then gradually bring them online. Ongoing maintenance can then be managed with a minimal impact on production.

For more information, refer to [Including projects](#).

## Redundancy

Redundancy can be implemented at different levels of your CitectSCADA system, depending on the reliability requirements of your project. The following types of redundancy are available:

### Device Redundancy

Multiple data paths to a device can be configured within CitectSCADA. This ensures that if a fault occurs on the primary path, data can still be monitored over the secondary path.

### Server Redundancy

Primary and Standby Alarm, Report, and Trend servers can be configured so that if a primary server becomes unavailable to process a client's request, the request can be channelled to a standby server for processing.



### LAN Redundancy

To avoid an interruption due to network failure, a redundant LAN can be implemented that will provide an alternative path to a server should it be required.

See Also [Building Redundancy into Your System](#)

## Building Your Project

Once you have determined your project requirements and design, you can begin implementing the design in CitectSCADA.

The following topics will assist you to identify the project components and options that you need to implement.

### Projects

You will first need to create a new CitectSCADA project, and familiarize yourself with tasks like storing, including, and archiving it.

Before running the project, the process of compiling it will alert you to any errors in the configuration.

See Also [.Building Your CitectSCADA Project](#)  
[Administering projects](#)  
[Compiling the Project](#)

### Setting up I/O Device Communication

CitectSCADA can be configured to communicate with I/O devices from a number of different vendors. To establish communications with an I/O device, you will need to perform the following steps:

- Install the relevant device driver
- Configure the hardware and software required by the device
- Set up a test project to test the communications channel
- Configure a variable tag for each data point on the I/O device that you want to communicate with.
- Configure the device in the project, either manually or using the Communications Express Wizard.

See Also [Using the Communications Express Wizard](#)  
[Communicating with I/O Devices](#)  
[Tagging Process Variables](#)

### Graphics Components

Graphics components are the means through which operators view and interact with the runtime system. Graphics pages can be designed to provide operators at different system areas or levels with relevant monitoring and control options.

To create graphics components that meet your operational requirements, you should be familiar with how to create graphics pages, use page templates, and configure graphical objects like Genies and Super Genies.

See Also [Defining and Drawing Graphics Pages](#)  
[Using Genies and Super Genies](#)  
[Using Objects](#)  
[Defining Common Object Properties.](#)  
[Understanding Object Types](#)

## Alarms

The CitectSCADA alarm system monitors your production processes and alerts operators to critical or unexpected events that may require attention.

There are two types of alarms that you may need to configure:

- **Hardware alarms** - alert you to equipment faults
- **Configured alarms** - allow you to specify relevant alarm conditions for your facility (for example, the value of a variable tag monitoring the level, temperature, or status of a specific piece of equipment). There are seven types of configured alarms, depending on the type of alarm condition you need to set up.

To help operators process alarms, you can create graphics pages that provide alarm information (such as the action an operator must perform to correct the situation).

See Also [Configuring and Processing Alarms](#)  
[Configured alarms](#)  
[Formatting an Alarm Display](#)

## Data Collection

Data collection in CitectSCADA incorporates two main aspects:

- **Trends** - The trends system allows you to collect and monitor plant data. Depending on your requirements, data can be collected on a periodic basis, or when a specific event occurs. The data can then be saved to disk for analysis or displayed on a graph or report. To use trends in your system, you will need to be familiar with how to configure trend tags and display trend data in a graph or report.
- **Reports** - Reports provide information on the status of your plant and processes. You can configure reports with the following information to ensure that they meet your operational requirements:
  - **Period/Trigger:** Reports can be run on a request basis, periodically, or when a specific event occurs.
  - **Report format:** You can use a text editor to create a file that specifies how a report is displayed.

- **Report output:** Reports can be output to a file, device, or displayed on a graphics page.

See Also [Logging and Trending Data Reporting Information](#)

## Users and Areas

You can design security for your CitectSCADA system which incorporates both of the following features:

- **Users** - User accounts allow you to restrict access to your runtime system. All users must log in to the system with a user name and password to gain access. User accounts can be set up for individuals or for groups of users.
- **Areas** - Areas allow you to define geographical or functional boundaries in your system. You can then control both the access users have to different parts of the project, and the tasks they can perform.

See Also [Using Security](#)

## System Components

CitectSCADA includes the following system components, which provide further options for monitoring, control, and user interaction:

- **Commands and Controls** - Configurable keyboard commands and slider controls allow operators to interact with the runtime system.
- **Events** - Events (such as variable tags or expressions) can be configured that trigger a specific action, like a command.
- **Accumulators** - Accumulators track incremental runtime data. The data is stored as variable tags in an I/O device, and updated regularly while the trigger is active.
- **Statistical Process Control** - SPC allows to you to track quality by collecting and interpreting process variables associated with a product.
- **Labels** - System wide substitutions can be configured for commonly executed commands and expressions.
- **Devices** - High-level CitectSCADA data (including reports and logs) can be transferred to other system elements such as printers, databases, or files.
- **Remote Access** - A CitectSCADA project can be accessed remotely or wirelessly in the following ways:
  - **CitectSCADA Web Client** - The CitectSCADA Web Client allows you to view a live CitectSCADA project within a Web browser.
  - **CitectSCADA Pocket** - CitectSCADA Pocket provides a mobile, wireless interface to CitectSCADA. Operators can move away from the control room and continue to monitor the SCADA system.

- **Internet Display Client** - An Internet Display Client can be used to run a runtime-only version of a CitectSCADA project over the Internet from a remote location.

See Also [Defining Commands and Controls](#)  
[Configuring Events](#)  
[Using Accumulators](#)  
[Understanding Statistical Process Control](#)  
[Using Labels](#)  
[Using Devices](#)  
[CitectSCADA Web Client](#)  
[Running Your System Over the Internet](#)  
[Exchanging Data with Other Applications](#)

#### Setting up CitectSCADA as an OPC data source

CitectSCADA OPC Server allows you to access all data available in the CitectSCADA runtime environment (e.g. from PLCs and databases) through any OPC Client application (v1.0 or v2.0).

When CitectSCADA is used to monitor and control a plant, data from PLCs is collected and displayed in the CitectSCADA runtime environment. OPC Clients can access device and tag information through the interface to the OPC Server, which in turn interacts with the CtAPI interface to the CitectSCADA Runtime. For details on how to configure the OPC Server, refer to [Using OPC Server DA2.0](#)

## Deployment

Once you have built your project, you need to configure each computer in your CitectSCADA system. The configuration information is stored on each computer in a citect.ini file. The information includes:

- The role the computer has in the Citect network
- The project being run
- The CPU Configuration
- The Citect Events enabled for each component
- The Cicode run for each component on startup
- The cluster configuration
- The security settings applied

The Computer Setup Wizard displays a series of pages where you can configure these settings. The selected options are written to the citect.ini file. You should run the wizard on each computer as the final step before running your project.

See Also [Running the Computer Setup Wizard](#)





# Chapter 9: Administering projects

---

CitectSCADA is a project-based application. This section of the help looks at the administrative tasks associated with creating, storing and maintaining your projects. It includes:

- [Managing your projects](#)
- [Archiving projects](#)
- [Including projects](#)
- [Working with the Project Editor](#)
- [Using Find and Replace in a project](#)

## Managing your projects

This section of the help explains how Citect Explorer is used to manage the administration of your projects. It includes the following topics:

- [Creating a project](#)
- [Editing the properties of an existing project](#)
- [Copying projects](#)
- [Printing project details](#)
- [Deleting a project](#)
- [Linking projects](#)

### Creating a project

To create a new project:

- 1 Start Citect Explorer.
- 2 Click the **New Project** button, or from the **File** menu, select **New Project**.
- 3 Enter a **Name** for the project you wish to create [mandatory].
- 4 Enter a **Description**, and the **Location** where the new project files are stored.
- 5 Select a **Template style** and **Template resolution** to set the appearance of the graphics pages.
- 6 Click **OK** to create the project, or **Cancel**.

See Also [New Project dialog](#)

### New Project dialog

This dialog box lets you [create a new project](#). To create a new project, you must at least complete the **Name** field (the others are optional), then click **OK**.

Once created, project properties can be viewed and edited using the Project Properties dialog, which contains the items described below.

#### **Name**

A unique name for the project. The project name is restricted to 64 characters. It can contain any characters other than the semi-colon (;) or the single quote ('). Since the project name is a unique identifier, CitectSCADA does not permit you to create or restore a project with the same name.

#### **Description**

A description of the project. This field is useful for giving an explanation of the role of the project. You are urged to complete this field.

#### **Location**

The directory path where the project files are stored. As the Name field is entered, the directory is automatically generated in the Location field. You can override this by manually entering the location or clicking **Browse**.

#### **[Page defaults] Template style**

The style (appearance) of the graphics pages in the runtime system. The style you select is the default style for any new pages you add to the project. You can change the style of existing pages and templates using the Page Properties, accessed through the Graphics Builder.

Most users prefer the Standard style. You can view the pre-defined styles by looking in the Include project under Graphics, Templates.

#### **[Page defaults] Template resolution**

The default screen resolution of the standard graphics pages (such as alarms pages and standard trend pages):

Screen Type	Screen Width (pixels)	Screen Height (pixels)
VGA	640	480
SVGA	800	600
XGA	1024	768
SXGA	1280	1024
User	****	****

#### **[Page defaults] Show template title bar**

Determines whether to display the Windows title bar (at the top of each [graphics page](#)). The title bar contains the title of the window, maximize, minimize and



close buttons (at the right hand end of the title bar), and the control menu button (at the left hand end of the title bar).

To display a page in full screen (without a title bar), the size of the page must be the same size as the display (or larger). If the page is smaller than the display, the title bar still displays, even if full screen mode is enabled. Standard templates styles are available for both page sizes.

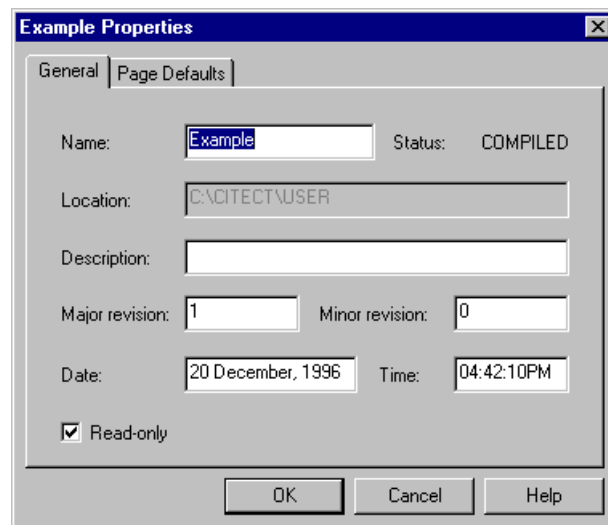
### [Page defaults] Background color

The background color that will be displayed in all newly created graphics pages.

## Editing the properties of an existing project

To edit the properties of an existing project:

- 1 Open Citect Explorer.
- 2 Select a project from the list.
- 3 Click the **Properties** button, or select **Project Properties** from the **File** menu.
- 4 Edit the properties in the Project Properties dialog. Click **Help** for more information about the fields.
- 5 Click **OK** to save your changes, or **Cancel** to abort.



See Also [Project Properties dialog](#)

Project Properties dialog

Use this dialog to [edit the properties of an existing project](#). Projects have **General** properties and **Page** properties.

**(General) Name**

The name of the project. This name is identical to the name that was used when the project was created. The project name is restricted to 64 characters. It can contain any characters other than the semi-colon (;) or single quote ('). Since the project name is a unique identifier, CitectSCADA will not permit you to create or restore a project with the same name.

#### **(General) Status**

The status of the project. This can be either **COMPILED** or **UNCOMPILED**.

#### **(General) Location**

The directory path where the project files are stored. This field cannot be edited.

#### **(General) Description**

A description of the project. This field is useful for giving an explanation of the role of the project. You are urged to complete this field.

#### **(General) Major revision**

CitectSCADA sets this property to one (1) when the project is first created. You can use this field to track major changes to the project. You can use an incremental revision history (e.g. 1, 2, 3, . . . or A, B, C, . . .) or the name of the person responsible for the last major revision.

#### **(General) Minor revision**

CitectSCADA sets this property to zero (0) when the project is first created. You can use this field in conjunction with the Major Revision to track your project's development.

#### **(General) Date and Time**

CitectSCADA will initially set these fields to the date and times at when the project was created. These fields are useful when used in conjunction with the Revision fields.

#### **(General) Project ID**

A unique number for the project. The project number can be between 1 and 1022.

If you enter an ID that has already been used for another project, CitectSCADA will detect this when it compiles the project.

The project number is part of the unique identifier (OID: Object ID) used by OPC drivers when reading from and writing to tags.

If you do not specify a project number, CitectSCADA will automatically generate one the next time you select this project in the Citect Explorer, or the next time you compile.

**Note:** If you enter 0, your project ID is automatically set the next time you compile.

### (General) Read-only

Specifies that no changes can be made to the project. If an attempt is made to modify the CitectSCADA project with this option selected, a message will prompt the user to disable the option before continuing.

**Note:** If you change any properties, you must click **OK** to save the changes to the project.

### (Page Defaults) [Template] Resolution

The default screen resolution of the standard graphics pages (such as alarms pages and standard trend pages):

Screen Type	Screen Width (pixels)	Screen Height (pixels)
VGA	640	480
SVGA	800	600
XGA	1024	768
SXGA	1280	1024
User	****	****

**Note:** You can override this default for your own pages at the time when you create them or any time afterward.

### (Page Defaults) [Template] Style

The style (appearance) of the graphics pages in the runtime system. The style you select is the default style for any new pages you add to the project. You can change the style of existing pages and templates using the Page Properties, accessed through the Graphics Builder.

Most users prefer the Standard style. You can view the pre-defined styles by looking in the Include project under **Graphics | Templates**.

**Note:** You can override this default for your own pages at the time when you create them, or any time afterward.

### (Page Defaults) [Template] Show title bar

Determines whether the Windows title bar displays (at the top of each graphics page). The title bar contains the title of the window, maximize, minimize and close buttons (at the right hand end of the title bar), and the control menu button (at the left hand end of the title bar).

To display a page in full screen (without a title bar), the size of the page must be the same size as the display (or larger). If the page is smaller than the display, the title bar still displays, even if full screen mode is enabled. Standard templates styles are available for both page sizes.

**Note:** You can override this default for your own pages at the time when you create them, or any time afterward.

## Copying projects

### (Page Defaults) Background color

The color that will display in the background of all new graphics pages.

You can copy the contents of one project into an existing or a new project.

#### To copy a project:

- 1 Open Citect Explorer.
- 2 Select the **Copy** icon, or select **Copy Project To** from the **File** menu.
- 3 In the Copy Project dialog box, select the source project from the drop-down list under **Project name**.
- 4 Select an existing destination project to copy to or select a new project.
- 5 Click **OK** to copy the project, or click **Cancel**.

See Also [Copy Project dialog](#)

#### Copy Project dialog

This dialog box lets you [copy](#) all the contents from one project into another. To copy a project, specify the source [From] and destination [To] projects, then click **OK**.

#### [From] Project name

The name of the source project being copied. If more than one project exists, you can choose a project name from the drop-down list.

#### [To] (Existing or New) project

You can copy to either an Existing or a New project name and location.

- **Existing Project:** The source project is written over (replaces) an existing project location under an existing project name.
- **New Project:** The source project is copied to the new location under a new project name. A new project must be given a new name not currently being used, and which complies with the naming requirements as detailed below.

#### [To] Name

The name of the destination project being copied to.

When copying to an existing project, you must choose a project name from the existing project names drop-down list.

When copying to a new project, you must create a new and unique name for the project. The project name is restricted to 64 characters, and can contain any characters other than the semi-colon (;) or single quote ('). Since the project name is a unique identifier, CitectSCADA will not permit you to create or copy to a project with an existing same name.

After the new project is created, you can change the Name through the Project Properties.

When copying to an existing project location, you can choose to delete the existing contents of the destination project, including subdirectories, before the source project is copied, by checking both the *Clear location before copying*, and the *Clear subdirectories* check boxes. This ensures that no residual files are left behind to interfere with the copied project. If you do not clear the project location before copying, only common files in the destination project are overwritten.

#### [To] Clear location before copying

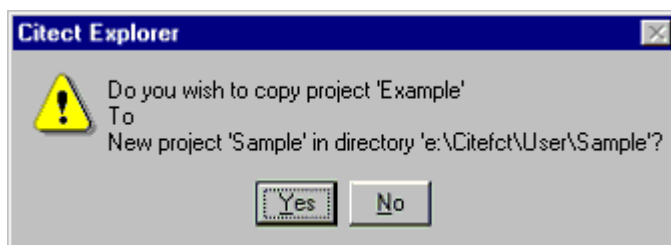
Specifies to delete the contents of the existing destination project before copying the source project to the destination location. This ensures that no residual files are left behind to interfere with the copied project.

#### [To] Clear subdirectories

Specifies to delete the contents of all the sub directories of the existing destination project before copying the source project to the destination location. This ensures that no residual files are left behind to interfere with the copied project.

#### Location

The directory path where the destination project files are stored. As the *Name* field is entered, the directory is automatically generated in the *Location* field. You might override this by manually entering the location or clicking **Browse**.



Check that the project names and location are correct. Click **Yes** to copy the project, or **No** to cancel.

## Printing project details

You can print configuration elements (database records, pages, [Cicode](#) files, etc.) in the current project. CitectSCADA prints to the Windows default printer.

#### To print project database details:

- 1 Open the Citect Project Editor
- 1 Select **Print** from the **File** menu.
- 2 Use the **Print selection list** to choose the elements you want to print.
- 3 Click **OK** to start printing, or **Cancel** to abort.

Before printing your database, print a small portion to test the results. You can change the default font, font size, and page size by choosing **Options** from the **Tools** menu. For other print options, refer to your Windows documentation.

See Also [Print \(project details\) dialog](#)

#### Print (project details) dialog

This dialog box allows you to [print](#) the configuration elements (database records, graphic pages, Cicode files, etc.) in the current project. Click **OK** to print the selection, or **Cancel** to abort printing.

#### [Print selection]

Lists all the elements in the project that can be printed. To select (or deselect) an element for printing, click the check box; a checkmark indicates it will be printed.

Click **Select All** to select every item in the list, or **Deselect All** to clear all your selections.

#### [Options] Graphics pages included in print selection

Specifies a particular page to print. Use the drop-down list to select a single page from the project. Choose the **<All pages>** entry to print all of the pages in the project.

#### [Options] Group printouts by graphics page

Print the objects database information with the related page. If this option is not set, then the objects database information is printed as continuous lists, with just a page reference.

You can only print the contents of the current project. Included projects will not be printed. You can specify the print font, font size, and page size in the **Options** for the Project Editor (in the **Tools** menu).

## Deleting a project

### To delete an existing project:

- 1 Open Citect Explorer.
- 2 Select a project from the list.
- 3 From the **File** menu, select **Delete Project**.
- 4 A message box asks you if you want to proceed. Click **Yes** to delete the project, or click **No** to cancel.

You cannot delete a project that is currently open or any installed project. You also cannot delete the Include project that is supplied with CitectSCADA.

**Note:** You cannot recover a deleted project that hasn't been backed up.

See Also [Linking projects](#)

## Linking projects

CitectSCADA installations on different computers over the same network can share the same project. After a project has been created on one computer, other computers on the same network can [link](#) to the same project, but only if the project location is on a shared or network drive. Once linked, the remote project is visible in the local Citect Explorer, and can be edited and compiled over the network. Only one version of a project ever exists, and is always kept on the computer it was created upon.

**Note:** Linking to a project provides the developer with full access and control to the project, even though it might be on a remote machine over the network. Be warned that it is possible to delete a linked project, even though it might be on a remote machine over the network. You should unlink a project rather than delete it over the network.

Linked projects will not be included into the compile of any other project unless they have specifically been Included into that project from within Project Editor. For details, see [Including projects](#).

### To link to a project:

- 1 Open the Citect Explorer.
- 2 Click the **Add Link** button, or select **Add Project Link** from the **File** menu.
- 3 Use the Select Project Directory dialog to choose a project location.
- 4 Click **OK** to link the project, or click **Cancel**.

If the new project has the same name as an existing one, you are prompted to change it before proceeding. Edit the properties in the Project Properties dialog.

### To remove a link to a project:

- 1 Open the Citect Explorer.
- 2 Select a project from the list.
- 3 Click the **Remove Link** button, or select **Remove Project Link** from the **File** menu.
- 4 You are prompted if you want to proceed. Click **Yes** to remove the link, or **No** to cancel.

See Also [Including projects](#)

## Archiving projects

Once you have configured your CitectSCADA system, you should back up (or archive) the project. This will avoid the loss of any configuration data in the case of a critical hardware failure.

**Note:** When you are developing a project, you should adopt a regular backup strategy. Before performing a backup, ensure that you have refreshed any linked tags in your project.

CitectSCADA lets you back up a project to a local drive (hard drive), network location, or removable media (floppy drive, memory stick).

This section of the help includes information on the following archiving tasks:

- [Backing up a project](#)
- [Backing up INI files](#)
- [Configuring a backup with password encryption](#)
- [Running a backup from the command line](#)
- [Restoring a project](#)

## Backing up a project

The CitectSCADA Backup program archives files using a standard compression routine, producing PKZip® v2.04g compatible files. The default extension for CitectSCADA backup files is .CTZ, though any extension (including .ZIP) can be used. This means you can also use the PKZip utility to extract files from a compressed CitectSCADA backup.

**Note:** Files produced with this backup program cannot be restored by CitectSCADA versions earlier than 5.10.

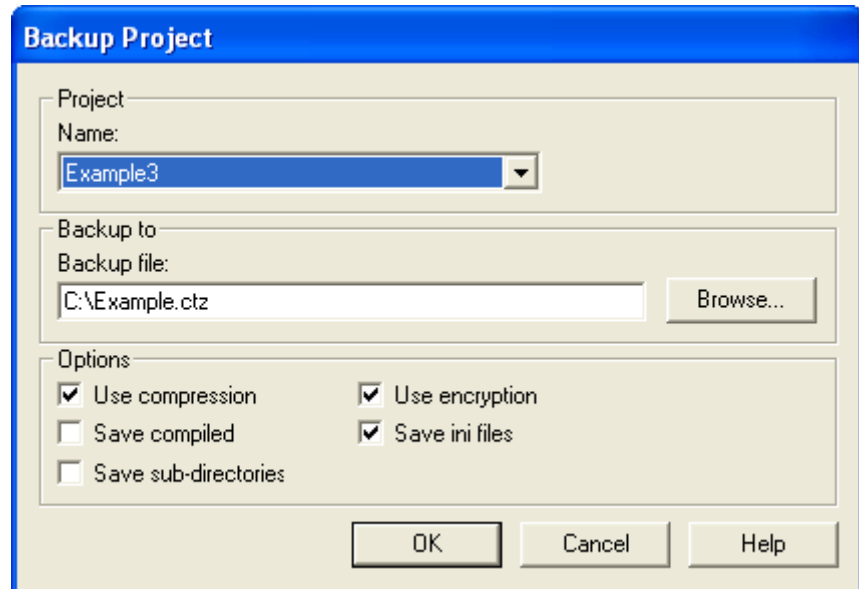
**To back up a project:**

- 1 Open Citect Explorer.
- 2 Click the **Backup** button,



or select **Tools | Backup**. The Backup Project dialog box displays:





- 3 In the **Name** field, select the name of the project to back up.
- 4 In the **Backup file** field, enter the path to the backup file location, including the file name. You can either type the path in directly or use the **Browse** button.

The backup file name defaults to <project>.CTZ. If the extension is omitted then .CTZ is used.

When you back up a project to a floppy disk, the backup program will ask you if you wish to delete the files on the floppy disk before starting the backup.

If the destination drive is configured as A: or B: and is detected as removable, you will have the option to delete any existing files on the disk.

- 5 Under **Options**, select the required options from the following list:
  - **Use compression:** You can use data compression when you are backing up a project to save space.
  - **Save compiled:** By default, CitectSCADA backs up the project in **uncompiled** mode. If you select this option, CitectSCADA backs up both the **compiled** and **uncompiled** projects, resulting in a larger backup file.
  - **Save sub-directories:** If you select this option, CitectSCADA also backs up all data in any sub-directories within the project directory. The directory structure is maintained in the backup, and you can choose to restore the sub-directories when restoring the project. For example, if

you wish to back up your Process Analyst Views, save them in a sub-directory of the project and select this option. When you restore the project, you will have the option to also restore the Process Analyst Views directory.

- **Use encryption:** As an added security measure, you can back up your project in an encrypted format. If you select this option, CitectSCADA requests a password. CitectSCADA writes the project to disk in a format that encodes the password to ensure that the project is protected. The project can only be restored if the password is entered.
- **Save ini files:** Select this option to back up citect.ini and CtTimeScheduler.ini from the WINDOWS directory.

**Note:** These two files are the only files backed up when you select this option. If you wish to back up any other files, save them in a sub-directory of the project and select the 'Save sub-directories' option.

6 Click **OK**.

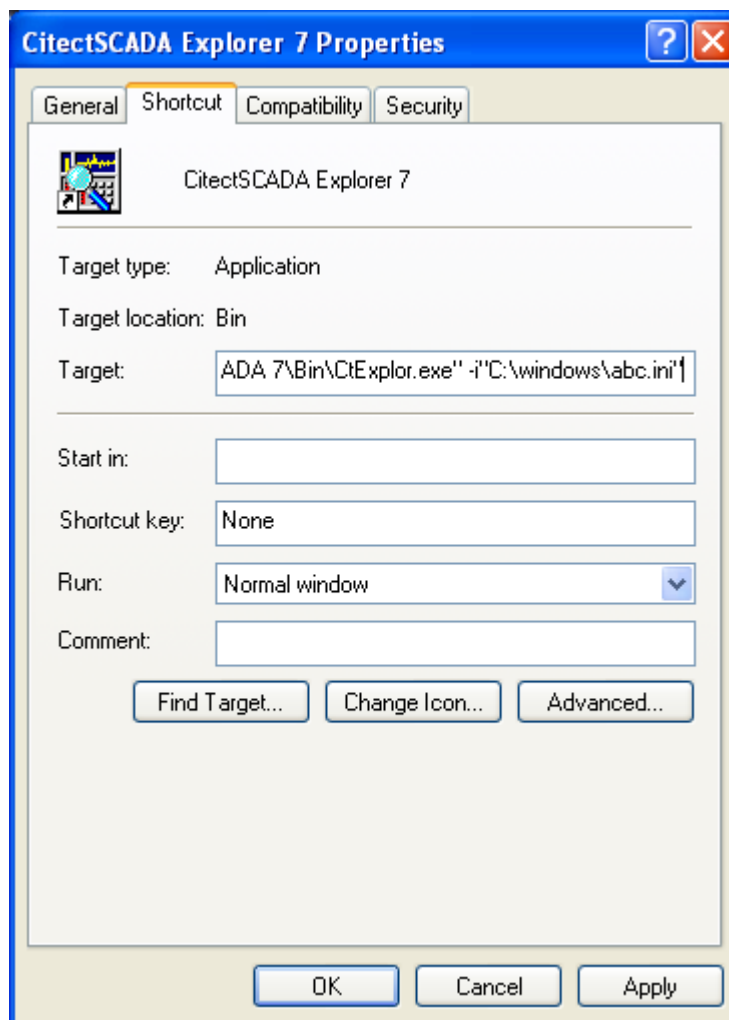
See Also [Backing up INI files](#)

## Backing up INI files

By default, when you select the 'Save ini files' option, citect.ini and ctTimeScheduler.ini are backed up from the WINDOWS directory.

If you are using a custom INI file (for example 'abc.ini'), you should make sure that it is saved in a sub-directory of the project. By selecting the 'Save sub-directories' option, your INI file will also be backed up.

**Note:** You can define a non-default INI file for CitectSCADA by passing a parameter through to Project Explorer (as shown below). If this is the case, the defined file will be backed up when you select the 'Save ini files' option; citect.ini will not be backed up.



If you run the backup program from the command line, and you specify an INI file as a parameter, the specified INI file will be backed up instead of citect.ini.

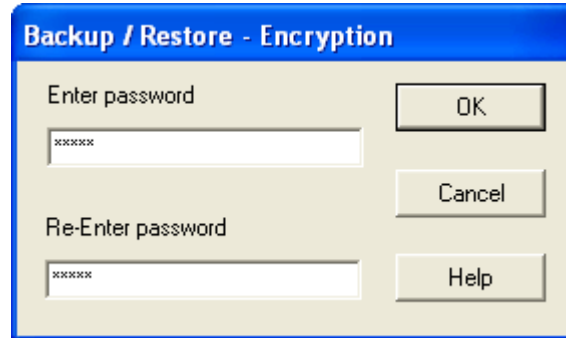
See Also [Configuring a backup with password encryption](#)

### Configuring a backup with password encryption

When you select the 'Use Encryption' backup option, CitectSCADA writes the project to disk in a format that encodes the password. The project can only be restored when the password is entered.

#### To use encryption:

- 1 Select the 'Use encryption' option on the Backup Project dialog box.
- 2 Click **OK**. The Backup/Restore-Encryption dialog displays:



- 3 In the **Enter Password** field, enter your password. Asterisks will display in place of the characters.
- 4 In the **Re-Enter Password** field, re-enter your password. CitectSCADA checks that you have typed the same password both times.
- 5 Click **OK**. The project will be backed up.

See Also

[Running a backup from the command line](#)

## Running a backup from the command line

You can execute the CitectSCADA backup program from the command line to back up and restore files other than CitectSCADA projects.

From version 5, the backup program is called CtBack32.exe. For older versions, it is called CtBackup.exe. By default, it is installed in the CitectSCADA project 'Bin' folder.

The CitectSCADA Backup program archives files using a standard compression routine, producing PKZip v2.04g compatible files. The default extension for CitectSCADA backup files is .CTZ, though any extension (including .ZIP) can be used. This means you can also use PKZip to extract files from a compressed CitectSCADA backup if you prefer.

When you execute a backup from the command line, if you specify an INI file as a parameter, it will be backed up instead of the citect.ini file in the WINDOWS directory.

The backup program reads the [citect.ini file](#) for any parameters set using the [BACKUP] category. These settings (if any, and their defaults if not) are over-ridden by any values passed as command line options.

The table below describes the backup command line options.

Option	Description
-d<name>	database name
-m<ext>	include extension
-x<ext>	exclude extension

Option	Description
-e	encrypt with password
-p<password>	encrypt/decrypt password
-s[+/-]	recurse subdirectories
-f<level>	format level, 0 only format if required, 2 always format disk. [obsolete since version 3.xx, 4.xx]
-u[+/-]	save uncompiled, use -u- to save compiled
-g[+/-]	show configure dialog
-c[+/-]	compress files
-b<path>	path to backup from
-r<path>	path to restore to
-i<filename>	ini file name
-f1	use old file format (truncates long filenames to 8.3). New for v5.10
-a	run in auto mode
	(NOTE: All required input must be in command line or INI file.)

### Examples

- To back up (in version 3) c:\data:use the following command:

```
CTBACKUP -g- -bc:\data
```

- To restore the above data use (in version 5):

```
CTBACK32 -g -rc:\data
```

- To backup a CitectSCADA database, eg backup demo use:

```
CTBACK32 -dDEMO -b -u- -c+ -d-
```

Ctbackup also uses the following parameters in the CITECT.INI file:

```
[BACKUP]
Database= ! database to backup or restore
BackupPath= ! file to backup to, for example c:\temp\example.ctz.
New for v5.10.
DrivePath= ! path to backup to or restore from.
[obsolete as of v5.10, use BackupPath instead]
FilePath= ! file path, used in not a database
BackupFile= ! file name on backup disk, default CTBACKUP.
[obsolete as of v5.10, use BackupPath instead]
Password= ! encryption password
Drive=0/1/2 ! 0=other, 1=A, 2=B
DiskSize=0/1 ! low density=0, high density=1
Encrypt=0/1 ! encrypt backup
FormatLevel= ! format level.
[obsolete since version 3.xx, 4.xx]
Configure=0/1 ! display configure dialog
Compress=0/1 ! compress backup
Overwrite=0/1 ! overwrite
SaveCompiled=0/1 ! save compiled
```

```
Recurse=0/1 ! recurse sub directories
DeleteAll=0/1 ! delete all before restore
SaveIniFiles=0/1! determines whether save ini files is checked
Operation=0/1 ! 0=backup, 1=restore
Include= ! include list
Exclude= ! exclude list, default DBK,_CI
CompiledFiles= ! compiled files, default RDB
FileFormat=0/1 ! 1= use old format (truncates long filenames to
8.3). New for v5.10.
```

## Restoring a project

You can restore backed up and archived projects using the Restore Project program. This program allows you to overwrite any current project with a backed up version, or restore a backed up project as a new project.

**Note:** Be careful when restoring files as all files in the destination and sub-directories will be deleted before restoring. If you accidentally set your restore path to the root directory of the drive, the program will delete your entire disk drive.

### To restore a project:

- 1 Open Citect Explorer.
- 2 Click the **Restore** button



or select **Tools | Restore**. The Restore Project dialog box will display:

- 3 In the **Backup file** field, enter the name of the project to restore.
- 4 Under **To**, select 'Current project', to overwrite a project with the backed up one, or 'New project' to restore a backed up project as a new one.
- 5 In the **Name** field, enter a name for the restored project.
- 6 In the **Location** field, enter the location of the project to restore, including the file name. You can either type in the path directly, or use the **Browse** button.
- 7 Under **Options**, select 'Ini files' to restore backed up INI files.
- 8 If you backed up the sub-directories under the project, the directories will be listed under 'Select sub-directories to restore'. You can choose to restore all or no sub-directories, or you can select specific sub-directories to restore.
- 9 Click **OK**.

See Also [Archiving projects](#)

## Including projects

With large systems, it might be more convenient to develop the application using a series of smaller projects, instead of one large project. For example, you could use a separate project for each section of the plant, or for each main process. This way, you can develop and test each of the smaller projects before including them in the main project.

CitectSCADA projects will not be included into the compile of any other project unless they have specifically been included into that project from within the Citect Project Editor.

**Note:** If a CitectSCADA project exists remotely on the same network as the local CitectSCADA installation and it is on a shared or network drive, it can be linked to the local Citect Explorer. This is different to including a project. Linking makes a project visible in the local Citect Explorer. Once linked, it can be selected as the current project for editing over the network.

Any linked project (visible in Citect Explorer) can be included within a local CitectSCADA project, and is subsequently included in the compile of the local Project.

Be careful not to confuse include files with included projects:

- **Include Files** contain CitectSCADA commands and/or expressions and are used as substitutions in a CitectSCADA command or [expression](#) property field.
- **Included Projects** are separate and (usually smaller) CitectSCADA projects that can be included in another CitectSCADA project so that they appear together as one project.

Each CitectSCADA system is supplied with a number of include project. These project contains pre-defined database records.

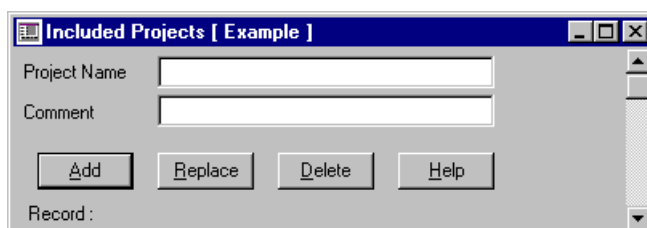
### Including a project in the current project

#### To include another project (in the current project):

- 1 Open the Citect Explorer.
- 2 Select a Project from the list.
- 3 Select the **System** icon and then **Included Projects**.
- 4 Complete the Included Projects form that appears.



- 5 Click **Add** to append a record you have created, or **Replace** if you have modified a record.



**Note:** Do not define circular references. That is, if project A includes project B, do not include project A in project B. Instead, create another project and include both A and B into this.

See Also [Included Projects dialog](#)  
[CitectSCADA's included projects](#)

#### Included Projects dialog

This dialog box lets you [include another project](#) in the current CitectSCADA project. With large systems, you should develop the application using a series of smaller projects instead of one large project.

You can include up to 240 projects. (You have to set [CtEdit]DBFiles to 310 in order to enable this limit.) All records in each project are globally accessible (i.e., a record defined in one project can be used in another).

#### Project Name

The name of the project to include in this project.

#### Comment

Any useful comment.

**Note:** Each CitectSCADA system automatically has an include project, which contains predefined database records and graphics libraries.

#### CitectSCADA's included projects

Each CitectSCADA installation is supplied with three predefined include projects, designed to help you develop your CitectSCADA project faster. They are:

- the **Include** project - a template project with trending and alarm pages.
- the **CSV\_Include** project - a Windows XP-styled set of templates with common toolbars and advanced visualization tools.
- the **CSV\_Instant Trend** project - created to support the CSV\_Include project's instant trending feature.

These projects contain pre-defined database records and graphics libraries that can be used as the foundation for the content within your own project.

**Note:** Do not modify the include project for use as a runtime project. It will not compile successfully, and should be set aside for use as a template for new projects. CitectSCADA upgrades install a new version of the CSV\_Include project, so you will lose changes you make to the project when this happens.

The include projects are hidden from the project tree in Citect Explorer by default.

**To show/hide the CitectSCADA Include project:**

- 1 Open the Citect Explorer.
- 2 Select **Show Include Project** from the **View** menu.

See Also [Introducing CSV\\_Include](#)

## Working with the Project Editor

The Project Editor is the primary tool used to configure the variable addressing, communications and system components of a CitectSCADA project. This section of the help looks at the components incorporated into the Project Editor to support this process.

- [Setting the Project Editor options](#)
- [Paste Tag dialog box](#)
- [Paste Function dialog box](#)
- [Find User Function dialog box](#)
- [Using Find and Replace in a project](#)

### Setting the Project Editor options

The Project Editor offers the option to change the way the project configuration environment operates.

**To set the Project Editor options:**

- 1 Launch the Project Editor
- 2 Select **Options** from the **Tools** menu.
- 3 Adjust the required option adjustments.
- 4 Click **OK**.

See Also [Project Editor Options dialog](#)

Project Editor Options dialog

This dialog box allows you to adjust the functionality of the Project Editor.

#### Show deleted

Enables the display of deleted records in the databases. When enabled, a check box at the bottom of the database form indicates if a record is deleted.

**Incremental compile**

Enables the incremental compilation of the project.

**Extended forms**

Enables the display of extended database forms. You can also use the F2 key on the keyboard to display extended forms.

**Inform record changed**

Enables the "Record has been changed" message window to appear when you add (or change) data in a database form and then try to close the form - before you add or replace the record.

**Note:** If you disable this option, you will lose data if you change a database record and forget to add or replace the record.

**Disable user functions search**

When you use a combo box to select a function (for a command or [expression](#) field), a list of in-built Cicode functions and user-written functions displays. If you disable user functions, only the in-built functions are displayed in the list.

**Confirm on project packing**

Enables the "Packing databases may take a long time" message window to appear when packing a database.

**Auto open error form**

Automatically displays the Compile Errors form if an error occurs when the project is compiled.

**Compile enquiry message**

Enables the "Do you want to compile?" message window to appear when the project has been modified and Run is selected from the File menu. Normally, CitectSCADA compiles the project automatically (if the project has been modified) when Run is selected.

**Compile successful message**

Enables the "Compilation Successful" message window to appear when the project has been compiled.

**Prompt on tag not exist**

Enables the "Variable tag not found. Do you wish to create this tag?" message window to appear when a variable tag is specified that does not exist in the database. With the message window enabled, you can create new variable tags as they are required.

**Prepare for Web deployment**

Automatically runs the Web Deployment Preparation tool every time you compile a project. Note that this dramatically increases the amount of time taken for each compile, particularly for large projects.

**Log deprecated warnings during compile**

If you select this option, the compiler will generate a warning to identify any deprecated elements it detects in a project, i.e. any functions, parameters, or Kernel commands that are no longer supported.

By unchecking this option, the warnings are still included in the displayed warning count, but they are not added to the error log.

**Info popup time**

The delay (in seconds) from the beginning of a database search until a search information window displays. The search information window displays the number of the traced records and allows you to cancel the search. You can cancel the search by selecting the Cancel button in the information window.

**Cicode Editor**

The text editor that is used for editing Cicode function libraries and report format files. You must enter the name of the executable file in this field. The default editor is the Cicode Editor (ctcicode.exe) supplied with CitectSCADA.

**Report Editor**

The editor that is used for editing Report Format Files. You must enter the name of the executable file in this field. The default editor is Write (write.exe). If you are using Rich Text Format (RTF) reports, make sure your editor is RTF capable.

**Print page size**

The number of lines (1 to 66) printed on each page when printing database records.

**Print font point**

The font size used when printing database records.

**Print font name**

The name of the font used when printing database records.

**Maximum list box items**

The maximum number of records that are displayed in drop-down combo boxes.

**Warn about unused tags during full compile**

Enables the generation of warning entries for unused tags that are not used directly in a CitectSCADA project. The warning entries are included in the

Project Editor's Compile Errors form when a full compile is run. By default this option is not selected.

**Note:** Warning entries are generated only for a full compile, not an incremental compile.

#### Log "tag not defined" warnings during compile

If you select this option, the compiler will generate a 'tag not defined' warning in the error log for any tags detected that are not defined in the variable database.

As CitectSCADA V7.0 now allows you to include undefined tags on your graphic pages, this warning may be redundant and impractical. By unchecking this option, the warnings are still included in the displayed warning count, but they are not added to the error log.

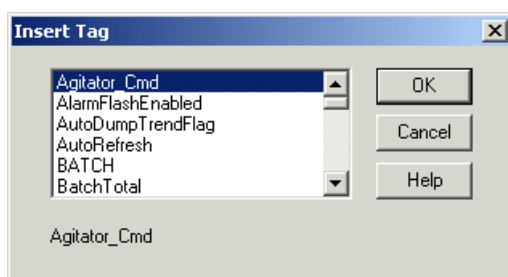
### Paste Tag dialog box

If you need to insert a variable tag into a tag or expression field, you can use the Paste Tag dialog box.

#### To insert a variable tag into a tag or expression field:

- 1 Select the location you wish to insert a tag in to such as an expression field in a form.
- 2 Select **Paste Tag** from the **Edit** menu to display the Insert Tag dialog box.
- 3 Select the tag name, and click **OK** or click **Cancel**.

The tag will be inserted in the tag or expression field at the location of the cursor.



### Paste Function dialog box

If you need to insert a function into a tag or expression field, you can use the Insert Function dialog box.

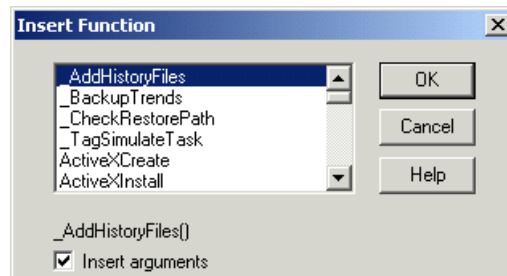
#### To insert a function into a tag or expression field:

- 1 Select the location you wish to insert a function in to such as an expression field in a form.
- 2 Select **Paste Function** from the **Edit** menu to display the Insert Function dialog box.
- 3 Select the function name, and click **OK** or click **Cancel**.

To insert the function with its [arguments](#) included, select the **Insert arguments** box.

The function is inserted in the current field at the location of the cursor.

**Note:** If the total length of the function and its parameters is greater than 254 characters, it won't appear in this dialog box. Instead, the message "Text Too Big" appears.



## Find User Function dialog box

If you need to locate a [Cicode](#) function in your Cicode source files, you can use the Find User Function dialog box.

**To locate a function within a Cicode source file:**

- 1 Select **Find User Function** from the **Edit** menu.
- 2 Enter the function name (or part of the function name) and click **OK** or click **Cancel**.

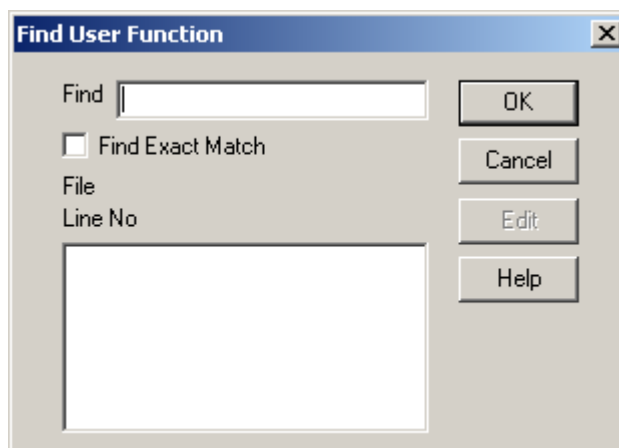
A list of functions that match your search criteria will be displayed.

**Note:** TIP: If you leave the Find field empty and click OK, a full list of functions appear in the list.

**To edit the Cicode file that contains the function:**

- 1 Select the function name from the list that appears when searching for the function (see above) and click **Edit** or click **Cancel**.

The file containing the selected function will be opened by the Cicode Editor.



## Using Find and Replace in a project

You can use the Find and Replace dialog to locate specified text in your projects. You can perform global text replaces in your CitectSCADA projects, as well as export search results.

This is explained in the following topics:

- [The Find and Replace dialog](#)
- [Specifying search coverage](#)
- [Using the results list](#)
- [Removing results](#)
- [Exporting results](#)
- [Jumping to a result \(Go To\)](#)
- [Replacing results](#)
- [Find and Replace error messages](#)

There is also a topic on [Troubleshooting Searches](#) to help determine if a search has been correctly configured when unexpected results are returned.

### The Find and Replace dialog

You open the Find and Replace dialog box from either the:

- **Project Editor:** You can find and replace text strings in your projects and included projects.
- **Graphics Builder:** You can find and replace text within a single graphics page, template, or Genie.

You can configure your search coverage, view your results, replace results, or open a search result for more information.

#### To display the Find and Replace dialog box:

- From the Project Editor or Graphics Builder, click **Edit | Find** or **Edit | Replace**. The dialog box appears with either the **Find** tab or **Replace** tab selected, depending on which command you selected.

#### To search text:

- 1 On the **Edit** menu in the Project Editor or Graphics Builder, click **Find**.
- 2 In the **Find** box, type the text string you want to search for. The search is not case-sensitive, so it doesn't matter whether you enter lower- or uppercase letters.

You can enter an entire string or a portion of the string you want to find. For example, typing BIT will return any string containing BIT, such as BIT\_1, BITE, HABIT, HABITS, and so on. You cannot enter wildcard characters, but you can include special characters, as well as spaces if you want.

- 3 Specify your [search coverage](#) using the **Look in** and **Search options** lists.
- 4 Click **Find**. Search results appear in the [results list](#) when the search completes. The status text under the results list indicates the progress of the search.

**Note:** When you start a search, the **Find** button changes to a **Stop** button you can use to exit the search. If you stop a search, a partial list of the results is displayed.

#### To replace text:

- 1 On the Edit menu in the Project Editor or Graphics Builder, click **Replace**.
- 2 In the **Find** box, type the text you want to search for.
- 3 In the **Replace with** box, enter the replacement text.
- 4 Specify your [search coverage](#).
- 5 Click **Find**.
- 6 View the [search results](#).
- 7 [Make your replacements](#) using **Replace** or **Replace All**.

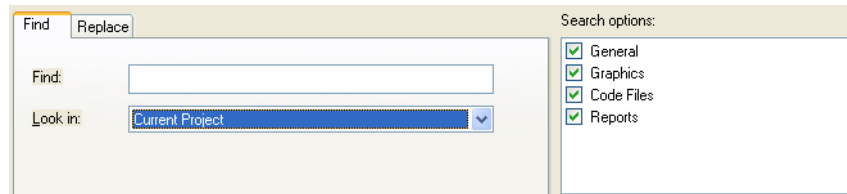
### Specifying search coverage

You specify search coverage using the **Look in** and **Search options** lists to determine which items in your projects you want to search for.

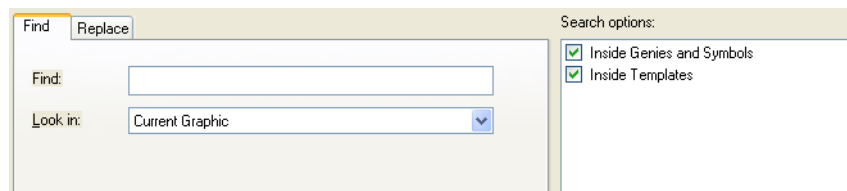
When you choose a **Look in** option, the **Search options** change. Each time you select a **Look in** option, all associated **Search conditions** are selected by default. The **Look in** options work with the **Search options** like this.



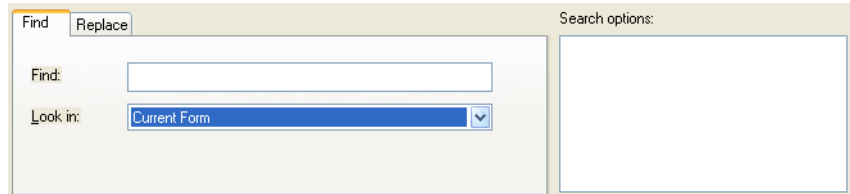
- Selecting **Current Project** or **Current Project and Include Projects** makes available the options described below.



- **General:** Searches all configuration databases associated with a project as well as included projects (if that option is selected).
- **Graphics:** Performs an “express search” for graphics pages only (the graphics page does not have to be currently open). *This search will not find text in symbols, Genies, or templates if they have not been used on a page.* If you don’t find the text you want, try the more comprehensive graphics search by opening a page, Genie, symbol, or template and selecting **Current Graphic** as the **Look in** option (see below).
- **Code Files:** Searches all Cicode/CitectVBA files in the current project (and included projects, if that option is selected).
- **Reports:** Searches all report files within the project folder and included projects (if that option is selected).
- Selecting **Current Graphic** makes available the options described below. (This search is a more extensive search than that performed by the **Current Project:Graphics** search described above, and will search graphics documents that are currently open.)

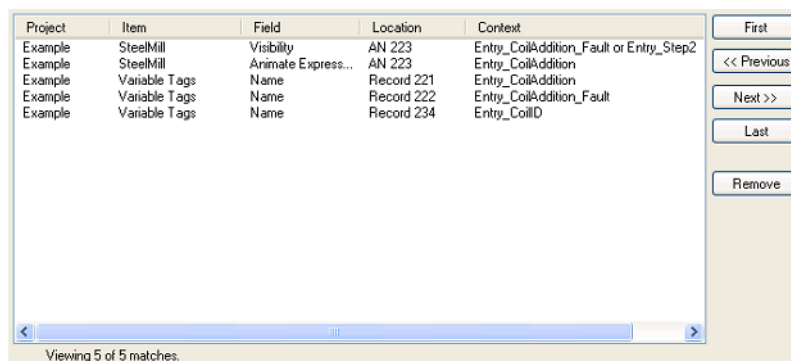


- **Inside Genies and Symbols** includes graphics objects contained within a Genie or symbol.
- **Inside Templates** includes objects contained within a page template.
- Selecting **Current Form:** Searches the current form.



## Using the results list

As matches are found they are listed the results list. The results list shows an overview of items that match the string entered in the search.



The results list can display a maximum of 200 results per page, sorted by project and then item (you cannot change the sort order). The results list contains the following columns:

Column	Description
<b>Project</b>	The name of the project in which the found text occurs.
<b>Item</b>	Depends on the type of document in which the item occurs. If the document type is a: <b>Database</b> - User-friendly name of the database. <b>Page</b> - Name of the page. <b>Cicode/VBA</b> - Name and path of the Cicode/VBA file. <b>Report</b> - Name of the report.
<b>Field</b>	Identifies that portion of the document/database in which the found item occurs in. For example, if the found item appears in a database, this refers to the column name in the database. Note that the search covers both expression/command as well as numeric properties.
<b>Location</b>	Shows the specific record number, AN, or line number on which the found item occurs within the document/database.
<b>Context</b>	An example of the context in which the found item occurs within the project. For example, if the document type is a: <b>Database</b> - a search result of BIT* might have a context of BIT_!. <b>Page</b> - BIT* might have a context of Toggle(BIT_!) <b>Cicode/VBA</b> - UserName might have a context of FUNCTION GetUserName() <b>Report</b> - PUMP* might have a context of @(Pump A)

If the number of results returned exceeds 200 items, use the **First**, **Previous**, **Next**, and **Last** buttons to navigate your results in groups of 200 results.

You can toggle between the Find and Replace functionality without losing the search results, but if you close the Results page, your search results are lost.

**Hint:** You can resize list columns by moving your mouse cursor onto the separator between the list columns. When the mouse cursor changes shape to a black bar with arrows, drag the column to the new size. You can also double-click the vertical bar between fields to resize that field to fit the widest item.

## Removing results

You can remove a search result from the Results window. Results that are removed are not included in exports or in replacement operations. Removing a result does not delete it, but merely removes it from the Results window.

### To remove a result:

- With the result you want to remove highlighted, click **Remove**. The result is removed from the Results window.

## Exporting results

You can export search results in a tab-delimited format to a specified location. Results are exported in the format

```
<Project> <Item> <Field> <Location> <Context>
```

If the Results window contains more than 200 results, all results are exported, not just the ones currently displayed. If you remove an item from the results list, it will not be exported. (For details on removing results, see [Removing results](#).)

If you export an item that has a context, the context string is tripped of tabs and new line characters.

Results exported are in Unicode format. Because of this, two leading characters and two trailing characters are added to the file, but in most cases will remain hidden. When exporting results, use Excel 2000 and later, which support the Unicode format.

### To export results:

- 1 With the search results you want to export listed in the Results window, click **Export**.
- 2 Specify the location in the dialog box and then click **Save**. If the file already exists, you're given the option to overwrite the file. Status text under the results list indicates the progress of the export.

**Note:** If you want to stop the export, click **Stop**. You cannot perform a partial export, so clicking **Stop** cancels the export entirely.

## Jumping to a result (Go To)

You can jump to an individual result to see where the result occurs. Depending on the type of document that contains the search result, the following occurs:

- **Database:** The form opens in the Project Editor and the text string is highlighted.
- **Cicode/VBA:** The document opens within the Cicode Editor and the text string is highlighted.
- **Graphic:** The page opens in the Graphics Builder and the Properties dialog box appears for the object that contains the text string. The property containing the text string is displayed. If the string occurs on or inside a Genie, the Genie form also appears.
- **Report:** The configured report editor opens and displays the report file, but the text string is not highlighted.

**To jump to a result:**

- With the search result you want to jump to highlighted in the Results window, click **Go To**. The document or form containing the occurrence opens.

See Also [Replacing results](#)

## Replacing results

You can replace single results or multiple results with the replacement text string you specified. You can also test a single result before replacing it. Depending on the type of document that contains the search result, the following occurs when a replacement is made:

- **Database:** The result is replaced with the replacement text and the database record updated. The form containing the search result is not opened; to see the location of the search result before or after the replacement is made, use the [Go To command](#).
- **Cicode/VBA:** The Cicode file containing the matched text loads (if it is not loaded already), the replacement is made, and the file saved.
- **Graphic:** The page opens in the Graphics Builder (if it is not already) and the replacement made. If the page is open and contains unsaved changes, you're instructed to save or discard the changes before making the replacement. If there are multiple changes to be made to the same graphics page, the page remains open until all the changes have been made.
- **Report:** The found text is replaced with the replacement text and the file is saved.

**Note:** Replacements cannot be undone once performed. You should take care to check your replacements before making them, especially when working with multiple replacements.

**To test a result:**

- 1 With the result you want to test highlighted, click **Test**. A dialog box appears showing the result of the text replace.

- 2 Click **Accept** to accept the text replacement, or click **Cancel**.

**To replace a single result:**

- With the result you want to replace highlighted, click **Replace**. The replacement is made and the result removed from the Results window. The next result in the list is then selected.

**To replace multiple results:**

- 1 With the search results you want to replace listed in the Results window, click **Replace All**. A confirmation dialog appears.
- 2 The replacements are made and removed from the Results window. (Replacements that are not made remain in the results list. This will occur if, for example, you try to replace a property that is read-only.)

**Note:** Clicking **Stop** during this process does not undo any replacements already made.

When attempting to make a replacement, you might encounter an error message that warns you of project-related issues you should be aware of before making a replacement. For details, see [Find and Replace error messages](#).

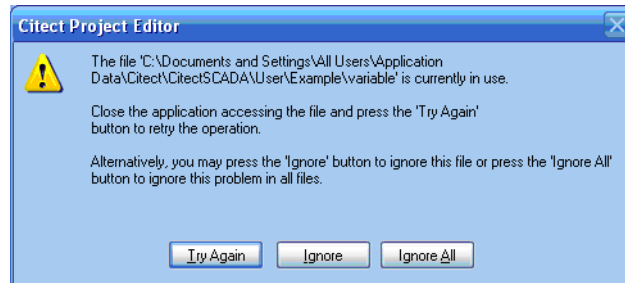
## Find and Replace error messages

Find and Replace will display one of the following errors if it cannot replace a text string:

- [File in use](#)
- [Replaced text truncated](#)
- [Original text not found](#)
- [Replaced text out of range](#)
- [Replacement text not numeric](#)
- [Field is read-only](#)
- [Undetermined error](#)

### File in use

This error appears if the database or file that is required for writing to has become unavailable. This may be the case if the database/file is being used by a third-party application.

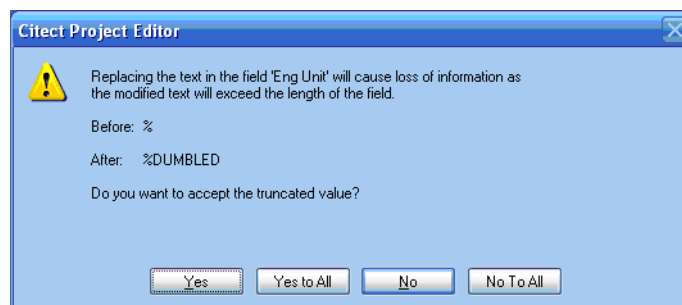


Do one of the following:

- Click **Try Again** (Default) to repeat the operation on the database/file.
- Click **Ignore** to skip the operation on this file.
- Click **Ignore All** button to skip any operations on files that are currently in use; this option causes this message not to reappear.

#### Replaced text truncated

This error appears if performing the replace the text in a DBF field will exceed the field width limits.



This error does not occur if searching the current graphics page. Here, the Engineering Units field containing % will be replaced with "%DUMBLEDORE". This field has a max width of 8, thus the truncation warning.

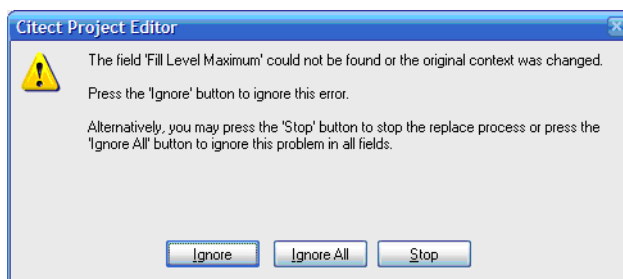
Do one of the following:

- Click **Yes** to commit the truncated text to the database. Usually this will generate a compilation error when the project is compiled.
- Click **Yes to All** to commit all changes to fields regardless if truncation exists without displaying the warning again.
- Click **No** (default) to leave the text as is.
- Click **No to All** button to leave all truncated fields as is.

### Original text not found

This error appears when attempting to replace an item on the current graphics page when the animation could not be found or the text could not be found. This would occur if the animation was deleted, or if the text found in an animation's field was changed after the find but before replacing the item.

In the example below, the Fill Level Maximum contained a value of 23, and the search text was 23. Before replacing the record, it was changed to 66.



Do one of the following:

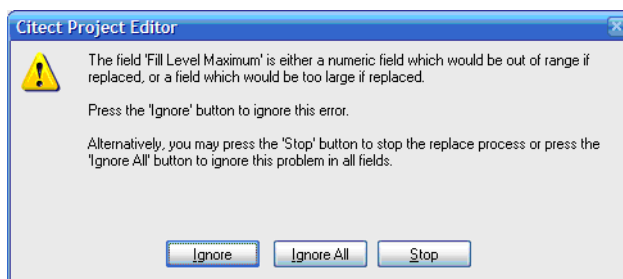
- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if it exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips any not found errors that occur during this replacement.
- Click **Stop** to stop the replacement at the current record.

### Replaced text out of range

This error appears when carrying out a replacement on the current graphics page in two different circumstances:

- 1 The field is text and is too long for the allowable field width.
- 2 The field is a numeric field, and would be out of the allowable range for a replacement value.

In the example below, the Fill Level Maximum allows a range of 0-100, and the value was 23 and is being replaced with 101, which would be out of range.



Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if one exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips out-of-range errors that occur during this replacement.
- Click **Stop** to stop the replacement at the current record.

#### Replacement text not numeric

This error will appear when carrying out a replacement on the current graphics page when the field being replaced is a numeric field, and the replacement text contains a non-numeric value.

In the example below, the Fill Level Maximum contained a value of '23', and the replacement text was 'fred'.



Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if it exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips any non-numeric errors that occur during this replacement.
- Click **Stop** to stop the replacement at the current record.

#### Field is read-only

This error appears when replacing an item on the current graphics page when the field being replaced is part of a linked object like a Genie or template.

In the example below, the Expression field was part of an object that was part of a genie.





Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if it exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips any read-only errors that occur during this replacement.
- Click **Stop** to stop the replacement at the current record.

#### Undetermined error

This error appears when carrying out a replacement on the current graphics page when a general error occurs, and should not happen in normal operation.



Do one of the following:

- Click **Ignore** to skip this operation, leave the entry in the list, and move on to the next replacement if it exists.
- Clicking **Ignore All** acts like the **Ignore** button, except that it skips any undetermined errors that occur during this replacement.
- Click **Stop** to stop the replacement at the current record.

## Troubleshooting Searches

If you don't find a result that you expected to find, check the following points, and then perform your search again:

- Did you spell the text string correctly?
- Did you include the correct number of spaces?

- Are you using the appropriate Look in option?
- Are you using the appropriate Search options?
- Are you searching in the correct project?
- Are you using the correct graphics search?
- If you are using the graphics page search, do you have the correct graphics page open?

# Chapter 10: Configuring Your System

---

Before you run your project you will need to configure each computer in your CitectSCADA system. Configuration information is stored on each machine in a `citect.ini` file based on your installation, database configuration and compiled project. Configuration is done with the Computer Setup Wizard.

The Computer Setup Wizard contains a series of pages allowing configuration of the most important computer specific settings including:

- The role the computer has in the Citect network
- The project being run
- The CPU Configuration
- The Citect Events enabled for each component
- The Cicode run for each component on startup
- The cluster configuration
- The security settings applied

The wizard uses the configuration stored in the project databases to provide information to you. Selected options are written to the `citect.ini` file.

The wizard must be run on each computer running CitectSCADA in your system. It should be run after compiling your project and as the last step before running the system.

See Also [Running the Computer Setup Wizard](#)

## Running the Computer Setup Wizard

To start the Citect Computer Setup Wizard:

- 1 Open Citect Explorer.
- 2 In the project list area, select **My Projects** - designated by a computer icon.
- 3 Double-click the **Computer Setup Wizard** icon, or choose **Tools | Computer Setup Wizard**. The Citect Computer Setup Wizard is displayed.
- 4 Select **Express Setup** or **Custom Setup**.

The pages that are displayed depend on the configuration of the machine and include:

- [Project Configuration](#)

- [Computer Role Configuration](#)
- [Network Model](#)
- [Internet Server Configuration](#)
- [Alarm Configuration](#)
- [Reports Configuration](#)
- [Trends Configuration](#)
- [CPU Configuration](#)
- [Events Configuration](#)
- [Startup Functions Configuration](#)
- [Cluster Connections Configuration](#) \*
- [Time Configuration](#) \*
- [Control Menu Security Configuration](#) \*
- [Keyboard Security Configuration](#) \*
- [Miscellaneous Security Configuration](#) \*
- [General Options Setup](#) \*

\* Only available in Custom Setup mode.

Each screen of the wizard is described in the sections that follow.

See Also [Project Configuration](#)

## Project Configuration

Select the project to run on this CitectSCADA computer. The Computer Setup Wizard will show you all the compiled projects defined in the project list, apart from the include projects.

If there is only one compiled project present, it will be automatically selected. If there are no compiled projects present, an error message is displayed and the wizard will terminate. If this occurs, return to Citect Explorer and confirm that the required project is saved locally and has compiled without errors.

See Also [Computer Role Configuration](#)

## Computer Role Configuration

Use the Computer Role Setup page to specify the role of the computer running CitectSCADA. Select one of the options described below.

**Note:** In order to use CitectSCADA's multi-process capabilities, networking *must* be enabled.

Option	Description
Server and Display Client	<p>This computer will be a standalone or networked CitectSCADA <a href="#">I/O server</a> and <a href="#">display client</a>. This option is disabled if this computer has no Server components assigned to it to run. Selecting this option enables the <b>Multi-Process</b> check box.</p> <p>Select the <b>Multi-Process</b> check box to separate your client and server components into individual processes. This option can be used for distributing the components across multiple CPUs.</p> <p>If you leave the <b>Multi-Process</b> check box unselected, CitectSCADA will run the client and all server components in one process.</p> <p>If the <b>Multi-Process</b> check box is selected the</p> <p>[General]MultiProcess parameter in the Citect.ini file is saved with the value 1. If not selected, the parameter is saved with the value 0.</p>
Display Client	<p>This computer will only be a display client. This option is disabled if this computer has been assigned a Server component to run. Selecting this option enables the <b>Full License</b> check box.</p> <p>Select the <b>Full License</b> check box if you want this display client to use a full license. This sets the [Client]FullLicense parameter in the Citect.ini file to 1 (the default value).</p>
Manager Client	<p>This computer will only be a manager client. This read only option is disabled if this computer has been assigned a Server component to run.</p>

Some of these options may be disabled depending on what servers have been configured to run on this computer. The Computer Setup Wizard cross-references your computer's network identification with the network addresses configured for each server in your project configuration.

See Also [Network Model](#)  
[CPU Configuration](#)

## Network Model

Select the network model to be applied to this CitectSCADA computer. Options include:

- No networking
- TCP/IP

The no networking option is not available when the multi-process option is selected. From Version 7.0 CitectSCADA uses TCP/IP to facilitate communications across a network.

**Note:** TCP/IP address information for Citect servers is configured within the Citect project itself. See [Network Address Definitions](#) for more information.

When you complete the Computer Setup Wizard, the chosen network model is written to the [LAN] section in the `citect.ini` file; for example:

```
...
[LAN]
TCP/IP=1
...
```

See Also [Internet Server Configuration](#)

## Internet Server Configuration

Select the **This computer is an Internet Server** option to make the computer an Internet Server. To allow communication with a remote [Internet display client](#), an Internet Server requires a permanent Internet connection and a static [IP address](#) (or hostname).

- 1 In the **Client Connection Information area**, type the **Internet Server IP address or hostname**. This adds a `Primary=<Internet Server IP address>` entry to the [DNS] section of the `citect.ini` file if there was no pre-existing entry. For details, see [\[DNS\]Primary](#).

**Note:** To determine the TCP/IP address of the Internet Server computer, choose **Start | Run**. Type **CMD** and press **Enter**. Then at the DOS prompt type **IPCONFIG** and press **Enter**.

- 2 Type the **Alternate Internet Server IP address or hostname**. This adds a `Standby=<Alternate Internet Server IP address>` entry to the [DNS] section of the `citect.ini` file if there was no pre-existing entry. For details, see [\[DNS\]Standby](#).

The Internet Display Client automatically connects to this alternate server if connection to the primary server fails. Note that this can only happen automatically if an initial connection has previously been made to the primary Internet Server.

See Also [Alarm Configuration](#)  
[DNS Parameters](#)

## Alarm Configuration

The Alarm Configuration page will only be displayed if this machine is configured as an Alarm Server in the Project Editor.

CitectSCADA has several options available for alarm processing:

Option	Description
Alarm scan time	Determines the rate at which alarms are scanned and processed. A value of 500 (the default value) indicates that CitectSCADA tries to process the alarms every 500ms. However, if CitectSCADA cannot read all the alarm data from the I/O device within 500ms, the alarms are processed at a slower rate. For example, if it takes 800ms to read all the alarm data from the I/O device, CitectSCADA processes the alarms every 800ms. If you reduce the alarm scan time, the alarms server uses less CPU (because it does not need to process the alarm records as often). The amount of data read from the I/O device is also reduced, so that other processes (Trends, Reports, and the current page) get their I/O device data more quickly. You can enter any value from 0 to 60000 (milliseconds).
Alarm save period	The period for saving alarm and event data (to disk). You can save alarm and event data periodically to ensure that the data is restored after a system crash or shutdown. Note that the smaller the period, the greater is the load on the system.
Summary length	The maximum number of alarm summary entries that can be held in memory. You can view these alarm summary entries on the alarm summary page. Note that each event requires 62 bytes of memory, plus the length of the comment. 32,000 events require at least 1.9 Mb of memory. If you use many events, you should have enough memory to keep them in RAM.
Summary timeout	The length of time that alarm summary entries remain in the alarm summary queue.
Primary alarms server save path	The path to the primary save files. CitectSCADA uses two save files for each alarms server, ALMSAV . DAT and ALMINDEXSAVE . DAT. The save primary path is the directory where the primary alarms server creates its save files. When restoring the files, the most recent (of the primary and secondary) save files will be used.
Standby alarms server save path	The path to the secondary save files.

To ensure there are no conflicts between alarm files used by multiple Alarm Servers from different clusters running on the same machine, the alarm files have a dynamic naming convention based on the following format:

```
<ProjectName>_<ClusterName>_<filename>.DAT
```

See Also [Reports Configuration](#)

## Reports Configuration

The Reports Configuration page will only be displayed if this machine is configured as a Reports Server in the Project Editor.

**Note:** For a networked computer to be a Reports Server it must also be the I/O Server or must be able to communicate with the I/O Server on the network.

CitectSCADA has several options available for report processing:

Option	Description
Startup report	Defines the name of the report to run when CitectSCADA starts up.
Inhibit triggered reports on startup	For example, you might have a report that is triggered off the rising edge of a bit on startup. The Reports Server notices the bit come on, and runs the report. If this option is checked, the Reports Server does not run this report until it has read the I/O devices a second time.
Run reports concurrently with primary Reports Server	Enables or disables tandem processing of reports. If this server is the standby Reports Server, it can process all reports in tandem with the primary server, or it can remain idle until called.

See Also [Trends Configuration](#)

## Trends Configuration

The Trends Configuration page will only be displayed if this machine is configured as a Trends Server in the Project Editor.

**Note:** For a networked computer to be a Trends Server it must also be the I/O Server or must be able to communicate with the I/O Server on the network.

CitectSCADA has one option available for trend processing:

Option	Description
Inhibit triggered trends on startup	You might have a trend that is triggered off the rising edge of a bit on startup. If this option is enabled, the trends server does not display the trend until it has read the I/O devices a second time.

See Also [CPU Configuration](#)

## CPU Configuration

The CPU Setup page is used to assign client and server components to specific processors in a multi-processor machine.

This page lists each component's full name, including the cluster to which it belongs, the priority and the CPU assignment. If the Multi-process option was not selected on the [Computer Role Configuration](#) page there will only be one entry listed, either Client or Client and Servers. If the Multi-process option was selected, you have the option of selecting specific CPUs for the Client, I/O Server, Alarm Server, Trend Server and Report Server.

**To assign a CPU to a component:**

- 1 Select one or more components from the list (hold the **Ctrl** key down to select multiple components).



- 2 Click **Modify**.
- 3 Type the number of the CPU and click **OK**.

When you complete the Computer Setup Wizard, the CPU assignments are written to each component section in the `citect.ini` file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Cluster1
...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Cluster1
...
```

See Also [Events Configuration](#)

## Events Configuration

Events are used to trigger actions, such as a command or set of commands. For example, an operator can be notified when a process is complete, or a series of instructions can be executed when a process reaches a certain stage. Select the **Enable Events on this computer** check box if events are to be enabled on this CitectSCADA computer.

The Events Setup page lists each component's full name, including the cluster to which it belongs, alongside a list of events that can be enabled for each component. If the Multi-process option was not selected on the [Computer Role Configuration](#) page there will only be one entry listed, either Client or Client and Servers. If the Multi-process option was selected, you have the option of enabling events for each component on this computer.

The Computer Setup Wizard only displays named events from the selected project. If you are using events in included projects you will need to edit your `citect.ini` file to add these under the `[Events]` section header.

**Note:** Events named 'Global' or events with no title will not appear as these are global events. These events will run on all computers that have events enabled. These events will run in the display client process.

### To enable an event for a component:

- 1 Select the component from the list.
- 2 Select the events you want to enable for that component, or click **Enable All** or **Disable All**.
- 3 Click **Next** when finished.

When you complete the Computer Setup Wizard, the events are written to each component section in the `citect.ini` file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Cluster1
Events=CSV_AlarmClient
...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Cluster1
Events=CSV_TrendXClient,CSV_TrendXServer
...
```

See Also [Startup Functions Configuration](#)

## Startup Functions Configuration

The Startup Functions Setup page is used to define the Startup Cicode that is executed by each CitectSCADA process.

The Startup Functions Setup page lists each component's full name, including the cluster to which it belongs, the priorities of the components and the startup function assigned to each component. If the Multi-process option was not selected on the [Computer Role Configuration](#) page there will only be one entry listed, either Client or Client and Servers. If the Multi-process option was selected, you have the option of assigning startup functions for each component on this computer.

If the StartupCode parameter value for a process is invalid, the CitectSCADA Runtime Manager will simply ignore it on start up.

### To assign a startup function to a component:

- 1 Select the component from the list. To select multiple components, hold down the Ctrl key as you select each item.
- 2 Click **Modify**.
- 3 Type the name of the Cicode function you want to call on startup for that component.
- 4 Click **OK**.

When you complete the Computer Setup Wizard, the events are written to each component section in the `citect.ini` file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
StartupCode=alarmServerStartup
```

```
...
[Trend.Cluster1.TrendServer1]
CPU=2
StartupCode=trendServerStartup
...
```

See Also [Runtime Manager Help - Startup Cicode Functions](#)  
[CPU Configuration](#)  
[Cluster Connections Configuration](#)

## Cluster Connections Configuration

The Cluster Connections Setup page is used to specify the clusters that each component connects to on startup. This controls what data streams a component can see in the system.

The Cluster Connections Setup page lists each component's full name, including the cluster to which it belongs, the priorities of the components and the clusters assigned to each component. If the Multi-process option was not selected on the [Computer Role Configuration](#) page there will only be one entry listed, either Client or Client and Servers. If the Multi-process option was selected, you have the option of assigning clusters for each component on this computer.

By default, all components will connect to all clusters unless otherwise modified.

If the Clusters parameter value for a process is invalid, then the CitectSCADA Runtime will simply ignore it on startup.

### To assign a cluster to a component:

- 1 Select the component from the list. To select multiple components, hold down the Ctrl key as you select each item.
- 2 Click **Modify**.
- 3 Select the clusters you want the component to connect to on startup.
- 4 Click **OK**.

When you complete the Computer Setup Wizard, the clusters are written to each component section in the `citect.ini` file; for example:

```
...
[Alarm.Cluster1.AlarmServer1]
CPU=1
Clusters=Sydney
...
[Trend.Cluster1.TrendServer1]
CPU=2
Clusters=Sydney,Tokyo
...
```

See Also [Implementing Clustering](#)

## [Time Configuration](#)

### Time Configuration

Select to synchronize this computer's time with the Time Server or select this CitectSCADA computer to be a Time Server.

**Note:** You cannot enter localized characters in the **Time server name** text box.

Choose one of the following options:

Option	Description
No time synchronization	Do not synchronize the time on this computer
Synchronize time with the Time Server	Causes the time on this computer to be updated regularly from the Time Server during runtime
This computer is the Time Server	Causes this computer to regularly update other CitectSCADA computers during run time if they have the Synchronize option (above) set

**Note:** A time server may be set up only on a CitectSCADA I/O server.

See Also [Control Menu Security Configuration](#)

### Control Menu Security Configuration

The CitectSCADA window property options allow you to control an operator's access to system features.

This allows for flexibility with system security at run time.

Option	Description
CitectSCADA configuration environment on menu	Allows the operator to use the control menu (top left-hand icon) to access the Citect Editor, Project Editor, Graphics Builder, and Cicode Editor from CitectSCADA at run time. Disabling this provides better security.
Display Title Bar	Allows the standard Windows title bar to be displayed at the top of CitectSCADA runtime windows. Disabling this option provides better security: your pages appear in full-screen state. Users will not have access to the title of the window, the maximize and minimize buttons (at the right hand end of the title bar), and the control menu button (at the left hand end of the title bar). A page in full screen state takes up the entire display area (assuming this does not affect its aspect ratio), and it cannot be resized. This option can be changed for individual pages by checking Title Bar at creation, or afterwards in Page Properties. Resizing pages can degrade picture quality. If this is unacceptable, redesign the page using the resolution you want.
Shutdown on menu	Allows the operator to use the control menu (top-left icon) to shut down CitectSCADA at runtime. The shutdown is not password- or privilege-protected. Disabling this provides better security.

Option	Description
Kernel on menu	Allows the operator to use the control menu (top left icon) to display the CitectSCADA Kernel at run time. Disabling this provides better security.

See Also [Keyboard Security Configuration](#)

## Keyboard Security Configuration

Windows has a set of standard task-swapping shortcut commands that are (optionally) supported by CitectSCADA at run time. This option allows the Alt-Space Windows command to be enabled or disabled at run time. Alt-Space provides access to the Windows control menu (even if the title bar has been disabled).

**Note:** The ability to disable Alt-Escape, Ctrl-Escape and Alt-Tab is not currently available.

See Also [Miscellaneous Security Configuration](#)

## Miscellaneous Security Configuration

Some standard Windows features may interfere with the secure operation of your CitectSCADA system. Use the Miscellaneous Security page to disable these features.

Option	Description
Inhibit screen saver while CitectSCADA is running	Stops the screen saver from blanking out important screens that should be always visible. Alternatively the screen saver password can add additional security features
Display Cancel button at startup	Provides the ability to stop CitectSCADA from starting up automatically. Automatic startup is a potential security problem

See Also [General Options Setup](#)

## General Options Setup

Use the General Options Setup page to specify general options.

Option	Description
Data Directory	The directory where the CitectSCADA data files are located. The CitectSCADA data files are the files that are generated at run time: trend files, disk PLC etc.
Backup project path	The backup directory that is used if a runtime database cannot be located (due to a disk failure or file loss).
Startup page	The Page Name of the <a href="#">graphics page</a> to display when CitectSCADA starts up.

Option	Description
Page scan time	<p>The delay (in milliseconds) between updating a graphics page and starting the next communications cycle. The Page Scan Time sets the default for how often your graphics pages are updated. When a page is updated, all relevant data (variable tags etc. represented on the graphics page) is scanned to determine if field conditions have changed. This setting is overridden by the Scan Time value specified in Page Properties (if applied). A value of 250 (the default value) indicates that CitectSCADA will try to update the page every 250ms. However, if CitectSCADA cannot read all of the data from the I/O device within 250ms, the page is processed at a slower rate. For example, if it takes 800ms to read all the data from the relevant I/O device, CitectSCADA processes the page every 800ms.</p> <p>Under some conditions, you might want to slow the update of your pages to reduce the load on the I/O servers. By reducing the page scan time, you allow more communication bandwidth to other CitectSCADA tasks or Clients. For example, you might want fast response on your main operator computers, while slowing the response time on manager computers. You can enter any value from 0 to 60000 (milliseconds).</p>

See Also [Finish](#)

## Finish

Click **Finish** to save the setup to the `citect.ini` file, **Cancel** to quit the wizard without saving, or **Back** to navigate to a page that requires adjusting.

# Chapter 11: Implementing Clustering

---

Once you have designed the clustering for your CitectSCADA system, including the configurations of servers you need, you can proceed to implement that design. You will need to configure:

- [Cluster Definitions](#)

Each cluster must be defined by giving it a unique name in the project.

- [Network Address Definitions](#)

Each physical server in your system must be identified with a unique name and IP address.

- [Alarm Server Definitions](#)

Each Alarm Server must be named, and assigned to a cluster and physical server. Each server should be identified as Primary or Standby

- [Report Server Definitions](#)

Each Report Server must be named, and assigned to a cluster and physical server. Each server should be identified as Primary or Standby

- [Trend Server Definitions](#)

Each Trend Server must be named, and assigned to a cluster and physical server. Each server should be identified as Primary or Standby

- [I/O Server Definitions](#)

Each IO Server must be named, and assigned to a cluster and physical server. Each server should be identified as Primary or Standby.

See Also [Rules of Clustering](#)  
[Assigning tags to a cluster at Runtime](#)

## Rules of Clustering

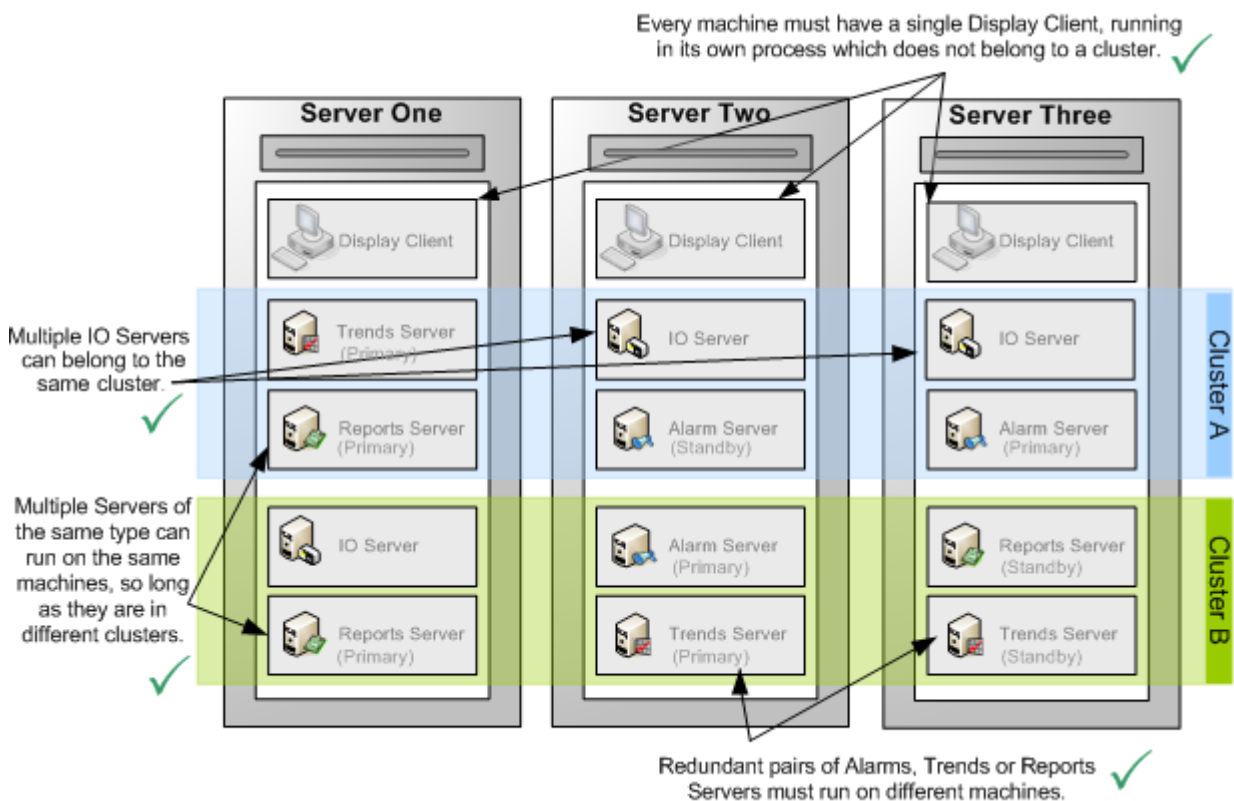
When configuring CitectSCADA the following clustering rules apply:

- Each cluster must have a unique name.
- Each server component must have a unique name.
- Each server component must belong to one cluster.
- Each cluster can contain only have one pair of redundant Alarm Servers. They must reside on different machines.

- Each cluster can contain only one pair of redundant Reports Servers. They must reside on different machines.
- Each cluster can contain only one pair of redundant Trends Servers. They must reside on different machines.
- Each cluster can contain an unlimited number of IO Servers.

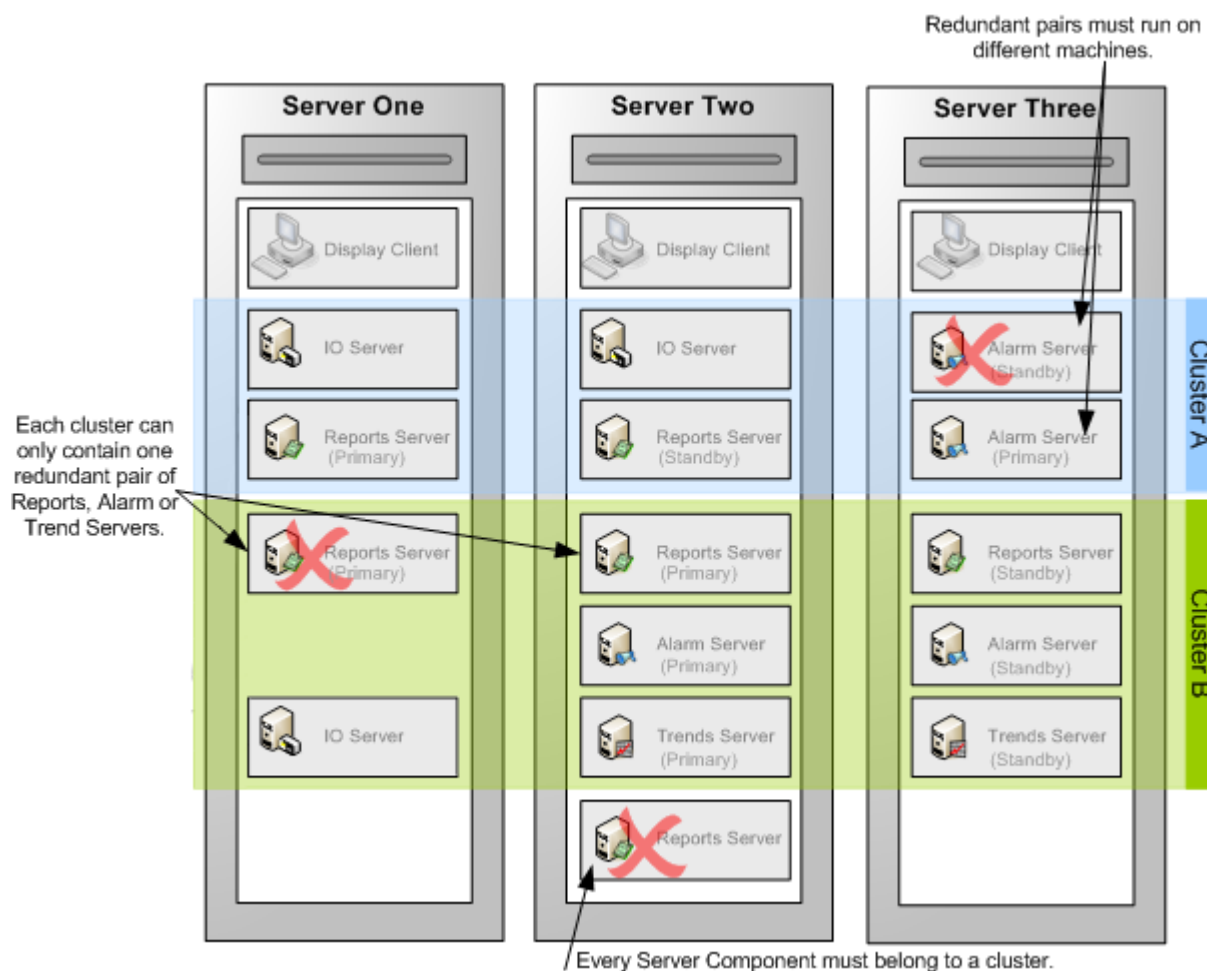
There are countless variations in how a CitectSCADA clustered system can be configured. The most appropriate configuration will depend on the **requirements** for the solution to be deployed and the **environment** in which it is being deployed. For more information, refer to [Typical system scenarios](#).

The diagram below is an example of a CitectSCADA system running with two clusters across three machines. All server and client components have been deployed in accordance with the clustering rules.





The next diagram demonstrates circumstances which break the clustering rules.



The CitectSCADA compiler or the CitectSCADA Runtime Manager detects when the rules of clustering are not being observed and advises the user accordingly.

See Also [About cluster context](#)

## Cluster Definitions

See [Rules of Clustering](#) for additional information.

To define a cluster:

- 1 In the Project Editor, choose **Servers | Clusters**.

- 2 In the Cluster dialog box, complete the cluster properties:

Option	Description
Cluster Name	The name of the cluster (maximum of 16 characters). The name must be unique to the project and must not contain spaces.
Comment	Any useful comment. This property is optional and is not used at runtime.

- 3 Click the **Add** button to append a new record, or **Replace** if you have modified a record.

See Also [Network Address Definitions](#)

## Network Address Definitions

To configure a network address:

- 1 In the Project Editor, choose **Servers | Network Addresses**.
- 2 In the Network Addresses dialog box, complete the properties:

Option	Description
Name	The name of the machine at the network address being configured (maximum of 16 characters). The name must be unique to the project and must not contain spaces.
Address	The IP address or computer name of the machine being configured.
Comment	Any useful comment. This property is optional and is not used at runtime.

- 3 Click the **Add** button to append a new record, or **Replace** if you have modified a record.

See Also [Alarm Server Definitions](#)

## Alarm Server Definitions

Note the default server port numbers under [Configure Servers](#).

To configure an Alarm Server:

- 1 In the Project Editor, choose **Servers | Alarm Servers**.
- 2 In the Alarm Servers dialog box, complete the properties:

Option	Description
Cluster Name	The name of the cluster to which this Alarm Server will belong. If there is only one cluster defined in the project, you can leave this field blank. The Alarm server will default to the defined cluster
Server Name	The name of the server (maximum of 16 characters). The name must be unique to the project and must not contain spaces.

Option	Description
Mode	The mode for this server, either Primary or Standby. If this property is left blank, the default value will be Primary. The Primary and Standby servers must run on different computers, and only one Primary and one Standby can be defined per cluster.
Network Addresses	The IP address or computer name of the machine being configured.
Port	The port this server will listen on. You can leave this field blank if you are running only one Alarm server on the machine, in which case the default port number will be used.
Comment	Any useful comment. This property is optional and is not used at runtime.

**Note:** The following fields are implemented with extended forms (press F2):

Option	Description
Publish Alarm Properties	TRUE or FALSE. By default this value is FALSE. When turned to TRUE the alarm properties are published and can be viewed as normal variable tags, and the Alarms Server listens as if it were an I/O connector.
Port	The port the Alarm Server will listen on if the Publish Alarm property is set to TRUE.

- 3 Click the **Add** button to append a new record, or **Replace** if you have modified a record.

See Also [Report Server Definitions](#)

## Report Server Definitions

Note the default server port numbers under [Configure Servers](#).

**To configure a Report Server:**

- 1 In the Project Editor, choose **Servers | Report Servers**.
- 2 In the Report Servers dialog box, complete the properties:

Option	Description
Cluster Name	The name of the cluster to which this Report Server will belong. If there is only one cluster defined in the project, you can leave this field blank. The Report server will default to the defined cluster
Server Name	The name of the server (maximum of 16 characters). The name must be unique to the project and must not contain spaces.
Mode	The mode for this server, either Primary or Standby. If this property is left blank, the default value will be Primary. The Primary and Standby servers must run on different computers, and only one Primary and one Standby can be defined per cluster.
Network Addresses	The IP address or computer name of the machine being configured.

Option	Description
Port	The port this server will listen on. You can leave this field blank if you are running only one Report server on the machine, in which case the default port number will be used.
Comment	Any useful comment. This property is optional and is not used at runtime.

- 3 Click the **Add** button to append a new record, or **Replace** if you have modified a record.

See Also [Trend Server Definitions](#)

## Trend Server Definitions

Note the default server port numbers under [Configure Servers](#).

To configure a Trend Server:

- 1 In the Project Editor, choose **Servers | Trend Servers**.
- 2 In the Trend Servers dialog box, complete the properties:

Option	Description
Cluster Name	The name of the cluster to which this Trend Server will belong. If there is only one cluster defined in the project, you can leave this field blank. The Trend server will default to the defined cluster.
Server Name	The name of the server (maximum of 16 characters). The name must be unique to the project and must not contain spaces.
Mode	The mode for this server, either Primary or Standby. If this property is left blank, the default value will be Primary. The Primary and Standby servers must run on different computers, and only one Primary and one Standby can be defined per cluster.
Network Addresses	The IP address or computer name of the machine being configured.
Port	The port this server will listen on. You can leave this field blank if you are running only one Trend Server on the machine, in which case the default port number will be used.
Comment	Any useful comment. This property is optional and is not used at runtime.

- 3 Click the **Add** button to append a new record, or **Replace** if you have modified a record.

See Also [I/O Server Definitions](#)

## I/O Server Definitions

Note the default server port numbers under [Configure Servers](#).

To configure an I/O Server

- 1 In the Project Editor, choose **Servers | I/O Servers**.

- 2 In the I/O Servers dialog box, complete the properties.

Option	Description
Cluster Name	The name of the cluster to which this I/O Server will belong. If there is only one cluster defined in the project, you can leave this field blank. The I/O Server will default to the defined cluster.
Server Name	The name of the server (maximum of 16 characters). The name must be unique to the project and must not contain spaces.
Mode	The mode for this server, either Primary or Standby. If this property is left blank, the default value will be Primary.
Network Addresses	The IP address or computer name of the machine being configured.
Port	The port this server will listen on. You may leave this blank in which case the default port number will be used.
Peer Port	The peer port is used for communication between IOServers to provide IODevice information updates and for standby writes. It is also used as a legacy port to allow pre v7 CitectSCADA Clients to still retrieve IO Data from the v7 IOServer. It defaults to a value of 2078. If you specify a different value for this port, v7 IOServers will still communicate correctly, but legacy CitectSCADA Clients will no longer be able to retrieve data.
Comment	Any useful comment. This property is optional and is not used at runtime.

- 3 Click the **Add** button to append a new record, or **Replace** if you have modified a record.

## Assigning tags to a cluster at Runtime

CitectSCADA includes functionality that allows you to assign the tags on a page to a specific cluster during Runtime.

If the tags on a page exist within a number of different clusters, you can use Cicode to pass a cluster name to the page as it opens. Any tags that do not have a cluster explicitly defined will then be assigned to the specified cluster as the page is launched.

This provides a practical solution for a replicated system, where your CitectSCADA project controls a number of identical sites or production lines. As the sites contain the same equipment, the tags representing the hardware architecture will potentially be the same for each. The ability to pass a cluster name to a page at runtime means just a single mimic page is required to represent multiple sites, reducing configuration time and simplifying project deployment.

For example, a menu page could be created to launch the mimic page for three identical production lines, each based on a the same page called "ProductionLine".

To achieve this, you would use the `PageDisplay` function to configure the following buttons on your menu page:

Button one	Text	<b>Production Line A</b>
	Command	<code>PageDisplay("ProductionLine","Cluster_A")</code>
	Comment	Display the mimic page in the context of production line A
Button two	Text	<b>Production Line B</b>
	Command	<code>PageDisplay("ProductionLine","Cluster_B")</code>
	Comment	Display the mimic page in the context of production line B
Button three	Text	<b>Production Line C</b>
	Command	<code>PageDisplay("ProductionLine","Cluster_C")</code>
	Comment	Display the mimic page in the context of production line C

In each case, `PageDisplay` would pass the name of the host cluster to the page, depending on which button is selected. As each production line shares a common architecture, any tags that do not have a cluster explicitly defined will be assigned to the specified cluster as the page is opened.

This functionality is supported by the following Cicode functions:

- [PageDisplay](#)
- [PageGoto](#)
- [WinNew](#)
- [WinNewAt](#)

You can also use the [PageInfo](#) function to determine the cluster context that has been set for a page.

This functionality extends to any Cicode that may be called from a graphics page. You can write Cicode that only specifies variable tag names, allowing the cluster to be defined by the current context of the page from which it is launched.

If the Cicode is executed by the function [TaskNew](#), you have the option to specify a cluster by setting the `ClusterName` parameter.

**Note:** The cluster context for Cicode cannot be changed once the code is running.

# Chapter 12: Building Redundancy into Your System

---

Redundancy in CitectSCADA can be defined at many different levels. When building redundancy for your system, it is important to consider a level that is appropriate to your requirements, by:

- Defining how critical your processes are;
- Determining if the potential downtime through failure is high;
- Deciding which components you would like to implement redundancy for; and
- To consider design and maintenance issues with a greater need of redundancy.

The section covers the following redundancy concepts:

- [I/O Server Redundancy](#)
- [I/O Device promotion](#)
- [Redundancy and Persistence](#)
- [Data Path Redundancy](#)
- [Alarms, Reports, and Trends Server Redundancy](#)

**Note:** Using the CitectSCADA Computer Setup Wizard will allow you to define the level of redundancy you require by defining the function of each computer (See [Running the Computer Setup Wizard](#)).

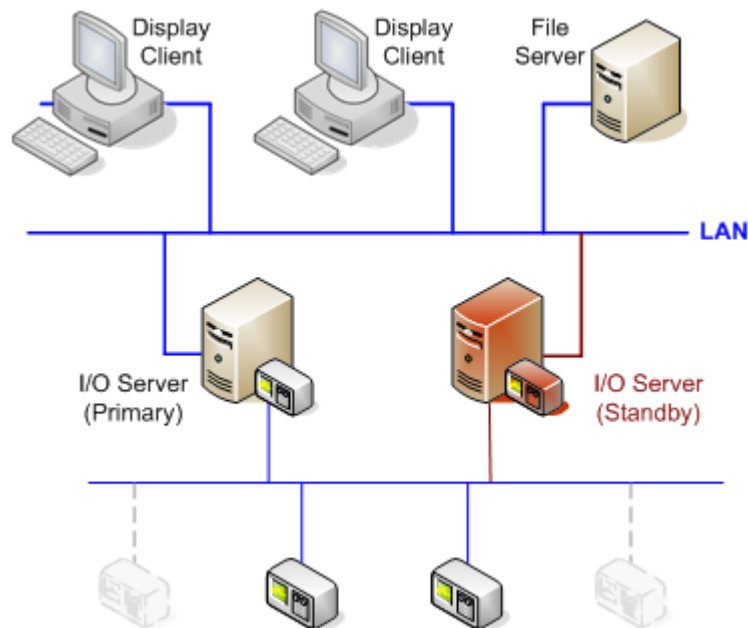
See Also

[Alarms, Reports, and Trends Server Redundancy](#)  
[How CitectSCADA handles file server redundancy](#)  
[How CitectSCADA handles FTP server redundancy](#)  
[Redundancy of Standalone Systems](#)

## I/O Server Redundancy

Systems with a single I/O Server have a single point of failure. If the primary server fails, control and monitoring of the system is lost. By introducing a second I/O Server and dedicating it to communicating with the same I/O devices, the single point of failure can be eliminated. You now have a *primary* and *standby* I/O Server in your system where the standby I/O Server will assume operations in case the primary I/O Server fails.

The diagram below illustrates the introduction of a standby I/O Server into the existing system. When the system is in operation, both I/O Servers are identically maintained.



**Note:** Although both I/O Servers are identical, it is important to note that the standby server is not duplicating the primary server's functions. If it were, the load on the PLC portion of the network would be double and would significantly reduce performance. Therefore, only the primary server communicates with the PLCs at any given time.

Redundancy is provided as follows:

- When the system is in operation and the primary I/O Server fails, or if you wish to perform some maintenance and take it offline, all clients will be reverted to the standby server without any interruption to the system.
- When the primary server is brought back online, the system returns control of the I/O Devices back to the primary server. This is done by copying the disk image from the operating standby server allowing the clients to reconnect to it, and resume control of the system.

When the system is running, you can also use redundant I/O Servers to split the processing load. This would result in higher performance as all I/O Servers would be running in parallel when servicing the I/O Devices.

See Also [I/O Device promotion](#)



## I/O Device promotion

I/O Device promotion refers to when a system failure forces a standby I/O device to assume a primary role during Runtime.

If there is more than one standby device available within a cluster, the order they are promoted in will depend on how your CitectSCADA project has been configured.

If required, you can manually set the order your standby devices are promoted in via the **Priority** field on the extended [I/O Devices Properties](#) form. If priorities are not set, the compiler automatically allocates a priority value to each standby device based on the following rules:

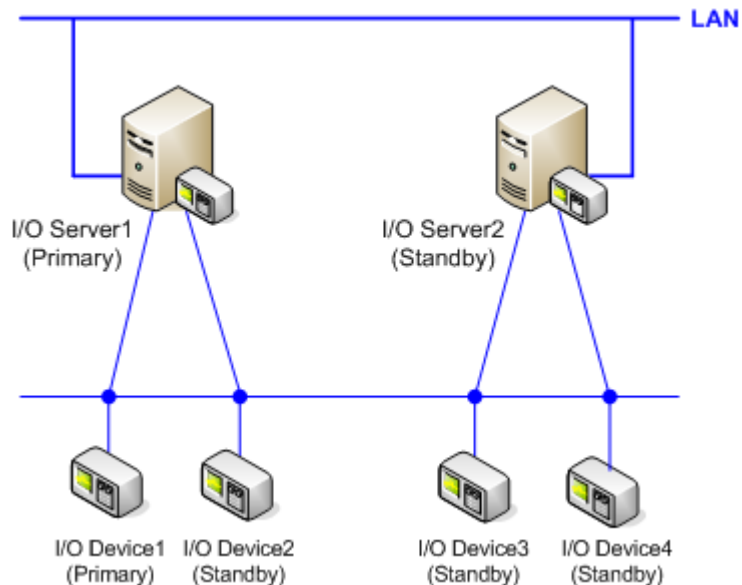
- user-configured priority settings take precedence
- any standby devices that do not have a priority setting will follow those that do, their priority being allocated in the order they were configured
- if no standby devices have a priority setting, they will be assigned priority according to the order they were configured.

These rules apply regardless of the connections between the I/O devices and I/O servers configured in a cluster.

The standby device priorities are confirmed and/or allocated during project compilation. Notification is provided for each allocation issued by the compiler. Compile errors will occur under the following circumstances:

- if a primary device has a priority setting other than the default value of 1
- if standby devices have duplicated priority values.

**Example** The following diagram shows four I/O devices connected to two I/O servers, with just one primary device configured.



If no priorities were set for the standby devices, the compiler would allocate the following:

- I/O Device1 is allocated **Priority 1** by default (no compiler warning)
- I/O Device2 is allocated **Priority 2** (compiler warning generated)
- I/O Device3 is allocated **Priority 3** (compiler warning generated)
- I/O Device4 is allocated **Priority 4** (compiler warning generated)

This presumes the devices were configured in numerical order.

If I/O Device3 has been manually set to priority 2, the compiler would allocate the following:

- I/O Device1 is allocated **Priority 1** by default (no compiler warning)
- I/O Device2 is allocated **Priority 3** (compiler warning generated)
- I/O Device3 is allocated **Priority 2** (no warning generated)
- I/O Device4 is allocated **Priority 4** (compiler warning generated)

In this case, the setting for I/O Device3 has taken precedence. The remaining standby devices are set for promotion based on the order they were configured.

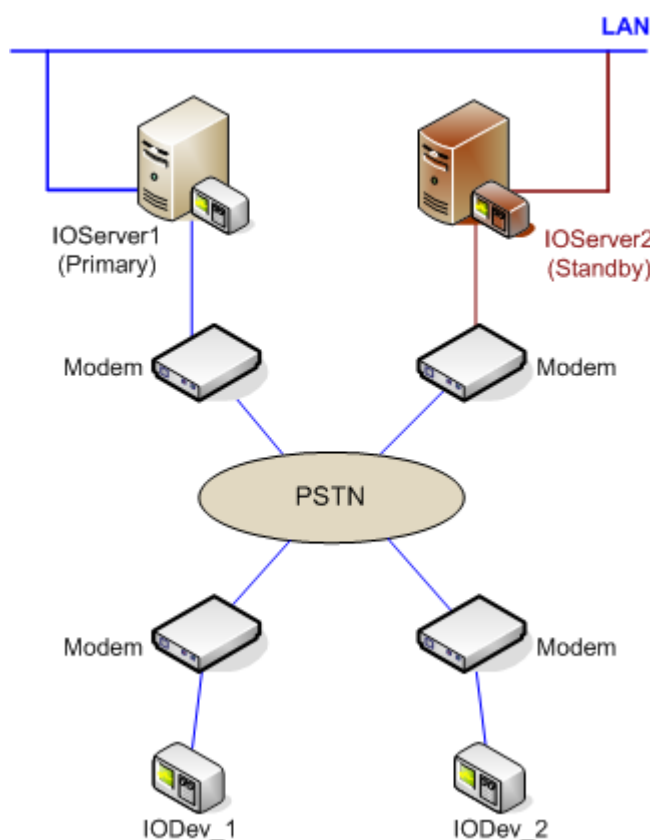
Note that the automated priorities work on an  $n+1$  equation; if I/O Device3 has been set to Priority 5, the remaining devices would have been allocated priority 6 and 7.

See Also [Redundancy and Persistence](#)

## Redundancy and Persistence

### Data Path Redundancy

If you are using Server redundancy, Persistence Caches (I/O Server cache) keep standby servers updated with the most recently read device data. A Persistence Cache, or I/O Server Cache, is created for each cached I/O device. The following diagram introduces the concept of a Persistence Cache.



The diagram shows that there are two I/O Servers, namely IOServer1 (primary) and IOServer2 (standby). Each connects to a PSTN via a modem, which is in turn connects to the I/O Devices, also over a modem. Persistence Caches work as follows:

- 1 Every IODevices->Cache Time period, all data from an I/O Device is stored temporarily in the memory of the I/O Server (I/O Server cache).
- 2 For every [IOServer]SavePeriod, IOServer1 saves its in-memory cache to disk.
- 3 The cache is saved in Persistence Caches -one for each cached device.

- 4 IOserver1 broadcasts to all other I/O servers the UNC path of the Persistence Caches (set with [IOserver]SaveNetwork).
- 5 From these Persistence Caches, IOserver2 updates its in-memory cache for its I/O devices.
- 6 Depending on the value of the CitectSCADA I/O Server parameter of '[IOserver]SavePeriod' (determines how often the Persistence Cache is saved to the hard disk in seconds), IOserver1 saves its in-memory cache to the hard disk every x amount of seconds.

**Note:** You can define an I/O Device on an I/O Server using the Express Communications Wizard, or by adding a device in the I/O Devices form in CitectSCADA's Project Editor.

You are not limited to just one Standby Server, since the UNC path name set in [IOserver]SaveNetwork is broadcast to all I/O Servers. Each I/O Server updates its cache from the Persistence Caches only for the I/O Devices defined on that server. It is then possible, therefore, set up several I/O Servers which update their in-memory caches with the most recently read data.

For example, we set the [IOserver]SaveFile and [IOserver]SaveNetwork parameters as follows:

On IOserver1	On IOserver2
[IOserver]	[IOserver]
SaveFile=C:\Data\IOserver1.dat	SaveFile=C:\Data\IOserver2.dat
SaveNetwork=\\IOserver1\Data\IOserver1.dat	SaveNetwork=\\IOserver2\Data\IOserver2.dat

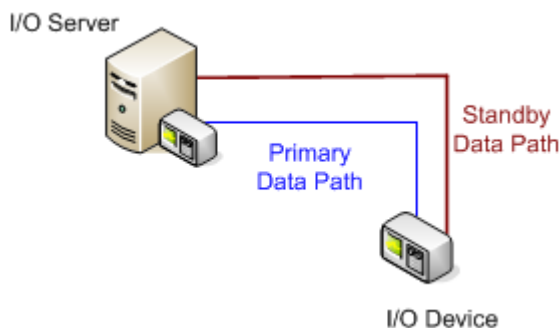
IOserver1 would broadcast the following UNC path of the Persistence Cache to all other I/O Servers: '\\IOserver1\Data\IOserver1.dat'. IOserver2 would then use the Persistence Caches to update its in-memory cache with the device data most recently read by IOserver1.

See Also [Data Path Redundancy](#)

## Data Path Redundancy

Data path redundancy is another form of redundancy involving defining data paths between the I/O Server and the connected I/O Devices. By providing a second (parallel) data path, you effectively ensure that if one data path to the I/O Device fails, the other can be used.

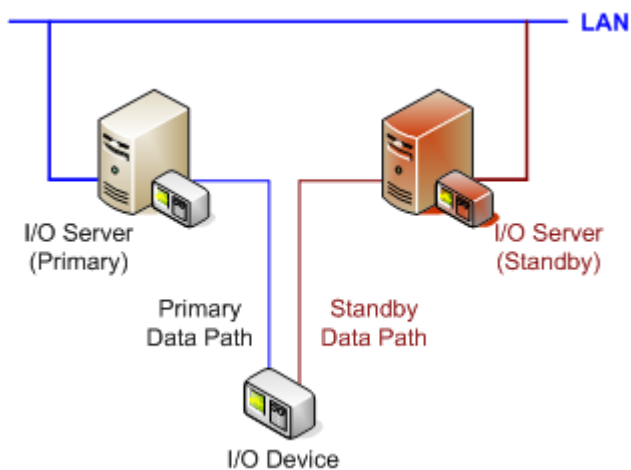
Most brands of PLCs have the facility to allow you to install a parallel data path from the I/O Server to the I/O Device.



The diagram above shows that an additional data path (running in parallel) has been defined. The redundancy is provided as follows:

- When you start your runtime system, CitectSCADA connects to the I/O Device using the primary data path.
- Should communications with the I/O Device fail at any time (e.g. if the communications cable is cut), CitectSCADA will switch to the standby data path without any interruption to the system.
- CitectSCADA reconnects through the primary data path when it is returned in to service.

On a larger CitectSCADA system (such as one running on a network), you can also use data path redundancy to maintain device communications with multiple I/O Server redundancy, as shown in the following diagram.



The redundancy is provided as follows:

- By using a redundant data path from the I/O Device (one path to each I/O Server), you can maintain I/O Device communication.

- Should communications with either the primary I/O Server or standby I/O Server fail, the I/O Device is still accessible.

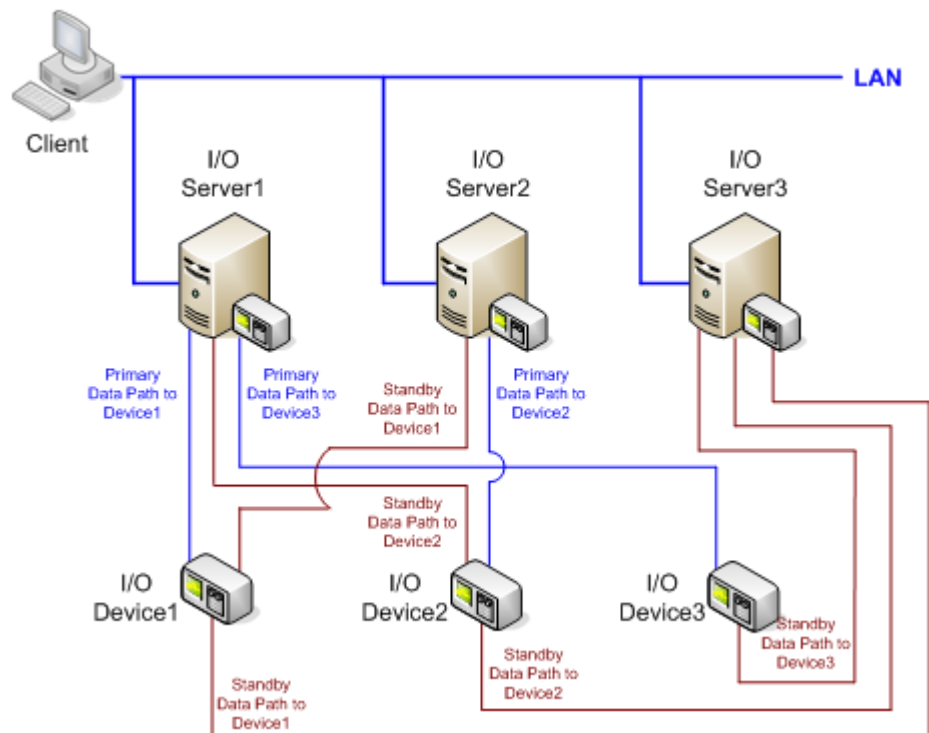
See Also [Multiple Device Redundancy \(Standby Data Paths\)](#)

### Multiple Device Redundancy (Standby Data Paths)

If your I/O Devices support peer-to-peer communication, you can add another level of redundancy to your system by duplicating the I/O Devices.

**Note:** Although I/O Servers do not adapt a Primary or Standby role, they are generally labeled “Primary” and “Standby”. A “Primary” I/O Server is the I/O Server with the Primary I/O Devices connected; a “Standby” I/O Server is the one with the Standby I/O devices connected. One I/O Server can connect to a mixture of Primary and Standby I/O Devices. The I/O Server can support any number of Standby Data Paths.

The following diagram demonstrates multiple device redundancy and Standby Data Paths:



In this scenario, we have three I/O Servers connected to three I/O Devices in the following manner:

I/O Server	I/O Devices Connected
IOServer1	I/O Device1 (Primary)
	I/O Device2 (Standby)
	I/O Device3 (Primary)
IOServer2	I/O Device1 (Standby)
	I/O Device2 (Primary)
IOServer3	I/O Device1 (Standby)
	I/O Device2 (Standby)
	I/O Device3 (Standby)

The following is known:

- CitectSCADA clients communicate with all configured I/O servers at the same time (on startup, the clients try to connect to each configured I/O Server. If they cannot establish communications with an I/O Server, a hardware error is generated).
- When all devices are running, CitectSCADA processes the I/O on the Primary I/O Devices (this reduces the I/O load on the I/O device (and PLC network), which is critical for best performance).
- The CitectSCADA client creates network sessions to all three I/O Servers.
- The client then sends requests for I/O Device1 and I/O Device3 to I/O Server1, and requests for I/O Device 2 to I/O Server2.

Redundancy is provided as follows:

- Should I/O Device1 fail on I/O Server1, the client will send requests for I/O Device1 to I/O Server2 through the standby data path. It continues to send requests for I/O Device3 to I/O Server1.
- If I/O Device also fails on I/O Server2, the client sends requests to I/O Server3.
- Should the connection between I/O Device1 and I/O Server1 be re-established, the client resumes sending requests for I/O Device1 to I/O Server1.

Since we can place primary and standby I/O Devices on various I/O Servers, you should share the primary I/O Devices between your I/O Servers to balance the loading across all the I/O Servers. However, this might not apply for all protocols because the loading could be dependent on the PLC network and not the I/O Server CPU. In this case, more than one active I/O Server on the same PLC Network can degrade the PLC network and therefore, slow the total response.

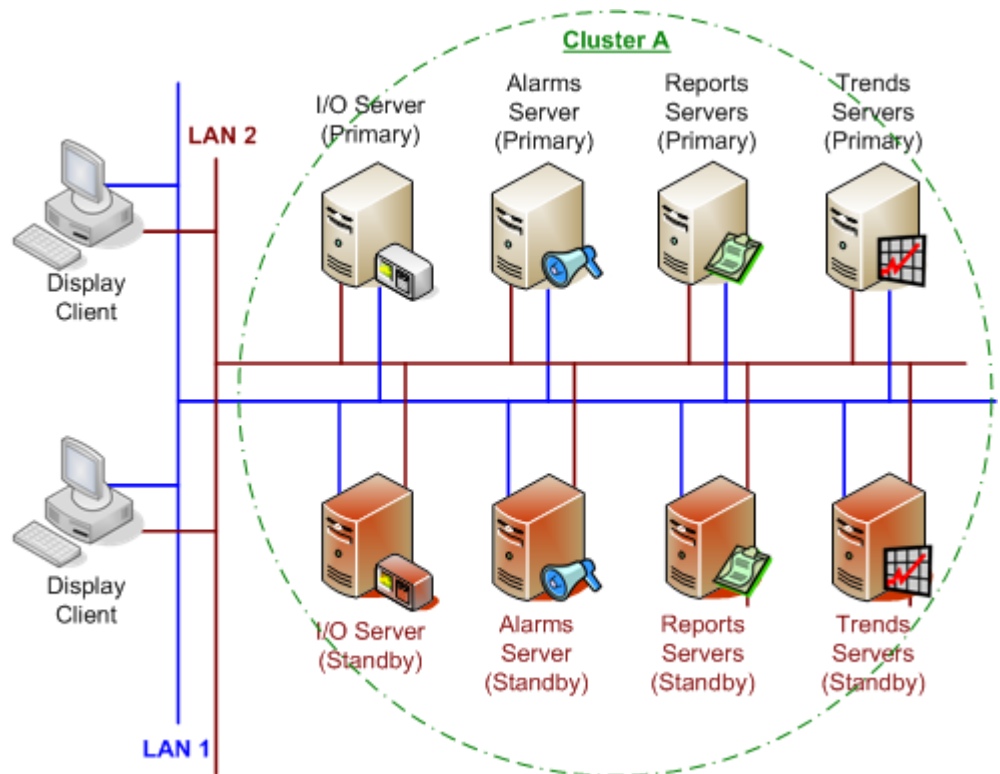
See Also [Alarms, Reports, and Trends Server Redundancy](#)

## Alarms, Reports, and Trends Server Redundancy

Redundancy of Alarms, Reports, Trends, and I/O Servers is achieved by adding standby servers, within a defined cluster, to provide redundant system components. You can also utilize the dual end point (or multiple network interfaces) capabilities of each component, effectively enabling you to specify a complete and unique network connection from a client to a server.

**Note:** TCP/IP now supports NIC redundancy.

Consider the following diagram:



The above example shows the following levels of redundancy:

- LAN Redundancy; and
- Server Redundancy.

### LAN Redundancy

Each of the system components in the cluster are connected to a second LAN (LAN2). This is achieved by utilizing the dual end points of each server. This will provide connection to two separate LANs to provide for LAN redundancy as follows:



- If LAN 1 should suddenly fail, each component in the cluster can easily maintain connection by using LAN 2.
- In turn, should LAN 2 fail, LAN 1 remains in operation.

#### Server Redundancy

Server redundancy of each component in the diagram is achieved by providing a corresponding standby server on the same LAN (LAN 1). Redundancy is provided as follows:

- If any of the servers fail or communications fail, their standby counterpart immediately assumes operation.
- For I/O Server redundancy, when the failed I/O Server comes back online, it resumes control.
- For Alarms, Reports, and Trends Server redundancy, when the failed primary server returns online, the clients stay on the standby Server (unless the standby Reports Server fails in which case it will revert back to the original primary server).

**Note:** Since CitectSCADA maintains the same data on both standby and primary servers, it is not important whether a client receives data from the primary or standby server, and it is quite normal for some clients to be communicating with the primary server, and others with the standby.

See Also

[How CitectSCADA handles alarms server redundancy](#)

#### How CitectSCADA handles alarms server redundancy

It is possible to configure two Alarms Servers in a CitectSCADA project. That is, a primary Alarms Server and a standby Alarms Server. With two Alarms Servers, you have full (mirrored) redundancy on your CitectSCADA system.

When both Alarms Servers are in operation, alarms are processed on both servers in parallel, and are logged by the primary Alarms Server. If the Primary alarms server fails, the Standby alarms server starts to log alarms to devices.

When an Alarms Server first starts up, it tries to establish a connection to an alternate Alarms Server. If it can connect, it transfers the dynamic alarm data from the running Alarms Server to the other Alarms Server (this data includes summary data and the current alarm states). If a connection to an alternate Alarms Server cannot be established, the Alarms Server opens the save file (defined with the [Alarm]SavePrimary parameter) and restores the data from the file. If two save files exist, one from the primary Alarms Server and one from the standby Alarms Server, CitectSCADA uses the save file with the later date, or in other words, the newest save file. If no save file is configured, the Alarms Server cannot obtain the initial status (state) of the alarms and no summary information will be available. If this is the case, the Alarms Server starts processing the alarms, and then acknowledges all the new alarms.

Whilst both Alarms Servers are active, they will both read data from the I/O Server and process the alarms. The on/off status of each alarm is not passed between the two servers. When operators perform functions on alarms (for example, acknowledge, disable, enable, add comments, etc.), this information is passed between the two Alarms Servers (if an operator acknowledges an alarm on one server, that server tells the other server to acknowledge the same alarm).

CitectSCADA clients connect to either the primary Alarms Server or standby Alarms Server. On startup, all clients try to establish a connection with the primary Alarms Server. If a connection with the primary Alarms Server can not be established, they will try to establish a connection with the standby Alarms Server. If the primary Alarms Server reactivates and becomes available, any clients connected to the standby Alarms Server remain connected to the standby Alarms Server (it is of no importance to CitectSCADA which Alarms Server the clients can communicate with, because they both contain the same (mirrored) data).

See Also [How CitectSCADA handles reports server redundancy](#)

#### How CitectSCADA handles reports server redundancy

It is possible to configure two Reports Servers in a CitectSCADA project. That is, a primary Reports Server and a standby Reports Server.

When both Reports Servers are in operation, the scheduled reports only run on the primary Reports Server. If the primary Reports Server fails, the scheduled reports run on the standby Reports Server (you can also configure the standby Reports Server so that it also runs the scheduled reports in parallel with the primary Reports Server). Note that no report data is transferred between the primary and standby Reports Servers (CitectSCADA does not synchronize the report data because reports can write their data to any type of device).

CitectSCADA clients either connect to the primary Reports Server or the standby Reports Server. On startup, all clients try to establish a connection with the primary Reports Server. If a connection with the primary Reports Server cannot be established, then an attempt to establish a connection with the standby Reports Server will be made. If the primary Reports Server reactivates and becomes available, any clients connected to the standby Reports Server remain connected to the standby Reports Server.

See Also [How CitectSCADA handles trends server redundancy](#)

#### How CitectSCADA handles trends server redundancy

It is possible to configure two Trends Servers in a CitectSCADA project. That is, a primary Trends Server and a standby Trends Server.

When both Trends Servers are in operation, trends are processed on both servers in parallel, and written to disk (each server must write to its own disk or its own private area on the [file server](#)).

When a Trends Server starts up, it tries to establish a connection to the other Trends Server. If it can establish a connection, it will transfer all the trend data

from the last time it was shutdown until the current time (this ensures that no trend data is lost).

CitectSCADA clients either connect to the primary Trends Server or the standby Trends Server. On startup, all clients try to establish a connection with the primary Trends Server. If a connection to the primary Trends Server cannot be established, the clients try to connect to the standby Trends Server. If the primary Trends Server reactivates and becomes available, any clients currently connected to the standby Trends Server remain connected to the standby Trends Server (it is of no importance to CitectSCADA which Trends Server the clients can communicate with, because they both contain the same (mirrored) data).

See Also [How CitectSCADA handles file server redundancy](#)

## How CitectSCADA handles file server redundancy

CitectSCADA allows for redundancy of the File Server. The [CtEdit]Backup parameter specifies a backup project path. If CitectSCADA cannot find a file in the Run directory (i.e. as specified by the [CtEdit]Run parameter), it will look in the backup path. If the file is found in the backup path, CitectSCADA will assume that the run path has failed (i.e. the file server has failed). CitectSCADA will then look for all relevant files in the backup before changing over. When CitectSCADA changes over to the backup path, it will call event number 11 and generate the hardware error: **File server failed, to Standby**.

File Server redundancy will only operate correctly if the redirector (or shell) on the computer can handle a failure of the File Server. The shell with Novell Netware cannot do this and will cause Windows to fail with fatal Network errors - when the file server fails. Microsoft LAN manager based networks and peer to peer networks will allow for File Server failure correctly. Therefore, CitectSCADA File Server redundancy will operate correctly with these networks.

**Note:** Only CitectSCADA switches to a backup path. Any other applications that are using files on the File Server will fail when the File Server fails. This may cause the computer to wait for long periods for the File Server (or to crash). This includes Windows itself, so you should install Windows on a local drive.

To enable File Server redundancy, set the [CTEDIT]Backup parameter to a backup database path. For example, if your primary path is **F:\CITECT\USER\DB**, set the backup path to another File Server or a local drive, such as **C:\CITECT\USER\DB**.

You should always make sure that the project in the Backup path is the same as the one in the Run directory - each time you compile the project in the run directory you should copy it into the backup directory.

See Also [How CitectSCADA handles FTP server redundancy](#)

## How CitectSCADA handles FTP server redundancy

CitectSCADA supports FTP Server redundancy. If the primary FTP Server goes down, CitectSCADA will attempt to connect to the FTP Server on the standby machine. This occurs independently of I/O Server redundancy, so the two FTP Servers must have the same passwords and the same directory structure.

FTP Server redundancy is configured by setting parameters in the [CLIENT] and [DNS] sections of the Primary FTP Server's Citect.ini file. These parameters are downloaded by the Internet Display Client (IDC) to its own Citect.ini file if the Primary FTP Server fails, provided the [INTERNET]Redundancy parameter has not been set to 0 (zero). The IDC then uses the downloaded redundancy information to connect to the standby FTP Server.

**Note:** Standby FTP Servers need not be Internet Servers. The Standby FTP Server can be any server using TCP/IP that the IDC can connect to, provided there are IDC licenses present in the network.

See Also [Redundancy of Standalone Systems](#)

## Redundancy of Standalone Systems

If you are using CitectSCADA as a standalone system, in the single process model, you can still achieve redundancy by implementing your standby systems on another CPU, provided you have a multi-CPU system. You may wish to implement this for load-balancing purposes, also.

See Also [Configuring Your System](#)

## Chapter 13: The Computer Setup Editor

---

The Computer Setup Editor is a utility designed to help you manage your CitectSCADA configuration files.

The Computer Setup Editor combines a graphical user interface with an extensive Help system. The graphical interface provides the user with a faster and easier way to view, search and modify parameters within the configuration files. The Help system navigates the user through the parameter reference documentation displaying the topic relevant to the current parameter.

The Computer Setup Editor provides a fast, easy and reliable way to manage your citect.ini configuration. It can be used to:

- Add or delete parameters.
- Change the value of existing parameters including the addition of optional comments.
- Save changes to disk, cancel all changes or backup to a new file.
- Generate a Comparison Report between two separate configuration files, visually highlighting their differences.
- Generate an Analysis Report on a configuration file, which provides a useful summary of parameter settings including validity checks on bounds and paths.

See Also [Getting Started with the Computer Setup Editor](#)  
[Before You Begin](#)  
[Using Computer Setup Editor](#)



## Getting Started with the Computer Setup Editor

This section is designed to get you started with the Computer Setup Editor. It deals with the following topics:

- [About the citect.ini file](#)
- [Before You Begin](#)
  - [Before editing a configuration file](#)
  - [Editing existing configuration files](#)
- [Using the Computer Setup Editor Interface](#)
  - [Expandable tree pane](#)
  - [Tree search pane](#)
  - [Parameter Details pane](#)
  - [Help pane](#)
  - [Parameter Reference pane](#)
  - [Parameter status pane](#)

### About the citect.ini file

The `citect.ini` file is a text file that stores values for a series of operating parameters which are used to fine-tune your system. These values are read in from `citect.ini` by CitectSCADA on startup and determine how the application operates.

During CitectSCADA installation a default `citect.ini` file is copied to the Windows directory. This configuration file contains a series of undocumented parameters used by CitectSCADA as well as a series of default settings.

**Warning!** Undocumented parameters should only be changed on the advice of Citect Support.

The `citect.ini` file has a defined structure made up of a [Header](#), a series of sections containing [Parameters](#) and their values, and [Comments](#). An extract from a `citect.ini` configuration file is shown below.

```
#
#Header Comment on citect.ini file
#[Alarm]
#Set to '1' by System Administrator on 30/06/2005 10:13:44 AM.
Primary = 1
SavePeriod = 600
SaveSecondary =
ScanTime = 500
Server = 1
```

See Also [Header](#)  
[Parameters](#)  
[Comments](#)

## Header

The header of the `citect.ini` file contains comments the user wants to document about the configuration settings contained within. Typically it would include a history of edits made to the file and reasons for those edits. For example:

```
# Originally set up by XX on XX for use on XX
# Modified by XX on XX at the recommendation
of Citect Support
```

Comments are located at the top of the file before the first definition of a section. Each line of the header starts with the “hash” or “pound” character (#).

You can add comments to a header using the Computer Setup Editor. For details see [Entering comments in the header](#).

## Parameters

You can choose from many configuration parameters to help you fine-tune your project. These parameters are documented in the Computer Setup Editor Help, as well as the CitectSCADA Help.

Parameters are broken into sections according to their purpose.

The syntax used in `citect.ini` to define a section is as follows:

```
[Section Name]
<parameter name1> = <parameter value1>
<parameter name2> = <parameter value2>
<parameter nameX> = <parameter valueX>
```

For example:

```
[Alarm]
Primary = 1
SavePeriod = 600
SaveSecondary =
ScanTime = 500
```

The following rules apply when the `citect.ini` is read at runtime:

- A section continues until a new section is defined or the end of file is reached.
- Each parameter definition finishes with a return.
- Any line starting with a hash or pound (#) character is considered to be a comment.
- Any line starting with an exclamation mark (!) is considered to be disabled and therefore treated as a comment.



## Comments

You can add comments to a section or to a parameter setting of the `citect.ini` file.

### Comments for a section

Comments for a section commence with a hash or pound (#) character and occur on the line just above the declaration of the relevant section.

For example:

```
#Alarm Section comment would go here
[Alarm]
```

Comments can be added to a section using the Computer Setup Editor. See [Commenting a section entry](#) for details.

### Comments for a parameter

Comments for a parameter commence with a hash or pound (#) character and occur on the line just above the declaration of the relevant parameter.

For example:

```
[Alarm]
#Set to '1' by System Administrator on 30/06/2005 10:13:44 AM.
Primary = 1
```

You can add comments to a parameter using the Computer Setup Editor. See [Commenting a parameter entry](#) for details.

## Before You Begin

**Before editing a configuration file** Stop your running projects before editing the `citect.ini` file.

Changes to configuration files will only take effect when a project is restarted. For some parameters, changes will only take effect when Citect Explorer is restarted.

### Editing existing configuration files

**Warning!** Computer Setup Editor might lose multi-line comments in existing configuration files. Before editing your configuration files, make sure you create a backup copy.

Computer Setup Editor saves its changes to configuration files such that:

- All parameters re-written to the file are done so in alphabetical order and with the preferred capitalization (if known) for all section and parameter names.
- All comments are limited to a single line.

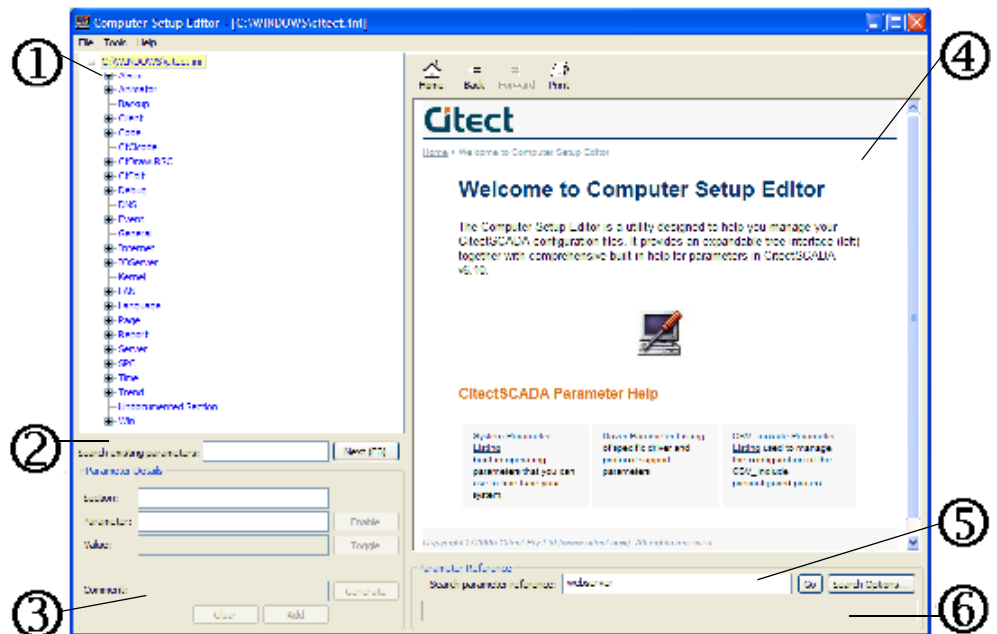
Any multi-line comments in your configuration file will be lost if you perform a save in Computer Setup Editor to the same filename, unless the comments occur

above the first section (denoted the “header”) or are one line long and exist before the entry in the INI file to which they pertain. This is because Computer Setup Editor has its own built-in comment specification whereby comments are assigned to the immediately following item in the INI file and can only be one line long.

To avoid losing comments, make a backup copy of your file before you edit it with Computer Setup Editor and enter the comments via the Computer Setup Editor.

## Using the Computer Setup Editor Interface

The Computer Setup Editor looks like this:



The interface consists of the following components:

- 1 The **expandable tree pane** used to display and navigate the contents of the `citect.ini` file using an expandable contents tree. See [Expandable tree pane](#) for details.
- 2 The **tree search pane** used to search for a particular element in the expandable tree pane. See [Tree search pane](#) for details.
- 3 The **Parameter Details pane** used to view and modify settings for parameters. See [Parameter Details pane](#) for details.

- 4 The **help pane** used to display context-sensitive parameter reference topics. See [Help pane](#) for details.
- 5 The **Parameter Reference pane** used to perform a full text-search on the parameter reference topics in the help pane. See [Parameter Reference pane](#) for details.
- 6 The **parameter status pane** used to display the current status of the parameter displayed in the help pane. See [Parameter status pane](#) for details.

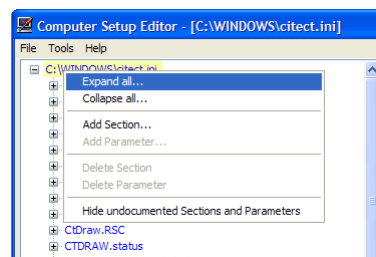
**Note:** You can open the Computer Setup Wizard from the Computer Setup Editor by choosing **Tools | Computer Setup Wizard**.

## Expandable tree pane

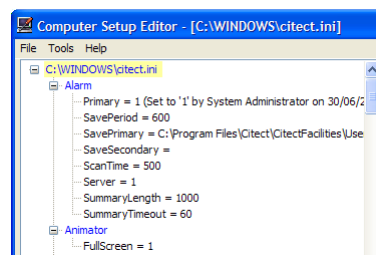
The expandable tree pane is used to display and navigate the contents of the `citect.ini` file using an expandable tree. Each branch on the tree represents a section within the `citect.ini` file, and each leaf item of that branch represents the parameters within that section. You can expand and collapse the tree nodes by clicking the “+” and “-” by each element, just as you would in Windows Explorer.

### To expand the tree:

- 1 Right-click the top element of the tree.
- 2 Choose **Expand all** from the shortcut menu.

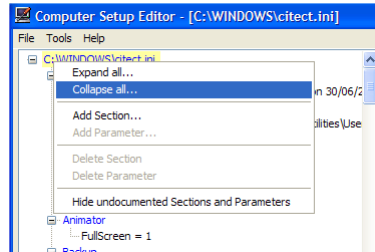


- 3 The configuration parameter tree expands to show all parameter entries.

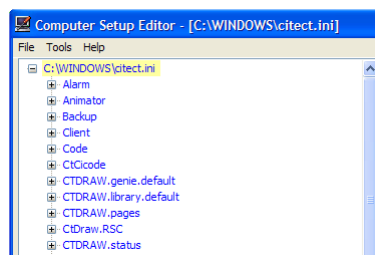


### To collapse the tree:

- 1 Right-click the top element of the tree.
- 2 Choose **Collapse all** from the shortcut menu.



- 3 The configuration parameter tree collapses to show only section entries.

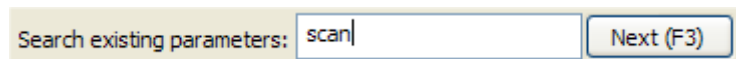


## Tree search pane

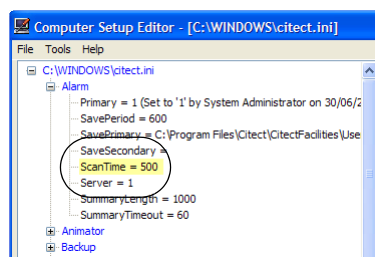
You can use the tree search pane to search for a particular element in the expandable tree pane.

To search for a specific parameter in the tree:

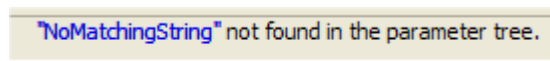
- 1 Type the search string in the tree search pane.



- 2 The tree expands and highlights the first occurrence of the text string searched on.



**Note:** If the search string is not found a message appears in the Parameter Reference pane. In this case, a hyperlink will appear in that pane which can be clicked to perform a search through the help reference topics.



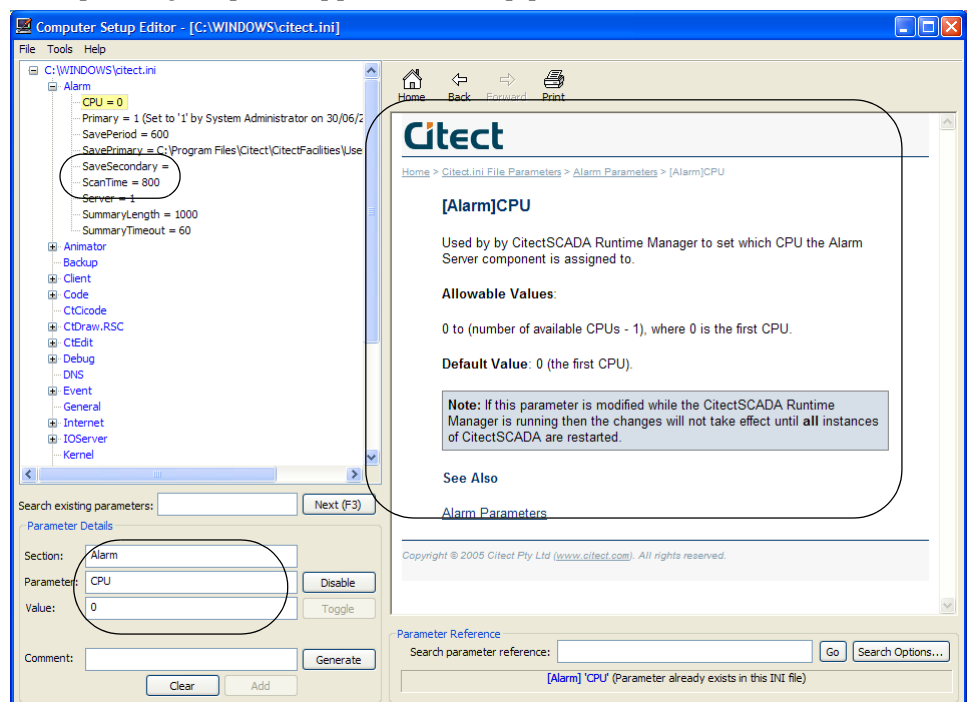
- 3 To continue to search for the next match within the parameter tree, click **Next** in the tree search pane, or press **F3**. To go to the previous instance, press **Shift + F3**.
- 4 If the top or bottom of the Parameter Tree is reached a message appears in the Parameter Reference pane.

End of tree reached whilst searching for "systems administrator"

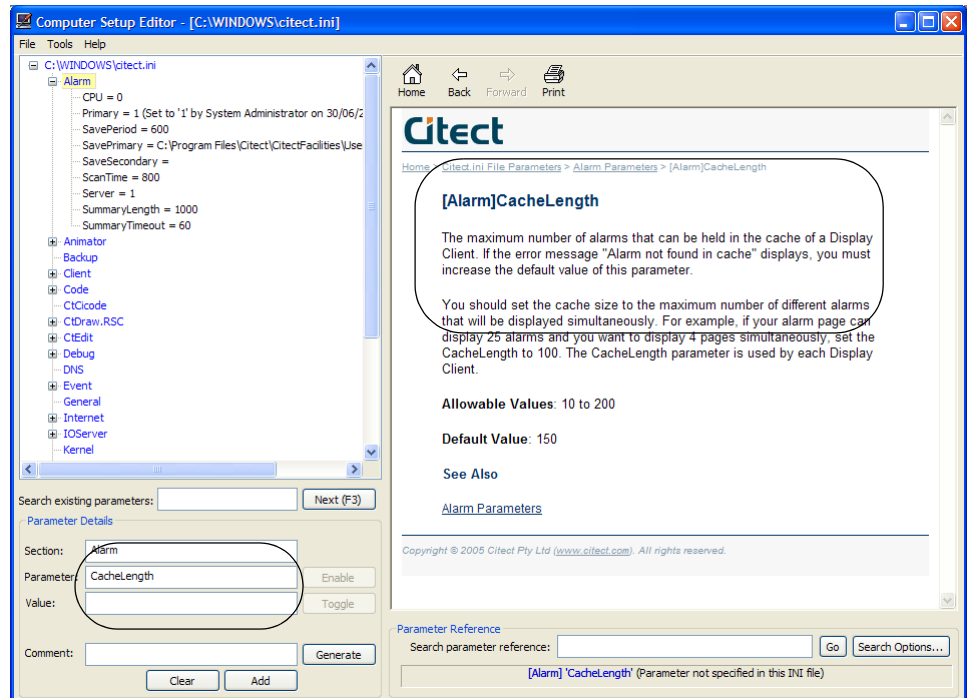
## Parameter Details pane

You can use the Parameter Details pane to view and modify settings and comments for parameters and sections. For details see [Maintaining Sections](#), [Maintaining Parameters](#), and [Commenting the citect.ini file](#).

When a parameter or section entry is highlighted in the tree pane the corresponding values populate the Parameter Details pane and the corresponding Help files appear in the help pane.



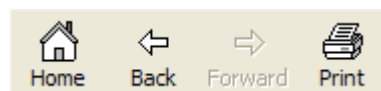
Similarly, the Parameter Details pane populates when a Help topic is selected. This occurs whether or not the parameter exists in the current configuration file.



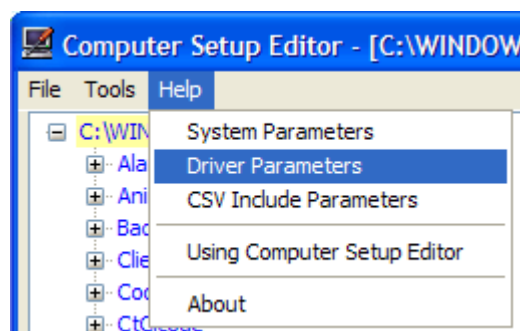
## Help pane

The Help pane displays context-sensitive parameter reference topics.

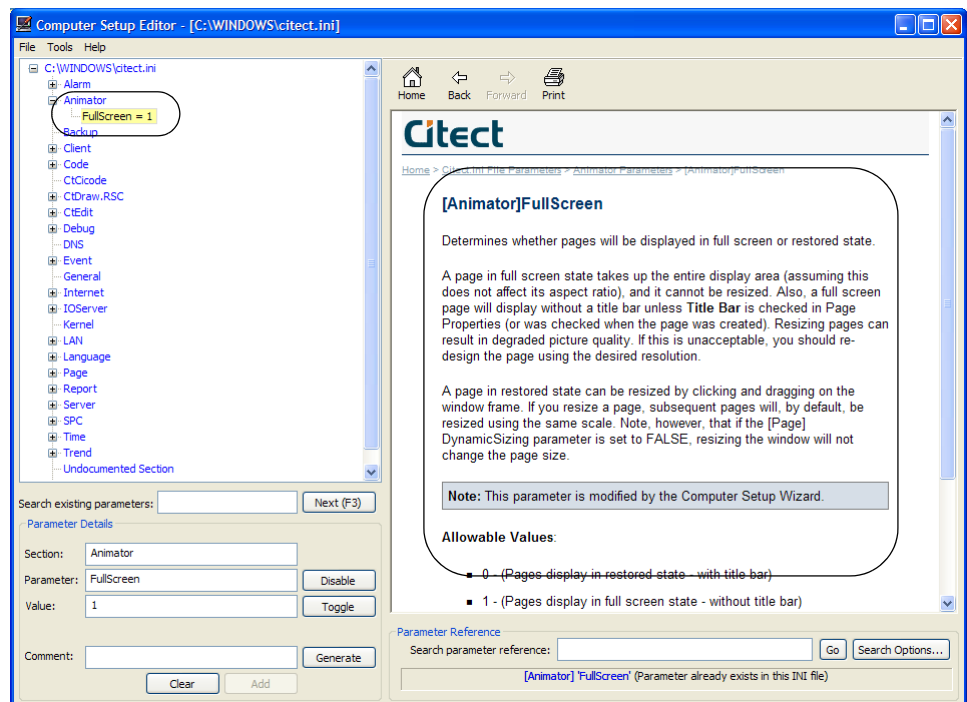
Use the **Back**, **Forward** and **Home** buttons shown in the pane (or the ones on your mouse) to navigate the Help. Click **Print** to print the current page.



To bypass the introductory Help pages and go straight to a specific help topic, choose the relevant Help topic from the **Help** menu.



When a section or parameter entry is selected in the expandable tree pane, the corresponding Help page appears in the help pane. If the selected item does not have a Help page, a message to this effect appears in the help pane.



**Note:** The converse of the above is also true: when a Help topic is selected in the Help pane, the corresponding entry in the expandable tree pane (provided it exists) is automatically selected.

## Parameter Reference pane

You can use the Parameter Reference pane to search the parameter reference topics in the help pane.

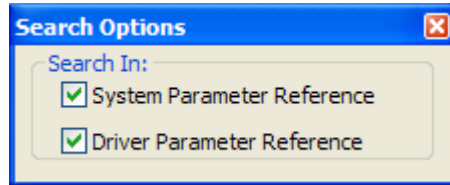
**To find the Help file:**

- 1 Go to the Parameter Reference pane.
- 2 Enter the search string in the Search Parameter Reference text box.



**Note:** The search string is limited to a single word.

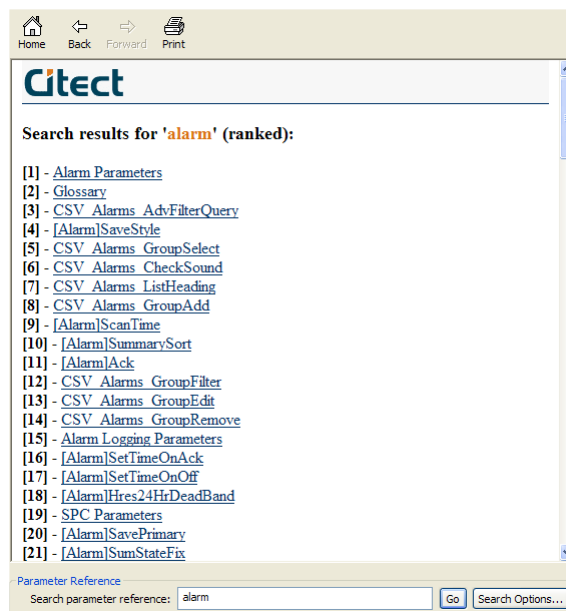
- 3 To limit the search to a particular set of documentation click **Search Options** and indicate which documentation set the search should be limited to, then close the window and continue.



**Note:** The CSV\_Include Parameters are not included in the system parameters search.

- 4 Click **GO**.

A full text search is conducted through the parameter reference help and the results appear in the help pane, ranked from most relevant to least relevant.



- 5 Click an entry in the search results; the corresponding Help topic appears in the help pane.

## Parameter status pane

The parameter status pane becomes active when either:

- A user selects a section or parameter entry in the expandable tree pane.
- A user selects a Help topic in the help pane.

Once active, the parameter status pane displays the name of the section entry being viewed, the name of the parameter entry being viewed (if applicable) and the status of that entry in the current configuration file.

The status shown will depend on whether a help topic exists for this parameter.



---

If there is no help topic the status pane will state the same.

[Time] 'Name' (No help available for this parameter)

If there is a help topic the status pane will state whether:

- The section already exists in this configuration file;
- The parameter already exists in this configuration file;
- The section is not specified in this configuration file; or
- The parameter is not specified in this configuration file.

[Time] 'RTsync' (Parameter not specified in this INI file)

**Note:** Section and parameter names are active links that cause the corresponding help topics to be displayed in the help pane.



## Using Computer Setup Editor

The following topics provide help on how to use the Computer Setup Editor:

<a href="#">"Opening the configuration file"</a>	
<a href="#">"Maintaining Sections"</a>	<a href="#">"Adding a section"</a> <a href="#">"Deleting a section"</a> <a href="#">"Disabling a section"</a> <a href="#">"Enabling a section"</a>
<a href="#">"Maintaining Parameters"</a>	<a href="#">"Adding a parameter"</a> <a href="#">"Editing a parameter"</a> <a href="#">"Deleting a parameter"</a> <a href="#">"Disabling a parameter"</a> <a href="#">"Enabling a parameter"</a> <a href="#">"Viewing undocumented parameters"</a>
<a href="#">"Commenting the citect.ini file"</a>	<a href="#">"Commenting a section entry"</a> <a href="#">"Commenting a parameter entry"</a> <a href="#">"Entering comments in the header"</a>
<a href="#">"Analyzing the citect.ini file"</a>	<a href="#">"Running a Configuration Analysis Report"</a>
<a href="#">"Using the Comparison Wizard"</a>	<a href="#">"Running the Comparison Wizard"</a> <a href="#">"Interpreting the Comparison Wizard"</a> <a href="#">"Copying values between the files"</a> <a href="#">"Saving changes to the compared file"</a> <a href="#">"Closing without saving changes"</a>
<a href="#">"Saving changes to the citect.ini file"</a>	<a href="#">"Saving the changes to your citect.ini file"</a> <a href="#">"Saving a backup of your changes to the configuration file"</a> <a href="#">"Abandoning the changes to your configuration file"</a>

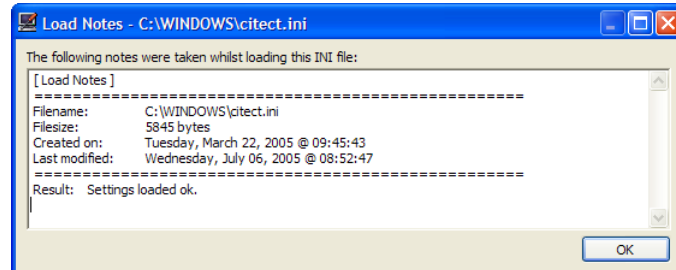
## Opening the configuration file

To open a configuration file:

- 1 Choose **Open** from the **File** menu.
- 2 Highlight the required configuration file from the file selection window, then click **Open**.
- 3 The configuration file will open and display in the expandable tree pane.

To view load notes:

- 1 Choose **View Load Notes** from the **File** menu.
- 2 A Load Notes window opens and displays any notes taken while loading the configuration file.



- 3 Click **OK** to close the window.

**Note:** The Load Notes can also be viewed from the INI Analysis Report. For details see [“Analyzing the citect.ini file”](#).

## Maintaining Sections

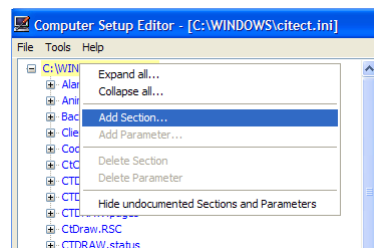
You can maintain the sections settings in your configuration file by:

- "Adding a section"
- "Deleting a section"
- "Disabling a section"
- "Enabling a section"

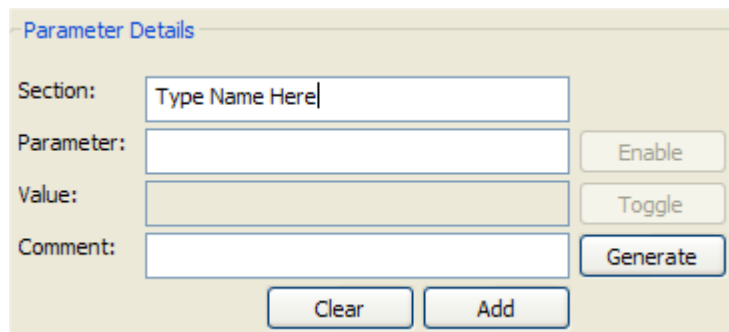
## Adding a section

**To add a section:**

- 1 Right-click on an entry in the expandable tree pane.
- 2 Choose **Add Section** from the shortcut menu.



- 3 The cursor moves to the **Section** text box in the Parameter Details pane. Type the name of the section you want to add.

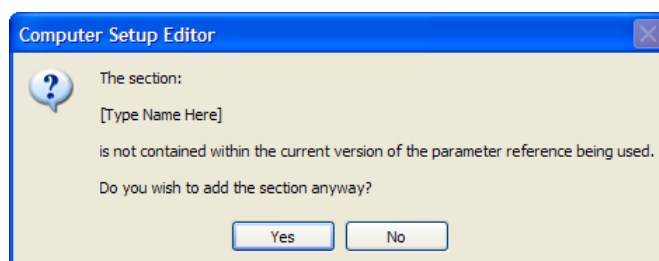


The **Parameter Details** dialog box contains the following fields and buttons:

- Section:** A text input field containing the placeholder text "Type Name Here".
- Parameter:** An empty text input field.
- Value:** An empty text input field.
- Comment:** An empty text input field.
- Buttons:**
  - Enable:** A button located to the right of the Parameter field.
  - Toggle:** A button located to the right of the Value field.
  - Generate:** A button located to the right of the Comment field.
  - Clear:** A button located at the bottom center.
  - Add:** A button located at the bottom center, to the right of the Clear button.

Two shortcuts exist to reach this step:

- Enter a new section name in the Parameter Details pane. The Computer Setup Editor activates the **Add** button.
  - Navigate the Help reference topics within the help pane to locate the topic for the section to be added to the parameter tree, and click the relevant topic. The Computer Setup Editor populates the Parameter Details pane accordingly.
- 4 Click **Add** to add the new section to the parameter tree.
  - 5 The Computer Setup Editor checks to see whether the section name is recognized in the parameter reference topics. A warning window opens if the name specified is not recognized. Click **Yes** to proceed.



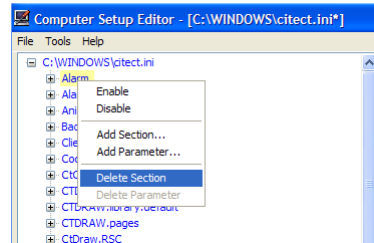
The new section appears in the parameter tree. It is highlighted as the currently selected entry, and its Help appears in the help pane.

## Deleting a section

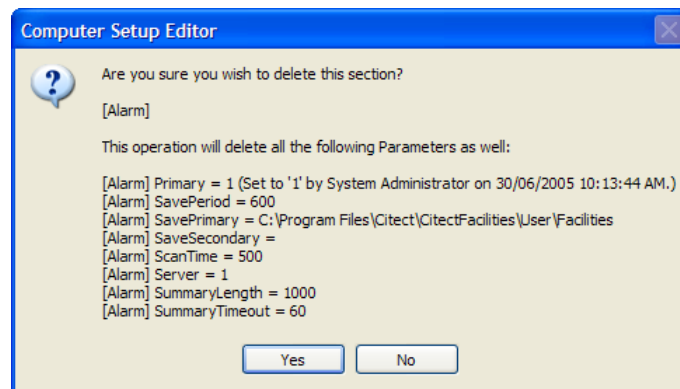
Deleting a section automatically deletes any parameters within that section.

### To delete a section:

- 1 Right-click the section to be deleted in the expandable tree pane.
- 2 Choose **Delete Section** from the shortcut menu.



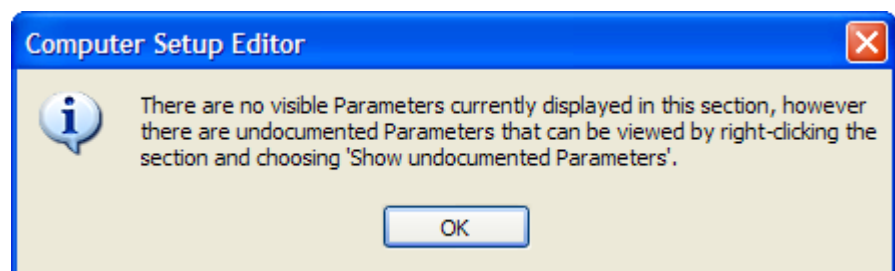
- 3 The Computer Setup Editor opens a warning window requesting confirmation of this action. Click **Yes** to continue.



**Warning!** If undocumented parameters are being viewed, they are also deleted from this section. See [“Viewing undocumented parameters”](#) for details.

- 4 The section is deleted from the parameter tree.

**Note:** If there are undocumented parameters not currently visible in the tree that exist in the file, an alert appears and the undocumented parameters are not deleted. See [“Viewing undocumented parameters”](#) for details.



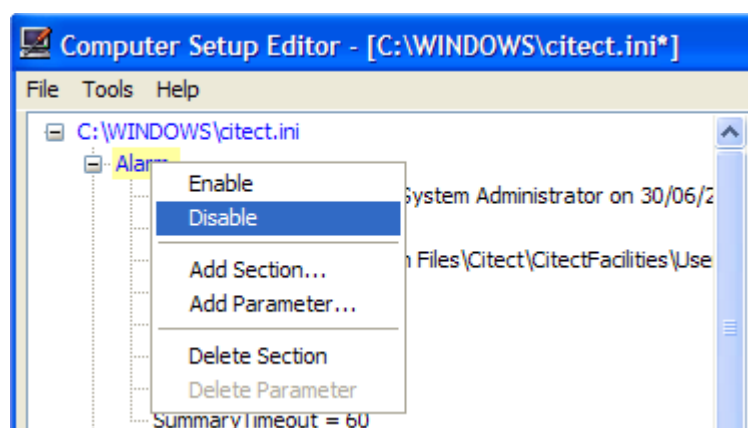
## Disabling a section

Disabling a section within the parameter tree causes the section and all associated parameters to be treated as a comment in the `citect.ini` file when the file is read at startup time.

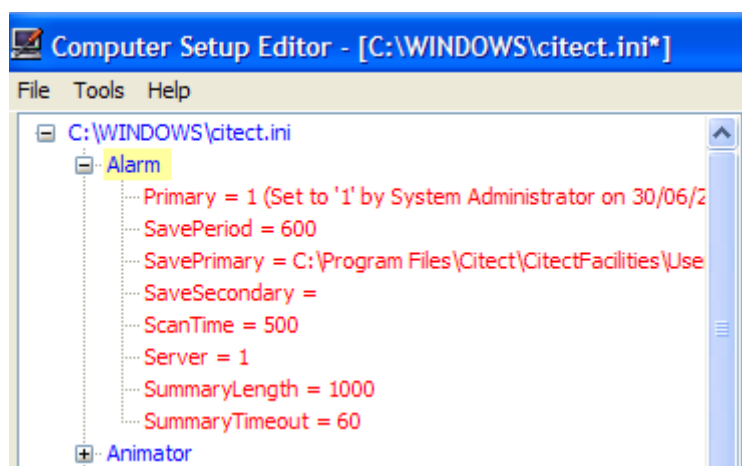
**Note:** Disabled sections are not deleted from the file and can be enabled later. See [“Enabling a section”](#) for details.

To disable a section:

- 1 Right-click the section you want to disable in the expandable tree pane.
- 2 Choose **Disable** from the shortcut menu.



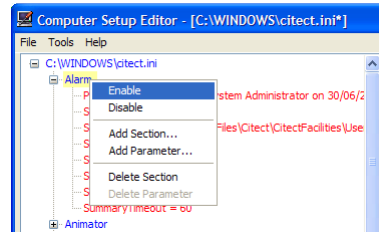
- 3 The section is disabled and the text in that section within the parameter tree changes to red to indicate its disabled status.



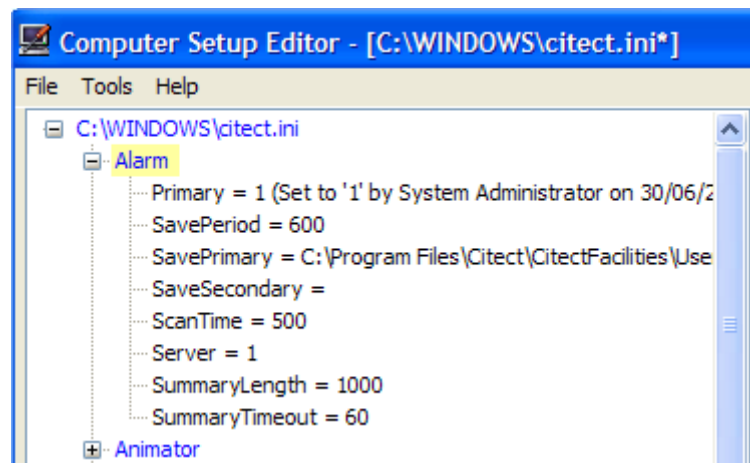
## Enabling a section

To enable a section:

- 1 Right-click the section you want to enable in the expandable tree pane.
- 2 Choose **Enable** from the shortcut menu.



- 3 The section is enabled and the text in that section within the parameter tree changes to black to indicate its enabled status.



## Maintaining Parameters

You can maintain the parameter settings in your configuration file by:

- [“Adding a parameter”](#)
- [“Editing a parameter”](#)
- [“Deleting a parameter”](#)
- [“Disabling a parameter”](#)
- [“Enabling a parameter”](#)
- [“Viewing undocumented parameters”](#)

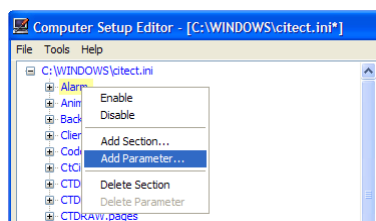
### Adding a parameter

To add a parameter:

- 1 Right-click the section in the expandable tree pane you want to add the parameter to.



- 2 Choose **Add Parameter** from the shortcut menu.



- 3 The section name pre-populates the Parameter Details pane. Type the name of the parameter you want to add in the **Parameter** text box.

 A screenshot of the 'Parameter Details' pane. It contains four text boxes: 'Section:' with 'Alarm' entered, 'Parameter:' with 'type new parameter name here' entered, 'Value:' which is empty, and 'Comment:' which is empty. To the right of the 'Parameter' box is an 'Enable' button, to the right of the 'Value' box is a 'Toggle' button, and to the right of the 'Comment' box is a 'Generate' button. At the bottom are 'Clear' and 'Add' buttons.

Two shortcuts exist to reach this step:

- Locate the section for the new parameter in the expandable tree pane, click it, then enter a new parameter name in the **Parameter** text box of the Parameter Details pane. The Computer Setup Editor activates the add button.
- Navigate the Help reference topics within the help pane to locate the topic for the parameter to be added to the parameter tree, and click the relevant topic. The Computer Setup Editor populates the Parameter Details pane. (In the case of Driver parameter reference topics - click on the **Add Parameter Wizard** link to populate the Parameter Details pane.)

## [ADISMC]Block

Add Parameter Wizard

A value (bytes) used by the I/O Server to determine if two or more packets can be blocked into one data request before being sent to the I/O Device. For example, if you set the value to 10, and the I/O Server receives two simultaneous data requests - one for byte 3, and another for byte 8 - the two requests will be blocked into a single physical data request packet. This single request packet is then sent to the I/O Device, saving on bandwidth and processing.

Allowable Values 5 to 256

Default Value 256

- 4 Enter the value for this parameter in the value field.

The Parameter Details pane changes to reflect the type the parameter:

- If the parameter has a path value a **Browse** button becomes available to help you set this parameter.

Parameter Details

Section:	<input type="text" value="Alarm"/>	
Parameter:	<input type="text" value="SavePrimary"/>	<input type="button" value="Enable"/>
Value:	<input type="text"/>	<input type="button" value="Browse..."/>
Comment:	<input type="text"/>	<input type="button" value="Generate"/>
<input type="button" value="Clear"/> <input type="button" value="Add"/>		

- If the parameter has a Boolean value, a **Toggle** button becomes available to help you set this parameter.

Parameter Details

Section:	<input type="text" value="General"/>	
Parameter:	<input type="text" value="CheckAddressBoundary"/>	<input type="button" value="Enable"/>
Value:	<input type="text"/>	<input type="button" value="Toggle"/>
Comment:	<input type="text"/>	<input type="button" value="Generate"/>
<input type="button" value="Clear"/> <input type="button" value="Add"/>		

- If the parameter has a string value, the button to the left of the field becomes unavailable. Type the value in the **Value** field.

The image shows a 'Parameter Details' dialog box. It contains four input fields: 'Section' with the value 'Alarm', 'Parameter' with the value 'Active', 'Value' which is empty, and 'Comment' which is empty. To the right of the 'Parameter' field is an 'Enable' button. To the right of the 'Value' field is a 'Toggle' button, which is circled in red. Below the 'Value' and 'Comment' fields is a 'Generate' button. At the bottom of the dialog are 'Clear' and 'Add' buttons.

- 5 Click **Add** to add the new parameter to the parameter tree.

The new parameter appears in the parameter tree. It is highlighted as the currently selected entry, and its Help appears in the help pane.

If the parameter being added to the tree belongs to a section yet to be defined in the tree, both the section and parameter are added to the parameter tree when you click **Add**.

If the parameter being added to the parameter tree is an undocumented parameter, it appears in the parameter tree. However, if you hide undocumented parameters, you'll only be able to see parameters when you view undocumented parameters. See ["To show undocumented parameters:"](#) for details.

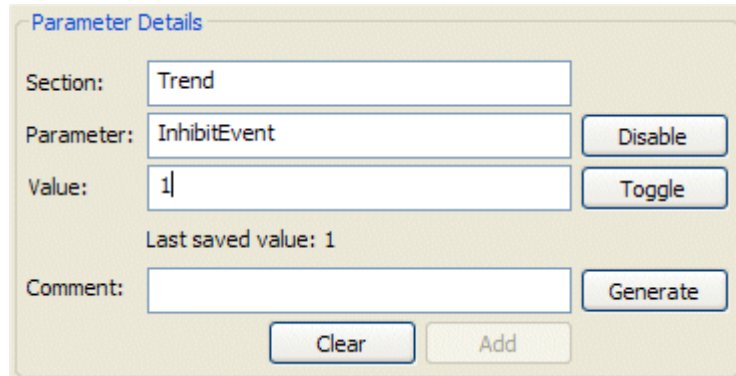
## Editing a parameter

To edit a parameter value:

- 1 Click the parameter in the expandable tree pane.
- 2 The current value populates the Parameter Details pane.

The image shows the 'Parameter Details' dialog box after editing. The 'Section' field now contains 'Trend', the 'Parameter' field contains 'InhibitEvent', and the 'Value' field contains '1'. The 'Comment' field is empty. The buttons to the right of the fields are 'Disable' (next to Parameter), 'Toggle' (next to Value), and 'Generate' (next to Comment). At the bottom are 'Clear' and 'Add' buttons.

- Click in the **Value** text box. The last saved value in the `citect.ini` file will be displayed in the area directly below the text box.



The **Parameter Details** dialog box contains the following fields and buttons:

- Section:** Trend
- Parameter:** InhibitEvent
- Value:** 1
- Last saved value:** 1
- Comment:** (empty text box)
- Buttons:** Disable, Toggle, Generate, Clear, Add

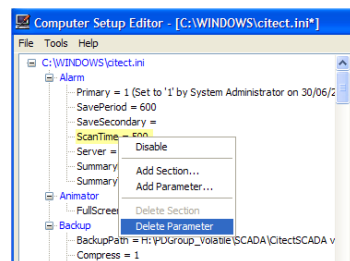
- Change the entry in the **Value** text box to the required value. The changes automatically appear in the parameter tree.

**Note:** The changes will not occur to the configuration file until they are saved. See [“Saving the changes to your citect.ini file”](#) for details.

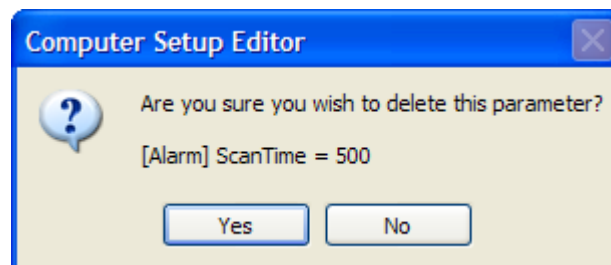
## Deleting a parameter

To delete a parameter value:

- Right-click the parameter in the expandable tree pane.
- Choose **Delete Parameter** from the shortcut menu.



- A window opens requesting confirmation to delete the parameter.



- Click **Yes**. The parameter is deleted from the parameter tree.

**Note:** Deleting parameters from the configuration file sets them to their default value if one exists.

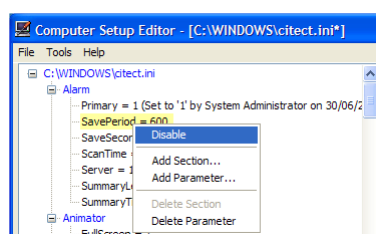
## Disabling a parameter

Disabling a parameter within the parameter tree causes the parameter to be treated as a comment in the `itect.ini` file when the file is read at startup time.

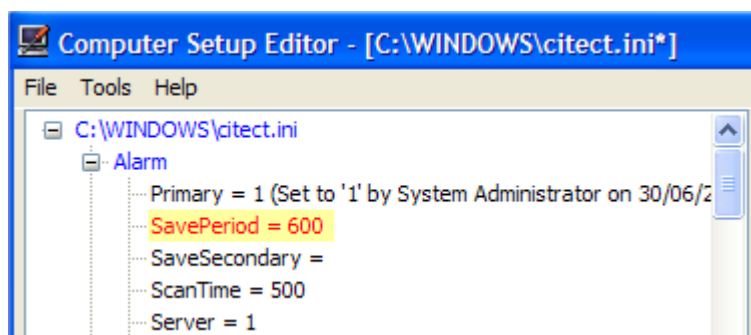
Parameter values which are disabled are not deleted from the file and can be enabled at a later time. See [“Enabling a parameter”](#) for details.

To disable a parameter:

- 1 Right-click the parameter you want to disable in the expandable tree pane.
- 2 Choose **Disable** from the shortcut menu.



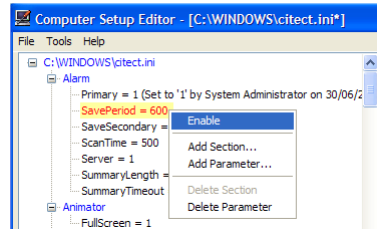
- 3 The parameter is disabled and the text for that parameter within the parameter tree changes to red to indicate it is now disabled.



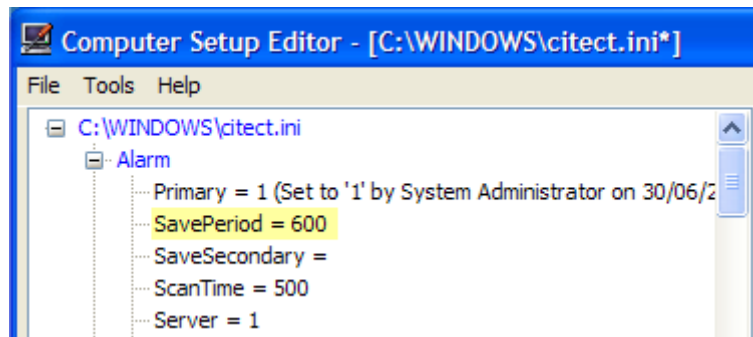
## Enabling a parameter

To enable a parameter:

- 1 Right-click the parameter you wish to enable in the expandable tree pane.
- 2 Choose **Enable** from the shortcut menu.



- 3 The parameter is enabled and the text for that parameter within the parameter tree changes to black to indicate it is now enabled.

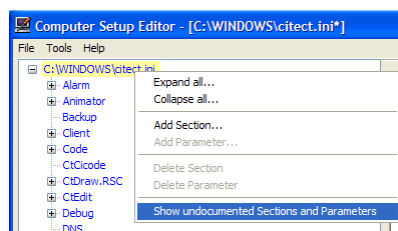


### Viewing undocumented parameters

When a configuration file is loaded by the Computer Setup Editor, only those parameters that are documented in the Parameter Help are visible in the parameter tree. There are, however, many other parameters (mostly used by CitectSCADA itself) for which no help topics exist; these are known as undocumented parameters.

To show undocumented parameters:

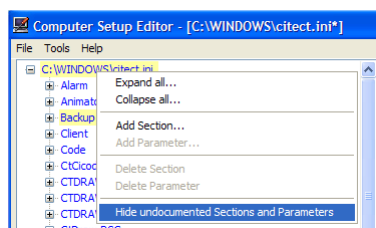
- 1 Right click the top element of the parameter tree.
- 2 Choose **Show undocumented Sections and Parameters** from the shortcut menu.



- 3 The parameter tree expands to show all undocumented sections and parameters.

To hide undocumented parameters:

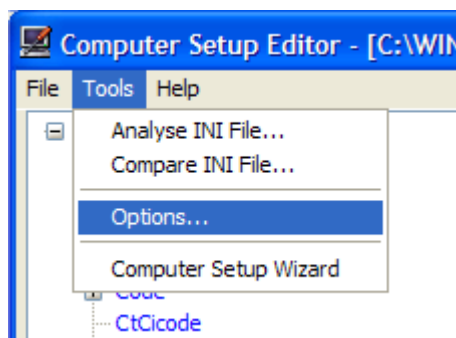
- 1 Right click the top element of the parameter tree.
- 2 Choose **Hide undocumented Sections and Parameters** from the shortcut menu.



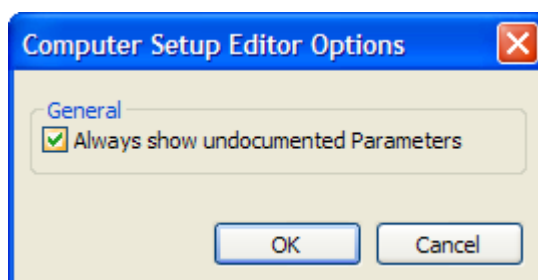
- 3 The undocumented sections and parameters are hidden.

Setting the default option for viewing undocumented parameters

- 1 Choose **Options** from the **Tools** menu.



- 2 The Computer Setup Editor Options window opens. Select the checkbox to display undocumented parameters by default.



- 3 Click **OK**. When Computer Setup Editor next opens a configuration file, this setting will be used to determine whether undocumented parameters are displayed in the parameter tree.

## Commenting the citect.ini file

You can comment your configuration file by:

- [“Commenting a section entry”](#)
- [“Commenting a parameter entry”](#)
- [“Entering comments in the header”](#)

### Commenting a section entry

To add a comment to a section entry:

- 1 Click the section in the expandable tree pane you want to add a comment to.
- 2 The Parameter Details pane populates with the section information.

Parameter Details

Section: Animator

Parameter:

Value:

Comment: Type Comment Here

Buttons: Enable, Toggle, Generate, Clear, Add

- 3 Type the comment in the **Comment** field, or, click **Generate** to insert the auto-generated comment “Added by {username} on {date}”.

Parameter Details

Section: Animator

Parameter:

Value:

Comment: Added by administrator on 5/07/2005 11:

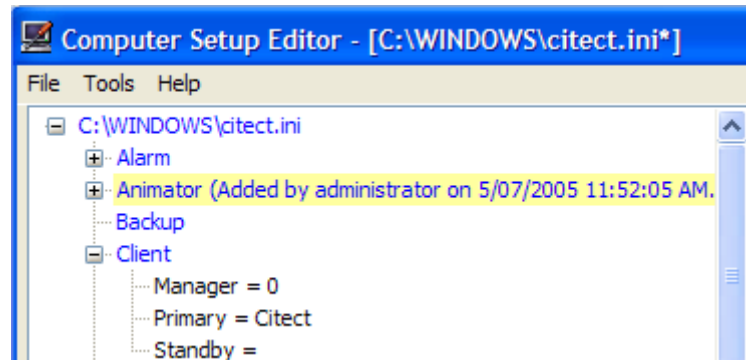
Buttons: Enable, Toggle, Generate, Clear, Add

**Note:** Comments can only be one line long. If the comment needs to be longer than one line, you should include it in the header of the file. See [“Entering comments in the header”](#) for details.

**Warning!** Using the **Generate** button replaces any existing comment made in relation to that section. To append comments type your text into **Comment** field.

- 4 The comment appears next to the section entry in the parameter tree.

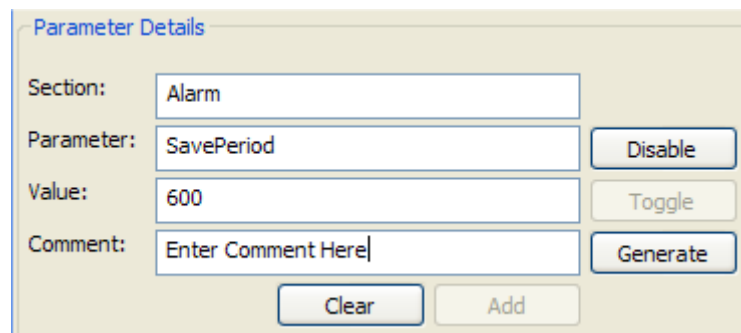




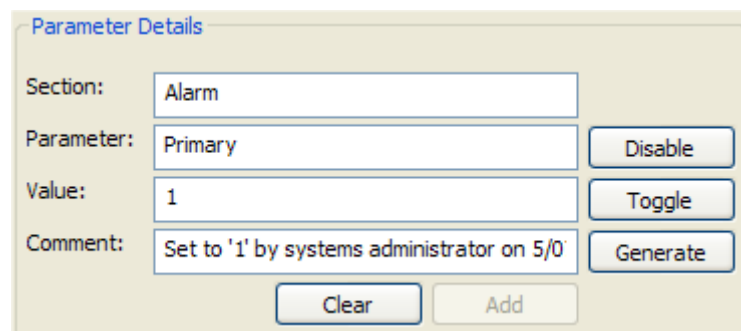
### Commenting a parameter entry

To add a comment to a parameter entry:

- 1 Click the parameter in the expandable tree you want to add a comment to.
- 2 The Parameter Details pane populates with the parameter information.

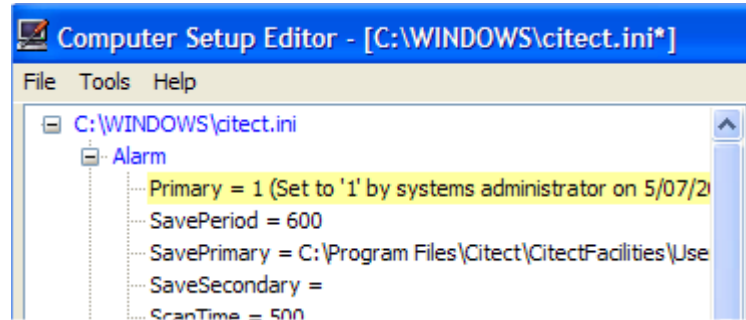


- 3 Type the comment in the **Comment** field, or, click **Generate** to insert the auto-generated comment "Set to {value} by {username} on {date}".



**Note:** Comments can only be one line long. If the comment needs to be longer than one line, you should include it in the header comment. See ["Entering comments in the header"](#) for details.

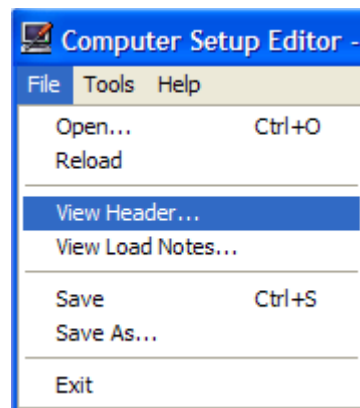
- 4 The comment appears next to the parameter entry in the parameter tree.



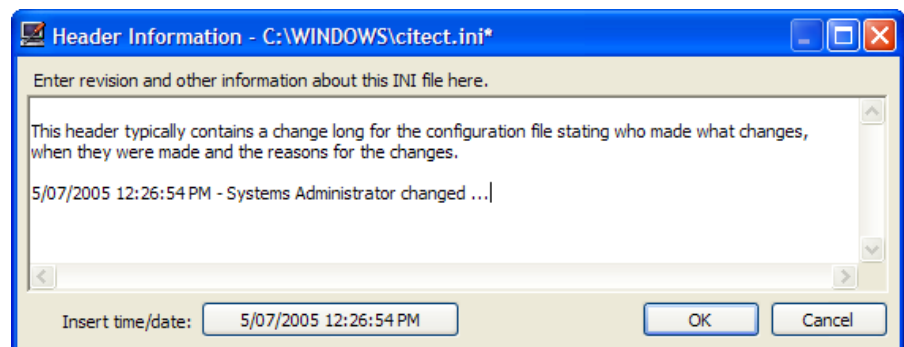
Entering comments in the header

To enter comments in the header:

- 1 Choose **View Header** from the **File** menu.



- 2 The Header Information window opens. Comments added in this window are stored in the header of the configuration file.



**Note:** Click **Insert time/date** to enter the current date and time where the cursor is placed in the text window.

- 3 Click **OK** to store the changes made to the header file.

**Note:** The Header information can also be viewed from the INI Analysis Report. See [“Analyzing the citect.ini file”](#) for details.

## Analyzing the citect.ini file

The INI Analysis Report is a visual summary of the contents of the `citect.ini` file that contains:

- A checkbox summary of server, client and network configuration.
- A listing of parameters that specify host (computer) names.
- A listing of drivers for which parameters have been specified.
- A listing of parameters that specify local and UNC paths.
- An analysis summary displaying the results of:
  - Bounds checking on numeric parameters to ensure the current value is within the allowable range.
  - Type checking on numeric parameters to ensure all assigned values are numeric.
  - Path checking on all path parameters to ensure that they are both valid and accessible from the current computer.

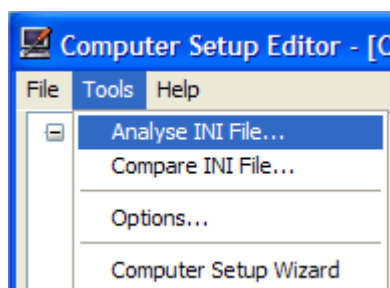
This information is intended to help diagnose the cause of runtime problems that may arise from invalid parameter values.

See [“Running a Configuration Analysis Report”](#).

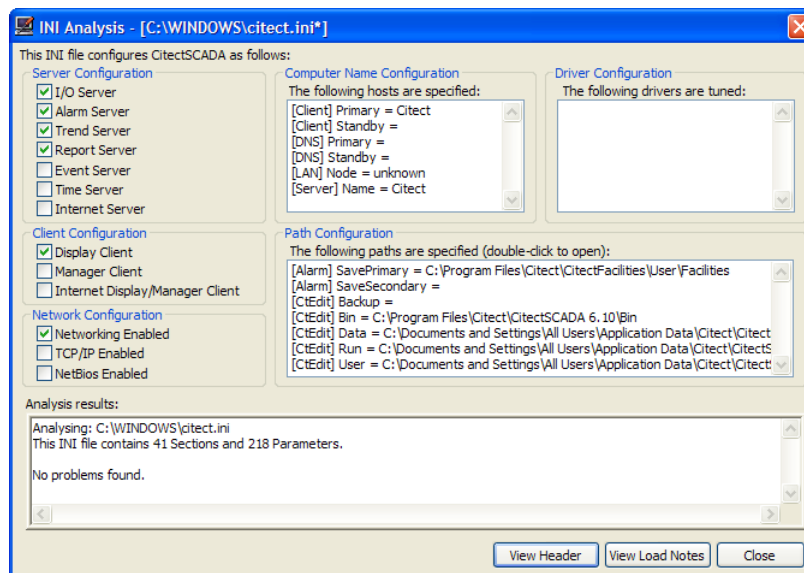
## Running a Configuration Analysis Report

### To run a Configuration File Analysis Report

- 1 Open the configuration file to be analyzed in the Computer Setup Editor.
- 2 Choose **Analyse INI File** from the **Tools** menu.



- 3 The INI Analysis Report opens.



- 4 Click **Close** to exit the report.

**Note:** Double clicking a driver entry in the Driver Configuration pane will shut the Analysis Report, select the driver section in the parameter tree and display the corresponding driver parameter help page in Computer Setup Editor.

## Using the Comparison Wizard

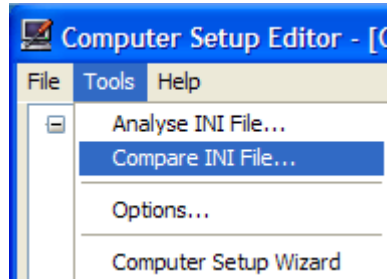
The Comparison Wizard allows you to compare the files across two separate configuration files. For further information see:

- [“Running the Comparison Wizard”](#)
- [“Interpreting the Comparison Wizard”](#)
- [“Copying values between the files”](#)
- [“Saving changes to the compared file”](#)
- [“Closing without saving changes”](#)

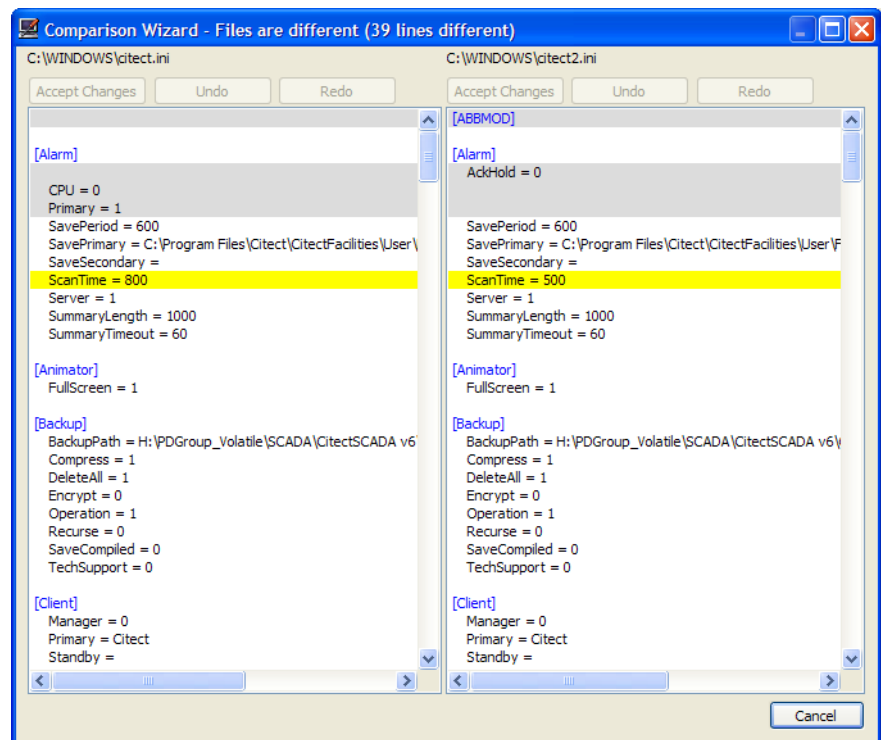
### Running the Comparison Wizard

#### To run the Comparison Wizard

- 1 Open the source configuration file in the Computer Setup Editor.
- 2 Choose **Compare INI File** from the **Tools** menu.



- 3 A **Choose comparison file** window opens. Navigate to the file you want to compare (**comparison file**) with the file currently loaded in the Computer Setup Editor (**source file**). Click **Open**.
- 4 The Comparison Wizard opens in a new window.

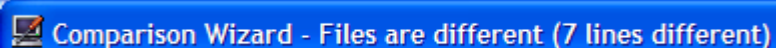


### Interpreting the Comparison Wizard

The Comparison Wizard displays the source file and the comparison file side by side.

Number of differences between the files

A summary of the number of lines different between the files is shown in the title of the window.



Comparison Wizard - Files are different (7 lines different)

Parameters with the same values

Lines that are not highlighted indicate that the parameters are set in both files and have the same values.



SummaryTimeout = 60      SummaryTimeout = 60

Parameters with different values

Lines that are highlighted in yellow indicate that the parameters are set in both files but have different values.



ScanTime = 800      ScanTime = 500

Parameters not set in both files

Lines that are highlighted in silver indicate that the parameters are set in only one of the files.



[Alarm]      [Alarm]  
Primary = 1      AckHold = 0

## Copying values between the files

Where parameter values are different, the Comparison Wizard allows changes to be made to either file to make them the same.

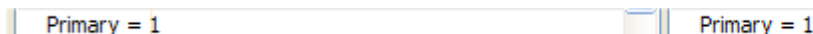
To copy a value from the source file to the comparison file:

- 1 In the source file locate the parameter you want to copy to the comparison file.
- 2 Right-click the line, it is highlighted in blue and a menu appears with **Copy to RHS**. Select this menu.



Primary = 1      Copy to RHS -->      SavePeriod = 600

- 3 The Parameter value copies to the comparison file and the line is no longer highlighted.

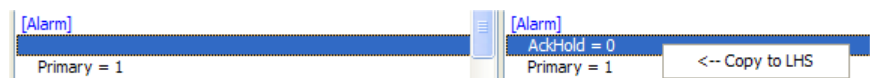


Primary = 1      Primary = 1

**Note:** To select multiple lines in the source file use the **Ctrl** and/or **Shift** keys in combination with the mouse, and then right-click any line of the selected lines and choose **Copy to RHS**.

To copy a value from the comparison file to the source file:

- 1 In the comparison file and locate the parameter you want to copy to the source file.
- 2 Right-click the line, it is highlighted blue and a menu appear with **Copy to LHS**. Select this menu.



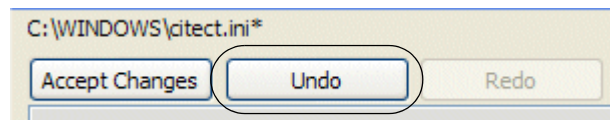
- 3 The Parameter value copies to the Source File and the line is no longer highlighted.



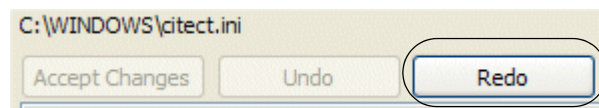
**Note:** To select multiple lines in the comparison file use the **Ctrl** and/or **Shift** keys in combination with the mouse, and then right-click any line of the selected lines and choose **Copy to LHS**.

## Undoing and redoing changes to the compared files

Where parameter values have been copied from one file to another, the **Undo** button becomes active providing you with an option to undo the change.



Undoing the change will return the file to the state before the copy was executed. The undo buffer is unlimited so all changes made in the file can be undone in the same sequence as they occurred. Once the Undo button has been used the **Redo** button becomes active.



Redoing the change will return the file to the state before the last change which was undone. The redo buffer is unlimited, so all undo actions can be redone in the same sequence as they occurred.

**Note:** As soon as a fresh copy between the files is executed the Redo buffer is reset and all previous undo actions can no longer be redone with the **Redo** button.

## Saving changes to the compared file

Changing either the source file or the comparison file activates the **Accept Changes** button. To save changes, click **Accept Changes** located above the relevant file.

Changes accepted to the comparison file are saved directly to the file.

Changes accepted to the Source File are only made to the copy rendered in the expandable tree pane. Use the **Save** command to save the changes to the file. See [“Saving the changes to your citect.ini file”](#) for details.

## Closing without saving changes

To close the Comparison Wizard without saving your changes, click **Cancel** located in the lower right corner of the window.

## Saving changes to the citect.ini file

For information on how the Computer Setup Editor saves changes to your configuration file see:

- [“Saving the changes to your citect.ini file”](#)
- [“Saving a backup of your changes to the configuration file”](#)
- [“Abandoning the changes to your configuration file”](#)

## Saving the changes to your citect.ini file

To save changes to the configuration file:

- 1 Choose **Save** from the **File** menu.
- 2 A warning windows opens requesting confirmation that the changes are to be saved. Click **OK**.
- 3 The changes to the configuration file are saved to file.

In some cases, after saving changes, the project will need to be recompiled before being run again.

**Warning!** Don't change your configuration file while your project is running.

## Saving a backup of your changes to the configuration file

To save a backup of changes to the configuration file:

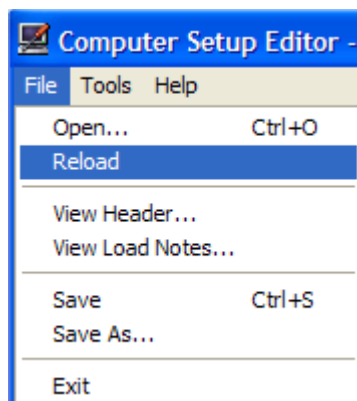
- 1 Choose **Save As** from the **File** menu.
- 2 A Save As windows opens. Enter the name and location of the backup file. Click **Save**.
- 3 A warning windows opens requesting confirmation of the name and location of the file. Click **OK** to proceed.
- 4 A confirmation window opens to confirm the backup has occurred. Click **OK** to continue.



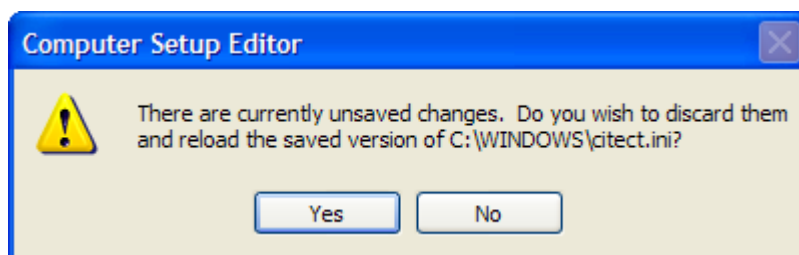
## Abandoning the changes to your configuration file

To abandon changes to the configuration file:

- 1 Choose **Reload** from the **File** menu.



- 2 If there are unsaved changes to the currently loaded file, a warning window opens requesting confirmation that the unsaved changes are to be discarded. Click **Yes** to proceed.



- 3 The configuration file will be reloaded without the unsaved changes.



# Chapter 14: Tagging Process Variables

---

You must assign a variable tag to each [I/O device](#) variable that CitectSCADA uses in your runtime system. To define your variable tags, you declare them in the variable tag database. The variable tag becomes a label, used to reference the address of the I/O device register. Using labels has several benefits:

- You do not have to remember the address every time you want to use the variable. You use the tag name, which should be logical and descriptive, and therefore less confusing.
- The address in the I/O device is defined only once. If you change the address, you only need to update the variable tag definition, not every instance in your configuration.
- You can scale the raw data to an appropriate range in the same declaration.

You must define your variables as a specific data type. The most common variables supported by I/O devices are digital and integer. CitectSCADA also supports real, string, byte, bcd, long, and longbcd data types.

After you have defined your variable tags, you can use them to:

- Display objects on a [graphics page](#).
- Store data for trending and analysis. (See [Trending Data](#).)
- Monitor Alarms. (See [Configuring and Processing Alarms](#).)
- Control equipment and processes. (See [Defining Commands and Controls](#).)
- Store data in memory (See [Configuring Local Variables](#).)

The most common variables supported by I/O devices are digital and integer variables, although some I/O devices support other numeric variables and strings.

See Also [Tag name syntax](#)  
[Using structured tag names](#)  
[Configuring Local Variables](#)  
[Configuring Variable Tags](#)  
[Formatting numeric variables](#)  
[Using arrays](#)

## Tag Naming

CitectSCADA puts a couple of restrictions on the names of variable tags:

- Tags are restricted to using a specific syntax. See [Tag name syntax](#).
- Tags should not have the same name as Cicode functions within the project or within any included projects. An error will result during compilation if a tag has the same name as a Cicode function and is placed on a graphic page.

In addition, using a tag naming convention will make your project easier and faster to design, configure, commission, and maintain. See [Using structured tag names](#) for recommendations about naming conventions.

## Tag name syntax

CitectSCADA tags (variable tags, alarm tags and trend tags) must have the following syntax:

```
[<alpha> | '_' ] * [<alpha> | <digit> | '\\' | '_']
```

That is, the tag name must begin with either an alpha character (A-Z or a-z) or the underscore character (\_). Any following characters must be either alpha characters (A-Z or a-z), digit characters (0 - 9), backslash characters (\), or underscore characters (\_). The use of any other characters will result in a compiler error.

For example, '\_MyTag123' and 'my\New\Tag' are both valid tag names, whereas '\NewTag\' is invalid.

Tag names that begin with a numeric character, such as '12TagName', are only valid if the INI parameter **[General]TagStartDigit** is set to 1 (the default value is zero).

**Note:** The name of an Alarm Tag must follow this syntax but the Alarm Name does not.

## Using structured tag names

The following naming convention is recommended for a CitectSCADA system, to obtain maximum benefit when using features such as Genies and Super Genies. (If you are already using a naming system that differs from the following convention, you can still use Genies and Super Genies supplied with CitectSCADA by modifying the Genies that you want to use.)

Each tag name can contain up to 79 characters. To establish a convention, you must divide the characters in the tag name into sections that describe characteristics of the tag, for example, the area where the tag is located, the type of variable, and any specific attributes. Four basic sections are suggested for a CitectSCADA naming convention:

*Area\_Type\_Occurrence\_Attribute*

### Area

The **Area** section identifies a plant [area](#), number, or name. If you use a prefix that identifies tags within a particular area, you can easily duplicate all CitectSCADA functions within the area. For example, if you have three boilers with the same controls on each boiler, you can configure the tags for boiler number one, and

copy the tags to boilers two and three. You then only need to change the area section in the tag names to the area of the second and third boiler. The remainder of the tags remain unchanged, for example:

Boiler 1	Boiler 2	Boiler 3
B1_TIC_101_PV	B2_TIC_101_PV	B3_TIC_101_PV

If you do not need this facility, you can omit the Area section of the Tag Name to reduce the number of characters in the tag.

#### Type

The **Type** section identifies the Type of parameter, process equipment, or control hardware. The ISA standard naming system is recommended.

Variable Tag	Meaning
B1_TIC_101_PV	Temperature indicating controller
B1_FIC_101_PV	Flow Indicating controller
B1_PUMP_101_PV	Pump
B1_VALVE_101_PV	Valve

#### Occurrence

The **Occurrence** section identifies the loop number.

Variable Tag	Meaning
B1_TIC_101_PV	Temperature Indicating Controller 101
B1_TIC_102_PV	Temperature Indicating Controller 102
B1_PUMP_101_PV	Pump 101
B1_PUMP_102_PV	Pump 102

#### Attribute

The **Attribute** section identifies the attribute or particular parameter that is associated with the loop.

Variable Tag	Meaning
B1_TIC_101_PV	Process Variable
B1_TIC_101_SP	Setpoint
B1_TIC_101_OP	Output
B1_TIC_101_P	Gain or proportional band
B1_TIC_101_I	Integral
B1_PUMP_101_CMD	Command signal to start pump
B1_PUMP_101_M	Auto/Manual mode
B1_TIC_101_V	Value (running/stopped)

### Recommended Attributes

Genies and Super Genies supplied with CitectSCADA use the following attribute convention. If you follow this convention, you can use the Genies without having to modify them.

Mnemonic	Discrete Control / Monitoring	Data Type	Range
_CMD	Command Signal to Start Device	Digital	0 = Off, 1 = On
_M	Control Mode	Digital	0=Man, 1=Auto
_V	Value	Digital	0=Off, 1=On
_FAIL	Device Failure	Digital	1=OK, 0=Failed
FAULT	Device Fault	Digital	1=OK, 0=Fault

Mnemonic	Process Alarms	Data Type	Range
_ALM	General Alarm	Digital	0=Active, 1=InActive
_HHALM	High High Alarm	Digital	0=Active, 1=InActive
_HALM	High Alarm	Digital	0=Active, 1=InActive
_LALM	Low Alarm	Digital	0=Active, 1=InActive
_LLAM	Low Low Alarm	Digital	0=Active, 1=InActive
_DALM	Deviation Alarm	Digital	0=Active, 1=InActive
_DLALM	Deviation Low Alarm	Digital	0=Active, 1=InActive
_DHALM	Deviation High Alarm	Digital	0=Active, 1=InActive
_HHTRIP	High High Alarm Trip Point	Analog	
_HTRIP	High Alarm Trip Point	Analog	
_LTRIP	Low Alarm Trip Point	Analog	
_LLTRIP	Low Alarm Trip Point	Analog	
_DTRIP	Deviation trip Point	Analog	
_LDTRIP	Low Deviation Trip Point	Analog	
_HDTRIP	High Deviation Trip Point	Analog	
_HHhyst	High High Alarm Hysteresis	Analog	
_Hhyst	High Alarm Hysteresis	Analog	
_Lhyst	Low Alarm Hysteresis	Analog	
_LLhyst	Low Low Alarm Hysteresis	Analog	
_LDhyst	Low Deviation Alarm Hysteresis	Analog	
_HDhyst	High Deviation Hysteresis	Analog	

Mnemonic	Analog Control / Monitoring	Data Type	Range
_PV	Process Variable	Analog	
_SP	Setpoint	Analog	
_RSP	Remote Setpoint	Analog	
_OP	Output	Analog	
_OPM	Output Mode	Digital	0=Manual, 1=Auto
_SPM	Setpoint Mode	Digital	0=Local, 1=Remote
_P	Gain (Proportional Band)	Analog	

Mnemonic	Analog Control / Monitoring	Data Type	Range
_I	Integral (Reset)	Analog	
_D	Derivative (Rate/Preact)	Analog	
_KP	Gain Modifier	Analog	
_KI	Integral Modifier	Analog	
_KD	Derivative Modifier	Analog	
_SPTK	Setpoint Track Mode	Digital	0=OFF, 1=Track
_OPTK	Output Track Mode	Digital	0=OFF, 1=Track
_SPB	Setpoint Bias	Analog	
_SPR	Setpoint Ratio	Analog	
_DEV	Deviation		
_TOT	Totalizer Value	Analog	
_COUNT	Counter Value	Analog	
_CRESET	Counter Reset Command	Digital	0=Counting, 1=Reset
_CLIMIT	Counter Preset Limit	Analog	
_TIME	Timer Value	Analog	
_TRESET	Timer Reset Command	Digital	0=Timing, 1=Reset
_EXP	Timer Expired	Digital	
_TLIMIT	Timer Limit	Analog	
_CALC1	Calculation Result 1	Analog	
_LINZ1	Linearized Signal 1	Analog	
_Q	Data Quality Flag	Digital	1=OK, 0=BAD

**Note:** To keep the tag names shorter you can omit the underscore, but you would sacrifice readability; for example: B1TIC101PV instead of B1\_TIC\_101\_PV.

## Configuring Local Variables

Local variables allow you to store data in memory when you start your runtime system. They are created each time the system starts, and therefore do not retain their values when you shut down. Local variables may be of any data type supported by CitectSCADA, including 2-dimensional arrays of all standard CitectSCADA types except for strings.

Local variables are useful when you need each process to have a separate copy of the data. Each process has its own copy of each local variable configured in the project, and the values in a local variable are available only to the process that wrote them.

### To configure a local variable:

- 1 In Project Editor, select **Tags | Local Variables**. The Local Variables dialog displays.

- 2 In the **Name** field, enter a name for the local variable (maximum 79 characters). Variable names cannot include the '-', '/', '%', or <space> characters.
- 3 In the **Data Type** field, select one of the following supported data types:

Data Type	Variable	Size	Allowed Values
BCD	Binary- Coded Decimal	2 bytes	0 to 9,999
BYTE	Byte	1 byte	0 to 255
DIGITAL	Digital	1 bit or 1 byte	0 or 1
INT	Integer	2 bytes	-32,768 to 32,767
UINT	Unsigned Integer	2 bytes	0 to 65,535
LONG	Long Integer	4 bytes	-2,147,483,648 to 2,147,483,647
LONGBCD	Long Binary- Coded Decimal	4 bytes	0 to 99,999,999
REAL	Floating Point	4 bytes	-3.4E38 to 3.4E38
STRING	String	256 bytes (maximum)	ASCII (null terminated)

Numeric and digital variables have a default value of 0 and string variables default to "" (empty string). If you do not specify a data type, the local variable will be treated as 16-bit integer.

- 4 In the **Array Size** field, enter the size of the array (number of elements) used to store the local variable. The array will be of the data type specified in the **Data Type** field. The array can be one or two-dimensional. The maximum number of elements is 32766 per dimension. When specifying a multi-dimensional array, separate the dimensions with a comma, e.g. "20,30".
- 5 In the **Zero Scale** field, enter the value of the local variable that represents the zero point for the data (maximum 10 characters). The zero scale value is used as the lower limit for trend and bar graphs, and values below the zero scale value will cause an "Out of Range" error in the runtime system.
- 6 In the **Full Scale** field, enter the value of the local variable that represents the full scale point of the data (maximum 10 characters). The full scale value is used as the upper limit for trend and bar graphs, and values above the full scale value will cause an "Out of Range" error in the runtime system.
- 7 In the **EngUnit** field, enter the engineering units that the value represents (e.g. %, deg, mm/sec, etc.). Maximum 16 characters. This property is optional. If you do not specify engineering units, no engineering units are used. Do not use this property for digital and string data types.
- 8 In the **Format** field, Enter the display format of the value (of the variable) when it is displayed on a graphics page, written to a file, or passed to a function (that expects a string). This property is optional. If you do not



specify a format, the format defaults to #####. Do not use this property for Digital and String data types. Maximum 11 characters.

- 9 In the **Comment** field, enter any useful comment. This property is optional and is not used at runtime.
- 10 Click **Add**.

See Also [Configuring Variable Tags](#)

## Configuring Variable Tags

To configure a variable tag:

- 1 Start Citect Explorer.
- 2 Click **Variable Tags**, or choose **Tags | Variable Tags**. The Variable Tags form displays.
- 3 Enter the properties of the variable tag.
- 4 Click **Add** to append a new record, or **Replace** to modify a record.

At a minimum you must complete the **Variable Tag Name**, **I/O Device Name**, **Data Type**, and **Address** fields.

You can paste any existing variable tag into forms in your project.

To select an existing variable tag:

- 1 Open the Project Editor.
- 2 Choose **Edit | Paste Tag**.

To configure a digital tag:

- 1 Open the Project Editor.
- 2 Click **Variable Tags**, or choose **Tags | Variable Tags**.
- 3 Complete the properties in the **Variable Tags** dialog box that appears, using **DIGITAL** as the **Data Type**.
- 4 Click **Add** to append a new record, or **Replace** if you have modified a record.

You must at least complete the **Variable Tag Name**, **I/O Device Name**, **Data Type**, and **Address** fields. Leave the following properties blank:

- **Raw Zero Scale, Raw Full Scale**
- **Eng Zero Scale, Eng Full Scale**
- **Eng Units, Format**

To configure an analog tag:

- 1 Start the Project Editor.

- 2 Click **Variable Tags** or choose **Tags | Variable Tags**. The Variable Tags form appears.
- 3 Enter the properties, using **INT** (or Real, BCD, Long, LongBCD) as the **Data Type**.
- 4 Click **Add** to append a new record, or **Replace** to modify a record.

You must at least complete the **Variable Tag Name**, **I/O Device Name**, **Data Type**, and **Address** fields.

See Also [Variable Tag Properties](#)  
[Formatting numeric variables](#)

Variable Tag Properties

You can use this dialog for [Configuring Variable Tags](#). Variable tags have the following properties:

**Variable Tag Name**

You can use any name for a tag (79 characters) provided it follows the [Tag name syntax](#) and isn't the same as the name of a Cicode function within the project or any included projects. If you have many tags, use a naming convention (see [Using structured tag names](#)). This makes it easier to find and debug your tags.

If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g., you cannot have the same variable tag name in more than one cluster).

**Data Type**

The type of I/O device variable (16 characters). I/O devices support several data types that are used to exchange data with CitectSCADA. Because of the lack of an industry standard, most I/O device manufacturers use individual naming conventions for their I/O device variables. However, all variables correspond to one of the following CitectSCADA data types:

Data Type	Variable	Size	Allowed Values
BCD	Binary- Coded Decimal	2 bytes	0 to 9,999
BYTE	Byte	1 byte	0 to 255
DIGITAL	Digital	1 bit or 1 byte	0 or 1
INT	Integer	2 bytes	-32,768 to 32,767
UINT	Unsigned Integer	2 bytes	0 to 65,535
LONG	Long Integer	4 bytes	-2,147,483,648 to 2,147,483,647
ULONG	Unsigned Long Integer	4 bytes	0 to 4,294,967,295
LONGBCD	Long Binary- Coded Decimal	4 bytes	0 to 99,999,999
REAL	Floating Point	4 bytes	-3.4E38 to 3.4E38

---

STRING	String	256 bytes (maximum)	ASCII (null terminated)
--------	--------	------------------------	-------------------------

**Note:** If you do not specify a range for your tag, then an out of range error occurs if you write a value which is outside the range of the type.

You must specify the correct CitectSCADA data type that corresponds to the data type of the I/O device variable you are configuring. Each data type has a unique address format. You must use this format when you are specifying the address of the variable (as the Address property).

Ensure that you only use data types that are valid for your I/O device. If you do not specify a data type, the variable will be treated as 16-bit integer.

CitectSCADA supports concatenation of I/O device registers. For example, you can define a real data type (in CitectSCADA) as two contiguous int data types (in the I/O device). CitectSCADA reads across the boundary of the two ints and returns a real.

If you use concatenation of registers, the address of the variables must be on all odd boundaries or all even boundaries. You cannot mix address boundaries. For example, V1, V3, V5 are valid addresses.

Be careful when using this feature: the I/O device must maintain the integrity of the second register. (If the I/O device writes to the second int, the value of the real could be corrupted.) The structure of some I/O devices might not support this feature.

### String variables

While numeric variables are more common, some I/O devices also support ASCII strings. You can use strings to store text data (for example, from a bar code reader).

All strings must be NULL-terminated in the I/O device. CitectSCADA uses the NULL character to check for the end of a string, and if no NULL character is present, CitectSCADA reads (and displays) any extra characters in memory, after the end of the string.

When you are using a disk I/O device, you can also specify a string data type for storage of recipes, or for operator display information.

Not all I/O devices support strings. However, if your I/O device does support integer data types, CitectSCADA can use these integer registers to store ASCII strings in your I/O device. CitectSCADA strings can only be stored in contiguous blocks (consecutive registers), and are stored as an array.

To display the data types for an I/O device, double-click the I/O devices book from the **Help**, select your I/O device from the list, and then select the **Data Types** topic.

### I/O Device Name

The name of the I/O device where the variable is stored (16 characters). If using I/O device redundancy, you must specify the primary I/O device name here, not the standby.

**Address**

The register address in the I/O device where the variable is stored (64 characters). The format and prefix of an address will depend on the protocol configured for the I/O device, which can be determined by checking the **Protocol** field on the I/O Devices form in Project Editor.

**Raw Zero Scale / Raw Full Scale**

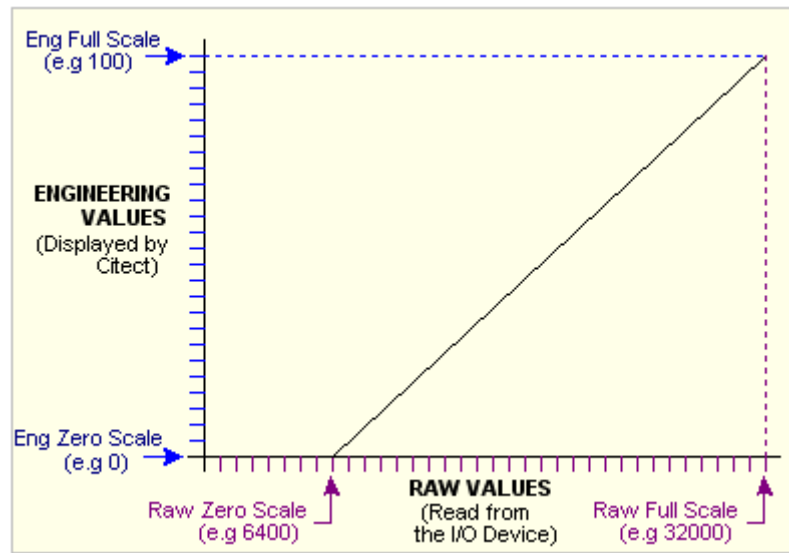
The unscaled (raw) values (of the variable) that represent the zero point and full scale point for the data (10 characters). The raw values are the values that CitectSCADA reads from the I/O device.

**Eng Zero Scale / Eng Full Scale**

The scaled values that CitectSCADA calculates from the raw values (10 characters). The Raw Zero Scale is scaled to the Eng Zero Scale and the Raw Full Scale is scaled to the Eng Full Scale. These properties are represented in engineering units and are used as the upper and lower limits of trends and bar graphs.

Most I/O devices return an integer to indicate the value of an analog input. To return a usable value, the I/O device converts an input signal (usually 4-20mA) to a raw scale variable, usually (but not always) in the range 6400 to 32000.

To present this variable as a meaningful value, you can specify a scaling calculation. CitectSCADA then scales all values accordingly, as in the following diagram.



The scaled value of the variable (engineering value), not its raw value, is used throughout the CitectSCADA system.

The scaling properties are optional. If you do not specify scaling, Eng Zero Scale defaults to Raw Zero Scale, and Eng Full Scale defaults to Raw Full Scale; that is, no scaling occurs.

A value that is below the specified Raw Zero Scale or above the specified Raw Full Scale causes an "Out of Range" error in your runtime system. Do not use a scaling factor for Digital and String data types.

### Eng Units

(16 characters.) The engineering units that the value represents (e.g. %, deg, mm/sec, etc.). This property is optional. If you do not specify engineering units, no engineering units are used. Do not use this property for digital and string data types.

### Format

(11 characters.) The display format of the value (of the variable) when it is displayed on a [graphics page](#), written to a file, or passed to a function (that expects a string). This property is optional. If you do not specify a format, the format defaults to ####.#. Do not use this property for Digital and String data types.

### Deadband

## Formatting numeric variables

The deadband is the percentage of the variable tags engineering range that a tag must be change by in order for an update to be sent through the system. Default value is 0.

For example, if there is a variable tag with an engineering zero of 0 and an engineering full scale of 10000, using a deadband set to 1%. If the current value of the tag is 5600, the value of the tag in the plc must change to a value above 5700 or below 5500 before an update will be sent through the system.

### Comment

Any useful comment (48 characters).

### Linked

The **Linked** field in the status line of the Variable Tags form reads either **Yes** or **No** and indicates whether or not the variable tag is linked to an external data source. When you program an I/O device using software other than CitectSCADA, an external data source is used to store tag data. Linked variable tags are updated whenever external tag data changes, meaning you do not need to enter the information again in CitectSCADA.

The value of a numeric variable (number) can be displayed on a [graphics page](#) or written to a file in many different formats (for example, 24, 0024, 24.000, or 24.0%).

### Format specifiers

Format specifiers are keyboard characters that you use to define the format for the numeric variable. The specifiers that you can use are:

Specifier	Description	Function
#	The hash character	The number of characters to display
0	Zero	Padding
-	Minus	Justification
.	Period	Decimal notation
EU		Engineering units
S		Exponential notation

### Specifying the number of digits

The hash character (#) specifies how many digits CitectSCADA displays. All numeric variables display to the right of an [animation point](#), for example:

Format: #####

In this example, CitectSCADA displays at least four digits (or spaces) to the right of the animation point. If the number contains more than four digits, the first

digit is located at the animation point. The following figure illustrates several numbers in the above format.

```

+   5
+  75
+1275
+5731275
  ↑
  |
  | Animation point (AN)

```

#### Padding with zeros

When a number contains fewer digits than your format specifies (5 or 75 in the above example), CitectSCADA only displays the meaningful digits. You can use the padding character, zero (0), as the second character in the format string, to fill the number with zeros, for example:

Format: #0##

This format string displays:

```

+0005
+0075
+1275
+5731275
  ↑
  |
  | Animation point (AN)

```

#### Changing justification

By default, numeric variables are right-justified (within their field). You can change the default justification by using a minus (-) sign as the second character in the format string, for example:

Format: #-###

This format string displays:

```
+5
+75
+1275
+5731275
  ↑
  Animation point (AN)
```

#### Specifying decimal notation

To specify decimal notation, use a period (.), for example:

Format: ###.##

In this example, CitectSCADA displays three digits before the decimal point and two digits after. If the variable is 12.3, CitectSCADA displays 12.30.

All numbers are automatically rounded, i.e. 12.306 displays as 12.31.

#### Specifying engineering units

You can specify engineering units (such as %, deg, rpm, M, mm/sec, etc.) when you define a variable tag. To include these units in the format of the number, type EU in the appropriate position.

Format: ####.##EU

**Note:** If you do not specify an engineering unit for the variable, only the number is displayed (or logged).

#### Specifying exponential notation

To specify exponential notation, include the exponential character (s), for example:

Format: #s###

In this example, CitectSCADA displays the number in exponential notation format, for example: 1.234e+012.

#### Combining format specifiers

You can combine format specifiers, for example:

Format: #0-###.##EU

This format string displays six digits before the decimal point and two digits after. The number is left justified, padded, and displayed with engineering units (if specified).



### Using shortform notation

As an alternative to the hash (#) notation, you can use shortform notation. With shortform notation, you use a number to specify the format of the numeric variable, for example:

Format: 3.2

This format string displays three digits before the decimal point and two digits after. This format string is equivalent to the `###.##` format specification. You can also include engineering units with shortform notation, for example:

Format: 6.0EU

This format string displays six digits before the decimal point and no digits after. The number is displayed with engineering units (if specified). This format string is equivalent to the `#####EU` format specification.

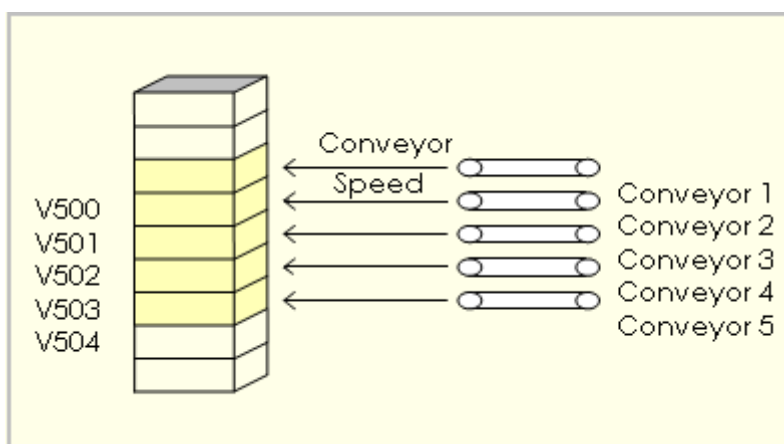
You cannot use padding or left justification with shortform notation.

**Note:** If the value of a numeric variable is extremely large, it is displayed in exponential notation (e.g. 1.2345E012). If no format is specified for a variable, the default `####.#` (or 4.1) is used.

See Also [Using arrays](#)

## Using arrays

An array is a collection of variables (all of the same data type) that are stored in consecutive memory registers in your I/O device. Numeric arrays are useful when you have separate devices (or processes) performing similar functions. You can program the I/O device to store, in consecutive memory registers, variables that relate to each of these processes, for example:



Here, five consecutive variables (V500, V501, V502, V503, and V504) store the conveyor speed of five conveyors (1 to 5). Instead of configuring five separate variable tags (one for each conveyor), you can configure a single tag, as an array.

To specify a single variable tag for an array, define the first address and add the size of the array (the number of consecutive addresses) to the register address, for example:

Variable Tag Name	Conveyor_Speed
Address	V500[5]

Here, five register addresses are referred to by the variable tag Conveyor\_Speed.

#### Referring to array elements

Each element of an array is referred to by an index. You can extract individual variables (from the array) by specifying the tag name and index:

<Tag Name>[ Index ]

For example, to refer to the third variable of the array in the above example (Conveyor 3), use the following syntax:

Variable Tag	Conveyor_Speed[2]
--------------	-------------------

The index of the first element of an array is always 0 (zero). In this example, Conveyor\_Speed[0] is the first element of the array (Conveyor 1), and Conveyor\_Speed[4] is the last element (Conveyor 5).

Note the following when using arrays:

- Do not define large arrays, because each time an array element is requested, CitectSCADA reads the entire array from the I/O device. With large arrays, system performance could be reduced.
- The size of the array must be less than the [maximum request length](#) of the [protocol](#). The I/O device description help topic (for your I/O device) displays the maximum request length of the protocol.
- You should declare all digital arrays on a 16 bit boundary. CitectSCADA rounds each digital array down to the nearest 16 bit boundary. Consequently, all elements in the array are changed. For example, if you declare an array Test to start at bit 35, and CitectSCADA starts the array at bit 32. The index to the array also starts at bit 32; Test[0] refers to bit 32, not bit 35.

#### String arrays

If you are using a CitectSCADA string data type, you must specify an array of integer data types. One int register stores two string characters.

To calculate the size of an array (for string data types), use the following formula:

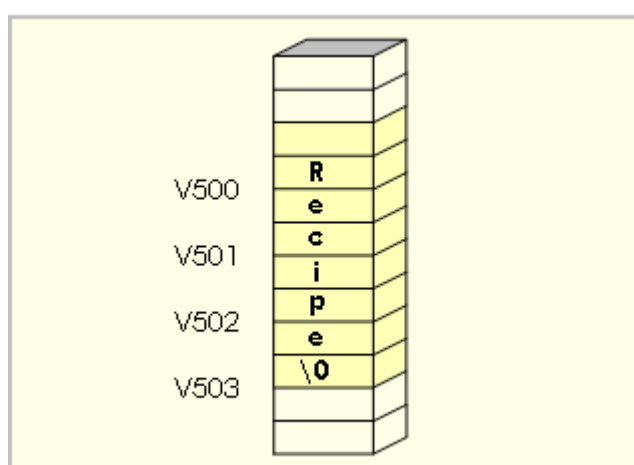
$$\text{Size of Array} = \frac{(\text{number of characters} + 1)}{2}$$

The last element of the array is always used to store the null character '\0'. This character marks the end of the string.

To store the word "Recipe", you must specify an array with 4 elements, for example:

Variable Tag Name	Recipe_Tag
Address	V500[4]

Two characters are stored in each register, i.e:



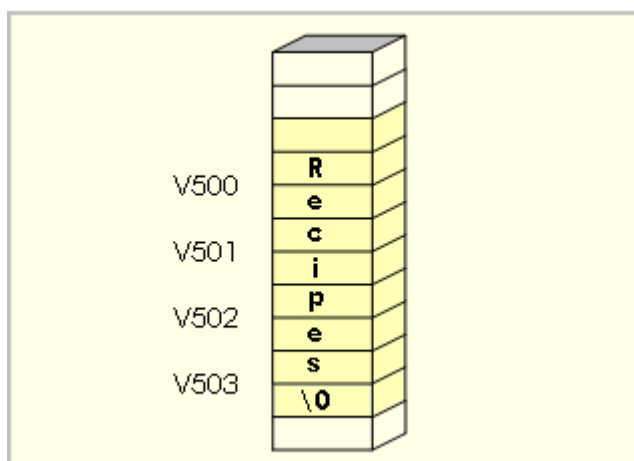
You can then refer to the entire string by specifying the tag:

<Tag Name>

For example:

Variable Tag	Recipe_Tag
--------------	------------

To store the word "Recipes", you would also specify an array with 4 elements. The characters are stored as follows:



**Note:** If your I/O device supports string data types, you must specify the address in the format determined by the I/O device you are using.

See Also [Using structured tag names](#)

# Chapter 15: Linking, Importing, and Exporting Tags

---

Because I/O devices are often programmed independently of CitectSCADA, CitectSCADA allows you to import, or link to, all the tags in an external data source. This means that you only have to define tag information once: when you program your I/O devices. You do not have to re-enter the same information again, in CitectSCADA. Because the necessary information is already saved in an external data source, you can just import it or link to it.

CitectSCADA also lets you export tags to an external file, specifying the destination and format of your choice. You might then import this file into a third party I/O device programming package database, or simply use it as a backup.

See Also

[Linking tags](#)  
[Importing tags](#)  
[Exporting tags](#)  
[Unity Support Matrixes](#)

## Linking tags

Linking is an I/O device specific operation. When you add an I/O device record in CitectSCADA (using the Express Communications Wizard), you can choose to link it to an external data source. The external data source is where all the tag data was saved when the actual I/O device was programmed. If you link to the external data source, CitectSCADA automatically creates variable tag records for every tag in the I/O device.

These variable tag records are dynamically linked to the tags in the external data source. CitectSCADA is updated whenever one of the external tags is changed. For example, you might program your I/O devices, configure your CitectSCADA project, then add some new tags or edit existing tags. In this situation, CitectSCADA is automatically updated with your changes.

This update occurs when you attempt to read the changed tags in CitectSCADA (e.g. you compile your project, display the tag using the Variable Tags dialog box, modify or paste the tag, or perform a manual refresh, etc.). For example, if you change the address of a tag using a third party I/O device programming package, the external data source is changed. Then, when you display the variable tag record or compile your project in CitectSCADA, the change is copied from the external database to CitectSCADA's variable tags database.

You can tell if a tag is linked because the status line at the bottom of the Variable Tags form will read **Linked: Yes**. If it is linked and you change a field, your change will not be overwritten when the link is next refreshed. Generally, however, any field which takes its value from the external data source is disabled.

**Note:** Some properties defined for the external tags will not be relevant to CitectSCADA. Also, some will not be in a format that CitectSCADA can read. Each I/O device has an associated format file in CitectSCADA. It is this file that determines what information is copied to CitectSCADA's variable tags database and how this information is to be converted. In most cases, you will not have to edit this file.

Note the following for Citect tags linked to Unity tags:

- 1 The addition of a new tag on either side, will result in the addition of a new tag on the other side.
- 2 The deletion of an existing tag on either side, will result in the deletion of the corresponding tag on the other side.
- 3 The modification of an existing tag on either side, will result in the modification of the corresponding tag on the other side.
- 4 The delete operation in either Unity or Citect will take precedence over other operations (such as modify).
- 5 If there is some contention between the Unity database and the Citect database for the same tag, that is modifications are done from both databases at the same time, the Unity database item will always take precedence over the Citect database item.
- 6 The linked I/O device live update synchronization will be triggered in the following cases:
  - 1 Unity configuration update.
  - 2 Citect configuration update.
  - 3 Manual refresh.

See [Linked Tags](#) for information about levels of support between Unity and CitectSCADA.

#### Breaking the link to the external data source

You can break the link to the external data source from the I/O devices form or through the Express Communications Wizard. If you break the link, you can choose to make a local copy of all the tags or you can simply delete them altogether.

#### Deleting the I/O device

If you delete an I/O device record which is marked as linked, you can choose to make a local copy of all the linked tags or you can simply delete them.

#### To link to tags in an external data source:

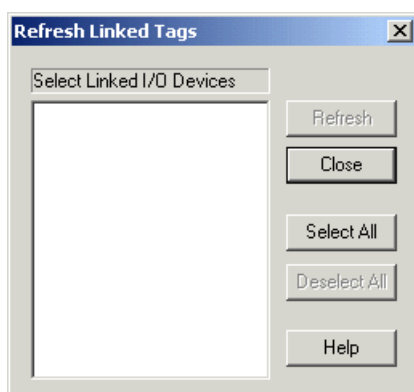
- 1 Start the Project Editor.
- 2 Choose **Communication | I/O Devices**.
- 3 Scroll to the relevant I/O device and choose **Linked | True**.
- 4 Complete the remaining fields as required.

#### To link to tags in an external data source using the Express Communications Wizard:

- 1 Start the Project Editor and choose **Communication | Express Wizard**. (Alternatively, you can open Citect Explorer, and then click **Express I/O Device Setup** in the **Communications** folder of the current project.)
- 2 Complete the wizard screens one by one, selecting the relevant I/O device, and so on. When the **Link to External Database** screen appears, select the **Link I/O Device to an external tag database** check box and complete the remaining details.

#### To manually refresh linked tags:

- 1 Open Citect Explorer.
- 2 Choose **Tools | Refresh Linked Tags**.



See Also [Refresh Linked Tags properties](#)  
[Importing tags](#)

#### Refresh Linked Tags properties

Use this dialog to refresh [linked tags](#). The dialog has the following field:

##### Select Linked I/O Devices

Every linked I/O device in your project (and included projects) are listed here. To refresh the tags for an I/O device, click the I/O device, then click **Refresh**. This updates your project with the latest tag values for the selected I/O device.

If you use CitectSCADA to modify any I/O device tags, your modifications will not be overwritten on refresh.

## Importing tags

Importing tags from an external data source lets you halve your data entry time. Instead of entering all your tag information once when you program your I/O device and once when you configure your project, you can program your I/O device, then import the tags straight into CitectSCADA, where they are treated as regular CitectSCADA tags. (CitectSCADA automatically creates variable tag records for every tag in the I/O device.)

Like linking, the importing of tags is an I/O device specific operation: you import all the tags for a particular I/O device. Unlike linking, however, imported tag records are not linked in any way to the tags in the external data source. Therefore, importing is typically a one-off operation. To update imported tags, you must import them again.

**Note:** Tags will be imported into the project in which the selected I/O device is defined. This could be either the project selected in the Explorer, or in one of its included projects.

For most external data sources, the import process involves two steps. First you export the data from the I/O device to a format that CitectSCADA can read, then you import the database into CitectSCADA. However, tag data saved using Mitsubishi or Unity FastLinX can be read directly by CitectSCADA. This means that no export is required.

When you import tags into CitectSCADA, you have two options for dealing with existing CitectSCADA tags:

- Delete all tags associated with that I/O device prior to the import.
- Update existing tags on import. Tags found in CitectSCADA and in the external data source are updated in CitectSCADA. Tags found in CitectSCADA but not in the external data source will remain untouched. All new tags are appended.

If you import a data source that is already linked, all of the tags in the data source are duplicated. That is, you will have two copies of each tag: one local and one linked.

See [Imported Tags](#) for information about levels of support between Unity and CitectSCADA.

Some properties defined for the external tags will not be relevant to CitectSCADA. Also, some will not be in a format that CitectSCADA can read. To define what information is copied to CitectSCADA's variable tags database and how this information is to be converted, you must edit the I/O device's format file.



### To import variable tags from an external database:

- 1 Open Citect Explorer.
- 2 Choose **Tools | Import Tags**.
- 3 Complete the **Import Variable Tags** dialog box as required.

See Also [Import variable tags properties](#)

## Import variable tags properties

Use this dialog for [Importing tags](#) from an external data source.

**Note:** If an I/O device is linked to an external data source the Database Type, External Database, Connection String, and Tag Prefix fields will be greyed out.

The Import Variable Tags dialog has the following fields:

### I/O Device

The I/O device for which you are importing tags. Use the menu to select an I/O device that has been defined using CitectSCADA.

**Note:** Tags will be imported into the project in which the selected I/O device is defined. This could be either the project selected in the Explorer, or in one of its included projects.

### Database type

The format of the data referenced by the external data source.

### External database (128 Chars.)

References the source for the external database. Click the Browse button to either navigate to the file or to specify configuration options. The external database can be:

- An explicit path and file (e.g., C:\Data\Tags.csv)
- An [IP address](#) and node (e.g., 127.0.0.1\HMI\_Scada)
- A URL (e.g. <http://www.abicom.com.au/main/scada>)
- A computer name (e.g. \\coms\data\scada)

Tag data can be read directly from Unity or Mitsubishi Fastlinx datasources. Click the browse button to specify configuration options. See [FastLinx for Mitsubishi Tag Browser Properties](#), or [Unity Link Configuration Properties](#).

### Connection string (128 Chars.)

Connection details for the data source. This is similar to an ODBC connection string. For example:

```
UserID = XXX; Password = YYY
```

or

ServerNode=111.2.3.44; Branch=XXX

Only certain drivers require a connection string:

- Mitsubishi FastLinX
- Unity Fastlinx (Static/Dynamic)
- OPC

Selecting one of these drivers and then browsing for the external database will automatically populate the Connection String field where required.

#### **Add prefix to imported tags**

Select this box to insert a prefix in front of the names of all imported tags in your *Variable.DBF*.

#### **Tag prefix**

The prefix (8 characters max.) that is inserted in front of the names of imported tags in the CitectSCADA variable tags database.

#### **Delete all I/O Device tags prior to import**

Select this box to delete all the I/O device's tags (from the CitectSCADA variable tags database) before importing.

If you do not select this checkbox, tags found in CitectSCADA and in the external data source are updated in CitectSCADA. Tags found in CitectSCADA but not in the external data source remain untouched. New tags are appended.

#### **Purge deleted tags not found in data source**

Select this box to delete tags which have been removed from the external database. In other words, if a tag is still part of the I/O device in your project, but not in the external database, it is deleted from your project.

### **FastLinX for Mitsubishi Tag Browser Properties**

Use the CitectSCADA FastLinX tag browser to:

- Browse the available Mitsubishi servers on your network.
- Generate the connection strings and server parameters that the Fastlinx database driver requires in CitectSCADA.

The FastLinX tag browser has the following fields:

#### **Database tree display**

Shows a hierarchical display of the MxChange servers (both local and networked), their Fastlinx databases, and associated nodes. To expand or collapse a tree entry, click the add [+] or minus [-] symbol, respectively, to the left of the entry.

When you open the browser, the information displayed depends on whether or not there is an existing FastLinX database with the same name as the current CitectSCADA project:

- If an FastLinX database exists with the same name as the CitectSCADA project and the database contains a GID node with the same name as the CitectSCADA I/O device, the browse dialog highlights the GID node of that name. You can then click **Open** to generate connection strings or server parameters.
- If no FastLinX database exists with the same name as the CitectSCADA project, the **Create a new database** node is highlighted. Click **Open** to generate the connection strings, or click the highlighted node to create a new database.

#### Server

Displays the server name on which the database, if present, exists. If no database is found, this box displays defaults to the name of the local PC. (Read-only)

#### Database

Displays the database name (i.e., name of the current CitectSCADA project). (Read-only)

#### IO Device

Displays the name of the GID node (i.e., CitectSCADA I/O device). (Read-only)

#### PLC Class

Indicates the type of project configured in the MxChange server.

#### PLC Type

Indicates the PLC type corresponding to the selected PLC class.

#### Password

Enter the password for the selected database to allow CitectSCADA to access the database for data retrieval. Note that the password must be valid for read/write access as assigned by your database administrator.

See Also [Defining Variable Tag Names for Mitsubishi FastLinX](#)

### Defining Variable Tag Names for Mitsubishi FastLinX

**Note:** The functionality described in this topic relates only to FastLinX for Mitsubishi. You must have purchased an appropriate CitectSCADA license for this functionality to be supported.

If you are using Mitsubishi FastLinX, you cannot use certain reserved words when defining your tag name variables. You cannot use:

■ Reserved words such as:

&	*	**	-
/	+	<	<=
>=	<>	=	>
ACTION	AND	ANY_NUM	ARRAY
BOOL	CASE	CONFIGURATION	CONSTANT
DATE	DATE	DINT	DO
DUT	DWORD	ELSE	EN
END_CASE	END_CONFIGURATION	END_FOR	END_IF
END_PROGRAM	END_VAR	END_WHILE	ENO
EXIT	FALSE	FOR	FUNCTION
IF	INT	INT	MOD
NOT	OF	OR	PROGRAM
REAL	REPEAT	STRING	STRING
TASK	TIME	TIME	TO
TRUE	TRUE	UNTIL	VAR_EXTERNAL
VAR_GLOBAL	VAR_IN_OUT	VAR_INPUT	VAR_OUTPUT
WHILE	WORD	XOR	

- Cicode function names. See [Functions Reference](#) for a complete list of Cicode functions.

Using reserved words or Cicode function names may cause a syntax error in Mitsubishi FastLink when you export tags to the FastLink tag name database.

See Also [FastLink for Mitsubishi Tag Browser Properties](#)

## Unity Link Configuration Properties

The Unity Link Configuration dialog specifies the parameters used to generate alarm and trend tags. It has the following fields:

### I/O Device & Protocol Driver

The name of the destination I/O device and its corresponding protocol. Unity Fastlink will only import tags to an I/O Device using either the MODBUS, MODNET, MBPLUS, UNITE, or OPC (OPC1 is NOT supported).

### Unity Database Type

The import/export database driver type selected from the Import Variable Tags dialog.

### PLC Family

The family name of the PLC you are importing from. It can be either Premium or Quantum. This option is only valid for the Unity Speedlink Static protocol.

### Unity File

Path of the Unity project file from which you want to import the tags. If the database type is Unity Speedlink Dynamic, this is a \*.stu file. If the database type is Unity Speedlink Static, this is a \*.xsy file.

**Unity Profile Enable**

Enable this if the Unity database is restricted by user profile and a user name and password combination are needed to gain access. See the Access Security Management section of the Unity Pro online help for further information.

**Unity Username and Password**

The username and password, if it is required, for the Unity database from which you are importing.

**DCOM Enable**

Enable this if the database host is a DCOM server.

**DCOM Server Name**

If enabled, the DCOM server that the client uses to hosts the database from which you are extracting tags.

**DCOM Username and Password**

If enabled, the username and password for the DCOM server.

**Log File Path**

The path name for log files generated during the import or export process.

**Logging Level**

The level of detail required in the import or export logs.

**Log Pool Size**

Specifies the maximum number of log files for the given device.

**Validate**

Click the validate button to check the Unity Link settings are correct. This button enables the OK button on the page.

## OPC Data Access Server Tag Browser Properties

The OPC Data Access Server Tag Browser dialog generates the strings that the OPC Data Access Server database driver requires in CitectSCADA.

The OPC Data Access Server Tag Browser dialog has the following fields:

**Machine Name**

The location of the OPC Server which can be an IP address or the network path server. Leave blank to select the local machine.

**Database tree display**

Displays all the Servers on the network that are running the OPC Server application, and the hierarchical database node structure for each.

Select a group of tags to import from an available database. OPC data can be either a tree (hierarchical) or a flat structure. The CitectSCADA OPC driver supports access to both types of storage. If the data is in a tree structure, it can be accessed to the first branch level down from the root node. Deeper levels of branching may be supported in the future.

**Note:** The authentication values must be valid for read/write access, as assigned by the appropriate database administrator.

## Exporting tags

The Export feature allows you to export I/O device data to an external data source, specifying the destination and format of your choice (e.g. RSLOGIX driver). This file might then be imported into a third party I/O device programming package database. Alternatively, you might use it as a backup.

**Note:** Exporting using the OPC driver is not supported.

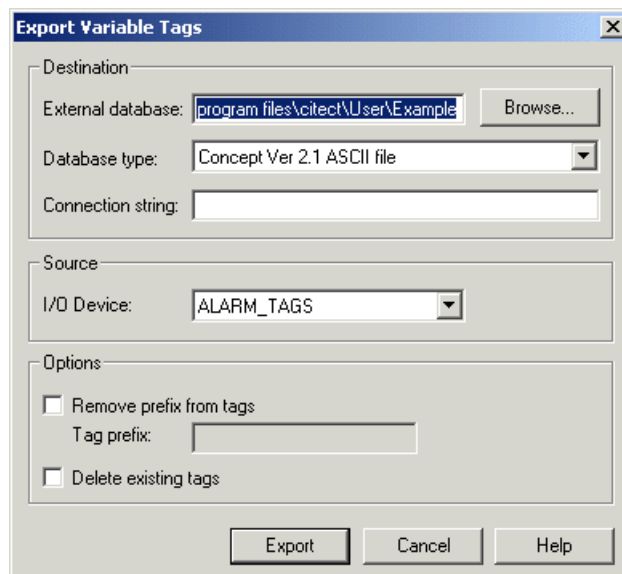
The file to which you are exporting must already exist; otherwise the export will not work. You can choose to delete its contents before exporting, or you can leave it and create duplicate tags.

**Note:** The export tags feature is not yet supported by all database types. If you have existing links to any external data source, the linked tags will also be exported. As the structure of each type of external data source differs, some tag data might not be exported. This is determined by the format file for the I/O device.

See [Exported Tags](#) for information about levels of support between Unity and CitectSCADA.

### To export variable tags to an external database:

- 1 Open Citect Explorer.
- 2 Choose **Tools** | **Export Tags**.
- 3 Complete the **Export Variable Tags** dialog box as required.



See Also [Export Variable Tags properties](#)  
[External data source](#)  
[Format file](#)

## Export Variable Tags properties

Use this dialog for [Exporting tags](#) to an external database. The Export Variable Tags dialog has the following fields:

### External database

A reference (128 characters max.) to the external data source to which your variable tags are exported. This can be:

- An explicit path and file (e.g., C:\Data\Tags.csv)
- An IP address and node (e.g., 127.0.0.1\HMI\_Scada)
- A URL (e.g., http://www.abicom.com.au/main/scada)
- A computer name (e.g., \\coms\data\scada)

**Note:** This data source must exist before the export can be performed.

### Database type

The format of the data referenced by the external data source.

### Connection string

Enter a connection string to provide connection details for the data source. This is similar to an ODBC connection string. For example:

```
UserID = XXX; Password = YYY
```

Not all data sources require a connection string.

### **I/O Device**

The I/O device for which you are exporting tags. Use the menu to select an I/O device that has been defined using CitectSCADA.

### **Remove prefix from tags**

Select this box to remove a known prefix from the front of the exported tag names.

### **Tag prefix**

The prefix (8 characters max.) to be removed from exported tag names.

### **Delete existing tags**

Select this box to delete any tags in the external database before exporting.

**Note:** For the Unity driver existing tags are not deleted, even if you select this check box.

## **External data source**

When setting up an import, export, or link, you need to provide a data source and the format of the data.

Your data source can be entered as:

- An explicit path and file (e.g., C:\Data\Tags.csv)
- An IP address and node (e.g., 127.0.0.1\HMI\_Scada)
- A URL (e.g., http://www.abicom.com.au/main/scada)
- A computer name (e.g., \\coms\data\scada)

The database type field specifies the format of the external data source. When CitectSCADA attempts to read from this data source, it will use the mechanism specified by the database type. The supported database types are:

- OPC (OPC1 is not supported)
- CSV (comma-separated values)
- Concept Ver 2.1 ASCII file
- Mitsubishi FastLinx (a specialized driver which allows you to connect directly to the PLC programming software)

### **To configure the external data source as a file**

The example uses a CSV file, in this case an RSLOGIX database driver

In the Import/Export or Links dialog box, enter details as follows:

- **External database** C:\Data\Tags.csv



- **Database type** RSLOGIX Driver
- **Connection string** (leave blank)

To configure the external data source using a specialized driver

**Note:** This example uses the supplied OPC driver.

In the Import/Export or Links dialog, enter details as follows:

- **External database:** The name of the OPC server process, e.g., FactorySoft.InProc
- **Database type:** OPC
- **Connection string:** The parameters are **ServerNode** or **Branch**, though both are optional. Their use depends on the location of the OPC server and the scope of the required data. **ServerNode** can be an IP address or the network path to the server. For example:
  - ServerNode=127.0.0.1
  - ServerNode=\\Server
  - ServerNode=www.server.com

For **Branch**, OPC data can be either a tree (hierarchical) or a flat structure. The CitectSCADA OPC driver supports access to both types of storage. If the data is in a tree structure, it can be accessed to the first branch level down from the root node, by entering the name of the branch. For example:

- Branch=device1

Deeper levels of branching might be supported in the future.

**Note:** This example uses the supplied Mitsubishi Driver

In the Import/Export or Links dialog box, enter details as follows:

- **External database:** 127.0.0.1\HMI\_Scada (you can also use the computer name instead of the IP address)
- **Database type:** Mitsubishi FastLinx
- **Connection string:** UserID=Citect;Password=Citect

The Connection String must be in the format which the specialized driver (in this case Mitsubishi FastLinx) expects; otherwise it might be ignored.

**Note:** Tags in your external data source must conform to the CitectSCADA standard. Tag names that are longer than 79 characters are truncated. If this truncation results in duplicate tags, you are informed when you compile your project. Characters other than (a to z, A to Z, and 0 to 9) and the underscore character "\_", are removed on import/link (and before any truncation).

See Also [Format file](#)

## Format file

The format file defines the import/export/linking rules. The file maps columns from the external data source format to the internal CitectSCADA database format. In other words, it determines what information is imported/exported/linked and how this information is modified during the operation.

The format file also provides the information to allow CitectSCADA to use the correct driver for accessing the external data source.

Some format files are provided with CitectSCADA; however, sometimes you might need to write or modify a format file using an editor such as Microsoft Notepad.

Following is an example of how format file rules work. (For more information, see [Format file layout](#).)

- 1 Column 3 in the external data source needs to be copied into the "Name" column in the CitectSCADA's tag database. (CitectSCADA names are restricted to alphanumeric characters (a to z, A to Z, and 0 to 9) and the underscore character "\_", but you do not need to worry about this in the format file; CitectSCADA does the conversion automatically). However, if Column 3 in the external data source happens to be blank, there is no need to copy this record across at all (it must be rejected).
- 2 Column 1 in the external data source needs to be copied straight into the "Addr" column in CitectSCADA's tag database, because they both mean exactly the same thing.
- 3 Columns 4, 5, 6, and 7 in the external data source all need to be copied into the "Comment" column in Variable.DBF. (It is not uncommon for external data sources to split the comments across several fields). The fields need to be copied in that order, so that if the data in Column 4 in the external data source is "**Loop**", Column 5 is "**1**", Column 6 is "**Process**", and Column 7 is "**variable**", these fields are copied across in order, so that the "Comment" column in Variable.DBF reads "**Loop 1 Process variable**". This process is called 'concatenation'. (For the "Comment" column, CitectSCADA automatically adds a space between each field from the external data source.)
- 4 The data in Column 1 in the external data source determines what CitectSCADA needs to write in the "Type" column. However the data cannot be copied across directly, because it would not make sense to CitectSCADA. Instead, it needs to go through a conversion (or filtering) process. This conversion needs its own set of rules, such as:
- 5 If Column 1 in the external data source is "BT%d:%d.%d" (where %d means "any decimal number"), CitectSCADA needs to write the string "DIGITAL" in the "Type" column.

- 6 If Column 1 in the external data source is "F%d:%d/%d" (where %d means "any decimal number"), CitectSCADA needs to write the string "DIGITAL" in the "Type" column.
- 7 If Column 1 in the external data source is "O:%e" (where %e means "any octal number"; that is, all digits from 0 to 7), CitectSCADA must leave the "Type" column blank. It still accepts the record (provided all other columns pass any filtering tests) but it does not write anything in the "Type" column. The assumption is that CitectSCADA currently does not have (or does not need) a suitable corresponding type.
- 8 If Column 1 in the external data source is "PD%d:%d.%d", CitectSCADA needs to write the string "REAL" in the "Type" column.
- 9 If Column 1 in the external data source is "ST%d:%d", CitectSCADA needs to write the string "STRING" in the "Type" column.
- 10 If there are no rules covering the contents of Column 1 in the external data source, CitectSCADA must reject the whole record and not copy it into the CitectSCADA database.

See Also [Format file layout](#)

## Format file layout

The format file is divided into sections:

- [\[General\] section](#)
- [\[Columns\] section](#)
- [\[ImportFilterMap\] and \[ExportFilterMap\] sections](#)

Each section consists of a section header (the section name enclosed in square brackets (e.g. "[my\_section]")) on a line by itself. This is followed by the body of the section, typically single line statements of the form:

```
"something = something_else -> something_else_again"
```

Any white space (or none at all) is acceptable around the "=" and the ">", but the whole statement must be on one line. Most statements within a format file follow this pattern, but in many cases there might be no ">" (the converter), or there might just be a converter without anything following it.

The following sections are required in all format files:

```
[General]
[Columns]
[ImportFilterMap]
[ExportFilterMap]
```

Other sections might be required depending on the complexity of the conversion between CitectSCADA and the external data source. This is determined by the contents of [ImportFilterMap] and [ExportFilterMap].

Comments can also be added within or between sections. To do this, place a semicolon ";" as the first character on the line. The rest of the line is then considered a comment, and is ignored by CitectSCADA. For example:

```
; I am putting the [General] section here
[General]
```

#### [General] section

The General section consists of 4 lines:

```
[General]
Name=name
Description= description
DriverName= driver name
DriverInst="a special string"
```

The *name* and the *description* are not currently used by CitectSCADA. CitectSCADA uses the *driver name* to load the correct driver for accessing the external data source. This driver might be one that is part of the CitectSCADA installation, or it might be a customized driver (including a driver that you have written yourself), for accessing a particular data source (which could be a protocol, type of hardware, server or file type). The driver must be an OLE DB-compliant driver.

The *special string* allows extra information to be passed to the driver. It is added to the connection string (in Citect Explorer and the Project Editor). So the connection string can be used for information that is likely to change often, and this *special string* can be used for more permanent information (such as the comma "," delimiter for a .CSV file). The main use of this string is as a delimiter for an input file. To specify that a comma "," is used by an input file as the delimiter, the following syntax would be used:

```
DriverInst="delimiter=,"
```

#### [Columns] section

The Columns section defines the format of the columns in the external data source. It is structured as follows:

```
[Columns]
External column name 1 = column width -> data type
External column name 2 = column width -> data type
..
External column name n = column width -> data type
```

The only restriction that CitectSCADA places on the data for *External Column name n* is that it must be unique within the section. For convenience, you can use the names that the external data source uses (such as "Description", "PLC\_id", "Iotype") or you can just make up names like "Column1", "Column2", etc. The order in which these entries appear is important: it must be the same as the order of the fields in the external data source. The names used for the External

Data Source columns in the [ImportFilterMap] and [ExportFilterMap] sections must come from this list.

*Column width* is the number of characters in the field, and *data type* is the type of data for that column. Currently the only acceptable data type is "STRING".

#### [ImportFilterMap] and [ExportFilterMap] sections

The [ImportFilterMap] and [ExportFilterMap] sections have identical syntax and functionality, except that the [ImportFilterMap] describes how to convert data from the External data source on import, while the [ExportFilterMap] describes the opposite conversion.

These are the most complex sections in the format file. (The rest of the text will just focus on the [ImportFilterMap], as the [ExportFilterMap] follows basically the same logic.)

The [ImportFilterMap] is structured as follows:

```
[ImportFilterMap]
Import Rule 1 = External column name 1 -> Citect Column i
Import Rule 2 = External column name m -> Citect Column j
..
Import Rule nn = External column name n -> Citect Column k
```

The values in *Import Rule nn* can be any name strings, but they must be unique within the section. Therefore, for convenience, you might want to use names like "ImportRule1", "ImportRule2", "Mapping1", "Filter1" etc., or you might want something that is descriptive of the conversion involved, such as "Description\_to\_comment".

The name used for *External column name n* must be identical to a name that appears in *External column name n* in the [Columns] section above.

The name used for *Citect Column k* must be the same as one of the columns in the CitectSCADA internal tags database, such as "Name", "Type", "Addr", "Comment" etc.

Thus the values for the external column and the CitectSCADA column provide information on how to transfer data from the external column to the Citect column during import.

For example:

```
ImportRule1 = Description -> Comment
```

This indicates that there is a relationship between the data in the "Description" field in the external data source and the data that needs to go into the "Comment" field in the CitectSCADA database.

The name that you use for *Import Rule nn* might be the same as the name of another optional section in the format file: here, the extra section provides CitectSCADA with more information. In the simplest case, if there is no section

with that name in the format file, the rule simply states that the data in *External column name n* is to be copied directly into *Citect Column k* without modification or filtering.

So if the "Description" column in the external database contains "Truck Position 1" and the above entry appears in the [ImportFilterMap] section, and there is no section called [ImportRule1], then after the import, the "Comment" column in the CitectSCADA database will contain the string "Truck Position 1".

#### Concatenation

To concatenate fields from the external database into one field in the CitectSCADA database, add separate entries to the [ImportFilterMap] section. Each section must contain the name of a relevant external column and the name of the destination column in CitectSCADA. *The entries must appear in the order in which the fields are to be concatenated.*

So, if the external data source has a field called "IOdev" containing the value "M", and another field called "IOaddr" containing the value "61", and you want to join them together so that the value "M61" is imported into the CitectSCADA "Addr" field, this is how it would be done:

```
[ImportFilterMap]
Addr1= IOdev -> Addr
Addr2= IOaddr -> Addr
```

Here, you must ensure that there are no sections in the format file called [Addr1] or [Addr2], unless you need some filtering or conversion.

See Also [Field conversion](#)

## Field conversion

To modify mapped data or to apply filtering (to reject certain records), create a new section using the name of the relevant line from the mapping section. For example, if you have the following mapping section:

```
[ImportFilterMap]
Test1_to_type = Test1 -> Type
```

and you want to convert the data from the Test1 column before importing it, somewhere in the file you must have a section called [Test1\_to\_type]:

```
Test1_to_type]
```

followed by the necessary conversion rule.

**Note:** The name must be from the import mapping section, not from the export mapping section. If you use a name from the export mapping section, the conversion applies to the export, not the import.

The basic format of this conversion/filtering section is as follows:

```
[Relevant mapping name]
Filtering Rule 1 = External Pattern 1 -> Citect String 1
..
Filtering Rule m = External Pattern m
..
Filtering Rule n = External Pattern n -> Citect String n
```

The name used for *Filtering Rule n* has no intrinsic significance to CitectSCADA (except that it uses it as a key to locate the entry). The only restriction is that it must be unique within the section, so you can use whatever is convenient.

The value in *External Pattern n* is a combination of characters which CitectSCADA will look for in the external data source column. This pattern can be any combination of the following:

Character in format file	Matches what string in external data source
<specific text>	<specific text>
*	Any string.
?	Any single character.
%d	Any decimal integer (nnn. . . where n is 0-9).
%e	Any octal number (0nnn. . . where n is 0-7).
%h	Any hexadecimal number (0xnnn. . . where n is 0-9, A-F or a-f).
%s	Any string.
{	Begin a "token string". Any characters enclosed by { } in the Input Pattern (including regular and special characters) represent a token string. The characters in the data stream that match a token string are referenced by the Output Data String and written directly to the output database as a group.
}	End of a token string.
\	Treat the following character as a literal. For example, if a literal * character was expected in the input data stream, you would use \* to denote this. If a literal backslash \ is expected, use \\.

Any other characters must literally match the same character.

If `External Pattern n` is found in the external column, `Citect String n` is written to the relevant column in CitectSCADA (as per the mapping).

In addition, two special characters can appear in the output data string:

Character in output string	Meaning
\$	The pattern <code>\$n</code> (where <i>n</i> is any integer) is replaced in the output data stream by the <i>n</i> th "token"; a token is a matching sequence of characters enclosed by { } in the input pattern. (An error will result if \$ not followed by a token number.)
!REJECT!	This sequence must appear by itself in the output data pattern. The whole record is rejected. As the record had already been matched to the input pattern, no further rules are checked.

Character in output string	Meaning
\	Treat the following character as a literal. This would be used if a literal \$ sign was required (use \\$) or if another digit immediately follows. For example, if the string "3August2001" must immediately follow the token, use "\$1\3August2001".
\ (at end of line)	Insert a literal space ' ' character at the end of the output line. Without this provision, the system could not distinguish between the end of the input line (which is likely to be followed by characters, such as spaces, that Windows will ignore) and a space being required at the end of the output line.

Other characters are written literally to the output database.

CitectSCADA works through each filtering rule in the section, looking for a match. If a rule does not match, the next one is tried, then the next, and so on, until a match is found. If no match is found, the whole record is rejected; none of the data from any field is copied to CitectSCADA.

For example, to convert the string "FLOAT" in the external data source to "DIGITAL" in CitectSCADA, you could use the following entry:

```
[ImportFilterMap]
Test1_to_type = Test1 -> Type
..
[Test1_to_type]
Rule1 = FLOAT -> DIGITAL
..
```

For a more complex example, let us assume that the external data source has a column called "Tag" which is equivalent to the "Name" field in CitectSCADA. Let us also assume that the external database has no direct equivalent of CitectSCADA's "Type" field, yet CitectSCADA needs this field to be filled in. We need to use the "Tag" field to decide what goes into the "Type" field of the CitectSCADA database.

If the "Tag" column in the external data source has the value "I:060/07", we have determined that we should write the string "DIGITAL" into CitectSCADA's "Type" field. In fact, if that field has "I:" followed by any octal value, followed by a slash "/", followed by any octal value, we want the string "DIGITAL" to appear in our "Type" field. How do we express all this in the format file?

Firstly, there are two sets of relationships to consider, one connecting the "Tag" field in the external data source to the "Name" field in CitectSCADA, and the other connecting it to the "Type" field in CitectSCADA. So we need two "mappings" (entries) in the [ImportFilterMap] section:

```
[ImportFilterMap]
Tag_to_Name = Tag -> Name
Tag_to_Type = Tag -> Type
..
```



As we want the data in the "Tag" field to be copied directly into the "Name" field, we do this by not having a [Tag\_to\_Name] section anywhere in the format file.

But because we are not copying directly from the "Tag" field to the "Type" field, but are just using the data to decide what goes into the "Type" field, we need a [Tag\_to\_Type] section.

Recall the desired outcome: If the "tag" field has "I:" followed by any octal value, followed by a slash "/", followed by any octal value, we want the string "DIGITAL" to appear in our "Type" field.

We express this in the format file as follows:

```
[Tag_to_Type]
Rule1 = I:%e/%e -> DIGITAL
..
```

This will match "I:060/07" or "I:0453/02343445602" (and cause the string "DIGITAL" to be written to CitectSCADA's Type field), but will not match "I:060/98" or "I:054".

To give a few examples of how the wild-card characters (%s, \* and ?) might be used, the pattern "HE%sLD" or "HE\*LD" in the format file would match "HELLO WORLD" or "HE IS VERY BOLD" in the external data source. The pattern "HE??????LD" would match "HELLO WORLD" but not "HE IS BALD", as each question mark "?" must match exactly one character.

CitectSCADA will also handle multiple wildcard patterns, such as "%s/%s:%s".

For an example more useful than "Hello World", imagine that we need to copy the data straight across without modification, but we want to ensure that no blank fields are copied across. The pattern "?%s" or "?\*" will match any string that has at least one character, but will not match a blank.

Sometimes only part of the input stream is required in the output, or the input might need to be split up into different output columns. In these situations "tokens" are useful.

In this example of an export problem, the "Addr" field in the CitectSCADA database needs to be split among two fields in the external database: the "IOdev" (whose value is always "D" or "M"); and "IOaddr" (whose value is a decimal integer of no more than 3 digits). Values in the "Addr" field of the CitectSCADA database are strings such as "D62", "M546", etc.

This problem could be solved by concatenation, i.e. using one mapping to write to the IOdev field, and three other separate mappings to copy each digit separately into the IOaddr field of the external database. But this would be complex and in some situations would not work.

It is better to use a token to solve the problem:

```
[ExportFilterMap]
..
Addr2IOdev = Addr -> IOdev
Addr2IOaddr = Addr -> IOaddr
..
[Addr2IOdev]
D = D* -> D
M = M* -> M
AnythingElse = * ->
..
[Addr2IOaddr]
Rule = ?{%d} -> $1
```

In the [Addr2IOaddr] section, the {%d} is the token string, and as it is the first (and only) token appearing in the rule, \$1 is used to reference it on the output stream side. So if the "Addr" field of the CitectSCADA database contains "D483", "D" is written to the "IOdev" field of the external data sink, and "483" (the token) to the "IOaddr" field.

Here is another example illustrating the use of multiple tokens. Suppose we need to: convert all period characters (.) to colons (:); remove the first two characters (which are blank); and remove any unrequired characters from the data we are expecting; that is, convert "..BJ6452.78....." to "BJ6452:78". This can be achieved by using the following rule:

```
Rule = ??{*%d}.{%d}* -> $1:$2
```

At this point, we introduce another feature of the format file. If you use the following rule:

```
[Relevant mapping name]
Filtering Rule = External pattern
```

i.e., without "-> Citect String" included, CitectSCADA interprets this as "check that the string matches the External Pattern, if it does, copy it across unchanged".

If this rule is used:

```
Filtering Rule = External pattern ->
```

i.e., without "Citect String", it would mean: "If the string pattern matches then accept the record but copy a NULL string to the CitectSCADA database."

Using the above example again, we can add the restriction that any records with no data (i.e. a blank or NULL string) in the Tag field of the external data source should not be imported into CitectSCADA. We would add a [Tag\_to\_Name] section, and would have just one rule: that we accept everything except for a blank.

```
[Tag_to_Name]
RejectBlanks = ?*
..
```

Recall that CitectSCADA checks the pattern in each filter rule sequentially until a pattern that matches the string is found in the external data source. With this in mind, a huge range of conversions and filterings are possible by ordering the rules correctly and, in some cases, by making use of concatenation.

For instance, if certain string types need to be converted but all others need to be copied unmodified to CitectSCADA, you could have a section with a set of rules at the top, followed by a final rule to let everything else through unmodified.

```
[Tag_to_Name]
Rule n = ..... -> .....
..
LetEverythingElseThru = %s
```

A single %s or \*, without anything else, matches anything and everything, including blanks.

For an example of how to reject a particular string or pattern, let's suppose we want to reject any tags starting with "DFILE"(another real-life example). We would simply use the following:

```
[Tag_to_Name]
Rule1 = DFILE* -> !REJECT!
..
LetEverythingElseThru = %s
```

Clearly, it is pointless having the !REJECT! rule not followed by other rules concerning patterns that you do want to accept, as anything that does not match an input pattern is rejected. The logic behind the order that the rules appear can become particularly important when using a !REJECT! rule. You would typically have any reject rule(s) as the first rule(s) in the mapping. There would never be any point in putting a !REJECT! rule as the last rule in the mapping.

!REJECT! rules can also be useful where some text file generated by another system contains some sort of header lines that are not wanted, but the rest of the data is required.

See Also [Having CitectSCADA recognize format files](#)

## Having CitectSCADA recognize format files

Your format file needs to be saved to the directory that the rest of CitectSCADA is running from, normally your \Bin directory (typically C:\Program Files\Citect\CitectSCADA 7\Bin).

For CitectSCADA to import, export, or link, it needs to know the name of the format file and the name of the driver that will access the external data source. It also needs the text of the string that will appear in the **Database type** field of the Import or Export dialog box. All this information is stored in another file, called tagdriv.ini, in the same directory.

The format of tagdriv.ini is simple and based on the odbc.ini format. When it is installed it already has the required information for the format files and

drivers that come shipped with CitectSCADA. You just need to copy the same layout for your new format file, and the driver that you are using.

The `tagdriv.ini` file has several sections. The first section is the `[External data sources]` section, with the following general layout:

```
[External data sources]
Section name 1 = the name you want to appear in the import/export/
link menu for entry 1
Section name 2 = the name you want to appear in the import/export/
link menu for entry 2
..
Section name nn = the name you want to appear in the import/export
menu for entry nn
```

Each entry in this section refers to a combination of format file and driver required for a particular import, export, or link operation.

The text on the left of the "=" sign must refer to the name of another section which must appear in `tagdriv.ini`.

The text on the right of the "=" sign is exactly what will appear in the menu under "**Database type**" for importing, exporting or refreshing variable tags.

The other sections each refer to a type of import or export described in the `[External data sources]` section, and give details about the format file and driver pair. The general layout of these sections is as follows:

```
[Section name matching an entry in [External data sources] ]
driverid = Driver ID
datastring = The name of the format file
Currently the Driver ID is always CiTrans.
```

So if we assume that the version of CitectSCADA you are installing contains 4 format files, there would be 4 sections in `tagdriv.ini`, as shown in the following example:

```
; This file contains the driver name, driver prog id, and format
file mappings
; The format file must reside in the \citect\bin directory
```

```
[External data sources]
CSV = CSV Driver
RSLOGIX = RSLOGIX Driver
Concept ver 2.1 Ascii = Concept Ver 2.1 ASCII file
MxChange = Mitsubishi FastLink
```

```
[CSV]
driverid = CiTrans
datastring = csv.fmt
```

```
[RSLOGIX]
```

```
driverid = CiTrans
datastring = rslogic.fmt
```

```
[Concept ver 2.1 Ascii]
driverid = CiTrans
datastring = concept ver 2_1.fmt
```

```
[MxChange]
driverid = CiTrans
datastring = MxChange.fmt
```

This adds four entries to the database type drop down menu: CSV, RSLOGIX, ASCII and Mitsubishi FastLinx.

The 4 entries in the menu match the strings on the right of the "=" sign in the [External data sources] section.

If you add another format file, you will also need to add a matching entry to tagdriv.ini. For example, if you add a new format file for a Simatic data source, you must add a line similar to this to the [External data sources] section:

```
SIMATIC = Siemens SIMATIC Driver
```

You must also add the following section to the bottom of the file:

```
[SIMATIC]
driverid = CiTrans
datastring = SIMATIC.fmt
```

Save the file, restart Citect Explorer, and "Siemens SIMATIC Driver " appears in the menu.

Selecting this entry causes the format file in the datastring entry under the [SIMATIC] section to be used for the import, export, or link; that is, simatic.fmt.

## Unity Support Matrixes

The following sections show the levels of support between Unity and CitectSCADA:

- [Imported Tags](#)
- [Exported Tags](#)
- [Linked Tags](#)

## Imported Tags

For Unity Fastlinx Dynamic and Static:

Unity PLCs	Unity Data Types	Citect Protocols	Support Status
QUANTUM	BOOL	Unite	Not Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Supported (except DT type)
	INT		
	DINT		
	UINT		
	UDINT		
	REAL		
	TIME		
	DATE		
	TOD		
	DT		
	STRING		
PREMIUM	BOOL	Unite	Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Supported (except DT type)
	INT		
	DINT		
	UINT		
	UDINT		
	REAL		
	TIME		
	DATE		
	TOD		
	DT		
	STRING		

**Note:** Citect recommends that you use the Unity Fastlinx Dynamic driver when importing tags from Unity Pro into a Citect project.

**Note:** Arrays are supported for all data types except DATE, TOD, DT and STRING data types.

In addition, the following Unity data types are not supported:

- Structured data types
- Arrays which are not zero-based
- Multi-dimensional arrays.

## Exported Tags

For Fastlinx Static:

Unity PLCs	Unity Data Types	Citect Protocols	Support Status
QUANTUM	BOOL	Unite	Not Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Not Supported
	INT		
	DINT		
	UINT		
	UDINT		
	REAL		
	TIME *		
	DATE *		
	TOD *		
	DT *		
	STRING		
PREMIUM	BOOL	Unite	Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Not Supported
	INT		
	DINT		
	UINT		
	UDINT		
	REAL		
	TIME *		
	DATE *		
	TOD *		
	DT *		
	STRING		

**Note:** Arrays are supported for all data types except DATE, TOD, DT and STRING data types.

In addition, the following Unity data types are not supported:

- Structured data types
- Arrays which are not zero-based
- Multi-dimensional arrays.

**Note:** Unity data types marked with \*, Citect does not have direct data types that matches. Instead you should use Citect supported data types and CiCode to convert and simulate those data types.

## Linked Tags

Linked I/O Device Live Update

For Unity Fastlinx Dynamic:

Unity PLCs	Unity Data Types	Citect Protocols	Support Status
QUANTUM	BOOL	Unite	Not Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Supported (except DT type)
	INT		
	DINT		
	UINT		
	UDINT		
	REAL		
	TIME		
	DATE		
	TOD		
	DT		
	STRING		
PREMIUM	BOOL	Unite	Supported
	EBOOL	Modnet	Supported (Except BYTE type)
	BYTE	Modbus	Supported (Except BYTE type)
	WORD	MBPlus	Supported (Except BYTE type)
	DWORD	OPC	Supported (except DT type)
	INT		
	DINT		
	UINT		
	UDINT		
	REAL		
	TIME		
	DATE		
	TOD		
	DT		
	STRING		

**Note:** Arrays are supported for all data types except DATE, TOD, DT and STRING data types.

In addition, the following Unity data types are not supported:

- Structured data types
- Arrays which are not zero-based
- Multi-dimensional arrays.



# Chapter 16: Securing CitectSCADA Projects

CitectSCADA projects represent a considerable investment. Once a commissioned CitectSCADA project has been delivered, it usually must remain in the delivered state until modifications are performed by an authorized person. In order to protect projects from unauthorized modification, CitectSCADA allows projects to be secured by an administrator as “read-only.”

This section describes the following:

- Characteristics of read-write and read-only projects (see [Overview](#)).
- Scenarios that describe [Securing a Top-level Project](#) and [Securing an Include Project](#).
- How to secure projects as read-only (see [Making a Project Read-Only](#)).
- Ensuring the CitectSCADA configuration applications and runtime environment can operate successfully (see [Using CitectSCADA with Windows Security](#)).
- The consequences of securing projects (see [Read-Only Privileges on Projects](#)).

## Overview

CitectSCADA has two types of project:

- **Read-write:** allows write and delete privileges to the project folder (or for any project file) for the current user.
- **Read-only:** projects that deny write and delete privileges to the project folder for the current user.

The table below shows the different characteristics of read-only and read-write projects:

Task	Read-only	Read-write
Create new files in project		■
Delete existing files in project		■
Modify existing files in project		■
Delete project		■
Rename project directly		■

Read-only projects cannot be compiled as top-level projects (i.e., projects that are the main (root) project as opposed to an included project) and online changes are not supported.

**Note:** If the project folder is read-only for the current user, but one or more files in the project have read-write access for the current user, the project is considered to be a hybrid read-only/read-write project. CitectSCADA does not support this type of project. Running a hybrid project may result in your system becoming unresponsive. (This note does not include those folders or files that require read-write access in order to operate at runtime; see [Using CitectSCADA with Windows Security](#) for details.)

The security model used in enabling read-only projects does not replace the existing CitectSCADA user accounts; instead, it works in conjunction with user accounts like this:

- CitectSCADA user accounts govern runtime security to project elements.
- Windows user accounts govern the security of configuration project elements.

## Securing a Top-level Project

This section describes a “real-world” situation that might require read-only privileges to be applied to a top-level CitectSCADA project.

**Note:** Before securing a top-level project, read the section [Read-Only Privileges on Projects](#) for details on operational constraints. Pay particular attention to the section [Read-only on top-level projects](#).

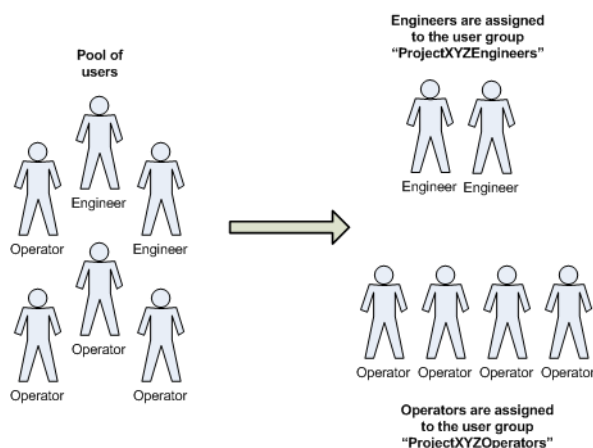
In this scenario, several onsite engineers are responsible for maintaining a top-level CitectSCADA project, ProjectXYZ. Consequently they require read-write privileges for all project folders.

The operators responsible for monitoring plant operations will use the CitectSCADA project at runtime only; consequently operators only have read-only access to the project.

The system administrator on site first identifies all those employees who will use the project, and then divides this pool of users into two user groups:

- **Project Engineers** - responsible for project configuration.
- **Operators** - responsible for the project’s runtime operations.

This is shown in the illustration below.



The administrator creates two user groups to make administering users easier: **ProjectXYZEngineers** and **ProjectXYZOperators**, and assigns engineers to the first group, operators to the second.

**Note:** Creating user groups is optional and makes it easier to handle privileges for multiple users. Creating user groups may be unnecessary if you only have a few users.

The administrator then assigns engineers read-write privileges to the top-level project folder, and operators read-only privileges, like this.

- 1 Select the project folder of the top-level project and display its properties.
- 2 Select the **ProjectXYZEngineers** user group and allow read-write privileges. (Remember that in order to use read-write CitectSCADA projects, read, write, *and* delete privileges must be assigned.)
- 3 Select the **ProjectXYZOperators** user group and deny write privileges. See the section [Making a Project Read-Only](#) for the specific privileges you should assign.
- 4 Apply and save the changes.
- 5 Review the changes to ensure engineers and operators have the correct privileges for their roles.

See Also [Securing an Include Project](#)

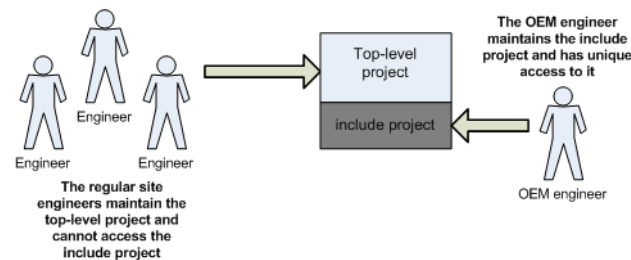
## Securing an Include Project

This section describes a “real-world” situation that might require read-only privileges to be applied to an include project.

**Note:** Before securing an include project, read the section [Read-Only Privileges on Projects](#) for details on operational constraints. Pay particular attention to the section [Read-only on include projects](#).

In this situation, an OEM has configured and delivered an include project that is part of a larger (top-level) project. Because the OEM engineer is solely responsible for maintaining the include (and *only* the include) project, the site administrator assigns the OEM engineer read-write access to the include project, but read-only access to the top-level project. Conversely, the site's regular engineers can access the top-level project but not the include project.

This scenario is shown here:



To set up this scenario the administrator does the following:

- 1 For ease of administration, assigns the site engineers to the user group **AcmeTopEngineers**.  
**Note:** Because there is only one OEM engineer, the administrator did not create a user group for this single user.
- 2 Selects the project folder of the top-level project and displays its properties.
- 3 Selects the **AcmeTopEngineers** user group and allows read-write privileges for this folder. (Remember that in order to use read-write CitectSCADA projects, read, write, *and* delete privileges must be assigned.)
- 4 Applies and saves the changes.
- 5 Selects the include project.
- 6 Selects the user name of the OEM engineer and allows read-write privileges for this folder.
- 7 Applies and saves the changes.
- 8 Reviews the changes made to ensure the correct privileges have been assigned. In particular, because of contractual obligations, the administrator ensures that the privileges assigned to the **AcmeTopEngineers** user group deny read-write access to the include project.

See Also [Securing a Top-level Project](#)

## Making a Project Read-Only

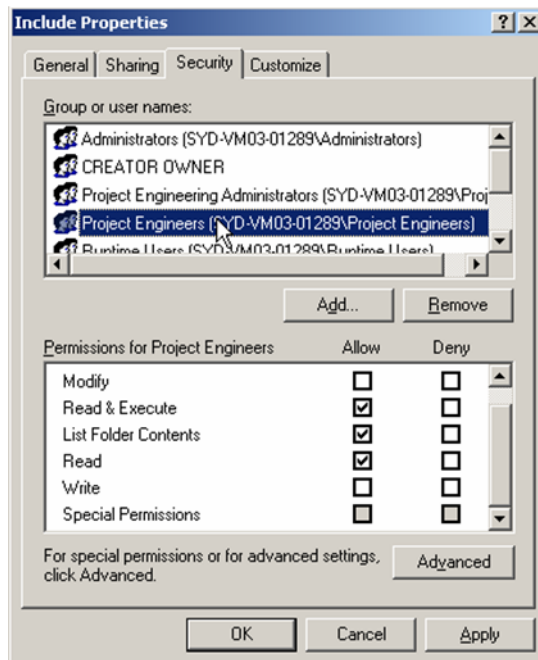
This section describes how to make a project folder read-only by modifying the file security settings for selected users and/or user groups. The read-only project configuration is identical whether you are using Windows 2003 Server, Windows XP, or Windows 2000.

### Notes

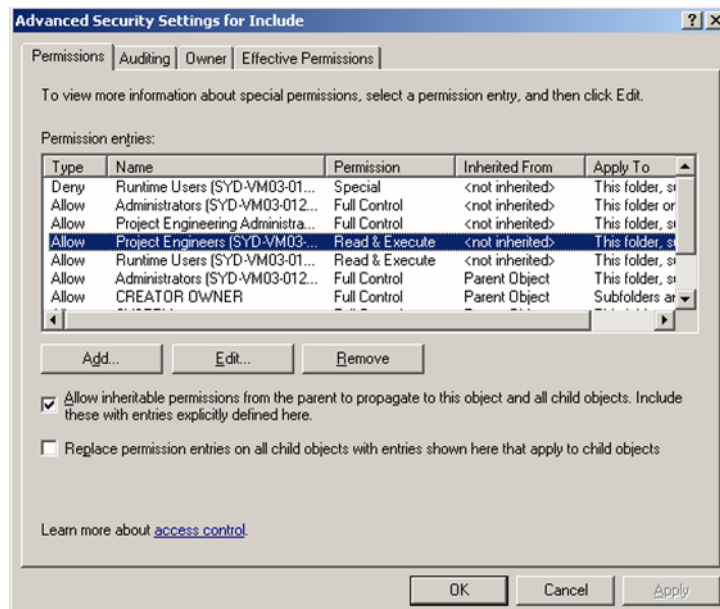
- This procedure assumes that you have already configured your users and user groups (optional), and added them to the User groups for the project folder list for the project you want to secure. For details on creating users and user groups, refer to your Windows documentation.
- Before making your project read-only, make sure that read, write, delete, and execute privileges have been applied appropriately to allow the CitectSCADA configuration and runtime environments to operate correctly. For details, see [Using CitectSCADA with Windows Security](#).
- If you are using Windows XP or Windows 2000 and your file system is FAT32, you must convert the file system to NTFS in order to be able to specify the correct user privileges on that workstation. For details, refer to your Windows documentation.
- To avoid unintended results, read [Read-Only Privileges on Projects](#) before making your project read-only.

### To make a project read-only:

- 1 In Windows Explorer, select the project folder you want to make read-only. By default project folders are located in the folder  
C:\Documents and Settings\All Users\Application  
Data\Citect\CitectSCADA
- 2 Right-click the folder and choose **Properties** from the context menu. The Properties dialog appears.
- 3 Select the **Security** tab.

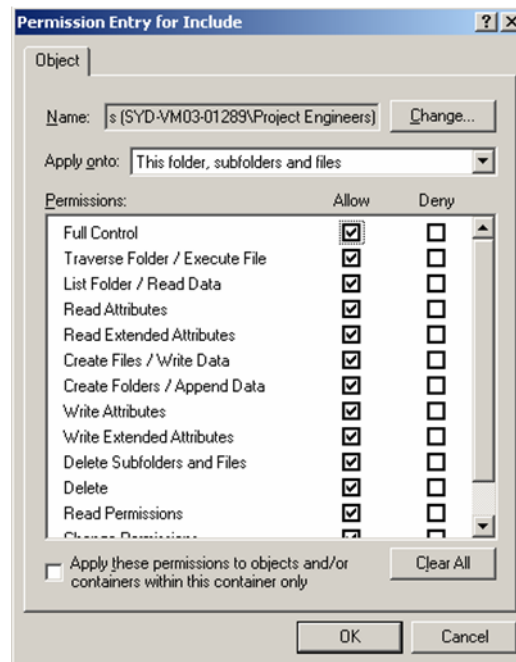


- 4 Select the user and/or user group you want to modify security settings for.
- 5 Click **Advanced**. The Advanced Security Settings dialog appears for the selected user/user group for the project folder.



Make sure the user or user group you want to modify permissions settings for is selected.

- 6 Click **Edit**. The Permission Entry dialog appears.



- 7 Click **Clear All** to clear the current selections and then select the **Allow** check box for the following options:
  - **Traverse Folder/Execute File**
  - **List Folder/Read Data**
  - **Read Attributes**
  - **Read Extended Attributes**
  - **Read Permissions**
- 8 Your selections should look like this once complete:

Full Control	<input type="checkbox"/>	<input type="checkbox"/>
Traverse Folder / Execute File	<input checked="" type="checkbox"/>	<input type="checkbox"/>
List Folder / Read Data	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Read Attributes	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Read Extended Attributes	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Create Files / Write Data	<input type="checkbox"/>	<input type="checkbox"/>
Create Folders / Append Data	<input type="checkbox"/>	<input type="checkbox"/>
Write Attributes	<input type="checkbox"/>	<input type="checkbox"/>
Write Extended Attributes	<input type="checkbox"/>	<input type="checkbox"/>
Delete	<input type="checkbox"/>	<input type="checkbox"/>
Read Permissions	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Change Permissions	<input type="checkbox"/>	<input type="checkbox"/>
Take Ownership	<input type="checkbox"/>	<input type="checkbox"/>

- 9 Click **OK**.
- 10 Click **Apply** to apply the permissions to the selected user/user group, and then click **OK** to dismiss the Advanced Security Settings dialog.
- 11 Click **OK** to dismiss the Properties dialog.

The project folder has now been specified as read-only for the selected user(s) and/or user group(s).

See Also [Securing a Top-level Project](#)  
[Securing an Include Project](#)

### Using CitectSCADA with Windows Security

The CitectSCADA configuration applications are located within the CitectSCADA\bin folder. To enable these applications (as well as the runtime environment) to operate correctly when CitectSCADA is used in an operating environment that uses Windows user security, an administrator must apply read, write, delete, and execute privileges to the following files and folders:

- CitectSCADA\bin folder
- Master.dbf file in the C:\Documents and Settings\All Users\Application Data\Citect\CitectSCADA\User folder

#### Notes

- By default, CitectSCADA gives all users full access to the CitectSCADA bin folder and User folder.
- All users that need to access the CitectSCADA configuration applications must have at minimum read access to *all projects*.



**Note:** Failure to assign appropriate privileges to these files and folders will prevent the CitectSCADA configuration and runtime environments from operating.

See Also [Making a Project Read-Only](#)

## Read-Only Privileges on Projects

This section describes the operational constraints of making a project read-only. This section also describes issues specific to securing top-level and include projects.

**Note:** Before [Making a Project Read-Only](#), make sure you're familiar with the issues described here. Also make sure that the correct privileges have been set in order for the configuration and runtime environments to operate; for details, see [Using CitectSCADA with Windows Security](#).

- [Startup](#)
- [General](#)
- [Graphics and pages](#)
- [Backup and restore](#)
- [Project upgrades](#)
- [Debugging](#)
- [Web deployment](#)
- [Runtime issues](#)

Most of the issues discussed above are common to both top-level projects and include projects. The sections listed below discuss issues specific to these types of projects:

- [Read-only on top-level projects](#)
- [Read-only on include projects](#)

### Startup

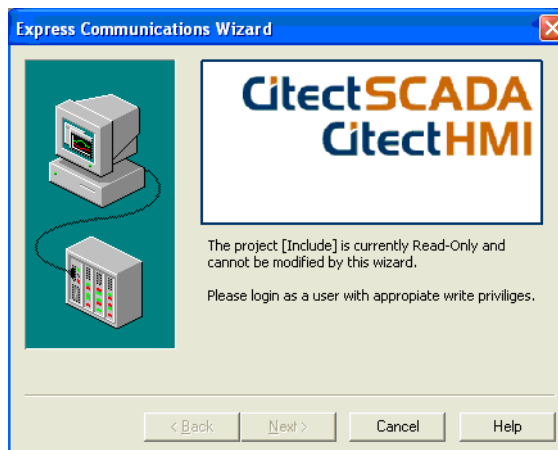
A project is determined to be read-only when Citect Explorer starts up. If the security permissions on the project folder are modified after Citect Explorer has started, *the CitectSCADA configuration applications may not be able to determine accurately that the project is read-only.*

See Also [Using CitectSCADA with Windows Security](#)

### General

When a read-only project is opened using the Graphics Builder, Project Editor, or Citect Explorer, the title bar shows the name of the project and a Read-Only message to indicate the project is read-only.

Opening the Express Wizard for a read-only project displays a message on the first page indicating the project is read-only:



In addition, any menu commands, toolbar buttons, and other operations that perform a write function are grayed out and/or unavailable. For example, the Copy command is available in the Project Editor for a read-only project, but the Cut and Paste commands are not.

When using the Process Analyst, you cannot create views to a project folder that is read-only and an error message is displayed.

### Graphics and pages

You cannot update graphics documents or pages in read-only projects. (You can, however, update pages in a read-write project if that top-level project includes one or more read-only projects.)

Opening an updated graphics page in a read-only project in Graphics Builder will attempt to show the upgraded symbol, but only at the presentation (not disk) level.

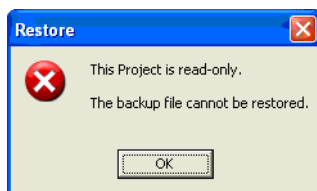
**Note:** Updating graphics documents and/or pages in mid-level include projects is not recommended (if these need to be updated, make all the relevant projects read-write first).

### Backup and restore

Read-only projects can be backed up by any user, regardless of their privileges. However, the backup functionality does not archive the current security

permissions. Consequently, if the project is restored on another machine, the security settings need to be reapplied.

Projects cannot be restored into an existing read-only project; attempting to do so displays the following error message:

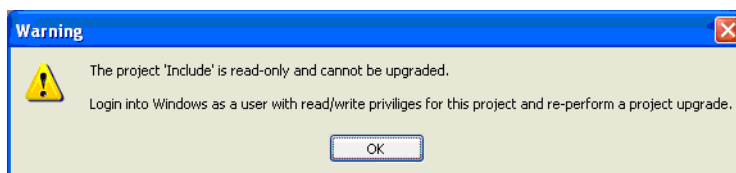


### Project upgrades

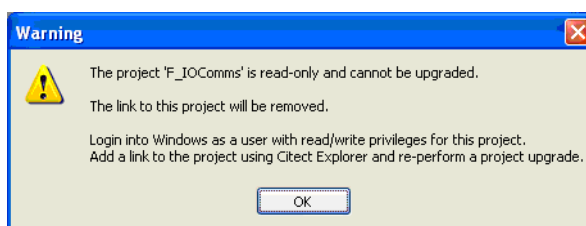
A project upgrade occurs when any of the following occurs:

- [CtEdit]Upgrade=1 is added to the `citect.ini` file.
- A project link is added via Citect Explorer.
- When a project is restored.

When CitectSCADA detects that the include, system, or CSV\_Include project is read-only and the version of CitectSCADA that the project was created under does not match the current version of CitectSCADA, the following error message appears:



In addition, when CitectSCADA detects that a user project is read-only and the version of CitectSCADA that the project was created under does not match the current version of CitectSCADA, the following error message appears:



Any links to the project will be removed and the project tree in Citect Explorer will be updated to indicate this.

If you plan to upgrade a top-level project, you must log on as a user with the appropriate read-write security privileges for this project, add a link to the project in Citect Explorer, and then perform the project upgrade again.

**Note:** Leaving non-upgraded projects in the system is potentially dangerous and could cause project corruption or failure of a configuration application.

#### Debugging

In read-only projects you can set breakpoints when debugging code, but these breakpoints aren't saved when you exit the Cicode Editor.

#### Web deployment

You cannot perform Web deployment with read-only projects.

#### Runtime issues

By default most output operations during runtime occur in the [DATA] or [RUN] location (see below for details). By default CitectSCADA configures the [RUN] location to the project directory. If you intend on making the project directory read-only, you must modify the path(s) to a suitable read-write location.

- **almsav.dat** - Alarm data by default is saved in the [RUN] location, which is usually the project folder. You must change this location if you intend on making the project folder read-only. Alternatively use the [Alarm]SavePrimary and [Alarm]SaveSecondary options in the citect.ini file to control the location of the output.
- **Disk PLCs** - If a user does not have the correct privileges for the [RUN] path, all communications will be offline for disk PLCs. You must change this location if intending to make the project folder read-only.
- **User Cicode functions** - Making a project read-only prevents the use of the following user functions: UserCreate, UserDelete, UserEditForm, UserSetPassword, UserSetPasswordForm. Attempting to use these functions results in an error code 262 (0x0106) ("Cannot open file").
- **Alarm Cicode functions** - Making a project read-only prevents the use of the following alarm functions: AlarmSetDelay, AlarmSetDelayRec, AlarmSetThreshold, AlarmSetThresholdRec. Attempting to use these functions results in a hardware alarm 400 (0x0190) ("Project or file is read-only"). You also cannot modify alarm properties such as threshold or delay.

Any files in your top-level project that require runtime read-write access should be located outside of the project folder.

#### Read-only on top-level projects

Before applying read-only to top-level projects, note the following:

- Before applying read-only to a top-level project, perform a full compile and then apply the relevant security attributes. Once you apply read-only

privilege, you cannot compile a top-level project (see following bullet). For projects that must manipulate files in the top-level project, you can modify the security of individual files to read-write; the project will be regarded as read-only as long as the project folder is denied write privileges.

**Note:** Any files in your top-level project that require runtime read-write access should be located outside of the project folder.

- Projects that are read-write that have read-only include projects as a component can be compiled as usual.
- You can only run a read-only project if it is a top-level project.
- Applying read-only to top-level projects prevents online changes being made to alarms, users, trends, and pages.

See Also [Making a Project Read-Only](#)  
[Securing a Top-level Project](#)

#### Read-only on include projects

Note the following before applying read-only to include projects:

- When compiling a top-level project with a read-only include project, the object IDs of the included projects are skipped and not changed.
- Updating graphics in a mid-level read-only include project is not recommended. If graphics pages need to be updated, make all the relevant projects read-write first.

See Also [Making a Project Read-Only](#)  
[Securing an Include Project](#)

## Securing Runtime Computers

The CitectSCADA runtime system is a Windows-based application that runs in the standard Windows environment. Typically, Windows allows you to run several applications at the same time, however, this may prove a problem if you require a computer to be dedicated to Runtime.

For example, an operator display panel may be used to present alarm notifications. You don't want the Runtime screen minimized or hidden behind another window.

There are several different ways can prevent access to software other than CitectSCADA.

See Also [Running a CitectSCADA server as a Windows service](#)  
[Running a CitectSCADA display client as a shell](#)  
[Disabling Windows keyboard commands](#)  
[Disabling control menu commands](#)  
[Removing the Cancel button](#)

## Running a CitectSCADA server as a Windows service

To prevent access to the operating system while CitectSCADA is running, you can run a server as a service. Once a computer is configured this way, CitectSCADA will start when Windows starts. It will not be necessary for any users to log on.

This is appropriate for situations where CitectSCADA is acting as server and does not require the display of screens or any input from the operator.

For information on how to set up a server as a Windows Service, contact your local support office.

See Also [Running a CitectSCADA display client as a shell](#)

## Running a CitectSCADA display client as a shell

To prevent operators from switching a display client over to a different application during runtime, you can configure Windows to launch with CitectSCADA running as the shell. This will deploy CitectSCADA Runtime as the only available interface for a computer, limiting access to within the context of the current project.

For information on how to set up a display client as a shell, contact your local support office.

[Disabling Windows keyboard commands](#)

## Disabling Windows keyboard commands

The Windows environment provides commands to switch between applications running on the computer at the same time. When using CitectSCADA, these commands might not be desirable - they allow an operator access to other Windows facilities without your direct control. You might be able to disable some of these commands with the *Computer Setup Wizard*. You should consult the Citect Knowledge Base for the latest information on disabling Windows keyboard commands.

See Also [Disabling control menu commands](#)

## Disabling control menu commands

The Control Menu (in the top left corner of an application window) provides commands to position and size the application window, and in some applications to control the application. The runtime system's Control Menu can be tailored to give access to several commands specific to CitectSCADA, such as Shutdown (to shut down the runtime system), or Kernel (to display the Kernel).

You can enable and disable these commands with the *Computer Setup Wizard*.

See Also [Removing the Cancel button](#)

## Removing the Cancel button

When the CitectSCADA runtime system starts, a message box displays the status of the system startup. This message box normally contains a **Cancel** button that allows you to cancel the startup. This button is useful when you are

---

debugging or testing the system. When you have completed testing, you can remove the **Cancel** button from the message box with the *Computer Setup Wizard*, to prevent an accidental cancellation of system startup.





# Chapter 17: Defining and Drawing Graphics Pages

---

A CitectSCADA runtime system usually comprises a series of graphics pages that display on your computer screen(s) and provide a “window into the process.” You can design your pages to provide your operators with control of an [area](#) (or all) of your plant. Your graphics pages can also display the status of your plant by using various graphical items known as objects.

See Also [Creating a New Graphics Page](#)  
[Using Page Templates](#)  
[Using a Browse Sequence](#)  
[Specifying a Startup Page](#)  
[Sizing the Page](#)  
[Defining Page Properties](#)  
[Setting Default Page Settings](#)  
[Understanding the Drawing Environment](#)  
[Using Find and Replace in a project](#)

## Creating a New Graphics Page

You can display your graphics pages on a monitor individually, or display several pages at a time. You can display them in any order, controlled by operator commands or controlled automatically.

To create a new page:

- 1 From the Graphics Builder, click **New**, or choose **File | New**.
- 2 Click **Page**.
- 3 Choose a **Template** upon which to base the page.
- 4 Choose a **Style** for the page.
- 5 Check or clear the **Linked** and **Title bar** as required.
- 6 Choose the **Resolution** for the page.
- 7 Click **OK**.

**Note:** If you create a new page using the Graphics Builder, you must edit the page record (with the Project Editor) to define a [browse sequence](#).

See Also [New Dialog Box](#)

## New Dialog Box

This dialog box is used to create a page, [template](#), symbol, [Genie](#), or Super Genie by selecting a button.

## Working with pages

### To open an existing page:

- 1 Click **Open**, or choose **File | Open**.
- 2 Choose **Type: Page**.
- 3 Select the **Project** where the page is stored.
- 4 Select the **Page**.
- 5 Click **OK**.

**Note:** To delete a page from the project, select the page name and click **Delete**.

### To save the current page:

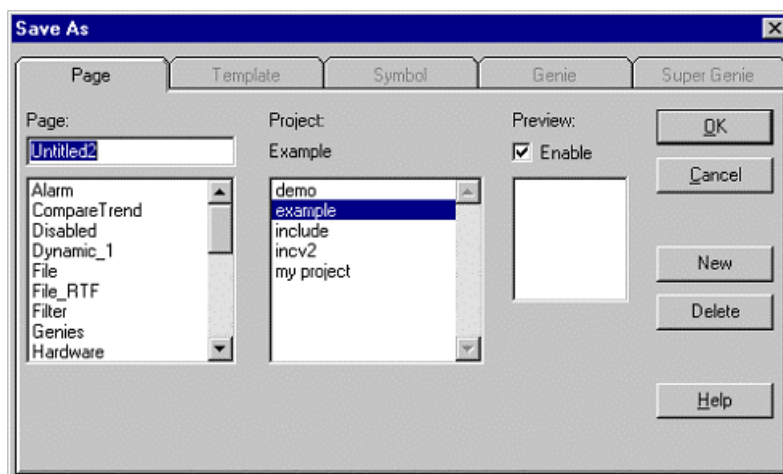
- 1 Click **Save**, or choose **File | Save**.
- 2 Select the **Project** in which to store the page. (The first eight characters of the name must be unique to this page.)
- 3 Click **OK**.

### To save the current page with a new name:

- 1 Choose **File | Save As**.
- 2 Select the project in which the page is stored.
- 3 Enter a name for the page in **Page**. The first eight characters of the name must be unique to this page.
- 4 Click **OK**.

To save all current pages that are open

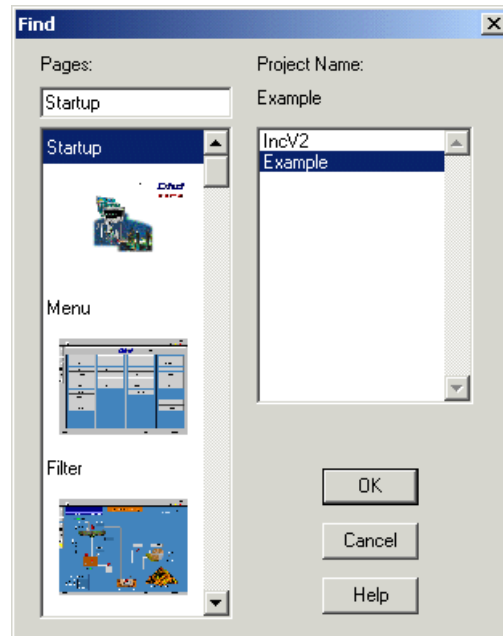
- Choose **File** | **Save All**. (The first eight characters of each page name must be different.)



See Also [Use Template \(new page/template\) dialog box](#)  
[Open/Save As dialog box](#)

To locate a graphics page:

- 1 Choose **File** | **Find**.
- 2 Select a project from the **Project Name** list.
- 3 Browse through the pages in the **Pages** list.
- 4 Click **OK** to view the page.



See Also [Using Find and Replace in a project](#)

To print a graphics page on your printer:

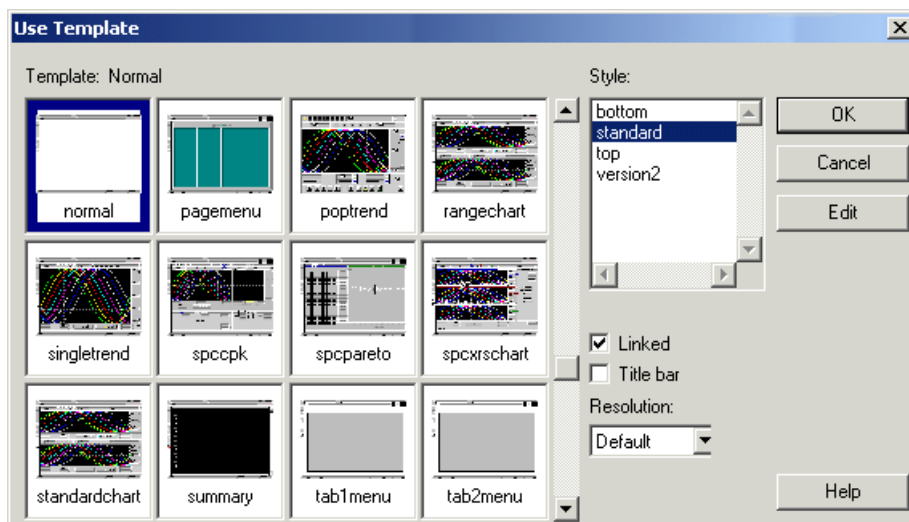
- Choose **File** | **Print**. (This prints without a confirmation dialog.)

To close an existing page:

- Choose **File** | **Close**.

## Use Template (new page/template) dialog box

This dialog box lets you create a new page or template based on an existing template.



### Template

A table of templates on which you can base the new page or template. Use the scroll bar to locate the thumbnail image of the template, then choose the template and click **OK** (or double-click the thumbnail image).

**Note:** To edit the template, select it and click **Edit**

### Style

The style of the page. CitectSCADA templates are grouped into several styles and are available in various page resolutions. When you create a new project, you can choose the style that most suits your taste and application. For details of each style, refer to the "Presenting CitectSCADA" booklet, supplied with your CitectSCADA system.

### Linked

To maintain the link with the original template, select this check box. A page or template that is linked with its original template is automatically updated if the template is changed.

**Note:** You can cut the link to the template at any time with the **Cut Link** command from the Edit menu, but you cannot re-link a page or template with its original template after the link has been cut.

### Title bar

The title to display in the title bar of the page or template.

## Open/Save As dialog box

### Resolution

The screen resolution of the page or template:

Screen Type	Width (pixels)	Height (pixels)
Default	Width of the screen on the computer currently in use	Height of the screen on the computer currently in use
VGA	640	480
SVGA	800	600
XGA	1024	768
SXGA	1280	1024
User	User-defined size	User-defined size

This dialog box lets you open or save a page, template, symbol, [Genie](#), or Super Genie. (To select the type of entity, click the appropriate tab.)

### Page/Symbol/Template/Genie

The name of the graphics page, template, symbol, Genie, or Super Genie.

If you are opening a page, template, symbol, Genie, or Super Genie, select its name from the large window.

If you are saving a page, template, symbol, Genie, or Super Genie, type a name into the smaller input box (or select a name from the large window if you want to overwrite an existing page, template, symbol, Genie, or Super Genie).

**Note:** If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g. you cannot have the same page name in more than one cluster).

### Project

The project in which to save the graphics page, template, symbol, Genie, or Super Genie.

### Library

(For symbols, Genies, and Super Genies only.) The library in which to save the symbol, Genie, or Super Genie. To create a new library, click **New**.

### Style

(For templates only.) The style of the template. To create a new style, click **New**.

### Preview Enable

Displays a thumbnail image of the page, template, symbol, Genie, or Super Genie.

### Title bar

(For templates only.) Specifies whether to include a space for the title bar. If you use a title bar, you will have slightly less display space on screen.

### Resolution

(For templates only.) The screen resolution of the template:

Screen Type	Width (pixels)	Height (pixels)
Default	Width of the screen on the computer currently in use	Height of the screen on the computer currently in use
VGA	640	480
SVGA	800	600
XGA	1024	768
SXGA	1280	1024
User	User-defined size	User-defined size

To delete a page, template, symbol, Genie, or Super Genie, select it and click **Delete**.

### Using Page Templates

You can use many different page types in your CitectSCADA runtime system.

- Mimic pages display the status of the plant, with buttons and other controls to give your operators control of processes within the plant.
- Alarm pages display alarm information.
- Trend pages display a visual representation of past and current activity in the plant.

**Note:** You must create default pages for your alarms (including alarm summary and hardware alarms pages).

To enable you to create your graphics pages quickly, CitectSCADA provides templates for all common page types. Templates help ensure that all pages in your project have a consistent 'look-and-feel'. (Consistency in your project reduces the time your operators need to learn how to use your runtime system.)

See Also [Choosing a page style](#)  
[Linking templates](#)  
[Creating your own templates](#)

### Choosing a page style

CitectSCADA templates are grouped into several styles and are available in various page resolutions. When you create a new project, you can choose the style that most suits your taste and application. For details, see the “Getting Started” booklet, supplied with your CitectSCADA system.

See Also [Linking templates](#)

## Linking templates

When using a template to create a new page, a link can be kept to the template. A page (or template) that is linked with its original template is automatically updated if the template is changed.

When a page is linked to a template, the objects that form the template cannot be accessed from the page by the usual double-click. To display the properties of these objects, hold the **Control** (CTRL) key down and double-click the object you want to view properties for. Alternatively, choose **Tools | Goto Object**, select the group, and click **OK**. However, most of these properties are read-only.

**Note:** You can cut the link to the template at any time using **Edit | Cut Link**, but you cannot re-link a page or template with its original template after the link has been cut.

See Also [Creating your own templates](#)

## Creating your own templates

If your project contains several pages that are similar (for example, menu pages, common processes, or common equipment), you can create your own template (containing all common objects) to use as a base for the pages. You can then create the pages based on the template, and add individual objects to each page.

If you want to delete or change the location of a common object, or to add a new common object, you do not have to change each page; instead, you can change the template. CitectSCADA automatically updates all pages based on the template.

**Note:** When you create a template, save it in the project directory. It is then backed up when you back up the project. Don't modify the standard templates that are supplied with CitectSCADA. When you edit a template, you must use the **Update Pages** command (from the **Tools** menu) to update each page based on the template. Note that the properties of the template are not updated automatically.

### To create a new template:

- 1 Click **New**, or choose **File | New**.
- 2 Click **Template**.
- 3 Choose a **Template** upon which to base the template.
- 4 Choose a **Style** for the template.
- 5 Check or clear the **Linked** and **Title bar** as required.
- 6 Choose the **Resolution** for the template.
- 7 Click **OK**.

### To open an existing template:

- 1 Click **Open**, or choose **File | Open**.
- 2 Select **Type: Template**.



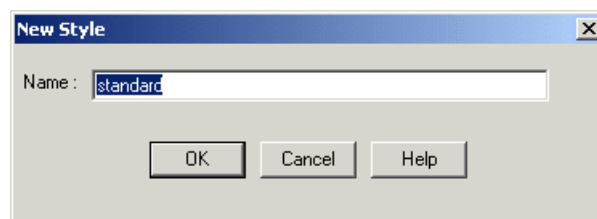
- 3 Select the **Project** where the template is stored.
- 4 Select the **Template**.
- 5 Click **OK**.

**Note:** To delete a template from the project, select the template name and click **Delete**.

**To save the current template:**

- 1 Click **Save**, or choose **File | Save**.
- 2 Select the **Project** in which to store the template.
- 3 Click **OK**.

**Note:** To create a new style for the template, click **New**. You create a new template style using the New Style dialog box.



See Also [New Style dialog box](#)

## New Style dialog box

This dialog box lets you create a new style of templates. Enter a name for your new style in the **Name** text box.

See Also [Creating your own templates](#)

## Using a Browse Sequence

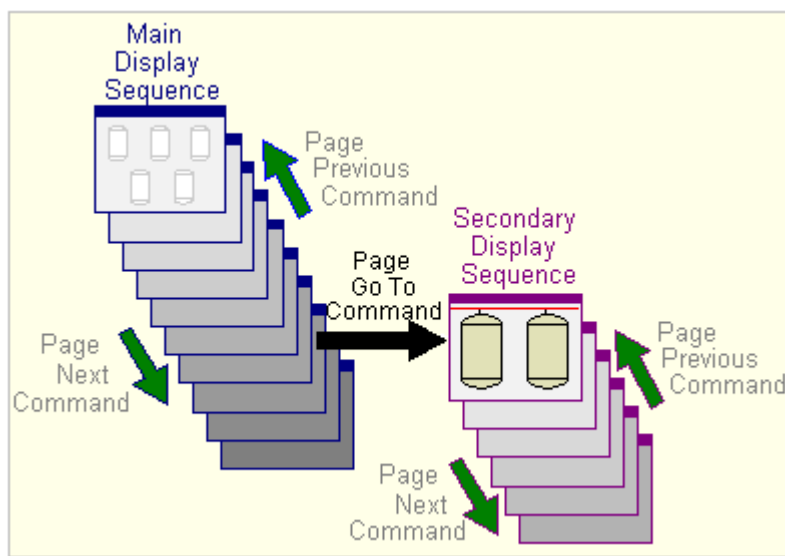
You can link related pages together with a browse sequence. A browse sequence creates a linear navigation sequence for the pages in your system.

When you define a graphics page, you can specify where in the browse sequence the page displays. Within a browse sequence, an operator can display a preceding or following page by choosing **Page Previous** and **Page Next** commands (or a similar set of buttons defined on each page).

When you save a page for the first time it is automatically added to the browse sequence.

You can also use multiple browse sequences by defining a Page GoTo command that displays a page in another (secondary) sequence. The Page Next and Page

Previous commands then display the next and previous pages in the secondary sequence, as in the following diagram:



You do not have to use a display sequence. You can define several Page GoTo commands that display specific pages in an hierarchical structure.

See Also [Specifying a Startup Page](#)

## Specifying a Startup Page

Every CitectSCADA system must have a startup page. When you start your runtime system, the startup page is the first page CitectSCADA displays on your screen. You might want to design your own startup page to display startup information, such as the company logo.

If you want to use a special startup page for the project, draw the page and save it with the name **Startup**. (By default, CitectSCADA always displays a page called "Startup" when your runtime system starts.)

**Note:** You do not have to specify a startup page. If you do not specify a startup page, CitectSCADA displays a default startup page. The default startup page contains command buttons that you can select to display your graphics pages. You can change the name of the default startup page with the Computer Setup Wizard.

See Also [Sizing the Page](#)

## Sizing the Page

By default, new pages in the Graphics Builder take up your entire display area. You can resize them if you want. You can:

- Specify the size of a page when you create it.
- Change the size of a page at any time after it is created.
- Specify that all new pages default to a different size.

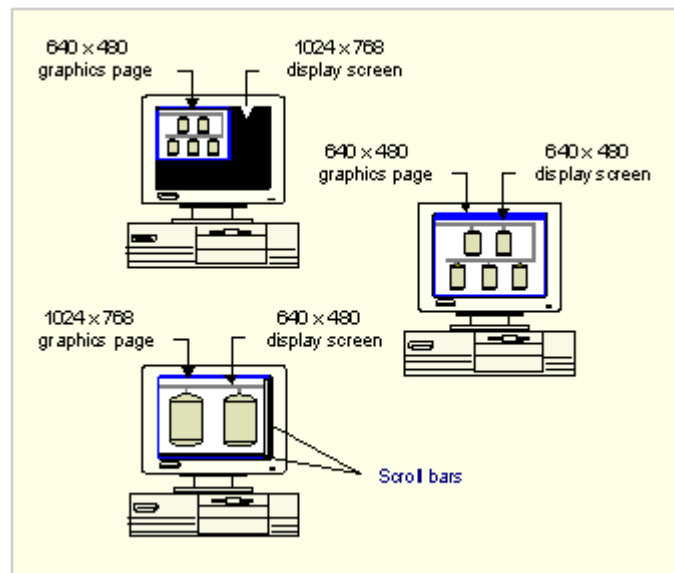
See Also [Page \(screen\) resolution](#)  
[Page size at runtime](#)

### Page (screen) resolution

When you draw a graphics page, determine the resolution of the computer you are using to draw the page, and the resolution of the screen that will display the page in your runtime system.

If a page displays on a screen with a resolution which is greater than the page's resolution, the page will be smaller than the display area. For example, if you draw a page on a VGA screen (640 x 480) and then display it on a XGA screen (1024 x 768), the image displays in the top left corner of the screen, and occupies a little more than half of the screen.

Conversely, if a page displays on a screen with a resolution which is lower than the page's resolution, the page will be larger than the display area. For example, if you draw a page on a XGA screen and then display it on a VGA screen, it occupies more than the entire screen; use the scroll bars to scroll to the area of the page that is not displayed.



See Also [Page size at runtime](#)

## Page size at runtime

By default, a page that is displayed at runtime:

- Appears the same size as when it was saved, unless its parent (the page it was called from) was resized.
- Displays in restored state (you can click **Maximize**).

You can resize the page by dragging the window frame.

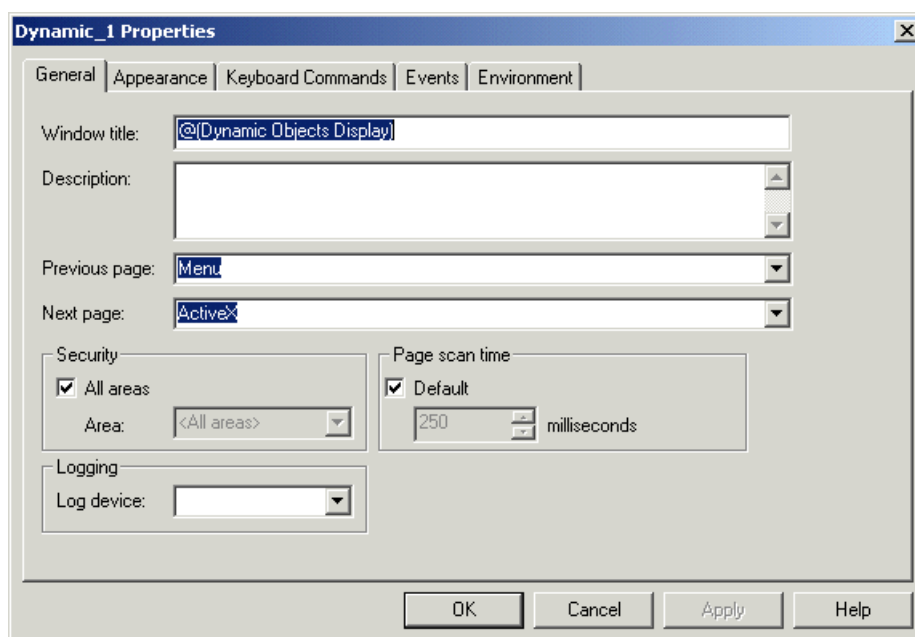
See Also [Sizing the Page](#)

## Defining Page Properties

You can define properties for your graphics pages.

To set up a page:

- 1 Open the page in the Graphics Builder.
- 2 Choose **File | Properties**.
- 3 Use the Page Properties dialog box to define the properties for your graphics pages.



See Also [Page Properties - General](#)  
[Page Properties - Appearance](#)  
[Page properties - Keyboard Commands](#)  
[Page Properties - Events](#)

[Page Properties - Environment](#)  
[Setting Default Page Settings](#)

Page Properties -  
General

Graphics pages have the following general properties:

**Window title**

The title to be displayed on the page at runtime. A window title can have a maximum size of 64 characters.

**Description**

Enter a description of the page, and its various functions and so on up to a maximum of 250 characters. The description is designed for comments only and does not affect how your system performs, nor does the description appear during runtime.

**Previous page**

The page that will precede the current page in the runtime browse sequence (maximum 64 characters). Choose an existing page from the menu or type in a page name.

This property is optional. If you do not specify a Previous page, the Page Previous command is inoperative while this page is displayed.

**Next page**

The page that will follow the current page in the runtime browse sequence (maximum 64 characters). Choose an existing page from the menu or type in a page name.

This property is optional. If you do not specify a Next page, the Page Next command is inoperative while this page is displayed.

**[Security] All areas**

Select the check box if you want the page to belong to all areas (the page can be seen by any operator with view access to at least one [area](#); see [User properties](#)).

**[Security] Area**

Enter the area to which this page belongs. Only users with access to this area can view this page. Click the menu to select an area, or type in an area number directly. If you do not specify an area, the page is accessible to all users.

**[Page scan time] Default**

The Page scan time defines how often this graphics page is updated at runtime. The Page scan time also determines the rate of execution of the While page shown events (i.e. the command(s) which are executed while the page is displayed at runtime).

Select this check box to use the default page scan time (as set using the [Page]ScanTime parameter); otherwise, leave it blank, and enter (or select) another value in the field below. For example, if you enter a page scan time of 200 milliseconds, CitectSCADA will try to update the page every 200 milliseconds, and any While page shown events are executed every 200 milliseconds.

**Note:** You can set the default page scan time using the Computer Setup Wizard.

#### **[Logging] Log device**

This is the device to which messages are logged for the page's keyboard commands. Click the menu to the right of the field to select a device, or type a device name.

**Note:** You must include the *MsgLog* field in the format of the log device for the message to be sent.

#### **[Cluster context] Inherit from caller**

Select this check box to make the current page inherit the default cluster context from the page or cicode task that opened this page (for example using the PageDisplay or WinNewAt Cicode functions). If this is checked you cannot select a specific default cluster. See [About cluster context](#) for more information.

#### **[Cluster context] Cluster**

Specifies a default cluster context for this page.

Click **Apply**, and then click **OK**. To define further properties for the page, select the relevant tabs.

See Also [Defining Page Properties](#)

## Page Properties - Appearance

You define the appearance of your graphics pages by using the controls on the **Appearance** tab.

Graphics pages have the following appearance properties:

#### **[Template] style**

The style (appearance) of the graphics page in the runtime system. You can set a default style (to be applied to all new pages) using the page defaults of existing pages and templates using the Page Properties.

Most users prefer the Standard style. You can view the pre-defined styles by looking in the Include project under Graphics, Templates.

#### **[Template] resolution**

The default screen resolution of the pages:

Screen Type	Width (pixels)	Height (pixels)
VGA	640	480
SVGA	800	600
XGA	1024	768
SXGA	1280	1024
User	****	****

#### **[Template] name**

The name of the template on which the page is based. Choose a template name from the menu.

**Note:** If you are looking for a template that you created yourself, make sure you entered the correct **Style** and **Resolution** above.

#### **[Template] show title bar**

Determines whether the Windows title bar displays (at the top of the page). The title bar contains the title of the window, maximize, minimize and close buttons (at the right hand end of the title bar), and the control menu button (at the left hand end of the title bar).

To display a page in full screen (without a title bar), the size of the page must be the same size as the display (or larger). If the page is smaller than the display, the title bar still displays, even if full screen mode is enabled. Standard templates styles are available for both page sizes.

#### **[View area] width**

The width (in pixels) of the area that the operator can view at runtime. Click the up and down arrows to increase and decrease the width, or type in another value directly.

#### **[View area] height**

The height (in pixels) of the area that the operator can view at runtime. Click the up and down arrows to increase and decrease the height, or type in another value directly.

#### **Background color**

The color that will display in the background of the graphics page.

The preview field to the right of this dialog displays a picture of the selected template. Click **Apply**, then click **OK**. To define further properties for the page, click the relevant tabs.

See Also [Defining Page Properties](#)

## Page properties - Keyboard Commands

The Keyboard Commands property lets you define keyboard commands for the page. A keyboard command is a particular key sequence which executes a command when it is typed in by the operator at runtime.

You can also define a message which will log every time the key sequence is entered.

Operators who do not satisfy the Access requirements specified under **Security** below cannot enter keyboard commands for this page at runtime.

Keyboard commands have the following properties:

### **Key sequence (32 Chars.)**

Enter the key sequences that the operator can enter to execute a command.

You can enter as many key sequences as you like. To add a key sequence, click **Add**, and type in the sequence or select one from the menu. To edit an existing sequence, click the relevant line, and click **Edit**. You can also remove key sequences by clicking **Delete**.

### **Key sequence command**

The commands (set of instructions) to be executed immediately when the selected key sequence is entered. The key sequence command can have a maximum length of 254 characters.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: **Insert Tag**, and **Insert Function**.

### **[Security] Same area as page**

Select this check box to assign the keyboard command to the same area as the page (see [Page Properties - General](#)). Only users with access to this area (and any required privileges) can issue this command or log the message. If you want to assign this keyboard command to another area, do not select this box: enter another area below.

### **[Security] Command area**

Enter the area to which this keyboard command belongs up to a maximum of 16 characters. Only users with access to this area (and any required privileges) can issue this command or log the message. For example, if you enter Area 1 here, operators must have access to Area 1 to issue this command.

Click an area from the menu or type in an area number.

### **[Security] No privilege restrictions**

Tick this box to disable privilege restrictions; otherwise, leave it blank, and enter another privilege below.

The consequences of not assigning a privilege restriction differ according to whether you have used areas in your security setup:



- **No Areas:** All operators have full control of the page.
- **Areas:** An operator will only need view access to the area assigned to this page to have full control over the page (see [User properties](#)).

### [Security] Privilege level

Enter the privilege level that a user must possess to issue this command or log the message (16 characters maximum). For example, if you enter Privilege Level 1 here, operators must possess Privilege Level 1 to issue this command. You can also combine this restriction with area restrictions (see above). For example, if you assign the keyboard command to Area 5, with Privilege Level 2, the user must be set up with Privilege 2 for Area 5 (see [User properties](#)).

Choose a privilege from the menu or type in an area number.

### [Logging] Log Message

A text message sent to the *MsgLog* field of the Log Device when the selected action is performed by the operator at runtime (32 characters maximum). The message can be plain text, Super Genie syntax, or a combination of the two.



If you want to include field data as part of a logged message, you must insert the field name as part of the device format when you configure the device. For instance, in the **Format** field of the **Devices** form, you could enter {MsgLog,20}{FullName,15}. This would accommodate the logging of messages such as **P2 started by John Smith**.

The log device to which the message is sent is specified through the General page properties.

Click **Apply**, and then click **OK**. Click **Clear Property** to clear property details, and disable the property. To define further properties for the page, click the relevant tabs.

See Also [Defining Page Properties](#)

## Page Properties - Events

You use the **Events** tab to assign commands to your graphics pages. These commands can be executed when the page is opened, closed, or whenever the page is open. You can also define different messages to log at the same time.

Page events have the following properties:

### Event

There are three events to which commands can be attached. You can select more than one type of event. Unique commands can be attached to each (i.e., you can perform one task when the page is opened, and another when it is closed).

**[Event] On page entry**

Select this option if you want a command to be executed when the page is first displayed. For example, you could execute a command to extract recipe data from a database into a Cicode variable, to be displayed on the page.

**[Event] On page exit**

Select this option if you want a command to be executed when the operator exits the page. For example, this command could be used to close a database that was opened at page entry.

Do not call the following functions: `PageGoto()`, `PageNext()`, `PagePrev()`, `PageDisplay()`, or `PageLast()`.

**Note:** If you shut down CitectSCADA, exit functions for the currently open pages do not execute. If a particular page exit code must run, call the code before calling the `Shutdown()` function.

**[Event] While page is shown**

Select this option if you want a command to execute continually for the entire time that the page is open. For example, the **While page is shown** command could be used to perform background calculations for this page.

**[Event] On page shown**

Select this option if you want a command to execute when a page is first displayed and all objects on it (including ActiveX controls) have been initialized. If you want to reference ActiveX controls in your command then you should use this event instead of "On page entry" to guarantee the controls are ready to be used. See also [PageInfo](#) Cicode function, type 25.

**Command**

The commands (set of instructions) to be executed immediately when the selected Event occurs (maximum of 254 characters).

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options; **Insert Tag**, and **Insert Function**.

Click **Apply**, and then click **OK**. Click **Clear Property** to clear property details, and disable the property. To define further properties for the page, click the relevant tabs.

See Also [Defining Page Properties](#)

## Page Properties - Environment

You use the **Environment** tab to define environment information for your graphics pages. You can add, remove, or edit page environment variables for a graphics page, template, or Super Genie.

For example, you can design all your loop pages to expand when clicked (a **Tune >>** button) to show tuning parameters. You can use the environment

variable to define the size of the expanded window. The `DspGetEnv()` function would be used as part of the Cicode for the button that expands the window. This means that the Cicode used to expand the window can be generic: you can use the same Cicode each time.

**Note:** If you add environment variables to a template, they are included with any pages created using the template. However, if the template's environment variables are subsequently changed, the corresponding variables of those pages will not be changed. To change a Super Genie's environment variables, see [Super Genie environment variables](#).

### Variables

The environment variables to be added to the page. When the page is opened during runtime, CitectSCADA creates these variables. Their values can then be read using the `DspGetEnv()` function. When the page is closed, the environment variable memory is freed (discarded). For the example above, you would add one variable to define the width of the page, and another to define the height.

To add an environment variable, click **Add**. To edit an existing environment variable, select it, and click **Edit**. (If you click **Add** or **Edit**, a small dialog appears, containing two fields, one for the property and one for its value.) To remove an environment variable, select it, and click **Remove**.

See Also [Defining Page Properties](#)

## Setting Default Page Settings

You can define settings that all new graphics pages will use.

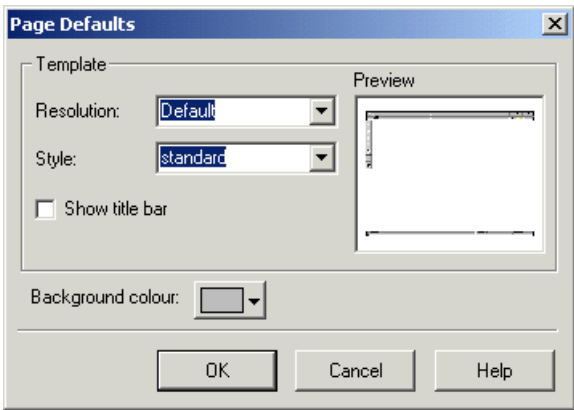
**To set the properties to be used for all new graphics pages:**

- 1 Choose **File | Defaults**.
- 2 Complete the **Page Defaults** dialog box, then click **OK**.

See Also [Page defaults](#)

Page defaults

You use the Page Defaults dialog box to define the screen resolution and style that all graphics pages will use.



You can override these defaults for your pages when you create them or edit them.

[Template] Resolution

Default screen resolution of the standard graphics pages (e.g., alarms pages and standard trend pages):

Screen Type	Width (pixels)	Height (pixels)
VGA	640	480
SVGA	800	600
XGA	1024	768
SXGA	1280	1024
User	****	****

[Template] Style

The style (appearance) of the graphics pages in the runtime system. The style you select is the default style for new pages you add to the project. You can change the style of existing pages and templates using the Page Properties.

Most users prefer the Standard style. You can view the pre-defined styles by looking in the Include project under Graphics, Templates.

[Template] Show title bar

Determines whether the Windows title bar displays at the top of each graphics page. The title bar contains the title of the window, maximize, minimize and close buttons (at the right hand end of the title bar), and the control menu button (at the left hand end of the title bar).

To display a page in full screen (without a title bar), the size of the page must be the same size as the display (or larger). If the page is smaller than the display,

the title bar still displays, even if full screen mode is enabled. Standard templates styles are available for both page sizes

You can override this default for your own pages at the time you create them, or later.

### **Background color**

The color that will display in the background of all new graphics pages.

### **Preview**

This dialog also displays a preview of your page with the defaults applied.

## **Understanding the Drawing Environment**

The Citect Graphics Builder is a feature-rich drawing environment that lets you develop a highly functional interface for your CitectSCADA projects.

See Also

[Grids](#)  
[Guidelines](#)  
[Options](#)  
[Colors](#)  
[Zooming](#)

## **Grids**

You can use a grid to align and place objects with precision. When the grid is active, any objects or groups of objects that you create, move, or re-size snap to the nearest grid intersection.

### **To display the grid:**

- 1 Choose **View | Grid Setup**.
- 2 Click the **Display Grid** check box.

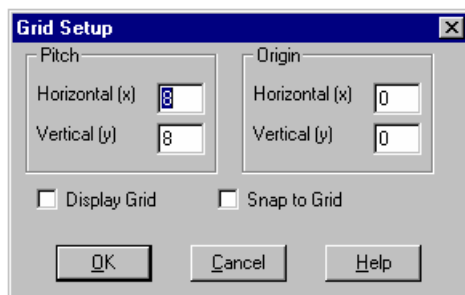
### **To snap to the grid:**

- Choose **View | Snap to Grid**.

By default, the grid is set to 8 x 8 pixels, with the origin located at the top-left corner of your page.

To change the default grid size and location:

- Choose **View | Grid Setup**. The Grid Setup dialog box appears.



See Also [Grid Setup dialog box](#)

## Grid Setup dialog box

This dialog box lets you define the origin and pitch (spacing) for [Grids](#). The grid allows you to align and place objects precisely.

### Pitch

The horizontal and vertical spacing of the grid points (in pixels). The smallest grid size you can specify is 3 x 3 pixels.

### Origin

Specifies the grid origin (base point).

### Display Grid

Displays the grid on screen.

### Snap to Grid (F8)

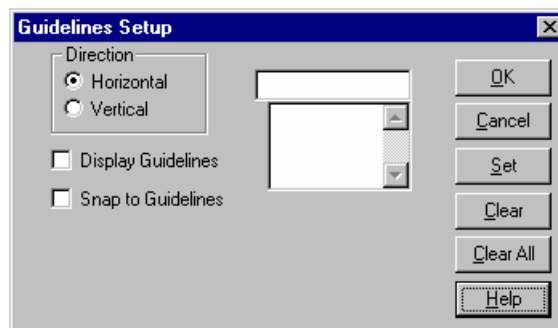
Activates the grid. When the grid is active, any object or group of objects that you create, move, or re-size snaps to the nearest grid intersection.

## Guidelines

You can use horizontal and vertical guides as a straight-edge, to align and place objects precisely. When an edge or the center of an object gets close to a guide, that edge or center automatically snaps to the guide.

To set up guidelines:

- 1 Choose **View | Guidelines Setup**.

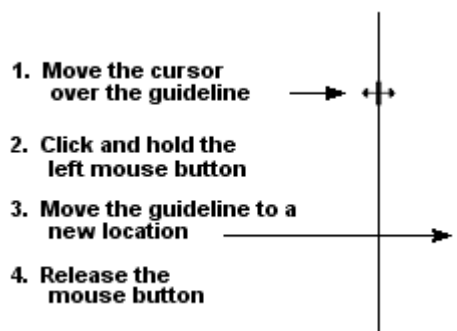


- 2 To run the guideline horizontally across your page, click **Horizontal**.
- 3 Enter a position (distance from the top of your page) for the guideline, and click **Set**.
- 4 To run the guideline vertically down your page, click **Vertical**.
- 5 Enter a position (distance from the left edge of your page) for the guideline, and click **Set**.
- 6 Click the **Display Guidelines** check box.
- 7 Click the **Snap to Guidelines** check box.
- 8 Click **OK**.

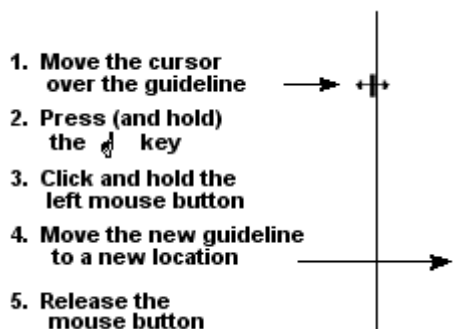
To turn off guidelines:

- Choose **View | Snap to Guidelines**.

To move a guideline:



To create a new guideline with the mouse:



To delete a guideline with the mouse:

- Drag the guideline to the edge of the page.

See Also [Guidelines Setup dialog box](#)

## Guidelines Setup dialog box

This dialog box lets you define 32 horizontal and 32 vertical [Guidelines](#). Guidelines act as a straight-edge, allowing you to align and place objects precisely. When an edge or center of an object gets close to a guide, that edge or center automatically snaps to the guide.

### Direction

The direction of the guideline (horizontal or vertical).

### Display Guidelines

Displays the guidelines on screen.

### Snap to Guidelines (F7)

Activates the guidelines. When the guidelines are active, any object or group of objects that you create, move, or re-size snaps to the nearest guideline.

## Options

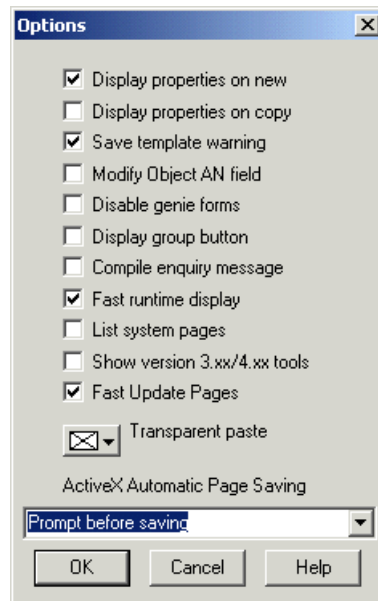
You can define general options for your drawing environment.

To define general options for the drawing environment:

- 1 Choose **Tools | Options**. The Options dialog box appears.
- 2 Enter the relevant details.



3 Click **OK**.



See Also [Options dialog box](#)

## Options dialog box

You use the Options dialog box to set [Options](#) for the project and the Graphics Builder.

### Display Properties on New

Enables the automatic display of the relevant properties dialog when you add an object to the page.

### Display Properties on Copy

Enables the automatic display of the relevant properties dialog when you copy an existing object on the page.

### Save Template Warning

Enables the display of a warning message when you modify and then save an existing template. When you modify an existing template, any graphics pages that are associated with the template are not updated until you perform an **Update Pages** to update each page based on the template.

### Modify AN Field

Allows you to modify the number of the [animation point](#) (AN) of any object. Note that you cannot change an AN to the same number as an existing AN on the graphics page.

### Disable Genie Forms

Disables the display of Genie forms when a new Genie is added to the page or an existing Genie is edited. With Genie forms disabled, a form for each native object in the Genie displays.

### **Display Group Button**

Enables the display of a group button on a Genie dialog. The group button displays a form for each native object in the Genie.

### **Compile Enquiry Message**

Enables the “Do you want to compile?” message window when the project has been modified and **Run** is selected. Normally, CitectSCADA compiles the project automatically (if the project has been modified) when **Run** is selected.

### **Fast Runtime Display**

Enables the fast display of graphics pages in the runtime system.

### **List System Pages**

Specifies that system pages will be included in:

- The list of pages in the Graphics Builder **Open** and **Save** dialog boxes.
- The Page Properties **Previous** and **Next** menus, used for defining a browse sequence for your pages.
- The list of files in the **Contents** area of Citect Explorer.

System pages are prefixed with an exclamation mark (!).

### **Show version 3.xx/4.xx tools**

Enables the old (version 3 and version 4) toolbox. This toolbox contains old tools (such as Slider and Bar Graph), which are no longer necessary, as they can be configured using the Object properties.

### **Fast Update Pages**

Affects the operation of Graphic Builder's “Update Pages” tool. If Fast Update Pages is checked, Graphics Builder only updates modified pages. If not checked, all project pages are updated.

### **Transparent Paste**

Allows you to specify a color that becomes transparent when a bitmap is pasted on a graphics page. This applies to bitmaps that are pasted from the clipboard, or imported from another application.

**Note:** Transparent [data bits](#) are not natively supported by other applications. If pasting a bitmap into an external application, transparent bits will appear as the transparent paste color.

### **ActiveX Automatic Page Saving**

The **ActiveX Automatic Page Saving** menu lets you save graphics pages before inserting an ActiveX control. This guards against the loss of data if you insert custom built ActiveX controls which cause the Graphics Builder to crash. With Automatic Page Saving enabled, if inserting an unstable ActiveX control causes a crash, you do not lose work. When you reopen the Graphics Builder, you can recover the saved page.

You have three choices:

- **Save page before inserting ActiveX controls:** The graphics page is automatically saved with the current page name.
- **Prompt before saving:** A query will display, asking if you want to save the page before inserting an ActiveX control.
- **Do not save automatically:** The graphics page is not saved automatically, and the query is not displayed.

## Colors

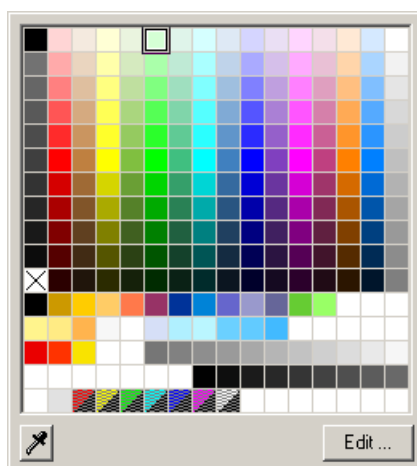
CitectSCADA supports True Color (16.7 million colors) for all static and animated objects, including page backgrounds, imported images, symbols, metafiles and bitmaps.

Wherever a particular page or object property has a color value, a current color button will appear.



To choose a different color to the one currently displayed on the button, click on the small black arrow to the right. This launches the Color Picker.

Color Picker



The first 11 rows of the Color Picker show a set of standard colors, including transparent (marked with a black cross). The remaining rows display any user

defined colors, referred to as **Color Favorites**. This includes flashing colors, represented by a two color block, divided diagonally.

To select one of the colors displayed on the color picker, simply click on it.

If the required color does not appear, you have the option to create a custom color, or match an existing color from one of your graphics pages.

**To match an existing color on a graphics page:**

- 1 Make sure the color you would like to match is present on the page currently displayed in Graphics Builder.
- 2 From the Color Picker, select the Color Selector tool:



- 3 Use the Color Selector to click on the color you would like to match.
- 4 The color you have chosen will now appear in the Color Value Display button.

**To create a customized color:**

- 1 From the Color Picker, click on the Edit button. This will make the Edit Favorite Color dialog appear.
- 2 Use the Edit Favorite Color dialog to create the color you would like to use (See [Edit Favorite Colors dialog box](#) for details).
- 3 **Name** the color if required.
- 4 Use the **Add** button to include the color with the Color Favorites displayed on the Color Picker.
- 5 Click **OK**. The color you have just created will now be selected.

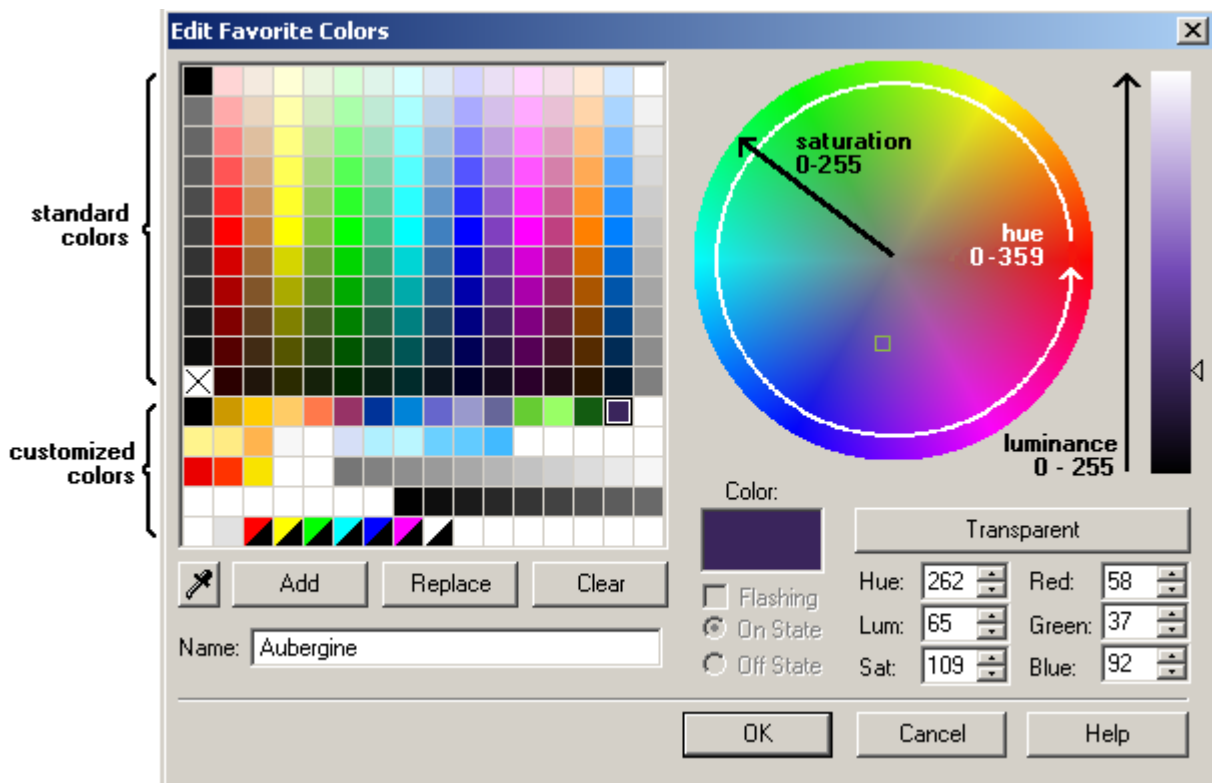
See Also [Edit Favorite Colors dialog box](#)  
[Swap Color dialog box](#)  
[Adjust colors dialog box](#)

## Edit Favorite Colors dialog box

With the Edit Favorite Colors dialog box, you can:

- Make a visual selection of a color you would like to use by clicking on the color wheel.
- Decide the brightness level for a color by clicking on the brightness bar.
- Create a color by entering hue, saturation and luminance values.
- Create a color by entering red, green and blue color values.
- Modify the colors available on the Color Picker by adding, replacing and removing colors on the color grid.

- Name a color, allowing it to be identified on the color grid via a tool tip.



The Edit Favorite Colors dialog contains the following elements:

#### Colors Grid

Displays a selection of predefined colors. The first 11 rows show a set of standard colors, including transparent (marked with a black cross). The remaining rows display the user-defined colors currently added as favorites. This includes flashing colors, represented by a two color block, divided diagonally.

The colors that appear in this grid represent those that are available from the Color Picker.

#### Add

Adds the color currently displayed in the **Color** panel to the user-defined favorites in the color grid. If there are no places available on the grid, you will have to use the **Replace** or **Clear** button instead.

#### Replace

This button allows to you alter a color on the color grid. Select the color you would like to change, adjust its color value, and then hit the **Replace** button. The selected color will be updated to reflect the changes that were made.

### **Clear**

Removes the selected color, leaving an “unused” position on the color grid.

### **Name**

Allows you to associate a name with a predefined color. The name can be viewed as a tool tip in the Color Picker, making it easy to distinguish a specific color among similar shades.

You can associate a name with a newly created color by typing in a name before clicking the **Add** button. You can also apply a name to an existing color by selecting the color, keying in a name and clicking the **Replace** button.

**Note:** The pre-defined color labels are already defined as color names.

### **Color**

The **Color** panel displays the color created by the current settings applied to the Edit Favorite Color dialog.

Note that the values displayed on the Edit Favorite Color dialog automatically adjust to correctly represent the color currently displayed in this panel. The values in each field are not independent.

### **Color wheel**

The color wheel allows you to visually select a color. It represents the full spectrum of colors in a cyclic layout, with color saturation increasing towards the outside edge. Simply click on the wheel to select a color.

### **Brightness bar**

Allows you to visually select the brightness you would like applied to a color. Click on the bar in the appropriate location to apply a brightness level. The bar represents luminance, as the colors move away from pure black as they progress up the bar to pure white.

### **Hue**

Specifies the hue value for the color currently displayed in the **Color** panel. Hue primarily distinguishes one color from another. The value can be between 0 (zero) and 359, representing degrees around the color wheel. For example, zero is a pure red to the right, 180 is pure cyan on the left.

### **Sat**

Specifies the saturation level for the color currently displayed in the **Color** panel. Saturation level increases the further the selected color moves away from gray scale to a pure primary color. The value can be between 0 (zero) and 255.

**Lum**

Specifies the luminance of the color currently displayed in the **Color** panel. Luminance represents the brightness of a color, the value increasing the further the color moves away from black towards pure white. The value can be between 0 (zero) and 255.

**Note:** When you create a color by using HLS values, you may find that the HLS values you specified for a color have changed when you reopen the dialog box. This happens because RGB values are less precise than HLS values, sometimes resulting in several HLS values being assigned the same RGB value. As a result, when the HLS values are generated from the RGB values, some values may change.

**Red**

Indicates the amount of red used to create the color currently displayed in the **Color** panel. The value can be between 0 (zero) and 255. Adjust this setting if you want to create a color using RGB values.

**Green**

Indicates the amount of green used to create the color currently displayed in the **Color** panel. The value can be between 0 (zero) and 255. Adjust this setting if you want to create a color using RGB values.

**Blue**

Indicates the amount of blue used to create the color currently displayed in the **Color** panel. The value can be between 0 (zero) and 255. Adjust this setting if you want to create a color using RGB values.

**Transparent**

Click this button to select transparent. Wherever transparent is used as a color, the background color, or color behind the transparent object, will be displayed.

**Flashing**

Check this box if you want to create a flashing color. A flashing color appears as a diagonally divided panel in the color grid. To create a flashing color, you will have to select an **On State** and **Off State** color.

**Swap Color dialog box**

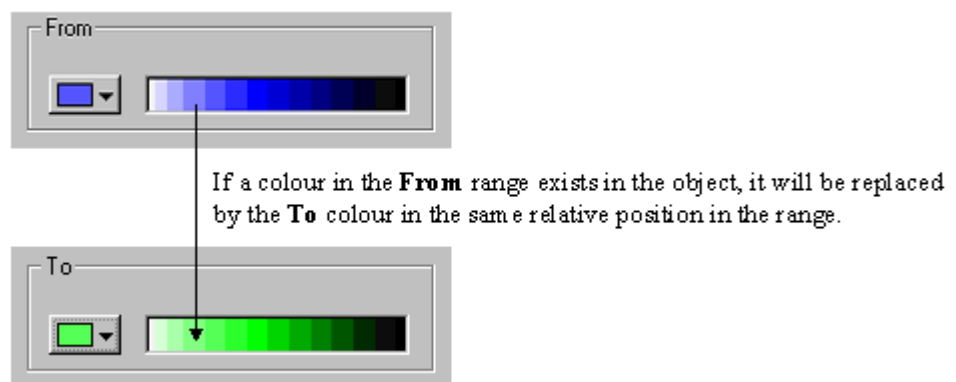
You use the Swap Color dialog box to swap the colors of an object (or group of objects, or a bitmap) to new colors.

**Note:** You may find that you get unexpected results when swapping colors in your bitmaps if they contain areas that have either very high or very low luminance values (that is, white or black areas). This is due to the fact that these areas contain almost no color, only luminance. For example, areas that represent specular highlights (the shine on a brightly illuminated steel conveyor belt, for example) will remain colorless no matter how the color for the rest of the bitmap

is adjusted. Consequently, before working with the Swap Color dialog box and other color tools, you should familiarize yourself with the concepts of hue, luminance, and saturation in order to avoid unexpected results with the images on your graphics pages.

### From

The current color of the object. If you click the **Swap range** check box, it presents a range of colors in varying degrees of brightness ranging from white to black. Any colors in this range that exist in the object are replaced by the corresponding colors from the **To** range as follows:



### To

The color the original object color will be changed to. If you click the **Swap range** check box, it presents a range of colors in varying degrees of brightness from white to black. This allows you to swap a whole range of colors at once.

### From any color

Specifies to change all colors in the object to the new color.

### Swap range

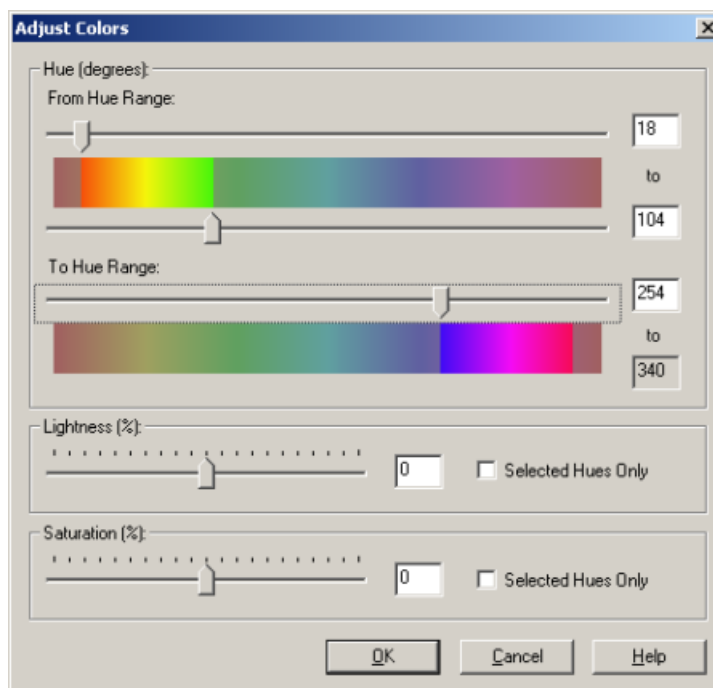
Specifies to swap a range of colors. The **From** and **To** boxes indicate the starting colors in the ranges.

**Note:** You cannot invert colors with **Swap Range** selected. This means, for example, that you could not swap dark red for light green and light red for dark green in one go.

## Adjust colors dialog box

The Adjust Colors dialog allows refined color remapping with Graphics Builder objects.





### Hue (degrees)

The Hue area allows the user to set the color hue range to map to and from. The bars displayed span values of 0 (zero) to 359 degrees, representing the cyclic nature of hue color values. A numeric value for each slider can be keyed in to the field to the right.

- The slider above the **From Hue Range** bar selects the start of the color range that will be mapped.
- The slider below the **From Hue Range** bar selects the end of the color range to be mapped.
- The slider above the **To Hue Range** bar selects the start point for the color range you'll be mapping to. Because the value range is cyclic, the selected area can span across to the left side of the bar.

Note that the range of colors that is excluded by your selection is grayed out, allowing for a visual assessment of the selected range.

### Lightness (%)

The lightness slider allows you to boost the lightness of colors across a range of (negative) - 100% to 100%. If the slider is increased above zero, colors will tend towards white. If the slider is set below zero, colors will move towards black. A numeric value for the slider can be entered into the field to the right.

The **Selected Hues Only** check box applies the lightness setting to only those colors that will be remapped. Leaving the box unchecked allows the lightness to be adjusted for all colors.

**Saturation (%)**

The saturation slider allows you to boost the saturation of color across a range of (negative) -100% to 100%. If the slider is increased above 0, colors will tend towards primary colors. If the slider is set below zero, colors will move towards grayscale. A numeric value for the slider can be entered into the field to the right.

The **Selected Hues Only** check box applies the saturation setting to only those colors that will be remapped. Leaving the box unchecked allows saturation to be adjusted for all colors.

Zooming

Use the Zoom command to view an enlarged view of your drawing. The Zoom command displays a zoom box on screen that magnifies the area around your cursor, enabling you to position or draw objects precisely.

Procedure	Description
To display the Zoom box	Choose <b>View   Show Zoom</b> .
To hide the Zoom box	Double-click the control menu box (on the Zoom box) or choose <b>View   Show Zoom</b> .

You can change the cursor into crosshairs that extend the full width and length of the drawing area. When you move away from the drawing area, the normal pointer reappears, allowing you to select commands and tools.

Procedure	Description
To toggle the cursor between a crosshair cursor and a pointer	Choose <b>View   Cross Hair Cursor</b> .
To hide the status bar	Choose <b>View   Show Status Bar</b> . Choose the command again to redisplay the status bar.
To hide the Toolbox	Double-click the control menu box (on the Toolbox) or choose <b>View   Show Tools</b> . Choose the command again to redisplay the Toolbox.
To hide the toolbar	Choose <b>View   Show Tool Bar</b> . Choose the command again to redisplay the tool bar.
To change the speed of the cursor when using the mouse	Choose <b>View   Mouse Slow Speed</b> .
To change the speed of the cursor when using the cursor keys	Choose <b>View   Cursor Keys Slow Speed</b> . <b>Hint:</b> Move the cursor on screen by using the left, right, up, and down cursor keys.

Using libraries

You can store frequently used objects or groups of objects (including bitmap objects) in a library as symbols. You can then paste these symbols onto your page.

After you paste a symbol from the library onto a graphics page, it can be moved, re-sized, re-shaped, brought to the front, and its properties can be edited, just like any other type of object.

You can paste a symbol from the library to the page:

- As an **unlinked** symbol; the pasted symbol is not updated with changes to the symbol in the library.
- As a **linked** symbol; the symbol on the page is updated when the symbol in the library is changed (to alter the properties of a symbol in the library, open the library and edit it in the library). If you edit the symbol from the page and then change the source symbol in the library, the pasted symbol changes. For example, if you double the size of a pasted symbol, then double the size of the symbol in the library, the pasted symbol doubles again. You can cut the link to the library by using the **Edit | Cut Link** command.

When you save an object in a library, the current properties of that object are saved with it. When you paste it as a symbol to a graphics page, they are used as defaults for the symbol. Pasted symbols have different Appearance properties to those of normal objects: you can only specify a visibility property.

When a symbol is pasted onto a page, the objects that form the symbol cannot be accessed from the page by double-clicking the symbol. To display the properties of these objects, hold the **Control** (CTRL) key down and double-click the specific object. Alternatively, you can select **Goto Object** from the **Tools |** the group, and click **OK**. However, if the link to the library is retained, most of these properties are read-only.

A symbol would be useful, for example, if you have defined a command button with a particular security classification, and you need to use it on quite a few graphics pages. You could save it as a symbol, and each time you want to use the button, paste it from the library. Each time it is pasted, it will have the same properties.

**Note:** CitectSCADA is supplied with a comprehensive range of symbols that you can use in your project(s). These symbols are stored in several libraries in the "Include" project. When a library is saved, the first eight characters of the library name must be unique to that library.

#### Copying an object to the library

You can copy an object to the library so that you can use it later on other graphics pages.

**To copy an object to the library (i.e. make it a symbol):**

- 1 Click **Select**.
- 2 Select the object (or group of objects).
- 3 Choose **Edit | Copy to Library**. The Copy To dialog box appears.

### Copy To dialog box

This dialog box lets you copy an object (or group of objects) to the library as a symbol.

Feature	Description
Symbol	A name for the symbol.
Library	The project library where the symbol is stored.
Project	The project where the library is stored.
Preview Enable	Displays a thumbnail image of the symbol.

**Note:** To edit the symbol, select it and click **Edit**. To create a new symbol, click **New**.

### New Library dialog box

This dialog box lets you create a new library for the symbol, Genie, or Super Genie. Enter a new **Name** for your new library. (The first eight characters of the library name must be unique to this library.)

## Using symbols

You can create symbols to use on your graphics pages.

#### To create a new symbol:

- 1 Click **New**, or choose **File | New**.
- 2 Select **Type: Symbol**

**Note:** You can also create an object (or group of objects on the page) and then choose **Edit | Copy to Library**.

#### To open an existing symbol:

- 1 Click **Open**, or choose **File | Open**.
- 2 Select **Type: Symbol**.
- 3 Select the **Project** where the library is stored.
- 4 Select the **Library** where the symbol is stored.
- 5 Select the symbol you want.
- 6 Click **OK**.

**Note:** To delete a symbol from the library, select the symbol name and click **Delete**.

#### To save the current symbol:

- 1 Click **Save**, or choose **File | Save**.
- 2 Select the Project in which the library is stored.
- 3 Select the Library in which the symbol is to be stored.

4 Enter a name for the symbol.

5 Click **OK**.

For details on using symbols on graphics page, see [Using Pasted Symbol Objects](#).

## Bitmaps

A bitmap image is a drawing object represented as an array of pixels (or dots), rather than as individual entities. A bitmap is treated as a single object that you can move, copy, and reshape. Because you can edit individual pixels in a bitmap, you can use bitmaps for vignettes and image blending.

You can create bitmaps with the Citect Bitmap Editor, or import bitmaps from any other Windows-based bitmap editor.

In a runtime system, CitectSCADA displays bitmaps differently to other objects. Bitmaps are mapped directly to the screen (i.e., each pixel in the image corresponds to a pixel on the screen). Objects are stored as a series of instructions, and are drawn on the screen in the same order as they were drawn on the graphics page.







### CitectSCADA Bitmap Editor

You use the Bitmap Editor to create and edit bitmap images. You can use bitmap images on your graphics pages, and as animated symbols.

The background color in the Bitmap Editor is always transparent, indicated as a white pixel with a black dot at the center. To draw with the background (transparent) color, click (and hold) the right mouse button.

Flashing colors are represented by a diagonally split pixel, indicating the on-state and off-state colors used.

The tool bar of the Bitmap Editor has the following buttons:

-  Exits the Bitmap Editor and saves editing changes.
-  Exits the Bitmap Editor and discards changes.
-  Zooms in on the image.
-  Zooms out on the image.
-  Selects a color from the image to set as the current color (keyboard shortcut is **Shift+P**). You can also select the current color from the color swatch.
-  Displays the **Bitmap Size** dialog box where you can view the image's current dimensions and edit the image's edge.

**To resize a bitmap:**

- 1 In Graphics Builder, click the bitmap.
- 2 Choose **Tools | Bitmap Editor**, or press **F9**.
- 3 Click **Resize**. The Bitmap Size dialog box appears.
- 4 Select a mode. Click **Grow** to enlarge the image, or **Shrink** to reduce it.
- 5 For each side of the bitmap, specify how many pixels you want to add or remove, then click **OK**.

**To set a color from a bitmap as the current color:**

- 1 In Graphics Builder, click a bitmap.
- 2 Choose **Tools | Bitmap Editor**, or press **F9**.
- 3 Click **Eye Dropper**, and then click a color in the image. The selected color becomes the current color, and is used when you click elsewhere on the image.

**To convert an object (or objects) to a bitmap:**

- 1 Select the object(s).
- 2 Choose **Tools | Convert to Bitmap**.

**Note:** The Convert to Bitmap operation is only supported in 8-bit (256) color mode.

**To invoke the Bitmap editor:**

- Choose **Tools | Bitmap Editor**.

**To paste a bitmap (from another application):**

- 1 Create the image in an external application.
- 2 Use the external application's copy command to copy the image to your computer's clipboard.
- 3 Switch to the graphics builder.
- 4 Choose **Edit | Paste**.

You can edit pasted bitmaps by selecting the object and then choosing **Edit | Properties**.

**To import a graphic:**

- 1 Choose **File | Import**. The Import dialog box appears.
- 2 Select the file you want to import by using the Import dialog box.
- 3 Click **OK** (or click the file that you want to import and drag it onto a page in Graphics Builder).

You can edit imported bitmaps by selecting the object and then choosing **Edit | Properties**.

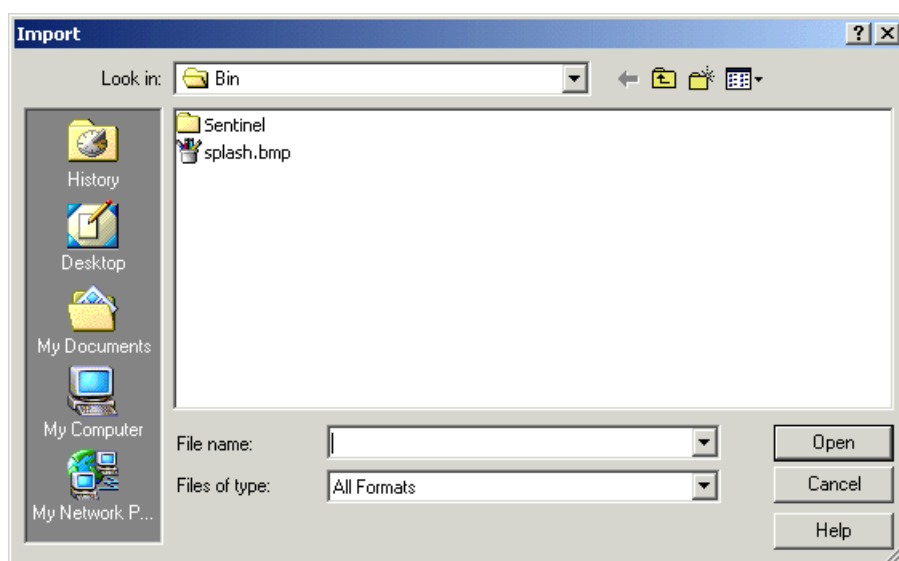
#### To import a flashing graphic:

- 1 Choose **File | Import as Flashing**. The Primary Import dialog box appears.
- 2 Select the first file you want to use for your flashing image.
- 3 Click **OK**. The Flashing Import dialog box appears.
- 4 Select the second file you want to use for your flashing image.

### Import dialog box

You use the Import dialog box to import a graphic produced with a different application.

If you have selected **Import as Flashing**, two Import dialog boxes appear in sequence, allowing you to choose two images that you want to implement as a flashing symbol. The **Import Primary** dialog allows you to select the initial image used; the **Import Flashing** dialog allows you to choose the second images used.



#### Look in

The drive and directory where the graphic is stored.

#### File Name

The name of the graphics file.

#### Files of Type

The type of graphics file. CitectSCADA supports the following file formats:

- Windows 3.0 bitmaps (\*.BMP, \*.DIB, \*.RLE)
- PCX format bitmaps (\*.PCX)
- Text files (\*.TXT)
- AutoCAD DXF Files (\*.DXF), 2D only. The binary format is also supported.
- Windows metafiles (\*.WMF)
- Encapsulated Postscript (\*.EPS)
- Fax Image (\*.FAX)
- Ventura Image (\*.IMG)
- JPEG (\*.JPG, \*.JIF, \*.JFF, \*.JGE)
- Photo CD (\*.PCD)
- Portable Network Graphic (\*.PNG). (PNG files with alpha channels are not supported.)
- Targa (\*.TGA)
- Tagged Image Format (\*.TIF)
- WordPerfect (\*.WPG)




















## Chapter 18: Using Objects

Objects are basic drawing entities that you add to your graphics pages. Objects are drawn using the tools in the drawing toolbox, and can be moved, reshaped, or copied after they are drawn. Objects are defined by a set of properties, which are assigned when the object is drawn, or afterwards, by double-clicking (these properties will override any conflicting Cicode Display functions).

Most objects can be assigned keyboard commands and access rights, and can be configured in such a way that they change dynamically at runtime when an [expression](#) returns a certain value, or a variable tag changes state. They can even be used as sliders to change the state of a variable.

**Note:** If an object is part of a group, part of a pasted [Genie](#) or symbol, or part of the page's template, you can still access its properties. Simply hold down the **Control** (CTRL) key and double-click the object. Alternatively, choose **Tools | Goto Object**, click the object, and then click **OK**. However, if there is still a link to the original Genie/symbol/page template, the object properties are mostly read-only.

CitectSCADA has 17 different types of objects, each with its own tool:

	Free Hand Line		Straight Line
	Rectangle		Ellipse
	Polygon		Pipe
	Text		Number
	Button		Symbol Set
	Trend		Cicode Object
	Pasted Symbol		Pasted Genie
	ActiveX		Process Analyst
	Database Exchange Control		

See Also [Using groups](#)

[Reshaping objects](#)  
[Using bitmaps](#)  
[Importing graphics](#)

## Using groups

CitectSCADA allows you to group multiple objects. A group has a unique set of properties (much the same set as an object) which determine the runtime behavior of the group as a whole. (The properties of the individual objects in the group remain unchanged.) By defining group properties, you can specify that the entire group changes dynamically under specific runtime conditions (for example, when an [expression](#) returns a certain value, or a variable tag changes state).

To edit or view the properties of a group, double-click it. If there are several groups on your page, choose **Tools | Goto Object**. This allows you to see which groups and objects are on your page, making it easier for you to select the object you want to edit. It also allows you to display the properties of the objects (or groups) that make up the group. (You can also edit the properties of an object in the group by holding down the **Control** (CTRL) key and double-clicking the object. Alternatively, select **Goto Object** from the **Tools** menu, select the object, then click **OK**.) This is useful if your page has groups within groups.

A group can be a mix of objects and other groups.

See Also [Reshaping objects](#)

## Reshaping objects

Pipe, Polyline, or Polygon objects can be edited to change their shape. Each of these objects consist of a continuous series of lines drawn between structural anchor points called nodes. Nodes are visible when an object is selected. Each node appears as a small square located at specific anchor points along the object. There is always a node located at the start and end of a polyline or pipe, and at every change of direction in an object's shape.

Pipe, Polyline, and Polygon objects can have their shapes changed in many ways. Their nodes can be selected individually or by group and moved to a different position, thus changing the shape of the object. The Pipe, Polyline, and Polygon objects also support node adding and deleting.

### Reshaping a line object

Line objects also have nodes, but behave in a more restricted manner than Pipe, Polyline, or Polygon objects.

A straight line can only consist of two nodes, (a start node point and an end node point). These can be individually selected to move the line to a different position, or at least change its direction. The Line object does not support node adding. To achieve the same result as adding a node to a Line object, create a

---

Polyline object instead. Deleting either of the (only two) nodes of a Line object will delete the whole Line object completely.

See Also [Using bitmaps](#)

## Using bitmaps

A bitmap image is a special object, represented as an array of pixels or dots, rather than as individual entities. Bitmaps are treated as single objects that you can move, copy, and reshape. You can create and edit bitmaps with the Citect Bitmap Editor. Because you can edit the individual pixels in a bitmap, you can use bitmaps for more 'artistic' images, such as vignettes and image blending.

See Also [Importing graphics](#)

## Importing graphics

The Graphics Builder has several file format filters to allow you to import graphics from other applications, such as drafting programs, illustration programs, presentation packages, scanners, etc. After a graphic is imported, you can use the graphics builder to edit the image.

Graphics files can be dragged from a third party application (such as Windows Explorer), and dropped onto a page in the Graphics Builder.

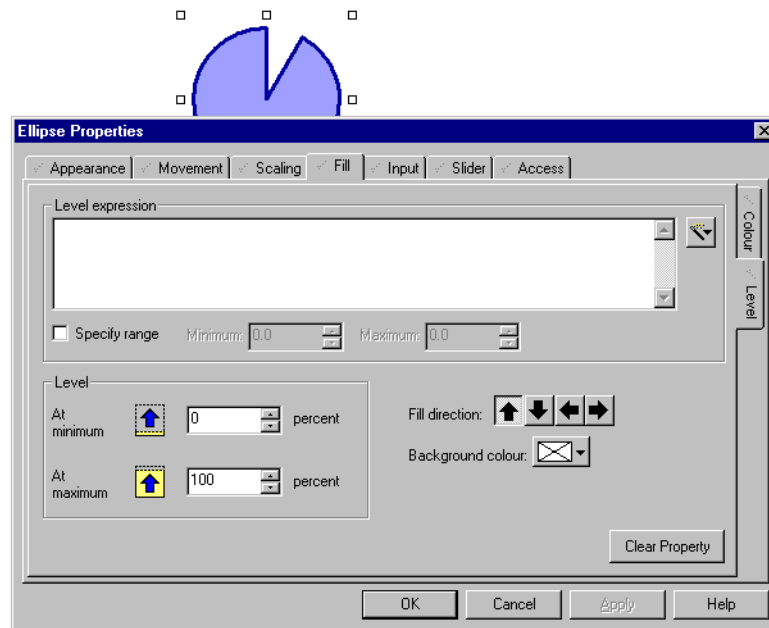
By default, unavailable colors in an imported bitmap are dithered. To disable this feature, choose **Tools | Options** in the Graphics Builder, and clear the **Dither bitmaps on paste** option. If dithering is disabled, unavailable colors are replaced by the closest match in your color palette.

## Object Properties

CitectSCADA enables you to define the properties of your [objects](#).

The properties of an object are defined in the Properties dialog box. When you draw an object, the properties dialog appear, allowing you to define the attributes.

You can also double-click the object (or choose **Edit | Properties**) to display the properties dialog. The following illustration shows a sample object and its associated properties dialog.



If an object is part of a group, part of a pasted Genie or symbol, or part of the page's template, you can access the object's properties by pressing **CTRL** and double-clicking the object. Alternatively, choose **Tools | Goto Object**, click the object, and then click **OK**.

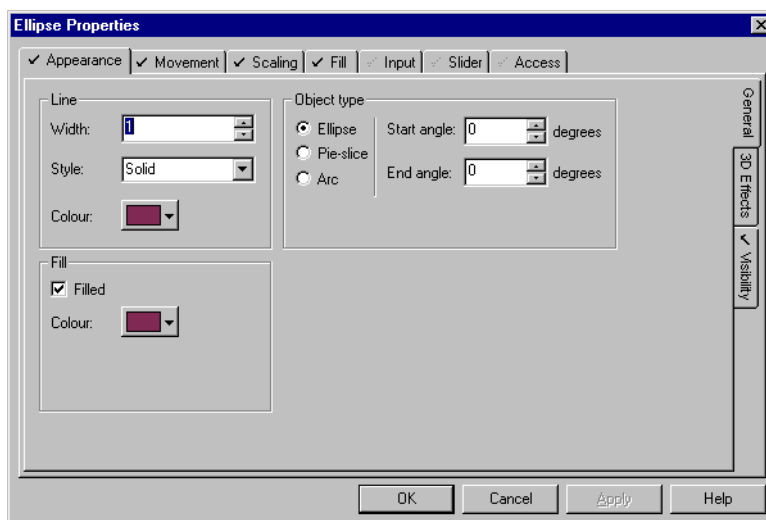
See Also [Appearance](#)  
[Movement](#)  
[Scaling](#)  
[Fill](#)  
[Input](#)  
[Slider](#)  
[Access](#)

## Appearance

Click the **Appearance** tab to define the appearance of the object, such as line style, and shadowing. You can also specify when the object will be hidden from the operator (e.g. when `DIGITAL_TAG` is OFF).

The check mark to the left of the Appearance tab tells you when an appearance property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.

Click the other tabs to define more properties for the object. Most properties work together; for example, an object could possess color fill, movement, and scaling properties simultaneously.



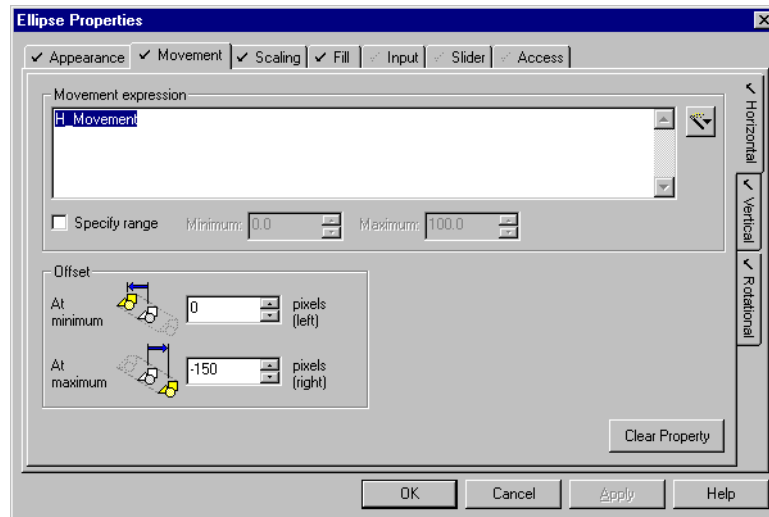
See Also [Object Properties](#)  
[Movement](#)

## Movement

Click the **Movement** tab to move the object vertically or horizontally, or to rotate it, depending on the return of an [expression](#), or the state of a tag.

The check mark to the left of the Movement tab tells you when a Movement property has been configured. The check marks in the tabs down the right of the dialog tell you which property is configured.

Click the other tabs to define more properties for the object. Most properties work together; for example, an object could possess color fill, movement, and scaling properties simultaneously.



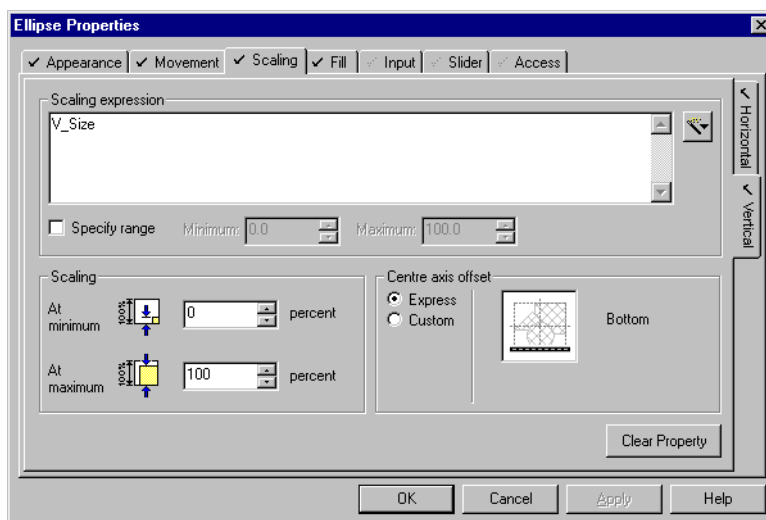
See Also [Object Properties](#)  
[Scaling](#)

## Scaling

Click the **Scaling** tab to scale the object both vertically or horizontally, depending on the return of an [expression](#), or the state of a tag.

The check mark to the left of the Scaling tab tells you when a Scaling property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.

Click the other tabs to define more properties for the object. Most properties work together, for example, an object could possess color fill, movement, and scaling properties simultaneously.



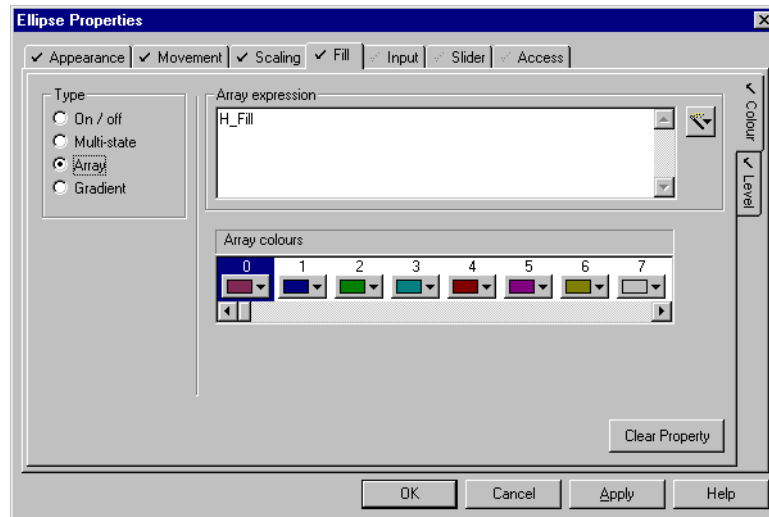
See Also [Object Properties](#)  
[Fill](#)

## Fill

Click the **Fill** tab to specify the color which is to fill the object, and the level to which the object will be filled. The fill properties can change dynamically, depending on the return of an [expression](#), or the state of a tag etc. (for instance, you could use this tab to visually reflect tank levels).

The check mark to the left of the Fill tab tells you when a Fill property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.

Click the other tabs to define more properties for the object. Most properties work together, for example, an object could possess color fill, movement, and scaling properties simultaneously.



See Also [Object Properties](#)  
[Input](#)

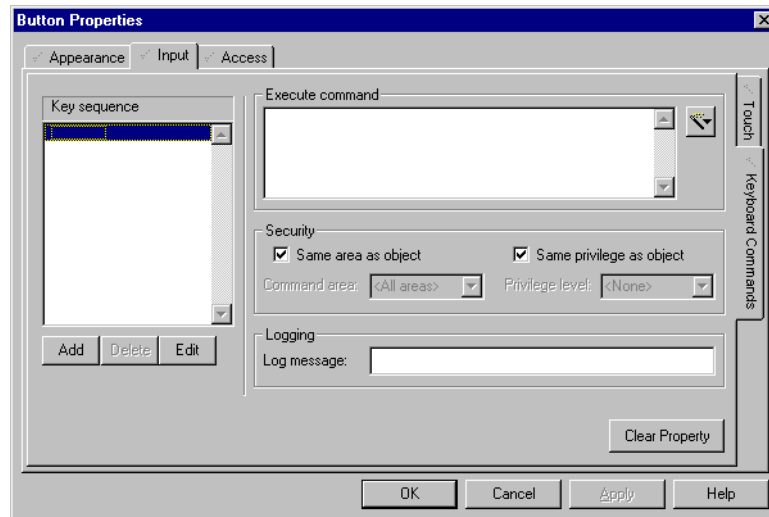
## Input

Click the **Input** tab to specify the command to be executed, and the message to be logged when an operator clicks on the object. You can also define keyboard commands for the object, and protect them with [area](#) and privilege security.

The check mark to the left of the Input tab tells you when an Input property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.



Click the other tabs to define other object properties. Most properties work together; for example, an object could possess color fill, movement, and scaling properties simultaneously.



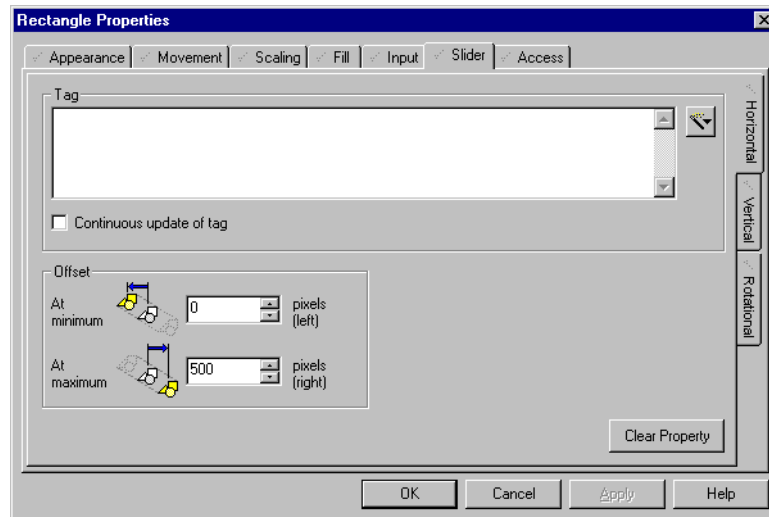
See Also [Object Properties](#)  
[Slider](#)

## Slider

Click the **Slider** tab to use the object as a slider. A variable can be associated with the object, and when the operator moves the object, the value of the variable will change. Objects can be set up to slide vertically and/or horizontally, or they can be rotated.

The check mark to the left of the Slider tab tells you when an Slider property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.

Click the other tabs to define other object properties. Most properties work together; for example, an object could possess color fill, movement, and scaling properties simultaneously.



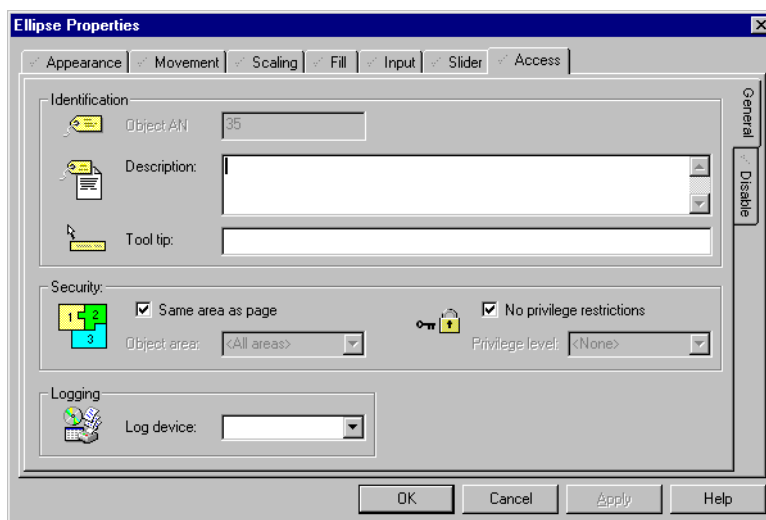
See Also [Object Properties](#)  
[Access](#)

## Access

Click the **Access** tab to assign an area or privilege to the object. Operators without appropriate access rights will not be able to use sliders, object specific keyboard commands etc. It also allows you to disable the object completely under certain runtime circumstances. This means that the object can be embossed, grayed, or even hidden.

The check mark to the left of the Access tab tells you when an Access property has been configured. The check marks in the tabs down the right of the dialog indicate which property is configured.

Click the other tabs to define more properties for the object. Most properties work together; for example, an object could possess color fill, movement, and scaling properties simultaneously.



### Clear Property

Click **Clear Property** to remove the configuration details for this property.

Clear Property

### Apply

Click **Apply** to bring your changes into effect. **Apply** allows you to view your changes without closing the Properties dialog.

Apply

See Also [Object Properties](#)

## Manipulating Objects

Objects can be manipulated in various ways, such as moving, resizing, and grouping.

See Also [Selecting objects](#)  
[Moving objects](#)  
[Resizing objects](#)  
[Deleting objects](#)  
[Locking/unlocking objects](#)  
[Grouping objects](#)  
[Copying and pasting objects](#)  
[Send to Back and Send Backwards](#)

[Bring to Front and Bring Forwards](#)

[Aligning objects](#)

[Rotating objects](#)

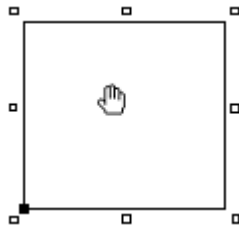
[Mirroring objects](#)

[Locate an object](#)

## Selecting objects

To select a single object:

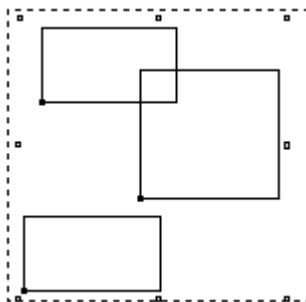
- 1 In Graphics Builder, click **Select**.
- 2 Click the object. The object's sizing handles appear, and the cursor changes from an arrow to a hand while on the object.



**Note:** To add other objects to the selection, hold the **Ctrl** key and click each object in turn. To deselect an object from a group selection, whilst still holding the **Ctrl** key, click the object again. To select all objects in the drawing, use **Select All** from the Edit menu. To deselect all objects, click anywhere other than on an object.

To select a group of objects using a marquee box:

- 1 In Graphics Builder, click **Select**.
- 2 Move the cursor outside the edge of the objects you want to group. This becomes the outer corner of a marquee bounding box.
- 3 Click and hold the left mouse button.



- 4 Drag the cursor outside the opposite edge of the objects you want to group. This creates a temporary visible marquee bounding box around the objects.
- 5 Release the mouse button.

**Note:** To add other objects to the selection, or remove objects from the selection, hold the **Ctrl** key and click each object in turn. To quickly select all objects in the drawing, you can use **Select All** from the Edit menu. To deselect all objects, click anywhere other than on an object.

See Also [Manipulating Objects](#)  
[Moving objects](#)

## Moving objects

You can move an object if you want by clicking the object and dragging it to the new location.

If you move an object as soon as you select it, an outline of the object boundary displays as you move it on your page. If you hold the mouse stationary for 1 sec or more before you move the object, the object itself displays as you move it, enabling you to better see the result for a more accurate placement.

See Also [Manipulating Objects](#)  
[Resizing objects](#)

## Resizing objects

You can resize objects simply.

### To re-size an object:

- 1 Select the object, and then move the cursor over a sizing handle. The cursor changes to a two-sided arrow showing the directions that you can drag the handle to resize the object.
- 2 Click and drag the handle to a new location. The object's bounding box appears as you drag.
- 3 Release the mouse button.

Select a sizing handle at a corner of the object to change the height and width at the same time. If you hold the **CTRL** key while you move a corner sizing handle, the object maintains its aspect ratio (that is, a square remains a square).

### To select an object's node:

- 1 Select the object. Node selection is only applicable to a Line, Pipe, Polyline, or Polygon object.
- 2 Position and hold the mouse pointer over the node. The mouse pointer will change to a small cross shape.
- 3 Click the left mouse button. The color of the selected node will change to the inverse of the background (light on a dark background, dark on a light background).

If you have a node selected and then click another node within the same object, the first node will deselect. To select multiple nodes, hold down the **Ctrl** key and click each node you want to select. (With the Ctrl key held down, you can click a previously selected node to deselect it.) Clicking the same node again toggles the

selection of the node alternately on and off. To deselect all nodes, click anywhere other than on a node.

**To move a node of an object:**

- 1 Select the node(s).
- 2 Position and hold the mouse pointer over the node. The mouse pointer will change to a small cross shape.
- 3 Click and hold the left mouse button. The cursor changes to a positioning symbol.
- 4 Drag the selected node(s) to the desired position.
- 5 Release the mouse button.

Selecting and moving multiple nodes maintains the aspect ratio of the graphic object between the selected nodes.

**To add a node to an object:**

- 1 Select the object.
- 2 Position and hold the mouse pointer directly over the graphic object at the exact point where the new node will be added. The mouse pointer will change to a pointing hand shape.
- 3 Press **Insert**, or either of the available plus (+) keys.

Depending upon the keyboard you're using, the plus key could be either on the number pad section, or accessed on the main keyboard via the **Shift** key.

**To delete a node from an object**

- 1 Select the node(s).
- 2 Press **Delete** or a minus (–) key.

If no nodes are selected, pressing the **Delete** or minus keys deletes the object.

See Also [Manipulating Objects](#)  
[Deleting objects](#)

## Deleting objects

You can delete unwanted objects.

**To delete an object (or a group of objects):**

- 1 Select the object (or group of objects)
- 2 Choose **Edit | Delete** or press the **Delete** key (or a minus (–) key).

See Also [Manipulating Objects](#)  
[Locking/unlocking objects](#)

## Locking/unlocking objects

On complex drawings (with many objects), selecting a discrete group of objects without including all objects (in the selected area) can be difficult (e.g. when an

object is hidden by another object). To prevent accidentally selecting an object, you can 'lock' it in position. When an object is 'locked', it cannot be selected, deleted, moved, or edited. Objects are locked only when the **Edit** menu **Break Lock Mode** option is not selected.

**To lock an object:**

- 1 Select the object.
- 2 Choose **Edit | Lock Object**.

**To unlock an object:**

- 1 Choose **Edit | Break Lock Mode**.
- 2 Select the object.
- 3 Choose **Edit | Unlock Object**.

See Also [Manipulating Objects](#)  
[Grouping objects](#)

## Grouping objects

You can group objects to make them easier to manipulate.

**To group objects:**

- 1 Click **Select**.
- 2 Select the objects to group, and then click **Group** (or choose **Arrange | Group Objects**).

**To ungroup objects:**

- 1 Click **Select**.
- 2 Select the objects to group, and then click **Ungroup** (or choose **Arrange | Ungroup Objects**).

**To change the properties of a group:**

- 1 Click **Select**.
- 2 Double-click the group. The Properties dialog box appears.
- 3 Change the properties as required.
- 4 Alternatively, choose **Tools | Goto Object**, select the group, and then click **OK**.

See Also [Manipulating Objects](#)  
[Copying and pasting objects](#)

## Copying and pasting objects

You can copy and paste objects onto other graphics pages.

**To copy an object to the clipboard:**

- 1 Click **Select**.

- 2 Select the object (or group of objects).
- 3 Click **Copy** or choose **Edit | Copy**.

**To paste an object (or group of objects) from the clipboard:**

- Click **Paste** or choose **Edit | Paste**.

**To cut (remove) an object:**

- 1 Click **Select**.
- 2 Select the object (or group of objects).
- 3 Click **Cut** or choose **Edit | Cut**.

**To paste an object (or group of objects) from the clipboard:**

- Click **Paste**, or choose **Edit | Paste**.

You can use the clipboard to transfer objects between different graphics pages and from other graphics applications. By default, unavailable colors in a pasted bitmap are dithered. To disable this feature, select **Options** from the **Tools** menu in the Graphics Builder, and remove the tick from the **Dither bitmaps on paste** option.

**To cancel your last drawing operation(s):**

- Click **Undo**, or choose **Edit | Undo**.

You can undo all operations performed during the current drawing session apart from edits to bitmaps.

**To cancel the Undo (or Redo) the last drawing operation(s):**

- Choose **Edit | Redo**.

See Also

[Manipulating Objects](#)  
[Send to Back and Send Backwards](#)

## Send to Back and Send Backwards

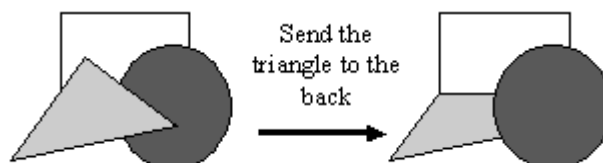
In the Graphics Builder, objects often overlap. New objects are placed in front of existing objects because CitectSCADA builds from the back to the front.

**To position an object (or group of objects) behind all other objects so that all objects overlap it:**

- 1 Click **Select**.
- 2 Select the object (or group of objects).



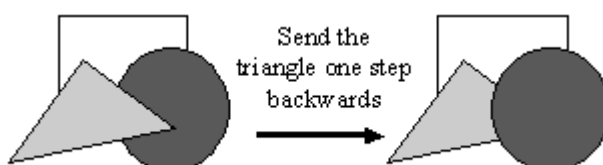
- 3 Click **Send to Back** or choose **Arrange | Send to Back**.



An object can be moved backwards and forwards one step at a time (rather than all the way to the back, or all the way to the front).

**To send an object (or group of objects) one step backwards:**

- 1 Click **Select**.
- 2 Select the object (or group of objects).
- 3 Click **Send Backwards** or choose **Arrange | Send Backwards**.



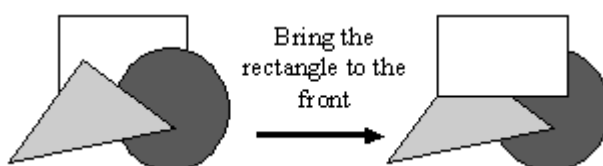
See Also [Manipulating Objects](#)  
[Bring to Front and Bring Forwards](#)

## Bring to Front and Bring Forwards

You can bring a selected object to the front.

**To bring an object to the front:**

- 1 Click **Select** tool.
- 2 Select the object (or group of objects).
- 3 Click **Bring to Front** or choose **Arrange | Bring to Front**.

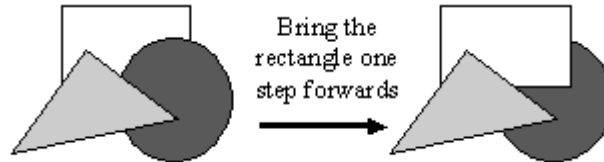


An object can be moved backwards and forwards one step at a time (rather than all the way to the back, or all the way to the front).

**To bring an object (or group of objects) one step forwards:**

- 1 Click **Select**.
- 2 Select the object (or group of objects)

- 3 Click **Bring Forwards** or choose **Arrange | Bring Forwards**.



See Also [Manipulating Objects](#)  
[Send to Back and Send Backwards](#)  
[Aligning objects](#)

## Aligning objects

You can precisely align a group of objects vertically, horizontally, or both.

If you select objects for the group individually, the first object you select maintains its position, and all other objects align with this object. If you select the objects using a marquee, the last object (in the selection) is the base object; all other objects align with this object. Because it is difficult to keep track of the order in which objects were created, it is usually easier to select objects individually.

**To align objects:**

- 1 Click **Select**.
- 2 Select the objects.
- 3 Choose **Arrange | Align**. The Align dialog box appears.



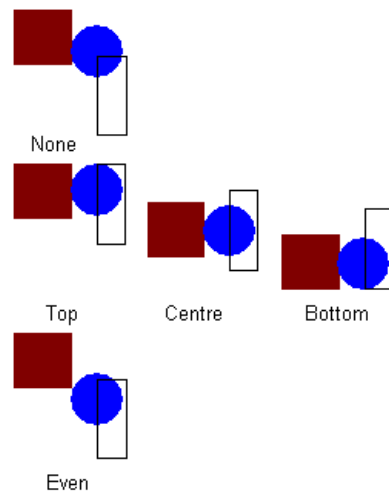
See Also [Align dialog box](#)  
[Manipulating Objects](#)

## Align dialog box

You use the Align dialog box to align a group of objects vertically, horizontally, or both.

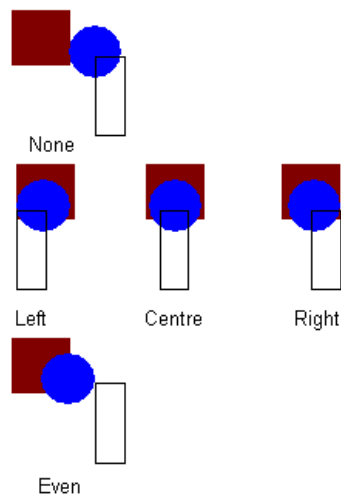
## Vertical

The alignment of the objects in the vertical direction:



## Horizontal

The alignment of the objects in the horizontal direction:



See Also [Manipulating Objects](#)  
[Rotating objects](#)

## Rotating objects

You can rotate an object 90° right (clockwise) or 90° left (counter-clockwise).

**To rotate an object (or group of objects):**

- 1 Click **Select**.

- 2 Select the object(s).
- 3 Choose **Arrange** | **Rotate**.

**To rotate a text object:**

- 1 Click **Select**.
- 2 Select the object(s).
- 3 Choose **Tools** | **Convert to Bitmap**.
- 4 Choose **Arrange** | **Rotate**.

See Also [Rotate dialog box](#)  
[Manipulating Objects](#)

## Rotate dialog box

The Rotate dialog box is used for [Rotating objects](#) (or groups of objects).

### Rotate

The direction to rotate the object (or group of objects). The object (or group of objects) are rotated 90 degrees in the direction you select. To rotate the objects (or group of objects) 180 degrees, click the direction button twice.

See Also [Manipulating Objects](#)

## Mirroring objects

You can mirror an object relative to its horizontal or vertical axis.

**To mirror an object (or group of objects) relative to its horizontal or vertical axis:**

- 1 Click **Select**.
- 2 Select the object(s)
- 3 Choose **Arrange** | **Mirror**.

See Also [Mirror dialog box](#)  
[Manipulating Objects](#)

## Mirror dialog box

This dialog box is used for [Mirroring objects](#) (or groups of objects) relative to a horizontal or vertical axis.

### Mirror

The axis about which to mirror the object (or group of objects).

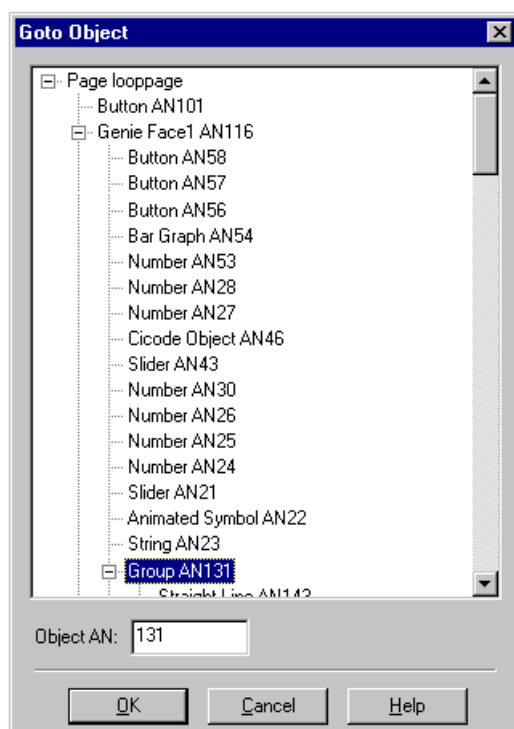
See Also [Manipulating Objects](#)

## Locate an object

You can locate a specific object on the [graphics page](#) you're currently working on.

To locate an object (or a group, Genie, symbol, or page template) on the current page:

- 1 Choose **Tools | Goto Object**.
- 2 Locate the object (or group, [Genie](#), symbol, or page template) in the tree structure or type the relevant AN in the **Object AN** box.
- 3 Click **OK**.



See Also [Goto Object dialog box](#)

## Goto Object dialog box

You use the Goto Object dialog box to precisely select and access the properties of objects, including page templates, and the objects, Genies, symbols, and groups on the page (or indeed, in the base template). The graphical elements that make up Genies, symbols, and groups are also made accessible.

The tree structure provides a simple way to locate graphical elements. Click an element to select it on the page, or double-click (or click **OK**) to access its properties. Page templates, symbols, Genies, and groups are made up of several objects (or other graphical elements) and have a plus sign (+) next to them; click the + sign to see these component objects.

**Object AN**

The AN to be located. When you enter an AN here, the corresponding object (group, [Genie](#), symbol, and so on.) is selected on the page and highlighted in the tree structure. You can then display its properties by clicking **OK**.

See Also [Manipulating Objects](#)

# Chapter 19: Understanding Object Types

---

CitectSCADA has many different object types, each with their own unique set of properties. For details of properties common to all object types, see [Defining Common Object Properties](#).

See Also

[Using Free Hand Line Objects](#)  
[Using Straight Line Objects](#)  
[Using Rectangle Objects](#)  
[Using Ellipse Objects](#)  
[Using Polygon Objects](#)  
[Using Pipe Objects](#)  
[Using Text Objects](#)  
[Using Number Objects](#)  
[Using Button Objects](#)  
[Using Symbol Set Objects](#)  
[Using Trend Objects](#)  
[Using Cicode Objects](#)  
[Using Pasted Symbol Objects](#)  
[Using Pasted Genie Objects](#)  
[Using ActiveX Objects](#)  
[Using the Process Analyst](#)  
[Using Database Exchange Control Objects](#)

## Using Free Hand Line Objects

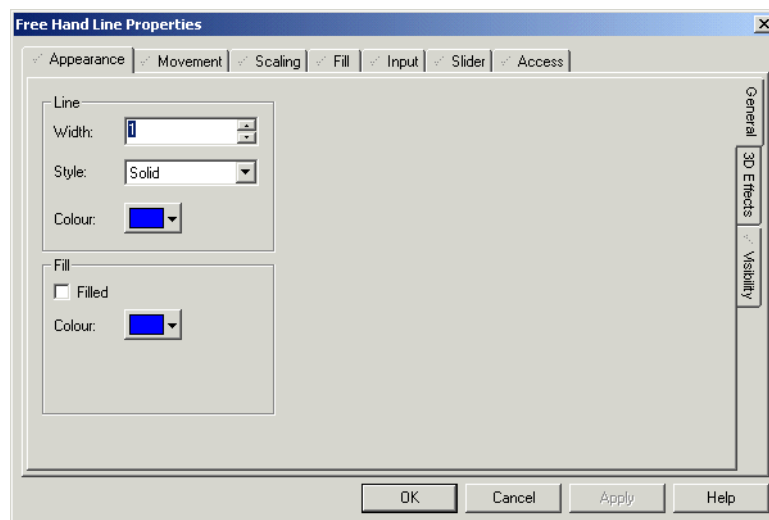
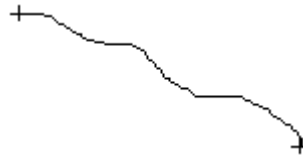
The Free Hand Line tool allows you to draw lines. Lines can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

**To draw a freehand line:**

- 1 Click the **Freehand** tool.



- 2 Move the cursor to where you want the line to start.
- 3 Click and drag the cursor to draw the line.



See Also [Freehand Line Properties - Appearance \(General\)](#)  
[Understanding Object Types](#)

### Freehand Line Properties - Appearance (General)

Free Hand Line drawings have the following general appearance properties.



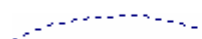


#### [Line] Width

The width of the line (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field.

If you make the line more than 1 pixel wide, it must be solid.

#### [Line] Style

The style of the line. You can choose one of the following line styles:

-  - Solid
-  - Dash
-  - Dot
-  - Dash Dot
-  - Dash Dot Dot



To change the style, choose a style from the menu to the right of this field.

#### [Line] Color

The color of the line.

#### [Fill] Filled

The Filled check box determines whether the object will be filled with a color. If you check this box, an invisible line is drawn from one end of your line to the other. Everything between the invisible line and your line will be filled.



- unfilled



- filled

#### [Fill] Color

The color with which the object will be filled. The color that you select as your fill color here is static.

To specify a fill color that changes with runtime conditions, click the **Fill** tab.

If you have enabled the Fill (Color) properties, note that the color you select here will override the OFF color for Fill Color (On/Off), the ABC color for Fill Color (Multi-state), Array Color 0 for Fill Color (Array), and the At minimum color for Fill Color (Gradient).

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

## Using Straight Line Objects

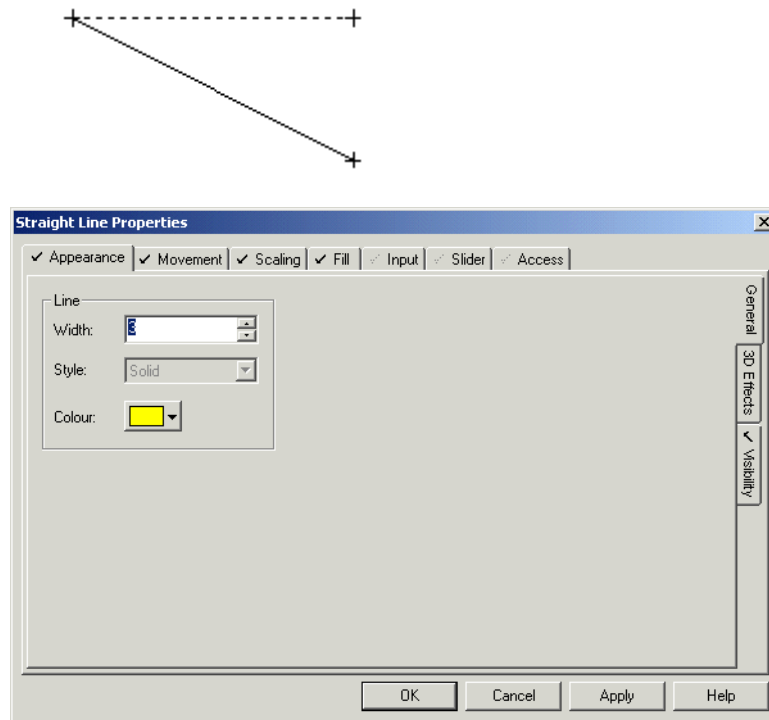
The Straight Line tool allows you to draw straight lines. Straight lines can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like any other type of object.

**To draw a straight line:**

- 1 Click the **Straight Line** tool.



- 2 Move the cursor to where you want to start the line.
- 3 Click and drag to draw the line. (If you hold the **Ctrl** key while drawing the line it is constrained to the vertical or horizontal.



See Also [Straight Line Properties - Appearance \(General\)](#)  
[Understanding Object Types](#)

## Straight Line Properties - Appearance (General)

Straight Lines have the following general appearance properties.

### [Line] Width

The width of the line (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field.

If you make the line more than 1 pixel wide, it must be solid.

### [Line] Style

The style of the line. You can choose from the following line styles:



To change the style, choose a style from the menu to the right of this field.

### [Line] Color

The color of the line.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

## Using Rectangle Objects

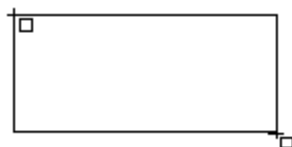
Use the Rectangle tool to draw rectangles and squares. Rectangles and squares can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

### To draw a rectangle:

- 1 Click the **Rectangle** tool.



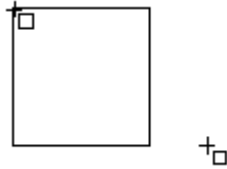
- 2 Move the cursor to where you want the rectangle to start.
- 3 Click and drag the mouse to the opposite corner of the rectangle and release the mouse button. If you hold the **Shift** key before you start drawing the rectangle, it is drawn from its center outwards.



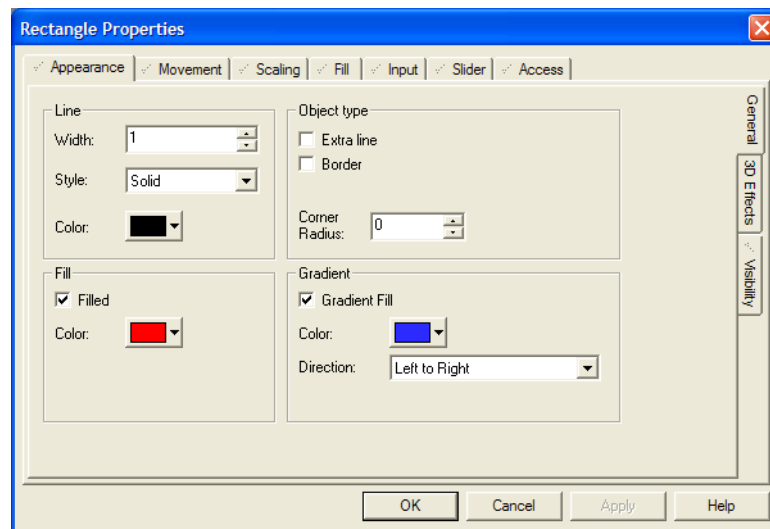
### To draw a square:

- 1 Click the **Rectangle** tool.
- 2 Click (and hold) the **Ctrl** key.

- 3 Move the cursor to where you want the square to start and click (and hold) the mouse button.



- 4 Drag the cursor to the opposite corner of the square and release the mouse button. If you hold the **Shift** key (and the **Ctrl** key) before you start drawing the square, it is drawn from its center outwards.



See Also [Rectangle Properties - Appearance \(General\)](#)  
[Understanding Object Types](#)

## Rectangle Properties - Appearance (General)

Rectangles have the following general appearance properties.

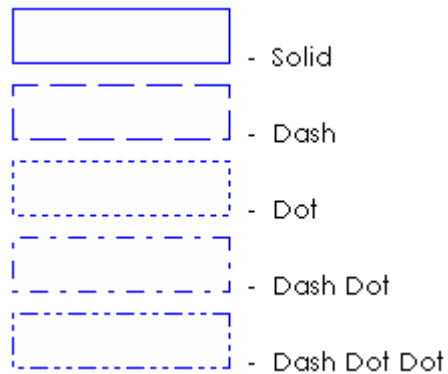
### [Line] Width

The width of the outline for the rectangle (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field.

If you make the line more than 1 pixel wide, it must be solid.

**[Line] Style**

The outline style of the rectangle. You can choose from the following line styles:



To change the style, choose a style from the menu to the right of this field.

**[Line] Color**

The outline color of the rectangle.

**[Fill] Filled**

The Filled check box determines whether the rectangle will be filled with a color.

**[Fill] Color**

The color with which the rectangle will be filled. The color that you select as your fill color here is static.

To specify a fill color that changes with runtime conditions, click the Fill tab. If you have enabled the Fill (Color) properties, note that the color you select here overrides the OFF color for Fill Color (On/Off), the ABC color for Fill Color (Multi-state), Array Color 0 for Fill Color (Array), and the At minimum color for Fill Color (Gradient).[Object type] Extra line

Adds an extra line (1 pixel width) of lowlight color to the rectangle, if the rectangle is defined as Raised or Lowered (click the 3D Effects tab).









**[Gradient] Color**

Controls the color of the gradient fill between the fill color and the gradient color. This option is available only when the **Filled** and **Gradient Fill** check boxes are selected. The gradient is updated at runtime to reflect the gradient between the two colors selected. Gradient fills support flashing colors.

Gradients do not rotate with an object; for example, if an object contains a left-to-right gradient fill and is rotated 90 degrees (either at runtime or in Graphics Builder), the gradient is still left to right.

### [Gradient] Direction

The direction to be used for the gradient color. Use the table below as a guide to choose the gradient color direction you want.

Example	Gradient Color Direction
	Left to Right
	Right to Left
	Top to Bottom
	Bottom to Top
	Horizontal Gradient to Middle
	Horizontal Gradient from Middle
	Vertical Gradient to Middle
	Vertical Gradient from Middle

### [Object type] Border

Adds an extra line (1 pixel width) of black to the perimeter of the rectangle.

### [Object type] Corner Radius

Controls the radius of the corners of the rectangle. Enter a value between 0 and 32. The higher the value, the more rounded the corners of the rectangle.

When the radius is greater than 0, the **Extra line** and **Border** options are not available.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

## Using Ellipse Objects

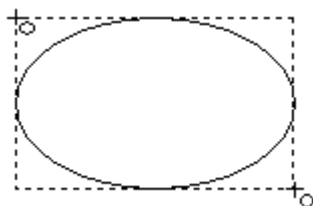
You use the Ellipse tool to draw ellipses, circles, arcs, and pie-slices. Ellipse objects can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

### To draw an ellipse:

- 1 Click the **Ellipse** tool.

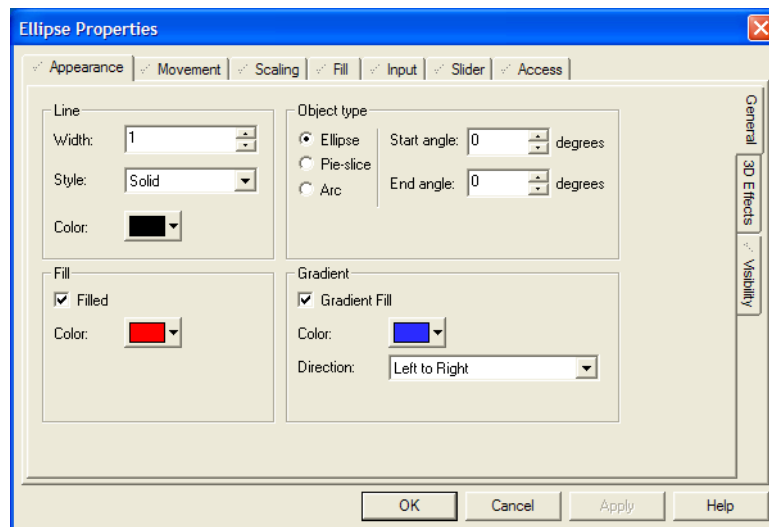
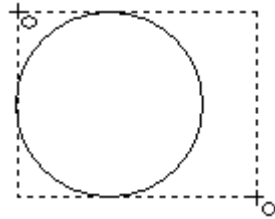


- 2 Move the cursor to a corner of the bounding rectangle (marquee) and click (and hold) the mouse button.
- 3 Drag the cursor to the opposite corner of the bounding rectangle and release the mouse button. If you hold the **Shift** key before you start drawing the ellipse, it is drawn from its center outwards.



**To draw a circle:**

- 1 Click the **Ellipse** tool.
- 2 Click (and hold) the **Ctrl** key.
- 3 Move the cursor to a corner of the bounding rectangle (marquee) and click (and hold) the mouse button.
- 4 Drag the cursor to the opposite corner of the bounding rectangle and release the mouse button. If you hold the **Shift** key and the **Ctrl** key before you start drawing the circle, it is drawn from its center outwards.



See Also [Ellipse Properties - Appearance \(General\)](#)  
[Understanding Object Types](#)

## Ellipse Properties - Appearance (General)

Ellipses have the following general appearance properties:

### [Line] Width

The width of the outline of the ellipse (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field. If you make the line more than 1 pixel wide, the line style will be solid.



**[Line] Style**

The outline style of the ellipse. You can choose one of the following line styles:



To change the style, choose a style from the menu to the right of this box.

**[Line] Color**

The outline color of the ellipse.

**[Fill] Filled**

The Filled check box determines whether the ellipse will be filled with a color.

**[Fill] Color**

The color with which the ellipse will be filled. The color that you select as your fill color here is static.

To specify a fill color that changes with runtime conditions, click the Fill tab. If you have enabled the Fill (Color) properties, note that the color you select here will override the OFF color for Fill Color (On/Off), the ABC color for Fill Color (Multi-state), Array Color 0 for Fill Color (Array), and the At minimum color for Fill Color (Gradient).









**[Gradient] Color**

Controls the color of the gradient fill between the fill color and the gradient color. This option is available only when the **Filled** and **Gradient Fill** check boxes are selected. The gradient is updated at runtime to reflect the gradient between the two colors selected. Gradient fills support flashing colors.

Gradients do not rotate with an object; for example, if an object contains a left-to-right gradient fill and is rotated 90 degrees (either at runtime or in Graphics Builder), the gradient is still left to right.

**[Gradient] Direction**

The direction to be used for the gradient color. Use the table below as a guide to choose the gradient color direction you want.

Example	Gradient Color Direction
	Left to Right
	Right to Left
	Top to Bottom
	Bottom to Top
	Horizontal Gradient to Middle
	Horizontal Gradient from Middle
	Vertical Gradient to Middle
	Vertical Gradient from Middle

**[Object type] Ellipse**

Select this radio button if you want the object to be a full ellipse.



For a full ellipse, you do not need to specify Start and End angles.

**[Object type] Pie-slice**

Select this radio button if you want to remove a section from your ellipse (i.e., you want it to resemble a pie-slice).

If you select this option, you can specify a Start angle, and an End angle:

**Start angle**

The angle (measured clockwise from 0°) of the section to be removed from the ellipse. For example, if you enter a start angle of 50°, your pie-slice would look something like this:



**End angle**

The angle (measuring clockwise from 0°) of the section of the ellipse which is to remain. For example, if you enter an end angle of 150°, your pie-slice would look something like this:



Start and End angles can be combined for various effects. For example, a Start angle of 270°, and an End angle of 150° would produce the following pie-slice:

**[Object type] Arc**

Select this radio button if you want to draw an arc.

If you select this option, you can specify a Start angle, and an End angle:

**Start angle**

The angle (measured clockwise from 0°) defining the segment to be removed from the ellipse, leaving an arc. For example, if you enter a start angle of 50°, your arc would look something like this:

**End angle**

The angle (measuring clockwise from 0°) defining the segment of the ellipse which is to remain. For example, if you enter an end angle of 150°, your pie-slice would look something like this:



Start and End angles can be combined for various effects. For example, a Start angle of 270°, and an End angle of 150° would produce the following arc:



For help with the other properties, see [Defining Common Object Properties](#).

## Using Polygon Objects

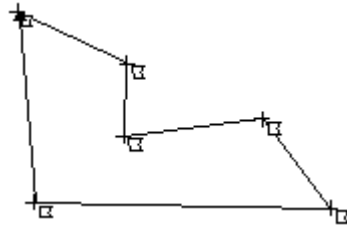
Use the Polygon tool to draw polygons and polylines. Polygons can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

### To draw a polygon:

- 1 Click the **Polygon** tool.



- 2 Move the cursor to where you want the polygon to start and click and hold the mouse button.
- 3 At the end of the first line segment, release the mouse button.
- 4 Move the cursor to each point on the polygon in turn, and click the mouse button (clicking and dragging is not required after the first segment).

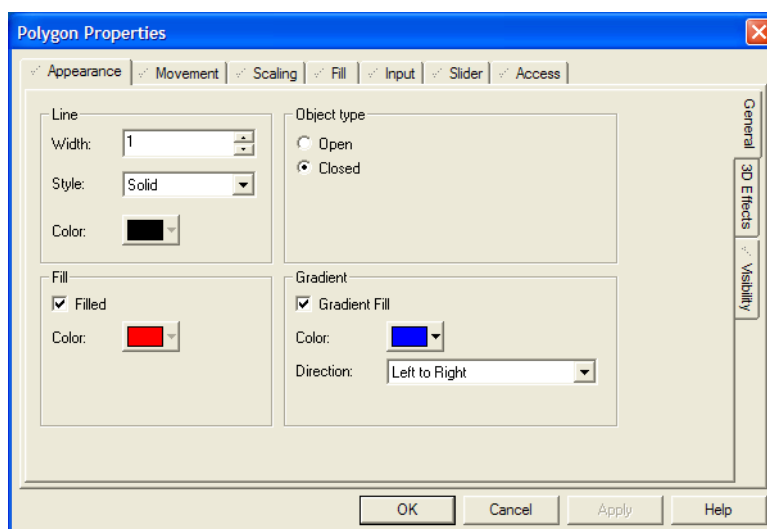
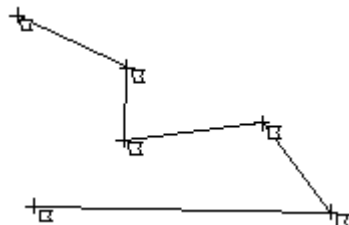


**Note:** To complete the polygon, double-click the mouse button. (To draw horizontally or vertically only, hold the Ctrl key down when you are drawing the polygon.)

### To draw a polyline:

- 1 Click the **Polygon** tool.
- 2 Move the cursor to where you want the polyline to start and click and hold the mouse button.
- 3 At the end of the first line segment, release the mouse button.
- 4 Move the cursor to each point on the polyline in turn and click the mouse button (clicking and dragging is not required after the first segment).
- 5 To complete the polyline, double-click the mouse button. Initially, the object will actually be a polygon. To change it to a polyline, double-click it, and define it as Object type - Open.

**Note:** To draw horizontally or vertically only, hold the **Ctrl** key down when you are drawing the polyline.



See Also [Polygon Properties - Appearance \(General\)](#)  
[Understanding Object Types](#)

## Polygon Properties - Appearance (General)

Polygons have the following general appearance properties.

### [Line] Width

The width of the outline of the polygon (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field.

If you make the line more than 1 pixel wide, it must be solid.

**[Line] Style**

The outline style of the polygon. You can choose any one of the following line styles:



To change the style, choose the style you want from the menu to the right of this field.

**[Line] Color**

The outline color of the polygon.

**[Fill] Filled**

The Filled check box, determines whether the polygon will be filled with a color.

**[Fill] Color**

The color with which the polygon will be filled. The color that you select as your fill color here is static.

To specify a fill color that changes with runtime conditions, click the Fill tab. If you have enabled the Fill (Color) properties, note that the color you select here will override the OFF color for Fill Color (On/Off), the ABC color for Fill Color (Multi-state), Array Color 0 for Fill Color (Array), and the At minimum color for Fill Color (Gradient).









**[Gradient] Color**

Controls the color of the gradient fill between the fill color and the gradient color. This option is available only when the **Filled** and **Gradient Fill** check boxes are selected. The gradient is updated at runtime to reflect the gradient between the two colors selected. Gradient fills support flashing colors.

Gradients do not rotate with an object; for example, if an object contains a left-to-right gradient fill and is rotated 90 degrees (either at runtime or in Graphics Builder), the gradient is still left to right.

### [Gradient] Direction

The direction to be used for the gradient color. Use the table below as a guide to choose the gradient color direction you want.

Example	Gradient Color Direction
	Left to Right
	Right to Left
	Top to Bottom
	Bottom to Top
	Horizontal Gradient to Middle
	Horizontal Gradient from Middle
	Vertical Gradient to Middle
	Vertical Gradient from Middle

### [Object type] Open

Defines the object as a polyline (the first point and the last point are *not* joined).



### [Object type] Closed

Defines the object as a polygon (the first point and the last point *are* joined).



For help on the remaining properties tabs, see [Defining Common Object Properties](#).

## Using Pipe Objects

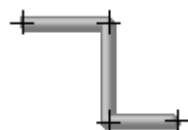
Use the Pipe tool to draw pipes with automatic three-dimensional shading. Pipes can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

**To draw a pipe:**

- 1 Click the **Pipe** tool.



- 2 Move the cursor to where you want the pipe to start, and click and hold the mouse button.
- 3 At the end of the first line segment, release the mouse button.
- 4 Move the cursor to each point on the path in turn and click the mouse button (clicking and dragging is not required after the first segment).



- 5 To complete the pipe, double-click the mouse button.

**Hint:** To draw horizontally or vertically only, hold the **Ctrl** key down when drawing the pipe.



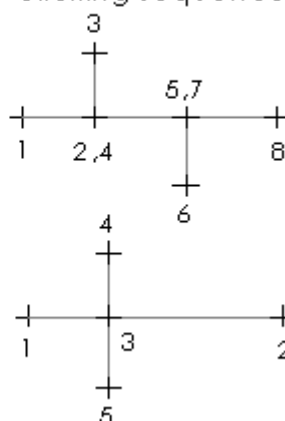
### Drawing Complex Pipe Arrangements

Use the Pipe tool to draw complex pipe arrangements (including "T" pieces and junctions). The illustration below shows some pipes, and the sequence of mouse clicks needed to draw each of them:

To draw this pipe ...



... follow this mouse clicking sequence



**Hint:** Use the grid to ensure accurate positioning for each click.

See Also

[Pipe Properties - Appearance \(General\)](#)  
[Understanding Object Types](#)

### Pipe Properties - Appearance (General)

Pipes have the following general appearance properties.

#### [Line] Width

The width of the pipe (in pixels). You can change the width by clicking the up and down arrows to the right of the field, or by entering another value in this field. All pipes must be at least 1 pixel wide.

#### [Line] Highlight Color

The color of the pipe where it is "in the light"; that is, the brightest color on the pipe.

#### [Line] Lowlight Color

The color of the pipe where it is "in shadow"; that is, the dullest color on the pipe.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

## Using Text Objects

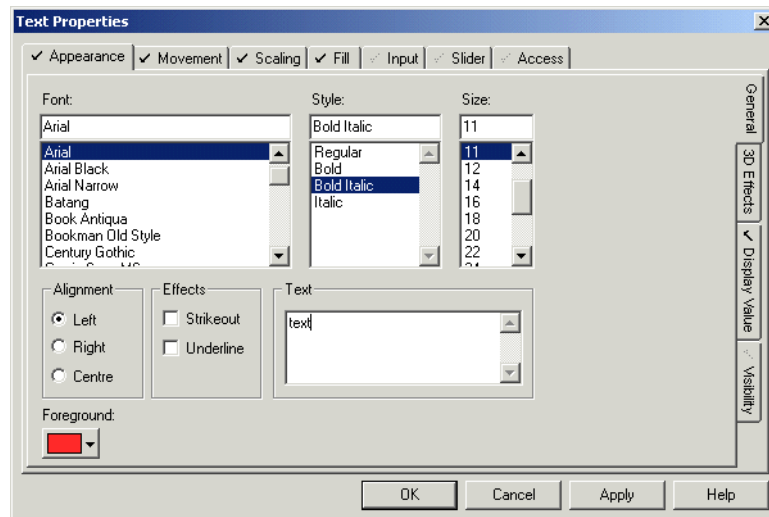
Use the Text tool to type text on the page. Text can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

To add text:

- 1 Click the **Text** tool.



- 2 Type your text on the keyboard. (Press **Enter** to start a new line.)
- 3 Move the cursor to where you want to position the text and click the left mouse button.



See Also [Text Properties - Appearance \(General\)](#)  
[Understanding Object Types](#)

### Text Properties - Appearance (General)

Text has the following general appearance properties

#### Font

The font used to display the text. Use the scroll bar to the right to view available fonts, or type all or part of a font name directly into this field.

#### Style

Select whether you would like the text to be Regular, **Bold**, **Bold Italic**, or *Italic*.

#### Size

---

Define the size of the text (point size). Available sizes might vary according to the selected printer and the selected font.

**[Alignment] Left**

Select this radio button to align the text to the left of the text box

**[Alignment] Right**

Select this radio button to align the text to the right of the text box:

**[Alignment] Center**

Select this radio button to align the text in the center of the text box

**[Effect] Strikeout**

Select this box to make the text will appear with a line through it.

**[Effect] Underline**

Select this box to underline the text.

**Text**

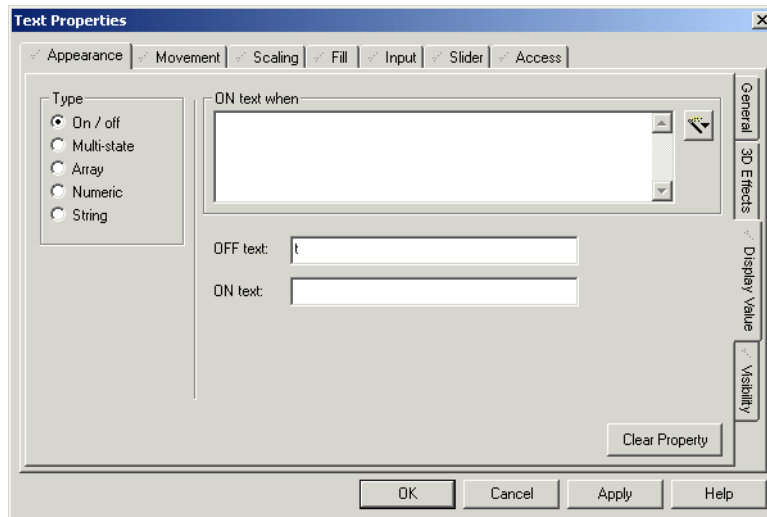
This field contains the text that will display on the page. You can enter any keyboard character(s). It is useful to edit text at this field, as you can apply text changes at the same time as you apply other font and color changes.

This text changes automatically depending on the Display Value properties that you define.

**Foreground**

The color of the text.

**Note:** There are several radio buttons in Display Value (On/Off, Multi-state and so on). When selected, these radio buttons change the appearance of the right hand side of the dialog. These radio buttons are only documented once below.



See Also

[Text Properties - Appearance Display Value \(On/Off\)](#)  
[Text Properties - Appearance Display Value \(Multi-state\)](#)  
[Text Properties - Appearance Display Value \(Array\)](#)  
[Text Properties - Appearance Display Value \(Numeric\)](#)  
[Text Properties - Appearance Display Value \(String\)](#)

### Text Properties - Appearance Display Value (On/Off)

Text has the following On/Off Display Value properties:

#### [Type] On / Off

Changes the text which displays when a particular condition is met, and another when it is not. For example, you could display an alarm message when a particular variable tag is in alarm, and a normal message when it is not.

#### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different text for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different message for each ON/OFF combination. In other words, you could display a different message for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

#### [Type] Array

Allows you to enter an [expression](#) which returns an integer. For each integer (from 0–255), you can display different text. For example, you could display a different message for each state of an analog tag.

#### [Type] Numeric

Displays the value of a tag or expression in numeric format (you can specify the format).

#### [Type] String

Displays the value of an expression as a string.

#### ON text when

The text entered in the **ON text** field (below) appears when the condition entered here is true. The text can be a maximum of 128 characters long.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options; **Insert Tag**, and **Insert Function**.

#### OFF text

The text that will display whenever the condition entered above is false. You can use any keyboard character(s) to a maximum length of 256 characters.

For example, you could display the message **Conveyor 110 Normal** when **CV110\_FAULT.On** is false (i.e., there is no alarm at conveyor 110).

#### ON text

The text that will display whenever the condition entered above is true. You can enter any keyboard character(s) to a maximum length of 256 characters.

For example, you could display the message **Conveyor 110 Alarm** when **CV110\_FAULT.On** is true (i.e. there is no alarm at conveyor 110).

Click **Clear Property** to clear property details and disable the property.

See Also

[Text Properties - Appearance Display Value \(Multi-state\)](#)

[Text Properties - Appearance Display Value \(Array\)](#)

[Text Properties - Appearance Display Value \(Numeric\)](#)

[Text Properties - Appearance Display Value \(String\)](#)

### Text Properties - Appearance Display Value (Multi-state)

Text has the following multi-state display value properties:

#### [Type] On / Off

Changes which text displays when a particular condition is met. For example, you could display an alarm message when a particular variable tag is in alarm, and a normal message when it is not.

**[Type] Multi-state**

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different text for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different message for each ON/OFF combination. In other words, you could display a different message for each of the following ON/OFF combinations **ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC**.

**[Type] Array**

The Array option allows you to enter an expression which returns an integer. For each integer (from 0–255), you can display different text. For example, if you had an analog tag, you could display a different message for each different state.

**[Type] Numeric**

Displays the value of an expression in numeric format (you can specify the format).

**[Type] String**

Displays the value of an expression as a string.

**Conditions**

The conditions you enter here will occur together in different ways, at different times. You can use each different combination to determine the text that will display.

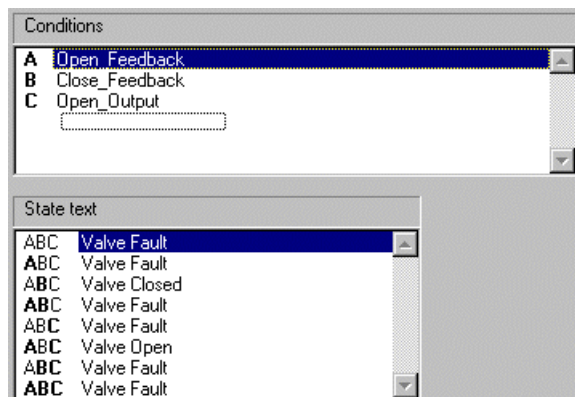
The default number of conditions is 3, but you can add more (to a maximum of 5 conditions, providing 32 combinations), using the **Add** button. You can also delete conditions using the **Delete** button, but there must always be a condition in this field. To enter a condition, click the relevant line (A, B, C, etc.), and click **Edit**. To insert a tag or function, click the **Wizard** button. This button displays two options: Insert Tag and Insert Function.

**State text**

The text that is to display for each combination of the above conditions. You can enter any keyboard character(s).

For example:

To display different messages about the status of a valve, you could fill out the **Conditions** and **State text** fields as follows:



In this example, **Open\_Feedback** and **Close\_Feedback** are variable tags representing digital inputs on the valve; **Open\_Output** is a variable tag representing an output on the valve. So, ABC means **Open\_Feedback** is on, and **Close\_Feedback** and **Open\_Output** are both off. For this combination, the text Valve Fault will display, because the valve is open when it should be closed. The same type of logic applies to the rest of the states.

Click **Clear Property** to clear property details and disable the property.

See Also

[Text Properties - Appearance Display Value \(On/Off\)](#)  
[Text Properties - Appearance Display Value \(Array\)](#)  
[Text Properties - Appearance Display Value \(Numeric\)](#)  
[Text Properties - Appearance Display Value \(String\)](#)

## Text Properties - Appearance Display Value (Array)

Text has the following Array Display Value properties:

### [Type] On / Off

Changes which text displays when a particular condition is met. For example, you could display an alarm message when a particular variable tag is in alarm, and a normal message when it is not.

### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different text for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different message for each ON/OFF combination. In other words, you could display a different message for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

**[Type] Array**

The Array option allows you to enter an expression which returns an integer. For each integer (from 0–255), you can display different text. For example, if you had an analog tag, you could display a different message for each different state.

**[Type] Numeric**

Displays the value of an expression in numeric format (you can specify the format).

**[Type] String**

Displays the value of an expression as a string.

**Array expression**

Enter the expression which is to return one or more integers. For each returned integer, a different piece of text is displayed.

If the return value is:

- Less than 0 (zero), it will be set to 0 (zero), and a runtime hardware alarm will be triggered.
- Greater than 255, it will be set to 255, and a runtime hardware alarm will be triggered.
- A real (non-integer) number, it will be rounded off to the nearest integer.

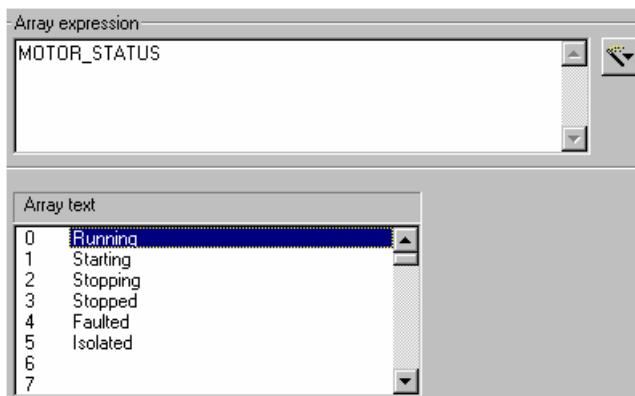
To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

**Array text**

The text that is to display for each integer returned by the Array expression entered above. You can enter any keyboard character(s).



For example, to display different messages about the status of a motor, you could fill out the **Array expression** and **Array text** fields as follows:



In this example, MOTOR\_STATUS is an analog variable tags representing the status of a motor. When the motor changes state, an integer is returned (0 = Running, 1 = Starting etc.) and the appropriate text displays.

Click **Clear Property** to clear property details and disable the property.

See Also

[Text Properties - Appearance Display Value \(On/Off\)](#)  
[Text Properties - Appearance Display Value \(Multi-state\)](#)  
[Text Properties - Appearance Display Value \(Numeric\)](#)  
[Text Properties - Appearance Display Value \(String\)](#)

## Text Properties - Appearance Display Value (Numeric)

Text has the following Numeric Display Value properties.

### [Type] On / Off

Changes the text which displays when a particular condition is met, and another when it is not. For example, you could display an alarm message when a particular variable tag is in alarm, and a normal message when it is not.

### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different text for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different message for each ON/OFF combination.

### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each integer (from 0–255), you can display different text. For example, if you had an analog tag, you could display a different message for each different state.

**[Type] Numeric**

Displays the value of an expression in numeric format (you can specify the format).

**[Type] String**

Displays the value of an expression as a string.

**Numeric expression**

The value of the expression entered here will be displayed on the [graphics page](#). It will be formatted according to the format selected below.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert tag and Insert Function.

**Format**

The value returned for the expression entered above will be displayed according to the format you enter here. For example, the analog variable tag **MOTOR\_STATUS** returns integers 0-5. If you enter this tag as the Numeric expression above, and enter **###** as your format, the display will alternate between **0.00, 1.00, 2.00, 3.00, 4.00, and 5.00**. You can select a format from the drop-down list, or type in your own. If the numeric expression is a single variable, its format is overwritten by the format you enter here.

Click **Clear Property** to clear property details and disable the property.

See Also

[Text Properties - Appearance Display Value \(On/Off\)](#)  
[Text Properties - Appearance Display Value \(Multi-state\)](#)  
[Text Properties - Appearance Display Value \(Array\)](#)  
[Text Properties - Appearance Display Value \(String\)](#)

## Text Properties - Appearance Display Value (String)

Text has the following String Display Value properties.

**[Type] On / Off**

Select this radio button to change the text which displays when a particular condition is met, and another when it is not. For example, you could display an alarm message when a particular variable tag is in alarm, and a normal message when it is not.

**[Type] Multi-state**

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different text for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different message for each ON/OFF combination. In other words, you could display a different message for each of the following ON/OFF combinations **ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC**.

**[Type] Array**

The Array option allows you to enter an expression which returns an integer. For each integer (from 0–255), you can display different text. For example, if you had an analog tag, you could display a different message for each different state.

**[Type] Numeric**

Select this radio button to display the value of an expression in numeric format (you can specify the format).

**[Type] String**

Select this radio button to display the value of an expression as a string.

**String expression**

The value of the expression entered here will be displayed as a string on the graphics page.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert tag and Insert Function.

Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click the relevant tabs.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

See Also

[Text Properties - Appearance Display Value \(On/Off\)](#)  
[Text Properties - Appearance Display Value \(Multi-state\)](#)  
[Text Properties - Appearance Display Value \(Array\)](#)  
[Text Properties - Appearance Display Value \(Numeric\)](#)

## Using Number Objects

Use the Number tool to represent a tag or expression as a number. When you place a number on your page, all you have to do is enter the relevant variable tag or expression. Numbers can be moved, resized, brought to the front, and so on, and its properties edited just like other types of object.

(The same functionality is also available through the Text tool.)

**To add a number to your graphics page:**

- 1 Click the **Number** tool.



- 2 Move the cursor to where you want the number to display and click the mouse button. The Text Properties dialog box appears where you enter the relevant variable tag or expression.

For help on the various Properties tabs, see [Text Properties - Appearance \(General\)](#) and [Defining Common Object Properties](#).

## Using Button Objects

Use the Button tool to draw buttons on [graphics page](#). You can then assign security rights and attach commands to it.

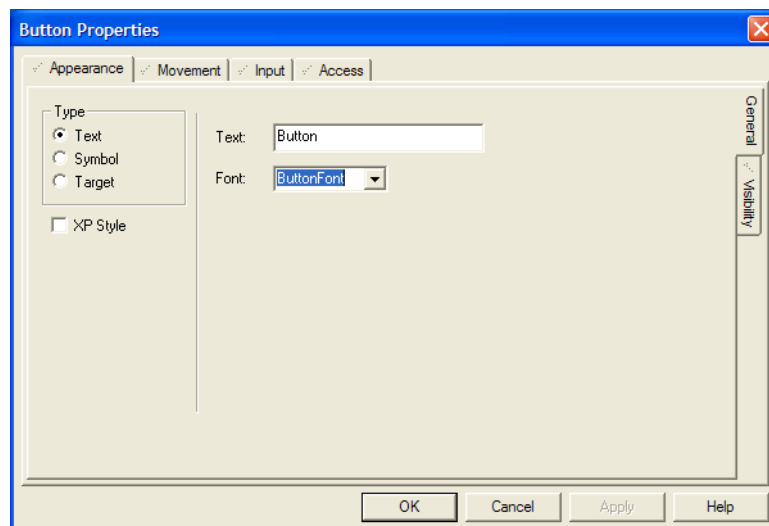
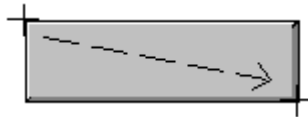
Buttons can be moved, resized, reshaped, brought to the front, and so on, and its properties edited just like other types of object.

**To draw a button:**

- 1 Click the **Button** tool.



- 2 Move the mouse to where you want the button to start and press (and hold) the mouse button.
- 3 Drag the mouse to where you want the button to finish and release the mouse button.



See Also [Button Properties - Appearance \(General\)](#)  
[Understanding Object Types](#)

## Button Properties - Appearance (General)

Buttons have the following General Appearance properties.

### [Type] Text

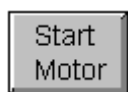
Select this option to display text on the button. If you select this option, the **Text** and **Font** fields will display to the right of the dialog.

If either the **Text** or **Symbol** type option is selected, you can use XP style buttons. To configure a button to use the Windows XP style, select the **XP Style** check box. During runtime an XP style button has a blue border when it has keyboard focus, and an orange border when the mouse is on the button. When you place a button on a page, the **XP Style** check box is selected by default.

### Text

The text to display on the button. You can use any keyboard character(s) to specify a name for the button; however, the following characters have special meaning:

**^n** - Wraps the text onto the next line. For example, **Start^nMotor** would display as:



### Font

Select the font to be used for displaying the button text.

### [Type] Symbol

Select this option to display a symbol on the button. If you select this option, the **Set** button will display to the right of the dialog.

Click **Set** to choose the symbol which is to display on the button. A picture of the selected symbol will also display.

### [Type] Target

When this option is selected, the button will not have any text or symbols on it, and it will have a transparent face.

### Mode

There are three different modes of transparent buttons:

- **BORDER\_3D**: The button is drawn with only the 3-D border (transparent face).
- **BORDER**: The button is drawn with only a thin line border.

- **TARGET:** The button is totally transparent. This constitutes a screen target. For help on the remaining properties tabs, see [Defining Common Object Properties](#).

## Using Symbol Set Objects

The Symbol Set tool allows you to represent changing runtime conditions with changing symbols. By clicking on this tool, then clicking on the [graphics page](#), you can define the symbols which are to display for each condition.

After a symbol set has been added to the page, it can be moved, re-sized, re-shaped, brought to the front etc., and its properties can be edited, just like any other type of object.

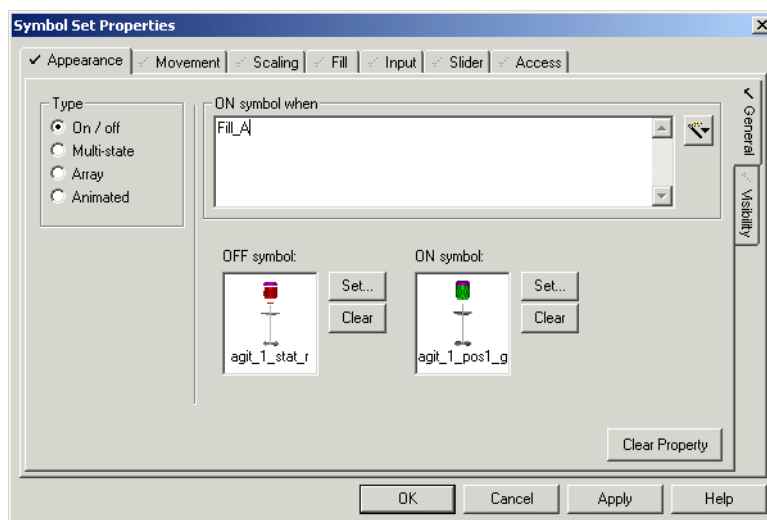
### To add a Symbol Set:

- 1 Click the **Symbol** tool.



- 2 Move the mouse pointer to the desired position on the page, and click with the left mouse button.
- 3 Fill out the relevant properties for the symbol set, and click **OK**.

When selected, the radio buttons on the dialog box change the appearance of the right hand side of the dialog. These radio buttons are only documented once below.



See Also [Symbol Set Properties - Appearance General \(On/Off\)](#)  
[Understanding Object Types](#)

## Symbol Set Properties - Appearance General (On/Off)

Symbol Sets have the following general appearance (On/Off) properties:

### [Type] On / Off

Select this radio button to display one symbol when a particular expression is TRUE, and another when it is FALSE. For example, you could display a red symbol when a particular variable tag is in alarm, and a green symbol when it is not.

### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different symbols for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different symbol for each ON/OFF combination. In other words, you could display a different symbol for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, **ABC**, ABC, ABC, **ABC**.

### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can display a unique symbol. For example, you could display a different symbol for each threshold of an analog alarm.

### [Type] Animated

Select this radio button to display an actual animation (several different symbols in sequence).

### ON symbol when (128 Chars.)

The symbol entered in the **ON symbol** field (below) will display whenever the condition entered here is TRUE. The symbol entered in the **OFF symbol** field (below) will display whenever the condition entered here is FALSE.

To insert a tag or a function, click the Wizard button to the right of this field. This button displays two options; **Insert tag**, and **Insert Function**.

### OFF symbol

The symbol that will display whenever the condition entered above is false. Click **Set** to select a symbol, or **Clear** to clear the current selection.

For example, you could display the OFF symbol when **MIX\_RUNNING** is false.

**stopped**

ON symbol

The symbol that will display whenever the condition entered above is true. Click **Set** to select a symbol, or **Clear** to clear the current selection.

For example, you could display the ON symbol when **MIX\_RUNNING** is true.

**running**

Click **Apply** or **OK** to bring your changes into effect, or **Cancel** to discard them and exit. Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click the relevant tabs.

See Also [Symbol Set Properties - Appearance General \(Multi-state\)](#)  
[Understanding Object Types](#)

### Symbol Set Properties - Appearance General (Multi-state)

Symbol Sets have the following general appearance (Multi-state) properties:

#### [Type] On / Off

Select this radio button to display one symbol when a particular expression is TRUE, and another when it is FALSE. For example, you could display a red symbol when a particular variable tag is in alarm, and a green symbol when it is not.

#### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different symbols for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different symbol for each ON/OFF combination. In other words, you could display a different symbol for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

#### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can display a unique symbol. For example, you could display a different symbol for each threshold of an analog alarm.

#### [Type] Animated

Select this radio button to display an actual animation.

#### Conditions

The conditions you enter here will occur together in different ways, at different times. You can use each different combination to determine which symbol will display.



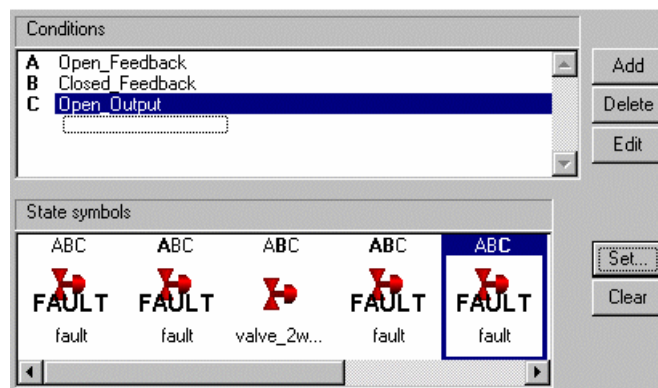
To enter a condition, click the relevant line (A, B, C, etc.), and click **Edit**. You can add more conditions (to a maximum of 5, providing 32 combinations), using the **Add** button. To insert a tag or function, click the **Wizard** button. This button displays two options; **Insert Tag** and **Insert Function**. You can also delete conditions using the **Delete** button, but there must always be a condition in this field. Conditions which are left black (instead of deleted) will be evaluated as permanently false at runtime.

### State symbols

The symbols that will display for each combination of the above conditions. Click the **Set** button to select a symbol, or **Clear** to clear the current selection.

For example:

To display different symbols each time the status of a valve changes, you could fill out the **Conditions** and **State symbols** fields as follows:



In this example, **Open\_Feedback**, and **Close\_Feedback** are variable tags representing digital inputs on the valve, and **Open\_Output** is a variable tag representing an output on the valve. So, **ABC** means **Open\_Feedback** is ON, and **Close\_Feedback** and **Open\_Output** are both OFF. For this combination, the fault symbol will display, because the valve is open when it should be closed. The same type of logic applies to the rest of the states.

Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click the relevant tabs.

See Also [Symbol Set Properties - Appearance General \(Array\) Understanding Object Types](#)

### Symbol Set Properties - Appearance General (Array)

Symbol Sets have the following general appearance (Array) properties:

#### [Type] On / Off

Select this radio button to display one symbol when a particular expression is TRUE, and another when it is FALSE. For example, you could display a red

symbol when a particular variable tag is in alarm, and a green symbol when it is not.

#### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different symbols for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different symbol for each ON/OFF combination. In other words, you could display a different symbol for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, **ABC**, ABC, ABC, **ABC**.

#### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can display a unique symbol. For example, you could display a different symbol for each threshold of an analog alarm.

#### [Type] Animated

Select this radio button to display an actual animation.

#### Array expression

Enter the expression which is to return one or more integers. For each returned integer, a different symbol will be displayed.

If the return value is:

- Less than 0 (zero), it will be set to 0 (zero), and a runtime hardware alarm will be triggered.
- Greater than 255, it will be set to 255, and a runtime hardware alarm will be triggered.
- A real (non-integer) number, it will be rounded off to the nearest integer.

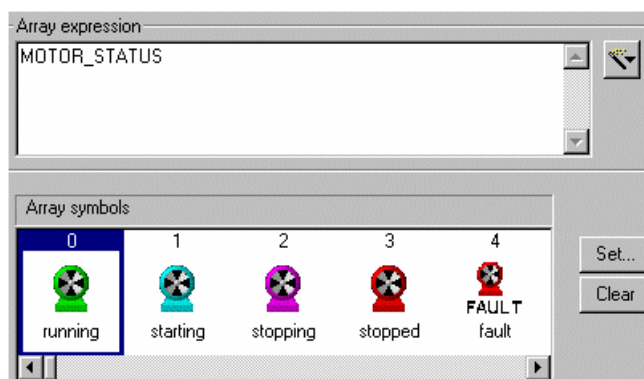
To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options; **Insert tag**, and **Insert Function**.

#### Array symbol

The symbol that is to display for each integer returned by the Array expression entered above (symbol 0 will be used when the expression returns integer 0, symbol 1 will be used when integer 1 is returned etc.).

Click the **Set** button to select a symbol, or **Clear** to clear the current selection.

For example, to display different symbols illustrating the various states of a motor, you could fill out the **Array expression** and **Array symbol** fields as follows:



In this example, **MOTOR\_STATUS** is an analog variable tag representing the status of a motor. Each time the motor changes state, an integer is returned (0 = Running, 1 = Starting etc.), and the appropriate symbol displays.

Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click tabs.

See Also

[Symbol Set Properties - Appearance General \(Animated\)](#)  
[Understanding Object Types](#)

### Symbol Set Properties - Appearance General (Animated)

Symbol Sets have the following general appearance (Animated) properties:

#### [Type] On / Off

Select this radio button to display one symbol when a particular expression is TRUE, and another when it is FALSE. For example, you could display a red symbol when a particular variable tag is in alarm, and a green symbol when it is not.

#### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to display different symbols for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can display a different symbol for each ON/OFF combination. In other words, you could display a different symbol for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

#### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can display a unique symbol. For

example, you could display a different symbol for each threshold of an analog alarm.

### [Type] Animated

Select this radio button to display an actual animation.

#### Animate when

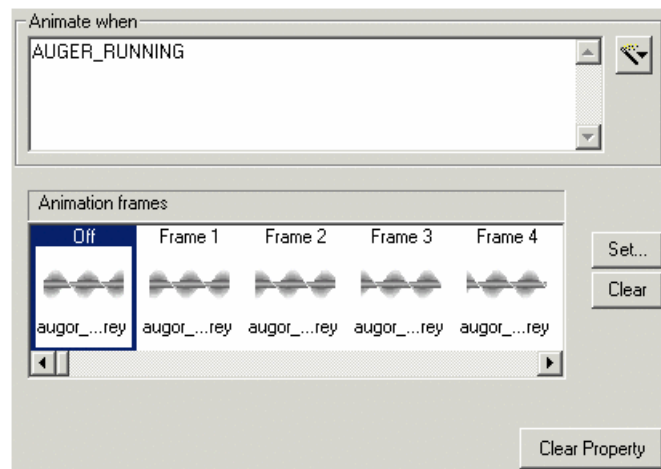
Whenever this expression is true, the animation will run. Whenever the expression is false, the Off frame (below) will display.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options; **Insert tag**, and **Insert Function**.

#### Animation frames

The symbols from Frame 1 onwards are those that will be used as the animation. They are displayed in sequence when the expression above is TRUE. The frequency at which the symbols are displayed is determined by the [Page]AnmDelay parameter. The symbol in the Off frame will display when the expression above is FALSE.

For example, to animate a running auger, you could fill out the **Animate when** and **Animation frames** fields as follows:



In this example, **AUGER\_RUNNING**, is a variable tag which is TRUE when the auger is running. The symbols in the animation frames (Frame 1 onwards) have been designed so that when displayed in sequence, they animate a running auger. The symbol in the Off animation frame will display when **AUGER\_RUNNING** is FALSE.

Click **Apply** or **OK** to bring your changes into effect, or **Cancel** to discard them and exit. Click **Clear Property** to clear property details, and disable the property. To define further properties for the object, click the relevant tabs.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

## Using Trend Objects

The Trend tool allows you to add a trend to the [graphics page](#) with the mouse (click and drag).

After a trend object is drawn, it can be moved, re-sized, re-shaped, brought to the front etc., and its properties can be edited, just like any other type of object.

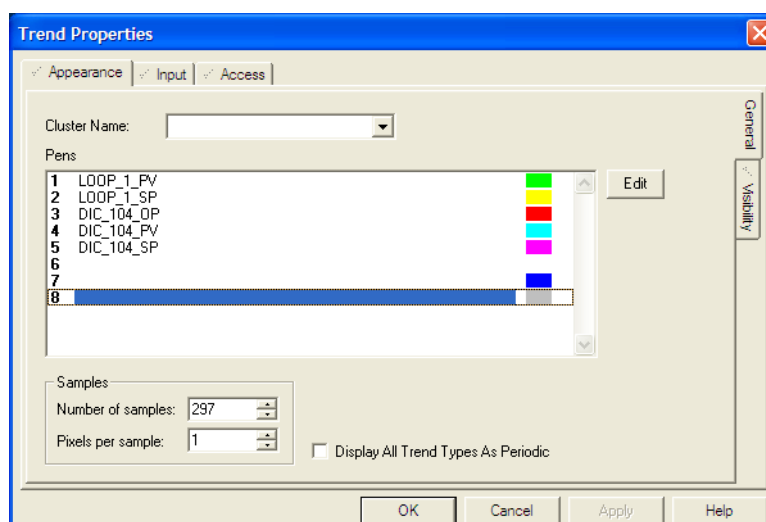
**To add a trend to a page:**

- 1 Click the **Trend** tool



or choose **Objects | Trend**.

- 2 Move the mouse to where you want the trend to start and click (and hold) the mouse button.
- 3 Drag the mouse to the opposite corner of the trend and release the mouse button.
- 4 The Trend Properties appears. Assign the Trend Tags to the pens, choosing appropriate colors.



See Also [Trend properties](#)  
[Insert Trend dialog box](#)  
[Understanding Object Types](#)

Trend properties

Trends have the following general appearance properties:

Cluster Name

The name of the cluster that runs the trends being graphed. Each trend graph can only communicate with one cluster, so you cannot mix trends from multiple clusters on a single trend graph.

**Note:** To mix trends from different clusters on a single trend graph you will need to use Process Analyst.

If there is only one cluster, or, the client is connected to only one cluster, this property can be left blank and the value of the single connected cluster is inferred.

If the client is connected to more than one cluster, then this field must be specified.

Pens

The pens (including color) to be displayed on the graph (31 characters maximum). You can use up to eight pens.

Double-clicking a selected pen or clicking **Edit** allows you to change the trend tag and pen color. To insert a trend tag, click **Wizard** to display the [Insert Trend dialog box](#).

If more than one trend tag is displayed in a trend window and each has a different sample period, the trend with the smallest sample period is used as the general [display period](#).

**Note:** If the trend object is part of a group, part of a pasted [Genie](#) or symbol, or part of the page's template, you can still access its properties: hold down the **Control** (CTRL) key and double-click the object. Alternatively, choose **Goto Object** from the Tools menu, click the object, then click **OK**. Note, however, that if it is part of a pasted Genie or symbol, or part of the template, you cannot edit existing pens, only new ones.

If you are configuring an SPC control chart, you must add a suffix to the trend tag to indicate the type of SPC. CitectSCADA has SPC templates that are easily configured through Genies. Use the Genies rather than defining these trend tags for yourself. The following table lists available SPC types:

SPC Definition	SPC Type
<tag name>.X	Mean of raw data in a subgroup (X - bar)
<tag name>.XCL	Center line of X - bar
<tag name>.XUCL	Upper control limit of X - bar

SPC Definition	SPC Type
<tag name>.XLCL	Lower control limit of X - bar
<tag name>.R	Range of raw data in a subgroup (R - bar)
<tag name>.RCL	Center line of R - bar
<tag name>.RUCL	Upper control limit of R - bar
<tag name>.RLCL	Lower control limit of R - bar
<tag name>.S	Standard deviation of raw data in a subgroup (S - bar)
<tag name>.SCL	Center line of S - bar
<tag name>.SUCL	Upper control limit of S - bar
<tag name>.SLCL	Lower control limit of S - bar

where <tag name> is any trend tag, for example:

Pen 1	PIC117_PV.XCL
Pen 2	PIC117_PV.XUCL
Pen 3	PIC117_PV.XLCL
Pen 4	PIC117_PV.X

If you are using the `PageTrend()` function to display this trend page, leave these fields blank.

### Display all Trend Types as Periodic

When selected, enables all trend pens (both periodic and event) to be displayed as periodic. Event and [periodic trend](#) data can then be displayed on the same graph. If this box is not selected, event and periodic pens have different styles and must be displayed on separate graphs.

**Note:** This option is set by default in the predefined CitectSCADA templates designed for use with periodic trends. It will only need to be enabled for customized templates.

### [Samples] Number of samples (5 Chars.)

The number of samples (1–32767) you can display in your trend window without scrolling (i.e., the width of your trend object). The default depends on the number of pixels per sample and your display resolution. The width of a trend object is equal to **Pixels per sample** × **Number of samples**.

**Note:** For a meaningful trend graph, **Pixels per sample** × **Number of samples** should be less than the width of the display. For example, an XGA screen has a width of 1024 pixels. If you use 10 pixels per sample, 102 samples can be displayed on the screen without scrolling.

### [Samples] Pixels per sample (2 Chars.)

The display width of each sample. The width of a trend object is equal to **Pixels per sample** × **Number of samples**. The default is 1 pixel.

Click **Clear Property** to clear property details, and disable the property. To define other properties for the object, click the relevant tabs.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

### Insert Trend dialog box

This dialog box lets you select a trend tag. To insert a trend tag, select the tag name, then click **OK**. The tag is inserted at the location of the cursor.

See Also [Using Trend Objects](#)  
[Understanding Object Types](#)

## Using Cicode Objects

The Cicode Object tool allows you to add a Cicode Object to the [graphics page](#) with the mouse (click and drag).

A Cicode Object can be any command (such as a function and so on). When the graphics page is displayed at runtime, the command is run continually. Cicode objects can also be assigned a key sequence, allowing you to enter keyboard commands when it is selected at runtime.

After a Cicode object is added, it can be moved and so on and its properties edited, just like other types of object.

**To add a Cicode Object to a page:**

- 1 Click the **Cicode Object** tool,

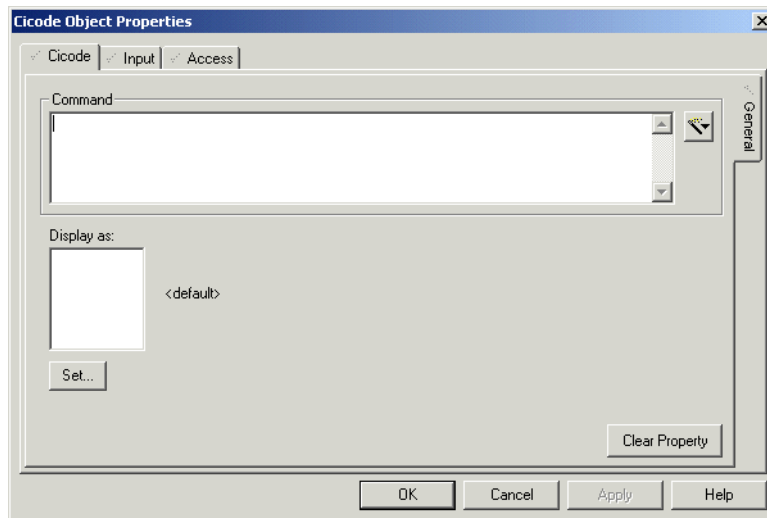


or choose **Objects | Cicode Object**.

- 2 Move the mouse to where you want to add the object, and click the left mouse button.



- 3 Define the relevant properties for the object, and click **OK**.



See Also [Cicode Object Properties - Cicode \(General\)](#)  
[Understanding Object Types](#)

## Cicode Object Properties - Cicode (General)

Cicode Objects have the following General properties:

### **Command** (254 Chars.)

A Cicode command that is continually executed. You can use any Cicode command, in-built Cicode function or user-written function. The command is executed continually (while the page is displayed), for example:

```
Command      DspSymAnm(25, "Pumps.Slurry1", "Pumps.Slurry2", "Pumps.Slurry3");
```

The command in this example uses the in-built function `DspSymAnm()`. The function displays three symbols ("Pumps.Slurry1", "Pumps.Slurry2", "Pumps.Slurry3") continually (at AN 25).

You can also write generic functions by using the Cicode function `DspGetAnCur()` to get the AN number, for example:

```
Command      DspSymAnm(DspGetAnCur(), "Pumps.Slurry1", "Pumps.Slurry2", "Pumps.Slurry3");
```

The command in this example displays three symbols ("Pumps.Slurry1", "Pumps.Slurry2", "Pumps.Slurry3") continually (at the current AN).

If you are using an actual animation, each symbol is displayed at a frequency that is set using the Computer Setup Wizard (also determined by the [Page] `AnmDelay` parameter). To add just an [animation point](#) to the page, add a Cicode Object, without a command.

Click **Clear Property** to clear property details, and disable the property. To define other properties for the object, click the relevant tabs.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

## Using animation points

Each point on a [graphics page](#) where an object is displayed is called an animation point. When you add an object (text, symbols, pipes, etc.) to your page, CitectSCADA automatically allocates a number (termed an AN) to the animation point.

The number of objects that you can use is limited by the performance of your computer, though this would rarely be a problem. A good rule of thumb is to try and keep the number of objects (and hence ANs) less than 3000.

CitectSCADA uses the first 2 ANs for automatically displaying system information such as messages, alarm information and page details. In some applications, such as trend pages, some other ANs are reserved.

You can add individual animation points to a graphics page by using the **Cicode Object** tool (add a Cicode Object without a command).

**Note:** You can locate any object on a page by referencing its AN. To locate an object by its AN use the **Goto Object** tool in the **Tools** menu.

## Using Pasted Symbol Objects

The Paste Symbol tool allows you to insert a symbol from a CitectSCADA library onto the [graphics page](#).



After a symbol is pasted using this tool, it can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like any other type of object.

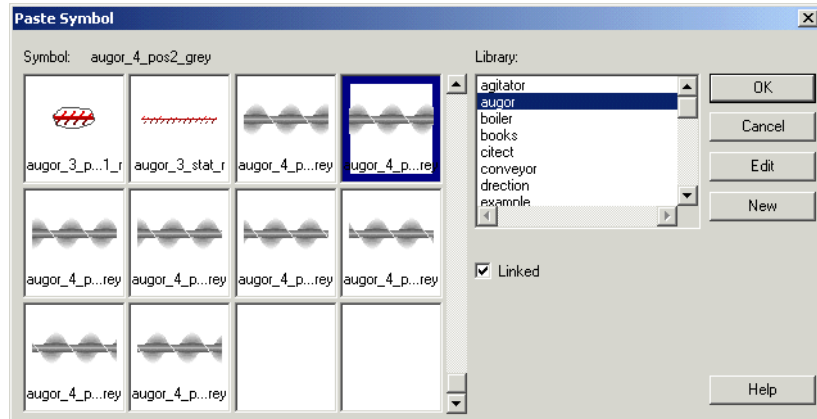
Pasted [library] symbols can be linked to their source, so that changes made to the original are inherited by the pasted symbol.

To display the properties of the objects in the symbol (after pasting), hold the **Control** (CTRL) key down and double-click the specific object. Alternatively, choose **Tools | Goto Object**, click the object, then click **OK**.

To learn more about creating symbols, see [Using libraries](#).

**To paste a symbol from the library to the page:**

- 1 Click the **Paste Symbol** tool or choose **Edit | Paste Symbol**



- 2 To paste a linked symbol, select the **Linked** check box. To paste an unlinked symbol, deselect the **Linked** check box.

To break the link:

- 1 Select the symbol.
- 2 Choose **Edit | Cut Link**.

See Also

[Paste Symbol dialog box](#)  
[Symbol Properties - Appearance \(General\)](#)

## Paste Symbol dialog box

This dialog box lets you paste a symbol from a CitectSCADA library to the [graphics page](#) (or template).

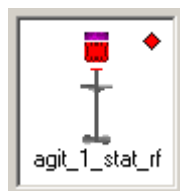
The Paste Symbol dialog box has the following properties:

### Symbol

A table of symbols in the project.

To add a symbol to a graphics page, use the scroll bar to locate the thumbnail image of the symbol, then select it and click **OK** (or double-click the thumbnail image). To edit the object in the library, select it and click **Edit**. To create a new symbol, click **New**.

**Note:** If the symbol has a small diamond-shaped badge next to it, it indicates it is a flashing symbol (see example below.)



### Library

## Symbol Properties - Appearance (General)

The library where the symbol is stored.

### Linked

To paste a symbol that maintains the link with its library, check this box. A symbol that is linked will be automatically updated if the symbol in the library is changed.

You can cut the link at any time with the **Cut Link** command from the Edit menu, but you cannot re-link a symbol with the library after the link has been cut.

If you have selected **Paste Symbol as Flashing**, two dialog boxes appear in sequence, allowing you to choose two images that you want to implement as a flashing symbol. The **Primary Select Symbol** dialog allows you to select the initial image used, the **Flashing Select Symbol** dialog the second image. If the bitmaps are different in size, the flashing symbol is scaled to the size of the primary image. If one of the symbols is itself a flashing symbol, only the primary state will be displayed.

This dialog displays a picture of the selected symbol, name, and path. Click **Set** to change the symbol, or double-click the image. The Select Symbol dialog appears, letting you select a new symbol.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

### To change the properties of an object in a pasted Symbol:

- 1 Click the **Select** tool.
- 2 Hold down the Control (CTRL) key and double-click the object.
- 3 Change the relevant properties in the dialog box. Alternatively, select **Tools | Goto Object**, select the object, and then click **OK**.

See Also [Using Pasted Genie Objects](#)  
[Understanding Object Types](#)

## Using Pasted Genie Objects

The Paste Genie tool allows you to insert a Genie onto the graphics page.



After a Genie is pasted using this tool, it can be resized, rotated, moved, copied, duplicated, pasted, brought to the front, and so on.

To display the properties of the objects in the Genie (after pasting), hold the **Control** (CTRL) key down and double-click the specific object.

## Using ActiveX Objects

CitectSCADA allows you to incorporate ActiveX objects into your CitectSCADA project. This means you can use components that have been developed independently of CitectSCADA.

The ActiveX tool can be used to insert ActiveX objects in graphics pages. After you have selected and positioned an ActiveX object, it can be moved, re-sized, re-shaped, brought to the front etc., and its properties can be edited, just like any other object.

### Managing associated data sources

If an ActiveX object has an association with a data source (for example, it stores data to a DBF file), you need to consider the impact of running a project that contains it on a different machine or via one of the Internet clients ([Internet display client](#) or WebClient).

If the path to the data source is hardcoded to a location on the local machine, the data source will not be found if the project is moved or run remotely. For example, the Database Exchange ActiveX control connects to a recipe.DBF file in the project path. If you restore a project that uses it on a different computer with a different installation path, you will need to recreate the data source to retrieve any recipes.

A solution to the problem is to locate any associated data sources in a central location on a network. For example, if the data source is located on a SQL server, it will be accessible from every machine on the common network.

#### To insert an ActiveX Control:

- 1 Click the **ActiveX** tool,



or choose **Edit | Insert ActiveX Control**.

- 2 Select an ActiveX Control and click **Insert**.

See Also [ActiveX Object Properties](#)  
[Tag Association](#)  
[Object Identification](#)

### ActiveX Object Properties

The two properties tabs common to all ActiveX objects are the **Tag Association** and **Visibility** tabs which appear vertically on the **Appearance** tab. The content and number of all other tabs depends on the individual design of each ActiveX object. This is determined by the amount of flexibility and support its creator has included.

For details on configuring these additional tabs, refer to the documentation provided with the ActiveX object.

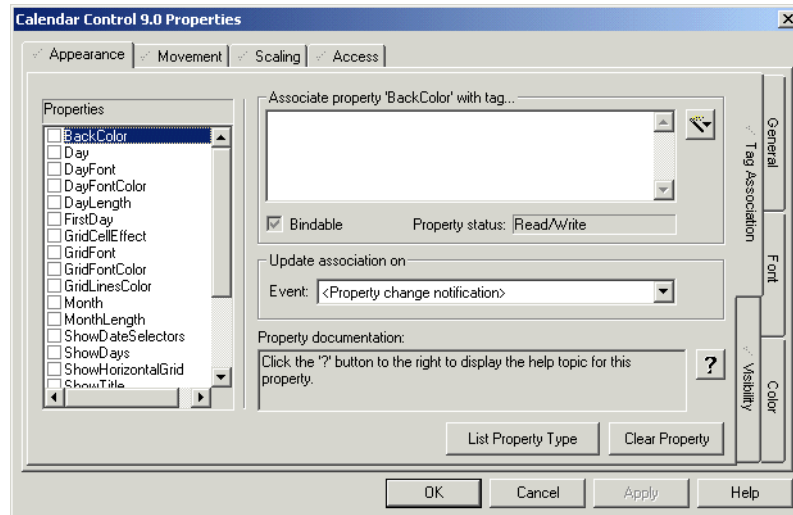
See Also [Tag Association](#)

## Tag Association

You can create an association between a property of an ActiveX object and a variable tag.

To create an association between a property of an ActiveX object and a variable tag:

- 1 Double-click the ActiveX object. The Properties dialog box appears.
- 2 Click the **Tag Association** tab.
- 3 Select a property from the **Properties** list.
- 4 Click the **Wizard | Insert Tag** button.
- 5 Select a tag from the list and click **OK**.



See Also [ActiveX Object Properties - Appearance \(Tag Association\)](#)  
[Understanding Object Types](#)

## ActiveX Object Properties - Appearance (Tag Association)

ActiveX objects have the following appearance properties:

### Properties

The "properties" of an ActiveX object relate to the elements that define the object's functionality and appearance. All the available properties for a selected ActiveX object are listed here, as defined by the object's creator.

The check box to the left of each item on the list indicates whether or not a tag has been associated with the Property. If the box is checked, it means an association has already been defined for the property. If you want to clear this tag association, simply uncheck the box, or clear the tag from the "Associate property with tag. . ." field.

### Associate property with tag

You can create an association between an ActiveX object property and a variable tag so that changing values in one are reflected in the other. To create an association, you need to first choose a property from the property list. The label **Associate property with tag** will change to display the property name.

Select the variable tag you would like to associate with a property by clicking on the Wizard button and choosing from the list of available tags. Alternatively, the tag name can be typed in to the **Associate property with tag** field.

**Note:** You can only use variable tag names in this field. Functions, expressions and constants are not supported when defining ActiveX property tag associations.

You can only associate a property with a variable tag if the tag type is compatible with the property. To display a list of compatible tag types, select the property, and click **List Property Types**. A list of compatible data types will display, or a message will inform you that there are no compatible types.

If there are no CitectSCADA types compatible with the property, the **Associate Property with tag** and **Update association on** fields will be disabled.

If you specify a tag which might be inappropriate for the selected ActiveX property, the **Type Evaluation dialog** will display a warning. This might happen if:

- The types compatible with the property are different to the tag type.
- The tag type is smaller than the types compatible with the property, meaning that data could be truncated or lost.

### Bindable

If an object property is “bindable”, it means it can automatically send notification of value changes to CitectSCADA, and acknowledge any value changes from an associated tag. This means both the property and an associated tag will automatically update whenever the value for either changes.

If a property is not bindable, the property/tag association can only be synchronized according to the event selected in the **Update association on Event** field.

Note that if an object property is bindable, the **Update association on Event** field will be automatically set to **<Property change notification>** by default. You can change this setting if you want the tag association to be updated by a more specific event.

For properties that are not bindable, you can mimic the behavior of **<property change notification>** by selecting **After update** for the **Update association on Event** field.

### Property status

This indicates the read/write status of the selected property. With ActiveX objects, some properties are permanently set, whereas others accept value changes. If a property is marked "read only", it indicates that its value is fixed and can only be read by CitectSCADA. If its status is read/write, can modify the property during runtime via a tag association.

### Update association on Event:

This defines when you want a value update to occur for the selected property and its associated tag. Use the menu to the right of the **Event** field to view events that can be used to trigger a tag association update.

The available events that can be used with a particular property are predefined by an object's creator. They typically include user interaction events (for example, mouse clicks), time events (such as a new day or new month), or value changes (such as "after update").

### Property documentation

Most ActiveX objects have documentation describing the object's controls and functionality. Some have a separate Help file included, others, simple text prompts to explain each property. This depends on what an object's creator has included.

The Property documentation field displays Help information for a selected property, or give instructions to obtain the Help required for a selected property. Usually the following message will appear:

"Click the '?' button to the right to display the Help topic for this property"

The **Help** button displays the ActiveX object Help file (if included), usually with the topic displayed that relates to the selected property. It should also provide information about settings provided on additional tabs the ActiveX object might call up on the properties dialog.

### Clear Property

The Clear Property button clears the tag associations for ALL the ActiveX object properties. To clear a tag association for a particular object property, clear the check box to the left of the property.

If you accidentally click **Clear Property**, you can restore your tag associations by clicking **Cancel** and reopening the ActiveX properties dialog.

## Object Identification

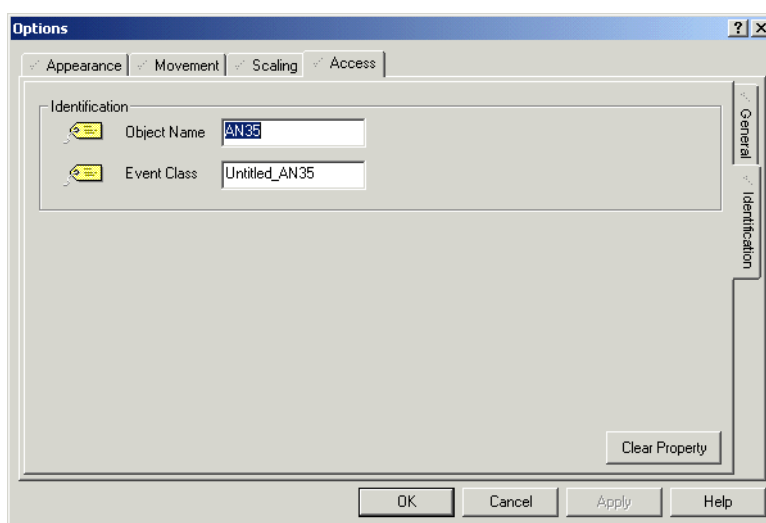
You can identify your ActiveX object.

### To identify your ActiveX object:

- 1 From Graphics Builder, double-click the ActiveX object. The Properties dialog appears.



- 2 Click the **Access** tab.
- 3 Click the **Identification** tab.
- 4 **Assign** your ActiveX object a name in the **Object Name** field.
- 5 **Assign** your ActiveX object an **Event Class**.
- 6 Click **OK**.



See Also [Object Properties - Access \(Identification\)](#)

## Object Properties - Access (Identification)

Objects have the following Access Identification properties:

### [Identification] Object Name

This field allows you to assign a name to your ActiveX object (32 characters maximum). It identifies the object when using the `ObjectByName()`, `ANByName()`, and `CreateControlObject()` Cicode functions.

The name can be any combination of alpha or numeric characters.

### [Identification] Event Class

Allocate a name for the event class of your ActiveX object (16 characters). You can then use this name to create a Cicode function to trap an event.

Don't change the default value if you want to access the ActiveX object using `CitectVBA`. If you do, `CitectVBA` can't access the object. If the Event Class is changed, you can reset it to the default value by clicking **Clear Property**.

### [Persistence] Persist ActiveX data between page transitions

Select this check box to allow changes made to the control to be persisted between pages. For example, you can select this check box if you want an

ActiveX edit control to keep the current text in the control, so that next time the page is entered, the same text is displayed.

The data is only persisted for that session. If CitectSCADA runtime is shut down and restarted, the updated data is not available.

**Note:** The data persisted depends on the ActiveX controls persistence implementation. Some controls will not persist certain data, therefore that data cannot be saved away and restored.

## Using Database Exchange Control Objects

The Database Exchange ActiveX control lets you connect to a data source (for example a database), and extract, display, and edit recipe values. The control is inserted and configured on a graphics page using the Citect Graphics Builder. For more information see the [Database Exchange Object online help](#).

# Chapter 20: Defining Common Object Properties

---

This section describes properties that are common to several object types. Some are also common to object groups.

See Also [3D Effects](#)  
[Visibility](#)  
[Movement](#)  
[Scaling](#)  
[Fill Color](#)  
[Fill Level](#)  
[Touch Commands](#)  
[Keyboard Commands](#)  
[Sliders](#)  
[Access](#)

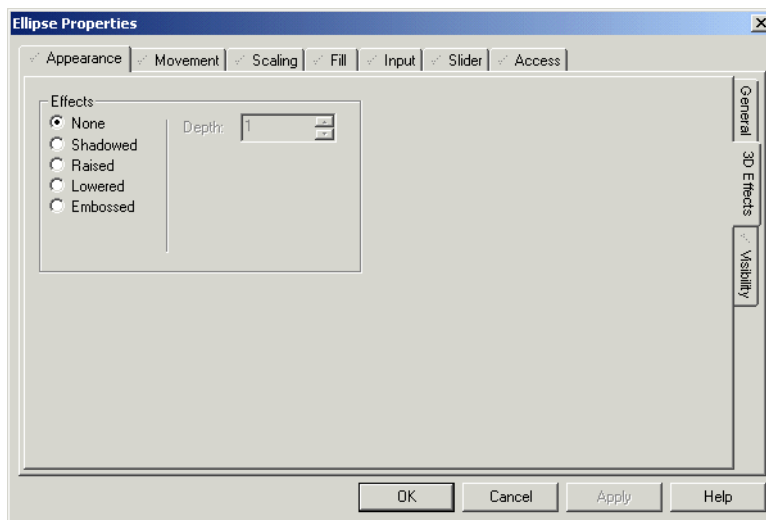
## 3D Effects

You can apply 3D effects to objects to make them more realistic.

**To apply 3D effects to an object:**

- 1 Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the Display properties on new option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you must double-click the group).
- 2 Click the **Appearance** tab.
- 3 Click the **3D Effects** tab (to the right of the dialog).
- 4 The dialog will then display several options to enable you to manipulate your object. To activate any of these options click the option (or the radio bullet to the left of the option).
- 5 By selecting certain options additional fields will display to enable you to further manipulate your object. Enter further details as required, using the Help button for detailed information about each field.

6 Click **OK**.



See Also [Object Properties - Appearance \(3D Effects\)](#)  
[Defining Common Object Properties](#)

## Object Properties - Appearance (3D Effects)

Objects have the following 3D Effects properties.

### [Effects] None

Select this option to display the object without any special effects (such as shadowing, embossing and so on).

### [Effects] Shadowed

Select this option to display the object with a shadow; for example:



### Depth

The distance (in pixels) that the shadow extends below and to the right of the object. This option alters the apparent distance between the object and its shadow, for example:



**Shadow color**

The color of the shadow. The shadow color will not change dynamically with runtime conditions.

**[Effects] Raised**

Select this option to display the object as a raised three dimensional solid, for example:

**Depth**

The distance (in pixels) that the sides of the object extend out from the raised surface. This option alters the apparent distance from the raised surface down to your [graphics page](#), for example:



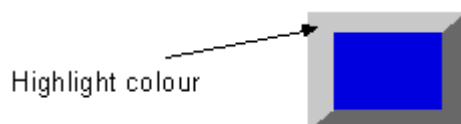
- Depth = 5



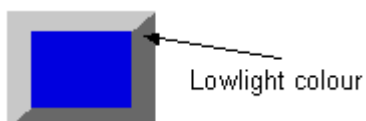
- Depth = 15

**Highlight color**

The color of the directly illuminated “edges” of the object.

**Lowlight color**

The color of the “edges” of the object that are in shadow.



**[Effects] Lowered**

Select this option to display the object as if it is actually lower than your graphics page, for example:

**Depth**

The distance (in pixels) that the sides of the object extend out from the lowered surface. This option alters the apparent distance from the lowered surface up to your graphics page, for example:



- Depth = 5



- Depth = 15

**Highlight color**

The color of the directly illuminated “edges” of the object.



Highlight colour

**Lowlight color**

The color of the “edges” of the object that are in shadow.



Lowlight colour

**[Effects] Embossed**

Select this option to display the object as if it has been embossed on your graphics page, for example:

# Embossed Text

### Depth

The distance (in pixels) that the embossed surface is lowered. This option alters the apparent distance from the embossed surface up to your graphics page, for example:

Embossed Text - Depth = 1

Embossed Text - Depth = 2

### Highlight color

The color of the right and lower edges of the object.

Embossed Text Highlight Colour

### Lowlight color

The color of the left and upper edges of the object.

Lowlight Colour Embossed Text

Click **Apply** or **OK** to save your changes, or **Cancel** to exit. To define further properties for the object, click the relevant tabs.

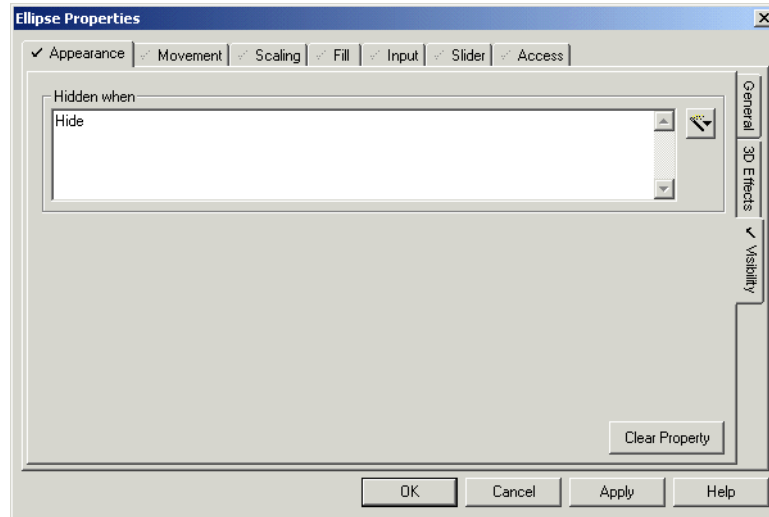
### Visibility

You can determine whether an object is visible or not.

To hide/unhide an object:

- 1 Double click the object you would like to hide.
- 2 Select the **Appearance** tab.
- 3 Select the **Visibility** tab (to the right of the dialog).
- 4 Click the **Wizard** button to the right of the **Hidden when** field.
- 5 Select either **Insert Tag** or **Insert Function** depending on which you would like to relate to your object.
- 6 Enter an [expression](#) in the **Hidden when** field. When this expression is true your object will be hidden.

7 Click **OK**.



See Also

[Object Properties - Appearance \(Visibility\)](#)  
[Defining Common Object Properties](#)

## Object Properties - Appearance (Visibility)

Objects and groups have the following visibility properties:

### Hidden when

The object/group will be hidden whenever the expression entered in this field is TRUE. Enter an expression of 128 characters or less. For example, if you want the object/group to be hidden for all operators except the superintendent, you could enter the following:

```
NOT GetPriv( _Super, _SectionA )
```

where `_Super` and `_SectionA` are labels.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

**Note:** If a group is hidden, all the objects (and other groups) in the group will also be hidden regardless of their individual properties. If the group is visible, its objects will behave according to their individual properties.

Click **Clear Property** to clear property details and disable the property.

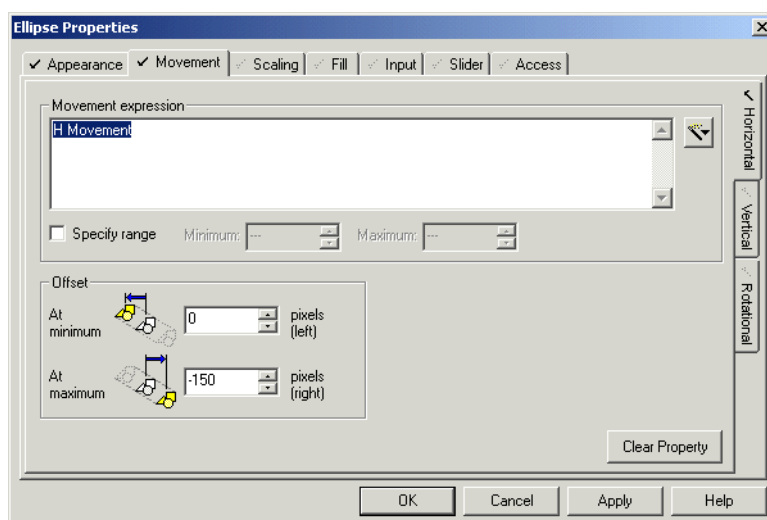
## Movement

You can control the movement of objects.



To configure an object or group that moves:

- 1 Draw the object/group (or paste a symbol). The properties tab dialog automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you must double-click the group.)
- 2 Click the **Movement** tab.
- 3 Click the **Horizontal**, **Vertical** or **Rotational** tab (to the right of the dialog).
- 4 Enter a Movement **expression** (the expression that will move the object/group at runtime).
- 5 Enter further details as required, using the **Help** button for detailed information about each field.
- 6 Click **OK**.



See Also [Object Properties - Movement \(Horizontal\)](#)  
[Object Properties - Movement \(Vertical\)](#)  
[Object Properties - Movement \(Rotational\)](#)  
[Group and object movement - examples](#)

## Object Properties - Movement (Horizontal)

Objects and groups can be moved from side to side during runtime, changing dynamically whenever the value of a particular expression changes. By default, as the value of the expression increases, the object/group will move (in increments) to the right. As the value of the expression decreases, the object/group will move (in increments) to the left.

This property could, for example, be used to display the position of a coal stacker moving along a stockpile.

**Note:** Horizontal movement cannot be used if the horizontal slider is enabled. A group and its objects can be configured with any movement combination (i.e., a group can move vertically while one of its objects rotates, and so on).

Objects and groups have the following horizontal movement properties:

#### **Movement expression**

The value of the expression entered in this field (128 characters maximum) will determine the horizontal movement of the object/group. By default, when the expression returns its minimum value, the object/group will shift hard to the left. When the expression returns its maximum, the object/group will shift hard to the right. For intermediate values, the object/group will move to the appropriate position between the minimum and maximum offset.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### **[Movement expression] Specify range**

Select this box to manually specify **Minimum** and **Maximum** values for the movement expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.

#### **[Movement expression] Minimum**

Enter the minimum value for the expression. When this value is returned by the expression, the object/group will shift to the left, by the **At minimum** offset. You can only enter a value here if you have selected the **Specify** range box.

#### **[Movement expression] Maximum**

Enter the maximum value for the expression. When this value is returned by the expression, the object/group will shift to the right, by the **At maximum** offset. You can only enter a value here if you have selected the **Specify** range box.

#### **[Offset] At minimum**

The distance (number of pixels from the original object/group center) that the object/group will shift to the left when the **Movement expression** returns its minimum value.

You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

#### **[Offset] At maximum**

The distance (number of pixels from the original object/group center) that the object/group will shift to the right when the **Movement expression** returns its maximum value.

You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

**Note:** You can shift the object/group right at minimum and left at maximum, by entering negative distances in the Offset fields, or by swapping the expression limits (in the **Minimum** and **Maximum** fields).

Click **Clear Property** to clear property details, and disable the property.

See Also

[Object Properties - Movement \(Vertical\)](#)

[Object Properties - Movement \(Rotational\)](#)

## Object Properties - Movement (Vertical)

Objects and groups can be moved up and down during runtime, changing dynamically whenever the value of a particular expression changes. By default, as the value of the expression increases, the object/group will move up (in increments). As the value of the expression decreases, the object/group will move down (in increments).

This property could be used to display the movement of an elevator.

**Note:** Vertical Movement cannot be used if the Vertical Slider is enabled. A group and its objects can be configured with any movement combination (i.e. a group can move vertically while one of its objects rotates, and so on).

Objects and groups have the following vertical movement properties:

### **Movement expression**

The value of the expression entered in this field (128 characters maximum) will determine the vertical movement of the object/group. By default, when the expression returns its minimum value, the object/group will shift down to its lowest position. When the expression returns its maximum, the object/group will shift up to its highest position. For intermediate values, the object/group will move to the appropriate position between the minimum and maximum offset.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

### **[Movement expression] Specify range**

Select this box to manually specify **Minimum** and **Maximum** values for the movement expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.

**[Movement expression] Minimum**

Enter the minimum value for the expression. When this value is returned by the expression, the object/group will shift down, by the **At minimum** offset. You can only enter a value here if you have selected the **Specify** range box.

**[Movement expression] Maximum**

Enter the maximum value for the expression. When this value is returned by the expression, the object/group will shift up, by the **At maximum** offset. You can only enter a value here if you have selected the **Specify** range box.

**[Offset] At maximum**

The distance (number of pixels from the original object/group center) that the object/group will shift up when the **Movement expression** returns its maximum value.

You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

**[Offset] At minimum**

The distance (number of pixels from the original object/group center) that the object/group will shift down when the **Movement expression** returns its minimum value.

You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

**Note:** You can shift the object/group up at minimum and down at maximum, by entering negative distances in the Offset fields, or by swapping the expression limits (in the **Minimum** and **Maximum** fields).

See Also

[Object Properties - Movement \(Horizontal\)](#)

[Object Properties - Movement \(Rotational\)](#)

## Object Properties - Movement (Rotational)

Objects and groups can be dynamically rotated during runtime, whenever the value of a particular expression changes. By default, as the value of the expression increases, the object/group will rotate clockwise (in increments). As the value of the expression decreases, the object/group will rotate anti-clockwise (in increments).

This property could be used to display an aerial view of the movement of a circular stacker in a coal mining operation.

**Note:** Rotational Movement cannot be used if the Rotational Slider is enabled. A group and its objects can be configured with any movement combination (i.e. a group can move vertically while one of its objects rotates, and so on).

Objects and groups have the following rotational movement properties:

### Angle expression (128 Chars.)

The value of the expression entered in this field will determine the rotation of the object/group. During runtime, when the expression returns its minimum value, the object/group will rotate to its anti-clockwise limit. When the expression returns its maximum, the object/group will rotate to its clockwise limit. For intermediate values, the object/group will rotate to the appropriate position between the minimum and maximum limits.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### [Angle expression] Specify range

Select this box to manually specify **Minimum** and **Maximum** values for the angle expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.

#### [Angle expression] Minimum

Enter the minimum value for the expression. When this value is returned by the expression, the object/group will rotate anti-clockwise, by the minimum offset. You can only enter a value here if you have selected the **Specify** range box.

#### [Angle expression] Maximum

Enter the maximum value for the expression. When this value is returned by the expression, the object/group will rotate clockwise, by the maximum offset. You can only enter a value here if you have selected the **Specify** range box.

#### [Angle] At minimum

The anti-clockwise angle (in degrees relative to 0°) that the object/group will rotate when the **Angle expression** returns its minimum value.

You can change the angle by pressing the up and down arrows to the right of the field, or by entering another value in this field.

#### [Angle] At maximum

The clockwise angle (in degrees relative to 0°) that the object/group will rotate when the **Angle expression** returns its maximum value.

You can change the angle value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

**Note:** You can rotate the object/group clockwise at minimum and anti-clockwise at maximum, by entering negative angles in the Angle fields, or by swapping the expression limits (in the **Minimum** and **Maximum** fields).

**[Center axis offset] Express**

Click this radio button for the quick and easy way of selecting the point about which the object/group will rotate. The express option gives you the choice of 9 points (Top Left, Bottom Right and so on), which are displayed in the picture field to the right of the dialog. To select one, just click it with your mouse.

**[Center axis offset] Custom**

Click this radio button to define your own center axis. When you select this radio button, two fields will display to the right, allowing you to plot the position of your center axis. Specify the distance to the right in the first field, and the distance down in the second. The Center axis is plotted based on these two values.

For example, if you enter 8 as the horizontal offset, and 13 as the vertical offset, the Center axis will be 8 pixels to the right, and 13 pixels below the center of the object/group.

Enter negative values in the offset distance fields to move the Center axis left instead of right, and up instead of down.

See Also

[Object Properties - Movement \(Horizontal\)](#)

[Object Properties - Movement \(Vertical\)](#)

## Group and object movement - examples

A group and its objects can be configured with any combination of movement (horizontal, vertical, and rotational). The following examples illustrate how some of these combinations work.

**Example 1: Rotating the group and moving the object left to right**

If your group is configured to rotate from 0° to 60°, and one of its objects is configured to move left and right, the object will do both. It will move left and right as per its own properties, and, at the same time, it will rotate as part of the group. Remember, however, that 'left' and 'right' are relative to the original orientation of the group, not the page. As the group rotates, 'horizontal' also rotates. When the group has rotated 15°, 'left' is actually 285° (not 270°), and 'right' is actually 105° (not 90°). When the group has rotated 50°, 'left' is 320°, and 'right' is 140°, and so on.

**Original state of group**

Group rotated right, and ellipse moved left



### Example 2: Rotating the group and moving the object up and down

If your group is configured to rotate from  $0^\circ$  to  $60^\circ$ , and one of its objects is configured to move up and down, the object will do both. It will move up and down as per its own properties, and, at the same time, it will rotate as part of the group. Remember, however, that 'up' and 'down' are relative to the original orientation of the group, not the page. As the group rotates, 'vertical' also rotates. When the group has rotated  $15^\circ$ , 'up' is actually  $15^\circ$  (not  $0^\circ$ ), and 'down' is actually  $195^\circ$  (not  $180^\circ$ ). When the group has rotated  $50^\circ$ , 'up' is  $50^\circ$ , and 'down' is  $230^\circ$ , and so on.

Original state of group



Group rotated right, and ellipse moved up



### Example 3: Rotating the group clockwise and rotating the object anticlockwise

If your group is configured to rotate from  $0^{\circ}$ – $60^{\circ}$ , and one of its objects is configured to rotate from  $90^{\circ}$ – $0^{\circ}$ , the object will do both. It will rotate as per its own properties, and, at the same time, it will rotate as part of the group. Remember, however, that the object's rotation is relative to the group, not the page. If the group rotates  $60^{\circ}$  to the right, and the object rotates  $90^{\circ}$  to the left, the object has only rotated  $30^{\circ}$  to the left relative to the page.

#### Original state of group



#### Group rotated clockwise, and ellipse rotated anticlockwise



**Note:** By moving the ellipse as shown in the above examples, you are actually changing the overall size of the group. It is important to remember this as it might affect object fill levels.

See Also [Movement](#)  
[Defining Common Object Properties](#)

## Scaling

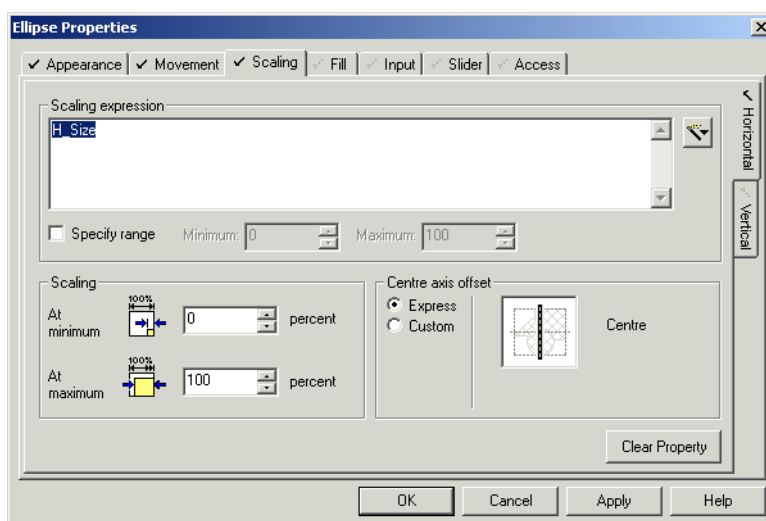
You can scale objects to the size you want.

#### To configure an object or group that changes size:

- 1 Draw the object (or paste a symbol). The object properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder.
- 2 Click the **Scaling** tab.



- 3 Click the **Horizontal** or **Vertical** tab (to the right of the dialog).
- 4 Enter a **Scaling expression** (the expression that will change the size of the object at runtime).
- 5 Enter further object property details as required, using the **Help** button for detailed information about each field.
- 6 Click **OK**.



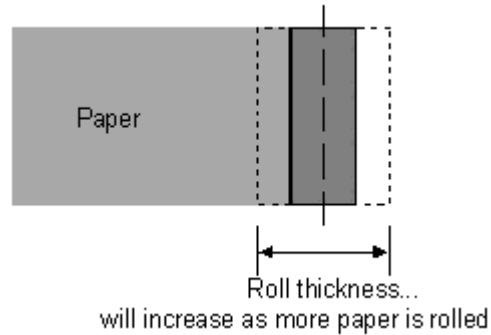
See Also

[Object Properties - Scaling \(Horizontal\)](#)  
[Object Properties - Scaling \(Vertical\)](#)  
[Defining Common Object Properties](#)

## Object Properties - Scaling (Horizontal)

The width of an object can be dynamically changed during runtime whenever the value of a particular expression changes. As the value of the expression increases and decreases, the width of the object increases or decreases accordingly as a percentage of the original width; that is, when it was added to the graphics page.

For example, an aerial view of a paper roll (in a paper mill), could display changing roll thickness:



Objects have the following horizontal scaling properties:

#### Scaling expression

The value of the expression entered in this field (128 characters maximum) will determine the horizontal scaling (width) of the object. By default, when the expression returns its minimum value, the object will display at its minimum width (as defined in the Scaling fields below). When the expression returns its maximum value, the object will display at its maximum width (as defined in the Scaling fields below).

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### [Scaling expression] Specify range

Select this box to manually specify **Minimum** and **Maximum** values for the scaling expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.

#### [Scaling expression] Minimum

Enter the minimum value for the expression. When this value is returned by the expression, the width of the object will be reduced to its minimum. You can only enter a value here if you have selected the **Specify** range box.

#### [Scaling expression] Maximum

Enter the maximum value for the expression. When this value is returned by the expression, the width of the object will be increased to its maximum. You can only enter a value here if you have selected the **Specify** range box.

**[Scaling] At minimum**

The minimum width of the object (as a percentage of its original width). The object will be reduced to this width when the **Scaling expression** returns its minimum value.

You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.

**[Scaling] At maximum**

The maximum width of the object (as a percentage of its original width). The object will grow to this width when the **Scaling expression** returns its maximum value.

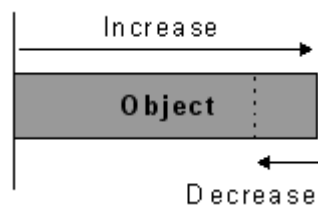
You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.

**Note:** You can increase the width at minimum, and decrease it at maximum, by swapping the percentages in the Scaling fields (i.e. put the high percentage in the **At minimum** field, and the low in the **At maximum**), or by swapping the expression limits (in the **Minimum** and **Maximum** fields).

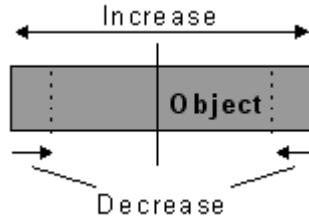
**[Center axis offset] Express**

Click this radio button for the quick and easy way of selecting one of three of the object's vertical axes (left, center, and right). These axes appear in the picture field to the right of the dialog.

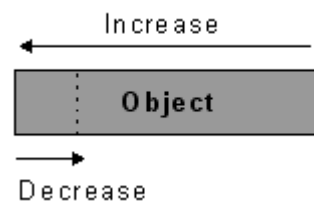
If you choose **Left**, all width changes occur to the right of the object. (i.e., the left edge remains anchored):



If you choose **Center**, width changes occur equally to both sides. (i.e., the vertical center axis remains anchored):



If you choose **Right**, all width changes occur to the left of the object. (i.e., the right edge remains anchored):



#### [Center axis offset] Custom

Click this radio button to define your own center axis. A field appears to the right of the dialog allowing you to specify how far from the object center (in pixels) you would like to place the axis. Although this option gives you the option to place the center axis anywhere on the object, once placed, the scaling process works in exactly the same manner as for the Express option (illustrated above).

For example, if you enter 20, the Center axis will be 20 pixels to the right of the object center.

Enter a negative value to move the center axis left instead of right.

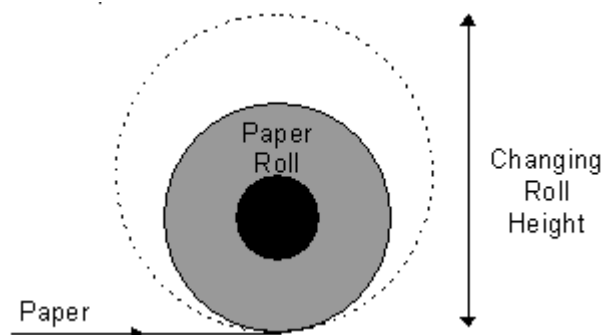
**Note:** If a group and its objects are configured to change size during runtime, the group scaling effects will be combined with the object scaling effects. For example, if a group is configured to double in size at runtime, and one of its object is configured to halve in size, the object will appear to remain the same size (it halves, then doubles). Remember, however, that the object's position might change as the group gets bigger.

See Also [Object Properties - Scaling \(Vertical\)](#)

## Object Properties - Scaling (Vertical)

You can change the height of an object during runtime. As the value of the expression increases or decreases, the height of the object increases or decreases accordingly as a percentage of the original height; that is, when the object was added to the graphics page.

For example, an elevation of a paper roll (in a paper mill), could display changing roll height (and width):



Objects have the following vertical scaling properties:

#### Scaling expression

The value of the expression entered in this field (128 characters maximum) will determine the vertical scaling (height) of the object. By default, when the expression returns its minimum value, the object will display at its minimum height (as defined in the Scaling fields below). When the expression returns its maximum value, the object will display at its maximum height (as defined in the Scaling fields below).

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### [Scaling expression] Specify range

Select this box to manually specify **Minimum** and **Maximum** values for the scaling expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.

#### [Scaling expression] Minimum

Enter the minimum value for the expression. When this value is returned by the expression, the height of the object will be reduced to its minimum. You can only enter a value here if you have selected the **Specify** range box.

#### [Scaling expression] Maximum

Enter the maximum value for the expression. When this value is returned by the expression, the height of the object will be increased to its maximum. You can only enter a value here if you have selected the **Specify** range box.

**[Scaling] At minimum**

The minimum height of the object (as a percentage of its original height). The object will be reduced to this height when the **Scaling expression** returns its minimum value. You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.

**[Scaling] At maximum**

The maximum height of the object (as a percentage of its original height). The object will grow to this height when the **Scaling expression** returns its maximum value.

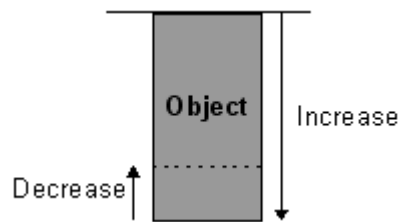
You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.

**Note:** You can increase the height at minimum, and decrease it at maximum, by swapping the percentages in the Scaling fields (i.e. put the high percentage in the **At minimum** field, and the low in the **At maximum**), or by swapping the expression limits (in the **Minimum** and **Maximum** fields).

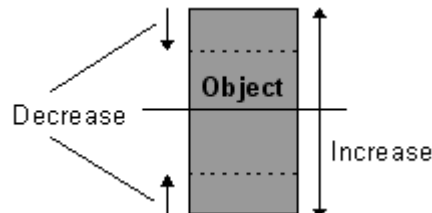
**[Center axis offset] Express**

Click this radio button to select one of three of the object's horizontal axes (top, middle, and bottom). These axes appear in the picture field to the right of the dialog. Click an axis to select it.

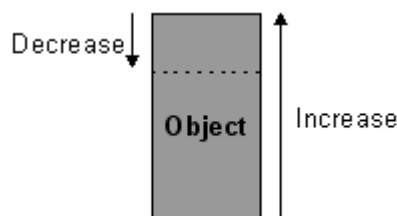
If you choose the top, all height changes will occur from the top of the object down. (i.e. the top edge will remain anchored):



If you choose the middle, height changes will occur equally above and below the axis. (i.e. the horizontal center axis will remain anchored):



If you choose the bottom, all width changes will occur to the top edge of the object. (i.e. the bottom edge will remain anchored):



#### [Center axis offset] Custom

Click this radio button to define your own center axis. A field will display to the right of the dialog, allowing you to specify how far from the object center (in pixels) you would like to place the axis. Although this option gives you the freedom to place the center axis anywhere on the object, once placed, the scaling process works in exactly the same manner as for the Express option (illustrated above).

For example, if you enter 20, the Center axis will be 20 pixels below the object center.

Enter a negative value to move the Center axis up instead of down.

#### Notes:

- If a group and its objects are configured to change size during runtime, the group scaling effects will be combined with the object scaling effects. For example, if a group is configured to double in size at runtime, and one of its object is configured to halve in size, the object will appear to remain the same size (it halves, then doubles). Remember, however, that the object's position might change as the group gets bigger.
- When the radio buttons in Object Properties - Fill Color (On/Off, Multi-state and so on) are selected, they change the appearance of the right-hand side of the dialog.

See Also [Object Properties - Scaling \(Horizontal\)](#)

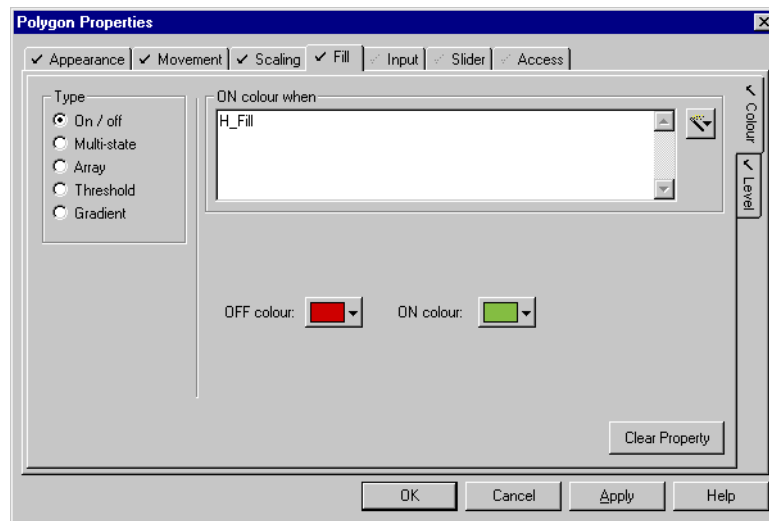
## Fill Color

You can control the fill color to use for your objects.

#### To configure an object or group with changing fill color:

- 1 Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you must double-click the group.)

- 2 Click the **Fill** tab.
- 3 Click the **Color** tab (to the right of the dialog).
- 4 Select the type of color change (On/Off, Multi-state and so on).
- 5 Enter the expression/conditions that will change the object's fill color at runtime.
- 6 Enter additional object property details as required.
- 7 Click **OK**.



See Also

[Object Properties - Fill Color \(On/Off\)](#)  
[Object Properties - Fill Color \(Multi-state\)](#)  
[Object Properties - Fill Color \(Array\)](#)  
[Object Properties - Fill Color \(Threshold\)](#)  
[Object Properties - Fill Color \(Gradient\)](#)

## Object Properties - Fill Color (On/Off)

Objects and groups have the following Fill Color (On/Off) properties:

### [Type] On / Off

Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.

### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.



For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

#### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.

#### [Type] Threshold

Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color should change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for all speeds in between.

#### [Type] Gradient

Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a fade from one color to another.

#### ON color when

The color selected as the **ON color** (below) will be used as the fill color whenever the condition entered here (128 characters maximum) is TRUE. The color selected as the **OFF color** (below) will be used as the fill color whenever this condition is FALSE. For example, you could fill an object/group with blue when **MIX\_RUNNING** is TRUE, and white when it is FALSE.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### OFF color

The fill color whenever the condition entered above is FALSE. For example, you could fill the object/group with white when **MIX\_RUNNING** is FALSE.

**Note:** The color that you select here will change any Fill color specified through Appearance (General) properties tab.

#### ON color

The fill color whenever the condition entered above is TRUE. For example, you could fill the object/group with blue when **MIX\_RUNNING** is TRUE.

**Note:** Group fill color is only applied if the individual objects in the group do not have their own fill colors defined.

See Also [Object Properties - Fill Color \(Multi-state\)](#)  
[Object Properties - Fill Color \(Array\)](#)  
[Object Properties - Fill Color \(Threshold\)](#)  
[Object Properties - Fill Color \(Gradient\)](#)

## Object Properties - Fill Color (Multi-state)

Objects and groups have the following Fill Color (Multi-state) properties:

### [Type] On / Off

Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.

### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0–255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.

### [Type] Threshold

Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color should change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for all speeds in between.

### [Type] Gradient

Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a fade from one color to another.

## Conditions

The conditions you enter here (using a maximum of 128 characters per condition) will occur together in different ways, at different times. You can use each unique combination to force a different fill color for the object/group.

To enter a condition, click the relevant line (A, B, C, and so on), click **Edit**, and type in the condition. You can add more conditions (to a maximum of 5, providing 32 combinations), using the **Add** button. To insert a tag or function, click the **Wizard** button. This button displays two options: Insert Tag and Insert Function. You can also remove conditions by clicking **Delete**, but there must always be at least one condition in this field. Conditions left blank (instead of deleted) are evaluated as permanently false at runtime.

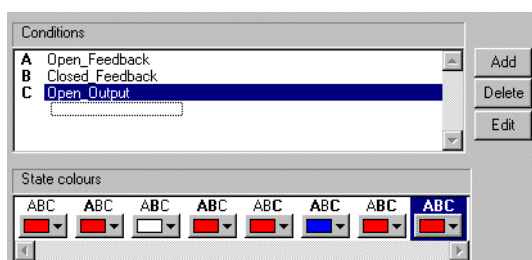
## State colors

The fill colors that will be used for each combination of the above conditions.

**Note:** The color that you select as **ABC** (all conditions false) will change any Fill color specified through Appearance (General) properties tab.

For example:

To fill the object/group with a different color each time the status of a valve changes, you could fill out the **Conditions** and **State symbols** fields as follows:



In this example, **Open\_Feedback**, and **Close\_Feedback** are variable tags representing digital inputs on the valve, and **Open\_Output** is a variable tag representing an output on the valve. So, ABC means **Open\_Feedback** is ON, and **Close\_Feedback** and **Open\_Output** are both OFF. For this combination, the red is used as the fill color to indicate a fault, because the valve is open when it should be closed. The same type of logic applies to the rest of the states.

**Note:** Group fill color is only applied if the individual objects in the group do not have their own fill colors defined.

See Also

[Object Properties - Fill Color \(On/Off\)](#)  
[Object Properties - Fill Color \(Array\)](#)  
[Object Properties - Fill Color \(Threshold\)](#)  
[Object Properties - Fill Color \(Gradient\)](#)

## Object Properties - Fill Color (Array)

Objects and groups have the following Fill Color (Array) properties:

### [Type] On / Off

Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.

### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0–255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.

### [Type] Threshold

Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color should change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for all speeds in between.

### [Type] Gradient

Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a fade from one color to another.

### Array expression (128 Chars.)

Enter the expression which is to return an integer. For each value returned, a different color will fill the object/group.

If the return value is:

- Less than 0 (zero), it will be set to 0 (zero), and a runtime hardware alarm will be triggered.

- Greater than 255, it will be set to 255, and a runtime hardware alarm will be triggered.
- A real (non-integer) number, it will be truncated (e.g. 8.1 and 8.7 would both be truncated to 8).

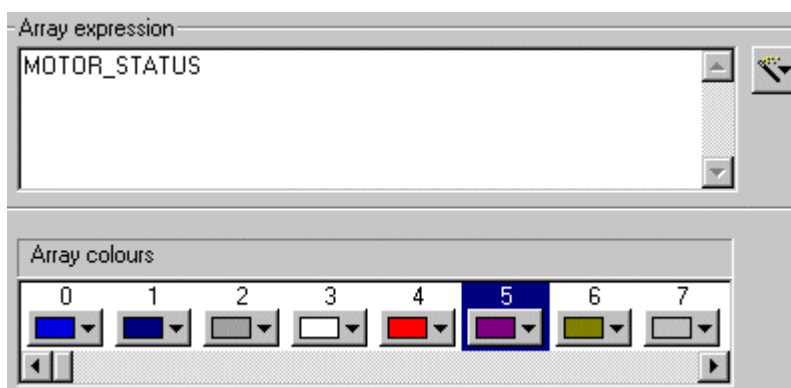
To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

### Array colors

The fill colors that will be used for each integer returned by the Array expression entered above (color 0 will be used when the expression returns integer 0, color 1 will be used when integer 1 is returned and so on).

**Note:** The color that you select for color 0 (zero) will change any Fill color specified through Appearance - General tab.

For example, to display different symbols illustrating the various states of a motor, you could fill out the **Array expression** and **Array symbol** fields as follows:



In this example, **MOTOR\_STATUS** is an analog variable tag representing the status of a motor. Each time the motor changes state, an integer is returned (0 = Running, 1 = Starting and so on), and the appropriate color fills the object/group. Color 5 onwards have no bearing on the fill color, because the tag only returns 5 unique integers (0–4).

**Note:** Group fill color is only applied if the individual objects in the group do not have their own fill colors defined.

See Also

[Object Properties - Fill Color \(On/Off\)](#)  
[Object Properties - Fill Color \(Multi-state\)](#)  
[Object Properties - Fill Color \(Threshold\)](#)  
[Object Properties - Fill Color \(Gradient\)](#)

## Object Properties - Fill Color (Threshold)

Objects and groups have the following fill color (threshold) properties.

### [Type] On / Off

Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.

### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0–255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.

### [Type] Threshold

Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color should change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for all speeds in between.

### [Type] Gradient

Select this radio button to dynamically (and smoothly) graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a smooth fade from one color to another.

### Color expression

The value of the expression entered in this field (128 characters maximum) determine the fill color of the object/group. i.e. When the value of this expression reaches a threshold value (as defined below), fill color will change.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

### [Color expression] Specify range

Select this box to manually specify **Minimum** and **Maximum** values for the Color expression, rather than using the default values. (Threshold values are percentages of the range between **Minimum** and **Maximum**.) For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.

### [Color expression] Minimum

Enter the minimum value for the expression. In terms of thresholds, the Minimum is **0%**. You can only enter a value here if you have selected the **Specify** range box.

### [Color expression] Maximum

Enter the maximum value for the expression. In terms of thresholds, the Maximum is **100%**. You can only enter a value here if you have selected the **Specify** range box.

### Thresholds (%)

The thresholds and their associated colors. A threshold is entered as a percentage of the expression range (the range of values that can be returned by the expression). For example, if the expression's minimum is 0 and its Maximum 200, the default thresholds would have the following effects:

Threshold	Associated Color	Meaning
< 5%	Bright Blue	When the expression returns <b>less than 10</b> , the color fill will be <b>Bright Blue</b> .
< 15%	Blue	When the expression returns <b>less than 30</b> , the color fill will be <b>Blue</b> .
> 85%	Red	When the expression returns <b>greater than 170</b> , the color fill will be <b>Red</b> .
> 95%	Bright Red	When the expression returns <b>greater than 190</b> , the color fill will be <b>Bright Red</b> .

You can add up to 100 threshold color combinations. To add a combination, click **Add** and enter the relevant details. To edit an existing combination, click the relevant line. You can also remove combinations by clicking **Delete**.

Any values not included in a range (e.g. between 15% and 85% in the example above) produce a static fill color as specified through **Appearance - General** tab.

**Note:** Group fill color is only applied if the individual objects in the group do not have their own fill colors defined.

See Also

[Object Properties - Fill Color \(On/Off\)](#)

[Object Properties - Fill Color \(Multi-state\)](#)

[Object Properties - Fill Color \(Array\)](#)  
[Object Properties - Fill Color \(Gradient\)](#)

## Object Properties - Fill Color (Gradient)

Objects and groups have the following Fill Color (Gradient) properties:

### [Type] On / Off

Select this radio button to fill the object/group with one color when a particular expression is TRUE, and another when it is FALSE. For example, you could fill an object/group with red when a particular variable tag is in alarm, and green when it is not.

### [Type] Multi-state

This option is useful when you have several possible conditions, occurring together in different combinations, at different times. Select this option to fill the object/group with a different color for each combination.

For example, three digital variable tags (A,B, and C) can each be ON or OFF at any time. You can fill the object/group with a different color for each ON/OFF combination. In other words, you could use a different fill color for each of the following ON/OFF combinations ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC.

### [Type] Array

The Array option allows you to enter an expression which returns an integer. For each unique integer (from 0 to 255), you can fill the object/group with a different color. For example, you could use a different fill color for each threshold of an analog alarm.

### [Type] Threshold

Select this radio button to dynamically change the fill color when an expression reaches a specific value (threshold). For example, you might decide that the fill color should change to red when the speed of a motor is greater than or equal to 4500 rpm, and to white when less than or equal to 100 rpm, but remains gray for all speeds in between.

### [Type] Gradient

Select this radio button to dynamically graduate the fill color, displaying a different color for each unique value returned by a particular expression. This option allows you to select two colors, to be used as the color limits. The color for each value returned is automatically selected from within the range defined by these limits. The result is a fade from one color to another.

### Color expression

The value of the expression entered in this field (128 characters maximum) will determine the fill color of the object/group. By default, when the expression returns its minimum value, the fill color will be the **At minimum** color (as defined below). When the expression returns its maximum value, the fill color



will be the **At maximum** color (as defined below). When the expression returns a value half-way between its minimum and maximum, a color will be selected from the half-way point of the range defined by the **At minimum** and **At maximum** colors.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### **[Color expression] Specify range**

Select this box to manually specify **Minimum** and **Maximum** values for the Color expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.

#### **[Color expression] Minimum**

Enter the minimum value for the expression. When this value is returned by the expression, the fill color of the object/group will be the **At minimum** color. You can only enter a value here if you have selected the **Specify** range box.

#### **[Color expression] Maximum**

Enter the maximum value for the expression. When this value is returned by the expression, the fill color of the object/group will be the **At maximum** color. You can only enter a value here if you have selected the **Specify** range box.

#### **At minimum**

The fill color of the object/group when the **Color expression** returns its minimum value.

**Note:** The color that you select here will change any Fill color specified through **Appearance - General** tab.

#### **At maximum**

The fill color of the object/group when the **Color expression** returns its maximum value.

**Note:** Group fill color is only applied if the individual objects in the group do not have their own fill colors defined.

See Also

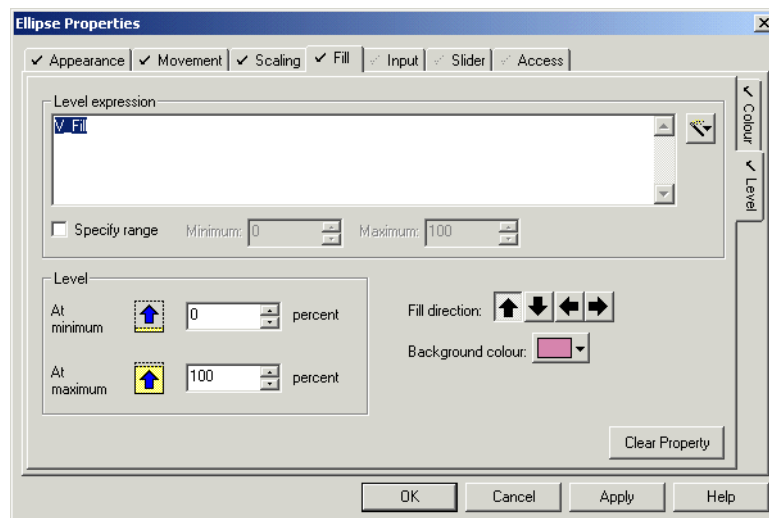
[Object Properties - Fill Color \(On/Off\)](#)  
[Object Properties - Fill Color \(Multi-state\)](#)  
[Object Properties - Fill Color \(Array\)](#)  
[Object Properties - Fill Color \(Threshold\)](#)

## Fill Level

You can control the fill level shown in your objects.

**To configure an object or group with changing fill level:**

- 1 Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you must double-click the group.)
- 2 Click the **Fill** tab.
- 3 Click the **Level** tab (to the right of the dialog).
- 4 Enter a **Level expression** (the expression that will change the fill level of the object/group at runtime).
- 5 Enter additional properties as required.
- 6 Click **OK**.



See Also [Object Properties - Fill \(Level\)](#)  
[Defining Common Object Properties](#)

### Object Properties - Fill (Level)

The fill level of an object/group can be changed during runtime, increasing or decreasing dynamically whenever the value of a particular expression changes. As the value of the expression increases and decreases, the fill level will increase and decrease accordingly (as a percentage of the full capacity of the object/group). If the object/group resizes at runtime, the fill level will adjust automatically in order to maintain the correct percentage.

The color that is used is set through either General Appearance, or Color Fill.

This property could be used to display temperature variations. You could even combine the Fill Color and Fill Level properties to produce a thermometer with mercury that rises and changes color with rising temperature.

Objects and groups have the following Fill Level properties:

#### **Level expression**

The value of the expression entered in this field (128 characters maximum) will determine the fill level of the object/group. By default, when the expression returns its minimum value, the object/group will be filled to the **At minimum** level. When the expression returns its maximum value, the object/group will be filled to the **At maximum** level. When the expression returns a value half-way between its minimum and maximum, the object/group will be filled to half-way between the **At minimum** and **At maximum** levels.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### **[Level expression] Specify range**

Select this box to manually specify **Minimum** and **Maximum** values for the Level expression, rather than using the default values. For an expression containing an analog variable tag, the defaults are the Engineering Zero and Full Scale values from the last variable tag in the expression. If the analog variable tag does not have Engineering Zero and Full Scale values, the defaults are 0 (zero) and 32000. For expressions without tags, the defaults are 0 (zero) and 100.

#### **[Level expression] Minimum**

Enter the minimum value for the expression. When this value is returned by the expression, the object/group will fill to the **At minimum** level. You can only enter a value here if you have selected the **Specify** range box.

#### **[Level expression] Maximum**

Enter the maximum value for the expression. When this value is returned by the expression, the object/group will fill to the **At maximum** level. You can only enter a value here if you have selected the **Specify** range box.

#### **At minimum**

The level to which the object/group will be filled when the **Level expression** returns its minimum value. For example, if you enter 30, the object/group will be 30% full when the expression returns its minimum value.

You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field.

#### **At maximum**

The level to which the object/group will be filled when the **Level expression** returns its maximum value. For example, if you enter 90, the object/group will be 90% full when the expression returns its maximum value.

You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field.

#### **Fill Direction**

The direction in which the color will spread when increasing. There are four options (each represented by an arrow): **Up, Down, Left, Right**. If you choose Up, the object/group will be filled from the bottom up. If you choose Left, the object/group will be filled from right to left, and so on

#### **Background color**

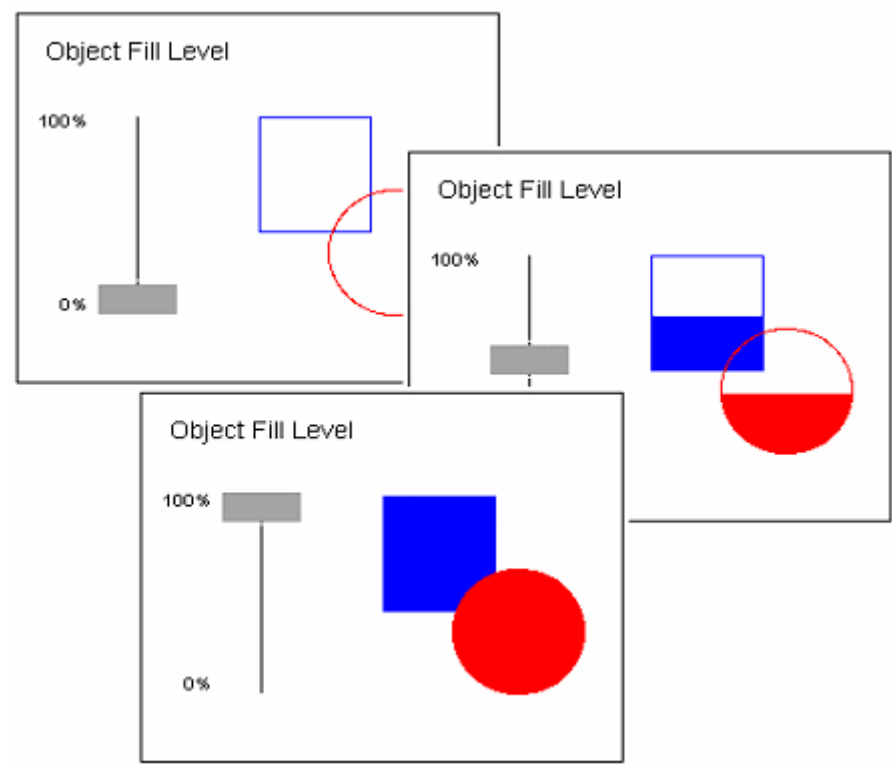
The color of any unfilled part of the object/group (for example, if the object/group is only 90% full, the unfilled 10% will be display using this color). The background is often made transparent. Using transparent, you would see the outline of the object/group, and anything behind the object/group on the page.

**Note:** If an object in a group is a slider, it might change the overall size of the group when used at runtime. If it does, the fill level of the group will adjust accordingly.

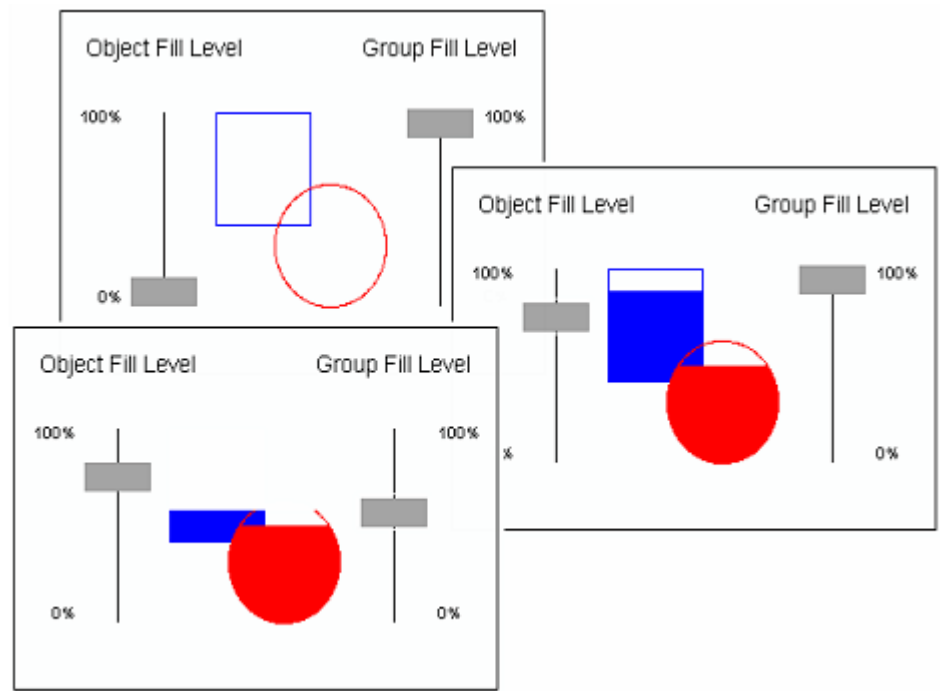
#### **Group and Object Fill Level: Examples**

A group and its objects can be configured with different fill levels. The group fill level, however, is best thought of as a reveal of the objects in the group. Group fill level and object fill level operate independently of each other; the group fill level just determines how much of the objects display.

**Example 1:** The fill level of the objects:



**Example 2:** Group the objects and configure a fill level for the group as well



In this example, the objects' fill levels can still be adjusted normally. The group's fill level determines how much of the objects you can see (and how much will be obscured by the groups background color; white, in this case).

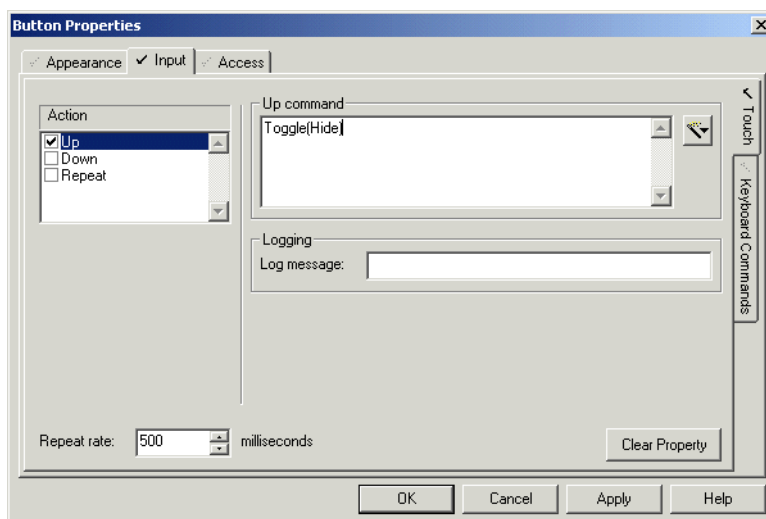
## Touch Commands

You can assign touch commands to an object or group.

**To assign a touch command to an object or group:**

- 1 Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you must double-click the group.)
- 2 Click the **Input** tab.
- 3 Click the **Touch** tab (to the right of the dialog).
- 4 Enter a command in the command field (the command that will be executed when the object/group is touched at runtime).
- 5 Enter further details as required, using the **Help** button for detailed information about each field.

## 6 Click OK.



See Also [Object Properties - Input \(Touch\)](#)  
[Defining Common Object Properties](#)

### Object Properties - Input (Touch)

The touch property lets you assign commands to the object/group. These commands are then executed when the object/group is touched at runtime (i.e., an operator clicks on the object/group). You can also define messages which will log at these times.

For example, a drive can be jogged by starting it when the mouse button is depressed and stopping it when the mouse button released; variables can be incremented while the mouse button is held, and so on. At the same time, it could log the time and date, and the name of the operator.

Operators who do not satisfy the access requirements cannot touch the object/group at runtime.

Objects and groups have the following input (touch) properties.

#### Action

There are three actions to which commands can be attached. You can select more than one type of action. Unique commands and log messages can be attached to each action (i.e. you can perform one task on the down action, and another on the up action, and log a separate message for each).

#### [Action] Up

Select this option if you want a command to be executed (and a unique message to be logged) when the operator positions the mouse pointer over the object/group, and clicks *and releases* the left mouse button.

As with standard Windows buttons, if the operator moves the cursor away from the object/group before releasing the mouse button, the command isn't executed (unless you also select the **Down** option).

#### [Action] Down

Select this option if you want a command to be executed (and a unique message to be logged) when the operator positions the mouse pointer over the object/group, and clicks the left mouse button. The command will execute as soon as the mouse button is clicked.

#### [Action] Repeat

Select this option if you want a command to execute continually (and a unique message to log continually) whenever the operator has the mouse pointer positioned over the object/group, and is holding the left mouse button depressed. If the operator moves the mouse pointer away from the object/group without releasing the mouse button, the command will stop executing, but will start again as soon as the mouse pointer is re-positioned over the object/group. The only exception is when you also have the Down option selected, in which case, the command will execute continually even if the mouse pointer has been moved away from the object/group.

To set the delay which precedes the first execution of the command (and the first log of the message), and the delay between each repeat.

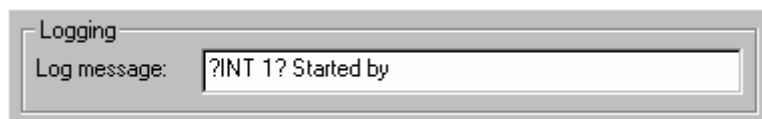
#### Up/Down/Repeat command

The command(s) (set of instructions) to be executed immediately when the selected action is performed. The command(s) can be a maximum of 254 characters long.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### [Logging] Log Message

The text message sent to the *MsgLog* field of the Log Device when the selected action is performed by the operator at runtime. The message is plain text, and can include tag name substitutions using [Genie](#) or Super Genie syntax. When using Super Genie syntax, the data type must be specified. The name of the tag will then be included in the text. The log message can be a maximum of 32 characters long.



The screenshot shows a window titled "Logging". Inside, there is a label "Log message:" followed by a text input field. The text input field contains the text "?INT 1? Started by".

To include field data as part of a logged message, insert the field name as part of the device format when configuring the device. For instance, in the **Format** field



of the **Devices** form, you could enter {MsgLog,20} {FullName,15}. This would accommodate the logging of messages such as **P2 started by John Smith**.

The log device to which the message is sent is specified through the General Access tab.

**Note:** If the object is part of a Genie or symbol, this property can be defined after the Genie/symbol is pasted onto a page. (Hold down the **Control** (CTRL) key and double-click the object.) If you define it before pasting (i.e. you define it for the original in the library), you cannot edit it after. Similarly, if the object is part of a template, it can be defined after a page has been created using that template (again, with Control + double-click). If you define it for the actual template, you cannot edit it for pages based on the template.

### Repeat rate

This option sets the delay which precedes the first execution of the command(s), and the delay between each subsequent repeat of the command(s).

You can change the rate by pressing the up and down arrows to the right of the field, or by entering another value in this field.

**Note:** If you define a touch command for an object in a group, the group's touch command will not work.

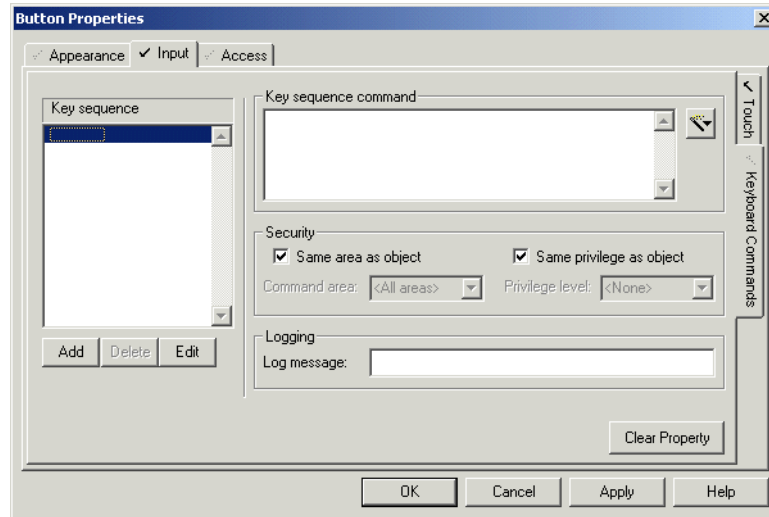
## Keyboard Commands

You can assign a [keyboard command](#) to an object or group.

**To assign a keyboard command to an object or group:**

- 1 Draw the object/group (or paste a symbol). The object properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you must double-click the group.)
- 2 Click the **Input** tab.
- 3 Click the **Keyboard Commands** tab (to the right of the dialog).
- 4 Enter the key sequence.
- 5 Enter a command in the command field (the command that will be executed when the key sequence above is entered by the operator at runtime).
- 6 Enter additional details as required.

## 7 Click OK.



See Also [Object Properties - Input \(Keyboard Commands\)](#)  
[Defining Common Object Properties](#)

### Object Properties - Input (Keyboard Commands)

The keyboard commands property lets you assign keyboard commands to the object/group. A keyboard command is a particular key sequence which executes a command when it is typed in by the operator at runtime. To execute an object/group keyboard command, the operator positions the cursor over the object/group and enters the key sequence using the keyboard.

You can also define a message which will log every time the key sequence is entered.

For example, the operator could change the water level in a tank by placing the mouse over the symbol representing the tank, and typing in the new level. At the same time, a message could be logged, listing the time and date, and the name of the operator.

Operators who do not satisfy the access requirements specified under **Security** below cannot enter keyboard commands for this object/group at runtime.

Objects and groups have the following keyboard commands input properties:

#### Key sequence

Enter the key sequences that the operator can enter to execute a command. For example, you might define the key sequence **### Enter**. During runtime, this key sequence would allow you to type in any three digit number, and click **Enter** to change variable tag values from mimic pages and so on

You can enter as many key sequences as you like. To add a key sequence, click **Add** and type in the sequence or select one from the menu. To edit an existing sequence, click the relevant line and click **Edit**. You can also remove key sequences by clicking **Delete**.

#### **Key sequence command**

The commands( set of instructions) to be executed immediately when the selected key sequence is entered. The commands can be a maximum of 254 characters long.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

#### **[Security] Same area as object/group**

Select this box to assign the keyboard command to the same area as the object/group. Only users with access to this [area](#) (and any required privileges) will be able to issue this command or log the message. If you want to assign this keyboard command to another area, do not select this box; instead, enter another area below.

#### **[Security] Command area**

Enter the area to which this keyboard command belongs. Only users with access to this area (and any required privileges) will be able to issue this command or log the message. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to issue this command.

Click the menu to the right of this field to select an area, or type in an area number directly.

**Note:** If the object is part of a Genie or symbol, this property can be defined after the Genie/symbol is pasted onto a page (**Ctrl** + double-click). Similarly, if the object is part of a template, this property can be defined after a page has been created using that template (**Ctrl** + double-click).

You can leave this field blank by selecting the **Same privilege as object/group** box.

#### **[Security] Same privilege as object/group**

Select this box to assign the keyboard command the same privilege as the object/group. Only users with this privilege level will be able to issue this command, or log the message. If you want to assign this keyboard command a different privilege, do not select this box; instead, enter another privilege below.

#### **[Security] Privilege level**

Enter the privilege level that a user must possess to be able to issue this command or log the message. For example, if you enter Privilege Level 1 here, operators must possess Privilege Level 1 to be able to issue this command. You

can also combine this restriction with area restrictions. For example, if you assign the keyboard command to Area 5, with Privilege Level 2, the user must be set up with Privilege 2 for Area 5.

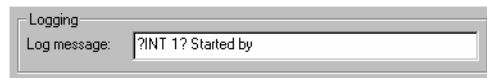
Click the menu to the right of this field to select a privilege, or type in an area number directly.

**Note:** If the object is part of a Genie or symbol, this property can be defined after the Genie/symbol is pasted onto a page (**Ctrl** + double-click). Similarly, if the object is part of a template, this property can be defined after a page has been created using that template (**Ctrl** + double-click).

You can leave this field blank by selecting the **Same privilege as object/group** box.

### [Logging] Log Message

The text message sent to the *MsgLog* field of the Log Device when the selected action is performed by the operator at runtime. The message is plain text, and can include tag name substitutions using Genie or Super Genie syntax. When using Super Genie syntax, the data type must be specified. The name of the tag will then be included in the text. The message can be a maximum of 32 characters long.



If you want to include field data as part of a logged message, you must insert the field name as part of the device format when you configure the device. For instance, in the **Format** field of the **Devices** form, you could enter {MsgLog,20}{FullName,15}. This would accommodate the logging of messages such as **P2 started by John Smith**.

The log device to which the message is sent is specified through the General Access tab.

**Note:** If a group and one of its objects are both assigned a keyboard command with the same key sequence, the object's command will take precedence (i.e., the group's command will not execute).

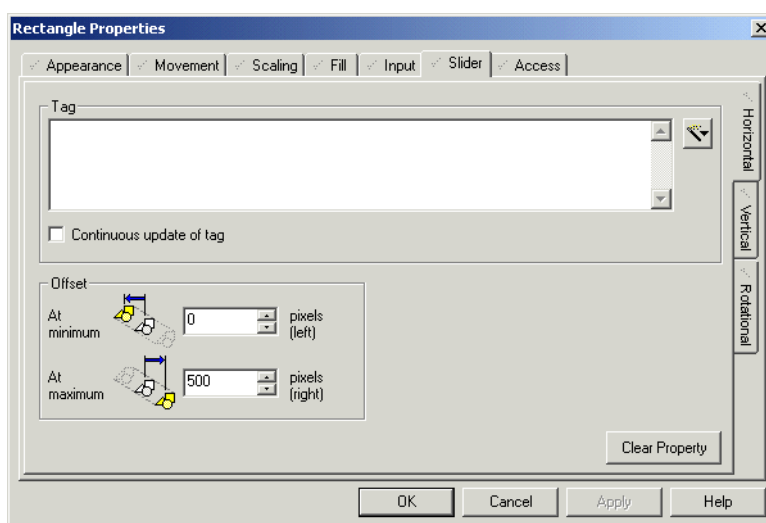
## Sliders

You can create sliders to have on your graphics pages.

### To configure a slider:

- 1 Draw the object/group (or paste a symbol). The properties tab dialog will automatically display, unless you have turned off the **Display properties on new** option in the Graphics Builder. (For a group, the properties dialog will not display automatically; you must double-click the group.)

- 2 Click the **Slider** tab.
- 3 Click the **Horizontal**, **Vertical** or **Rotational** tab (to the right of the dialog).
- 4 Enter the **Tag** to link to the slider.
- 5 Enter any additional details.
- 6 Click **OK**.



See Also [Object Properties - Slider \(Horizontal\)](#)  
[Object Properties - Slider \(Vertical\)](#)  
[Object Properties - Slider \(Rotational\)](#)

## Object Properties - Slider (Horizontal)

Objects and groups can be linked to variable tags in such a way that horizontal sliding of the object/group changes the value of the tag. As the slider moves to the right, the variable tag increases in value. As the slider moves to the left, the variable tag decreases in value. The slider also moves automatically to reflect the changing values of the tag.

**Note:** The horizontal slider cannot be used if the rotational slider is enabled, or if horizontal movement is enabled.

Objects and groups have the following horizontal slider properties.

### Tag

The value of the tag entered in this field (79 characters maximum) will change when the slider is moved left or right. You can define two slider limits on your graphics page. The object/group will not slide beyond these two points. During runtime, when the slider reaches its left-hand limit (**Offset At minimum**), the tag value changes to its minimum limit. When the slider reaches its right-hand limit (**Offset At maximum**), the tag value changes to its maximum limit.

To insert a tag, click the **Wizard** button to the right of this field.

**[Tag] Continuous update of tag**

Select this box if you want the variable tag to be updated continuously while the slider is being moved. If you do not select this box, the tag will only be updated when the slider has been released (i.e., it has been moved, and the operator has released the mouse button).

**[Offset] At minimum**

The distance (number of pixels from the original object/group center) that the object/group can slide to the left. When it reaches the point defined by this distance, the tag value changes to its minimum limit.

You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

**[Offset] At maximum**

The distance (number of pixels from the original object/group center) that the object/group can slide to the right. When it reaches the point defined by this distance, the tag value changes to its maximum limit.

You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

You can increase the value of the tag with a left-slide, and decrease it with a right-slide, by entering negative distances in the Offset fields.

**Note:** If an object in a group is a slider, it might change the overall size of the group when used at runtime. If it does, the fill level of the group will adjust accordingly. If a group and one of its objects are both defined as sliders, and they slide in the same direction or one is rotational, the object will take precedence (i.e. only the object will operate as a slider).

See Also

[Object Properties - Slider \(Vertical\)](#)

[Object Properties - Slider \(Rotational\)](#)

## Object Properties - Slider (Vertical)

You can link objects and groups to variable tags in such a way that vertical sliding of the object/group changes the value of the tag. As the slider moves to the up, the variable tag increases in value. As the slider moves to the down, the variable tag decreases in value. The slider also moves automatically to reflect the changing values of the tag.

**Note:** The vertical slider cannot be used if the rotational slider is enabled, or if vertical movement is enabled.

Objects and groups have the following vertical slider properties:

## Tag

The value of the tag entered in this field (79 characters maximum) will change when the slider is moved up or down. You can define two slider limits on your graphics page. The object/group will not slide beyond these two points. During runtime, when the slider reaches its upper limit (**Offset At maximum**), the tag value changes to its maximum limit. When the slider reaches its lower limit (**Offset At minimum**), the tag value changes to its minimum limit.

To insert a tag, click the **Wizard** button to the right of this field.

### [Tag] Continuous update of tag

Select this box if you want the variable tag to be updated continuously while the slider is being moved. If you do not select this box, the tag will only be updated when the slider has been released (i.e., it has been moved, and the operator has released the mouse button).

### [Offset] At maximum

The distance (number of pixels from the original object/group center) that the object/group can slide up. When it reaches the point defined by this distance, the **Tag** value changes to its maximum limit.

You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

### [Offset] At minimum

The distance (number of pixels from the original object/group center) that the object/group can slide down. When it reaches the point defined by this distance, the tag value changes to its minimum limit.

You can change the offset value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

You can increase the value of the tag with a down-slide, and decrease it with an up-slide, by entering negative distances in the Offset fields.

**Note:** If an object in a group is a slider, it might change the overall size of the group when used at runtime. If it does, the fill level of the group will adjust accordingly. If a group and one of its objects are both defined as sliders, the object will take precedence (i.e. only the object will operate as a slider).

See Also

[Object Properties - Slider \(Horizontal\)](#)  
[Object Properties - Slider \(Rotational\)](#)

## Object Properties - Slider (Rotational)

Objects and groups can be linked to variable tags in such a way that rotational sliding of the object/group changes the value of the tag. As the slider rotates clockwise, the variable tag increases in value. As the slider rotates anti-clockwise, the variable tag decreases in value. The slider also moves automatically to reflect the changing values of the tag.

**Note:** The rotational slider cannot be used if either of the other sliders is enabled, or if rotational movement is enabled.

Objects and groups have the following rotational slider properties.

### **Tag**

The value of the tag entered in this field (79 characters maximum) will change as the slider is rotated. You can define two slider limits on your graphics page. The object/group will not rotate beyond these two points. During runtime, when the slider reaches its anti-clockwise limit (**Offset At minimum**), the tag value changes to its minimum limit. When the slider reaches its clockwise limit (**Offset At maximum**), the tag value changes to its maximum limit.

To insert a tag, click the **Wizard** button to the right of this field.

### **[Tag] Continuous update of tag**

Select this box if you want the variable tag to be updated continuously while the slider is being moved. If you do not select this box, the tag will only be updated when the slider has been released (i.e. it has been moved, and the operator has released the mouse button).

### **[Angle] At minimum**

Enter an anti-clockwise angle (in degrees relative to 0°). The slider cannot rotate anti-clockwise beyond this limit. When it reaches this limit, the **Tag** value changes to its minimum limit.

You can change the angle value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

### **[Angle] At maximum**

Enter an clockwise angle (in degrees relative to 0°). The slider cannot rotate clockwise beyond this limit. When it reaches this limit, the **Tag** value changes to its maximum limit.

You can change the angle value by pressing the up and down arrows to the right of the field, or by entering another value in this field.

**Note:** You can increase the value of the tag with an anti-clockwise slide, and decrease it with a clockwise-slide, by entering negative distances in the Angle fields.

### **[Center axis offset] Express**

Click this radio button for the quick and easy way of selecting the point about which the object/group will rotate. The express option gives you the choice of 9 points (Top Left, Bottom Right and so on), which are displayed in the picture field on the dialog. To select one, just click it with your mouse.



### [Center axis offset] Custom

Click this radio button to define your own center axis. When you select this radio button, two fields will display to the right, allowing you to plot the position of your center axis. Specify the distance to the right in the first field, and the distance down in the second. The Center axis is plotted based on these two values.

For example, if you enter 8 as the horizontal offset and 13 as the vertical, the center axis will be 8 pixels to the right and 13 pixels below the center of the object/group.

Enter negative values in the offset distance fields to move the center axis left instead of right, and up instead of down.

**Note:** If a group and one of its objects are both defined as sliders, the object will take precedence (i.e., only the object will operate as a slider).

See Also [Object Properties - Slider \(Horizontal\)](#)  
[Object Properties - Slider \(Vertical\)](#)

## Access

You can determine the kind of access you want to have to your objects.

- [General Access to Objects](#)
- [Disable Access to Objects](#)

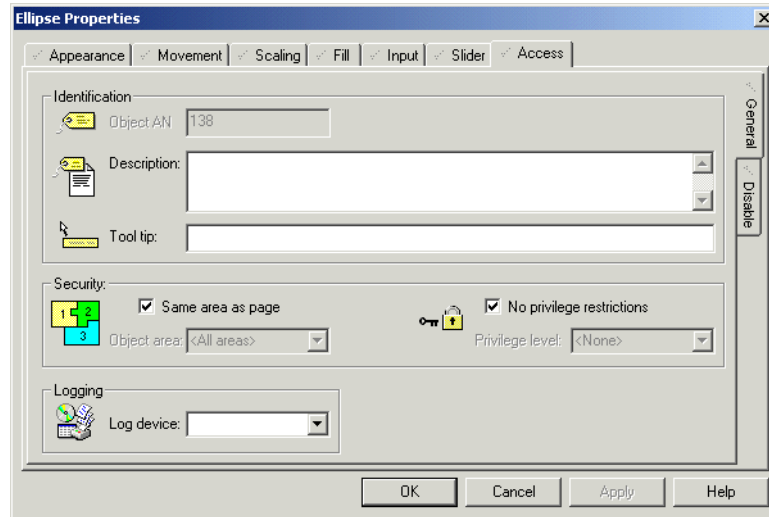
### General Access to Objects

Use the General tab to define the general access characteristics of objects.

**To define general access properties for objects:**

- 1 Double-click the object.
- 2 Click the **Access** tab.
- 3 Click the **General** tab.

- 4 Enter details as required, then click **OK**.



See Also [Object Properties - Access \(General\)](#)  
[Defining Common Object Properties](#)

## Object Properties - Access (General)

Use the Object Properties dialog to set up general identification, security, logging, and privilege parameters for your ActiveX object.

Objects and groups have the following general access properties.

### [Identification] Object/Group AN

Displays the Animation number of the object/group. The AN uniquely identifies the object/group, and can be used in Cicode functions and so on.

**Note:** If the object is part of a Genie or symbol, the following properties can be completed after the Genie/Symbol is pasted onto a page (**Ctrl** + double-click). Similarly, if the object is part of a template, the properties can be defined after a page has been created using that template (**Ctrl** + double-click).

### [Identification] Description

Enter a description of the object/group, and its various functions and so on. This field is purely for the entry of information which you consider beneficial to the smooth running and maintenance of your system. It will not affect the way the system runs, and it will not display during runtime.

### [Identification] Tool tip

Enter a short, meaningful description (48 characters maximum) of the object/group. During runtime, this description appears when the operator moves the cursor onto the object/group. The message can be plain text, Super Genie syntax

or both. When using Super Genie syntax, the data type must be included. The name of the tag will then be included in the text.

If an object in a group has a tool tip, this tool tip will always be displayed, not the group's tool tip. If the object does not have a tool tip, the group tool tip will display. In this instance, if the object is a member of a group, and that group is part of another group, the tool tip for the first group will display.

If you place an object behind a group, its tool tip will not display. Remember, however, that group boundaries are rectangular, no matter what shape is formed by the objects in the group. This means that 'blank' spaces between objects in a group are actually part of the group. Even if you can see the individual object, if it is behind the group, its tool tip will not display.

#### **[Security] Same area as page**

Select this box to assign the object/group to the same area as the page on which it has been drawn; otherwise, leave it blank, and enter another area in the **Object/Group area** field (below).

#### **[Security] Object/Group area**

Enter the area to which this object/group belongs. Users without access to this [area](#) (and any required privileges) will not be able to make full use of the object/group. They will not be able to use touch command, keyboard commands, movement, sliders and so on. (In order to avoid confusion for such operators, it is sometimes a good idea to disable the object/group when it is unavailable due to lack of security rights. Disabled objects/groups can be grayed, hidden or embossed.) For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to make use of this object/group.

Click the menu to the right of this field to select an area, or type in an area number directly.

#### **[Security] No privilege restrictions**

Select this box to disable privilege restrictions; otherwise, leave it blank, and enter another privilege below. The implications of not assigning a privilege restriction depend upon whether you have used areas in your security setup:

- **No Areas:** All operators have full control of the object/group.
- **Areas:** An operator will only need view access to control the object/group if it does not have privilege restrictions.

#### **[Security] Privilege level**

Enter the privilege level required for an operator to use this object/group. Operators without the required privileges will not be able to make full use of the object/group. They will not be able to use touch command, keyboard commands, movement, sliders and so on. (To avoid confusion for such operators, disable the object/group when it is unavailable due to lack of security

rights. Disabled objects/groups can be grayed, hidden or embossed.) For example, if you enter Privilege Level 1 here, operators must possess Privilege Level 1 to use of this object/group. You can also combine this restriction with area restrictions. For example, if you assign the object/group to Area 5, with Privilege Level 2, the user must have Privilege 2 for Area 5.

Click the menu to the right of this field to select a privilege, or type in an privilege number directly.

**Note:** If an object is part of a group, users must have access to the group in order to have access to the object.

#### **[Logging] Log device**

This is the device to which messages will be logged for the object/group's keyboard and touch commands. Click the menu to the right of the field to select a device, or type a device name.

**Note:** You must include the *MsgLog* field in the format of the log device for the message to be sent.

See Also [Disable Access to Objects](#)

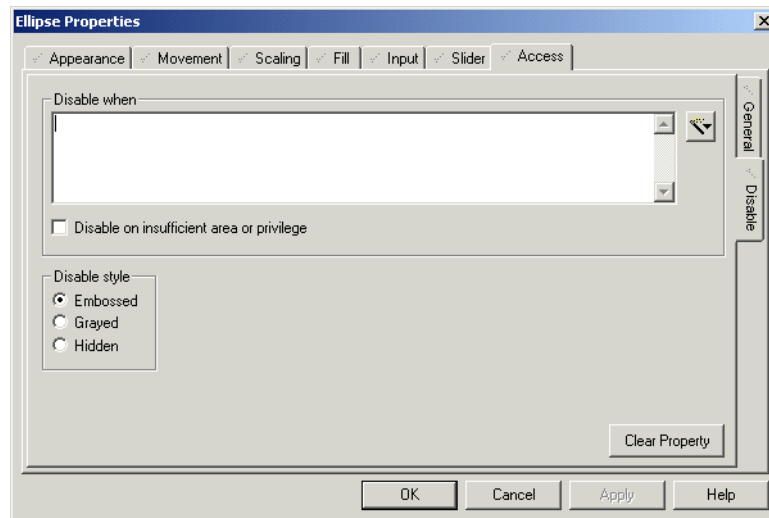
## Disable Access to Objects

If you need to, you can disable access to objects.

**To disable access to an object:**

- 1 Double-click the object.
- 2 Click the **Access** tab.
- 3 Click the **Disable** tab.
- 4 Specify the condition which will disable the object as well as the appearance of the object when disabled.

5 Click **OK**.



See Also [Object Properties - Access \(Disable\)](#)  
[Defining Common Object Properties](#)

## Object Properties - Access (Disable)

Objects and groups have the following access (disable) properties

### Disable when

The object/group will be disabled whenever the expression entered here (128 characters maximum) is true. If the object/group is disabled, the operator will not be able to use any form of input, such as sliders, touch commands, keyboard commands and so on.

To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: **Insert Tag** and **Insert Function**.

There are three ways of indicating a disabled object/group: Embossed, Grayed, and Hidden.

### [Disable when] Disable on insufficient area or privilege

The object/group will be disabled for operators whose area and privilege rights do not satisfy the requirements defined in the Access.

### Disable style

**Embossed:** When disabled, the object/group will look as if it has been embossed on the graphics page.

- **Grayed:** When disabled, the object/group will be grayed out (all color detail will be removed).
- **Hidden:** When disabled, the object/group will be entirely hidden from view.

**Note:** If a group is disabled, all objects in that group are also disabled.

# Chapter 21: Defining Commands and Controls

---

## Defining Commands and Controls

Commands allow your operators to interact with the CitectSCADA runtime system. You can define three types of direct command controls:

- [Touch commands](#) that your operators issue by clicking specific graphics object (displayed on a [graphics page](#)).
- [Keyboard commands](#) that your operators issue by typing instructions on the keyboard.
- [Slider controls](#) that your operators use to change the values of analog variables.

You can assign privileges to all commands and controls, and send a message to the command log each time an operator issues a command.

### Touch commands

You can assign Touch commands to the objects that you create on your graphics pages. Touch commands allow the operator to send commands to the runtime system, by clicking (with the mouse or similar) on an object on the graphics page. (For buttons, the command can be executed by highlighting the button with the cursor keys on the keyboard and pressing Enter.)

You can define several commands for an object, one command to execute when the operator clicks on the object, another for when the operator releases the mouse button, and another to operate continuously while the operator holds the mouse button down. For example, a drive can be jogged by starting it when the mouse button is depressed and stopping it when the mouse button released; variables can be incremented while the mouse button is held, and so on.

**Note:** You can define a disable condition for any object on a page (including buttons). When the condition is active, the object is grayed and the operator cannot select it. You can also associate a tool tip (Help text) with an object; if the operator holds the mouse pointer over the object, the tool tip will display in a pop-up window.

See Also [Touch Commands](#)

### Keyboard commands

Keyboard commands consist of a key sequence that an operator enters on the keyboard, and an instruction (or series of instructions) that executes when the key sequence is entered.

You can define keyboard commands that operate:

- For any graphics page displayed on the computer screen (system keyboard commands).
- Only when a specific graphics page is displayed (page-keyboard commands).
- Only when an operator positions the mouse pointer on an object on a graphics page. You can associate tool tips with any object; if the operator holds the mouse pointer over the object, the tool tip displays in a pop-up box.

Object keyboard commands have precedence over page keyboard commands (which have precedence over system (global) keyboard commands). If you define a keyboard command for an object that is identical to a page keyboard command, the object keyboard command executes when key sequence is entered, but the page keyboard command is ignored.

See Also [Keyboard Commands](#)  
[System Keyboard Commands](#)  
[Keyboard Keys](#)  
[Keyboards](#)  
[Defining Key Sequences for Commands](#)

## Slider controls

Slider controls allow an operator to change the value of an analog variable by dragging an object on the graphics page. Sliders also move automatically to reflect the value of the variable tag.

See Also [Sliders](#)

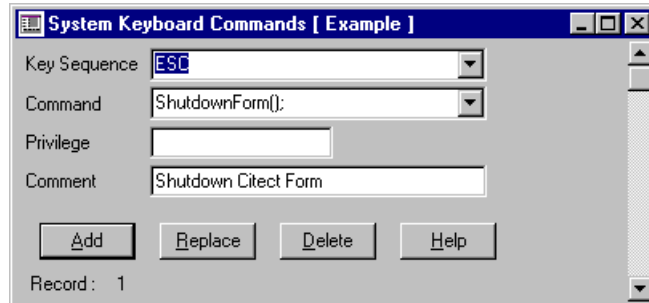
## System Keyboard Commands

You can define system keyboard commands.

**To configure a system keyboard command:**

- 1 If you plan to use any special keys, you must first define your keys.
- 2 In the Project Editor, choose **System | Keyboard Commands**.
- 3 Complete the properties in the **System Keyboard Commands** form that appears. You need to enter a key sequence and a command.
- 4 Click **Add** to append a record you have created, or **Replace** to modify a record.





See Also [System keyboard command properties](#)  
[Defining Commands and Controls](#)

## System keyboard command properties

System keyboard commands have the following properties:

### **Key Sequence (32 Chars.)**

The key sequence for the command.

### **Command (64 Chars.)**

The commands (set of instructions) to execute when an operator enters the key sequence.

### **Privilege (16 Chars.)**

The privilege required by an operator to issue the command.

### **Comment (48 Chars.)**

Any useful comment.

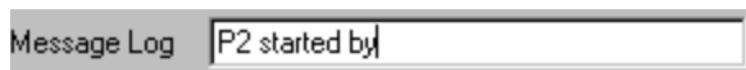
**Note:** The following fields are implemented with extended forms (press **F2**).

### **Area (16 Chars.)**

The area to which the command belongs. Only operators who belong to this [area](#) will be able to issue this command. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to issue this command.

### **Message Log (32 Chars.)**

A text message sent to the *MsgLog* field of the Log Device when the selected action is performed by the operator at runtime. The message must be plain text:



If you want to include field data as part of a logged message, you must insert the field name as part of the device format when you configure the device. For

instance, in the **Format** field of the **Devices** form, you could enter {MsgLog,20}{FullName,15}. This would accommodate the logging of messages such as **P2 started by John Smith**.

The log device to which the message is sent is specified in the Log Device field below.

#### Log Device (16 Chars.)

The device to which the **Message Log** is sent when the command is issued.

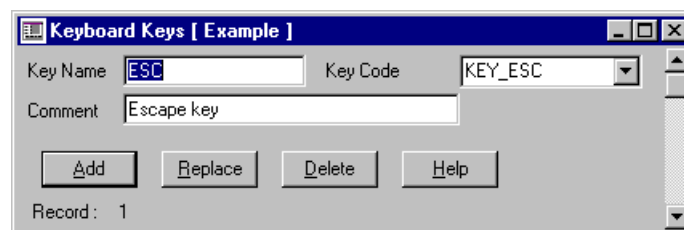
**Note:** You must include the *MsgLog* field in the format of the log device for the message to be sent.

## Keyboard Keys

You can define keyboard keys to make it easier to issue commands.

To define a keyboard key:

- 1 Choose **System | Keyboard Keys**. The Keyboard Keys dialog box appears.
- 2 Enter the **Key Name** and **Key Code** (and **Comment** if applicable).
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.



See Also [Keyboard keys properties](#)  
[Defining Commands and Controls](#)

### Keyboard keys properties

Keyboard Keys have the following properties:

#### Key Name

The name assigned to the key. Enter a value of 16 characters or less. You can define a meaningful name for any key on your keyboard, and use these keys in any keyboard command. For example, you can define the F1 key as the Login key. When the Login key is subsequently referenced in the CitectSCADA system, it refers to the F1 key.

You can assign a key name to any key on the keyboard (including the alphanumeric keys A-Z, a-z, and 0-9). However, after a key is defined as a command key it can only be used as a command key. If you do assign a definition to an alphanumeric key (for example the character A), then that key

can never be used as a data key. Each time it is pressed, CitectSCADA recognizes the definition for the key and not the keyboard character itself. Keyboard key definitions are usually only used with non-alphanumeric characters (e.g. the function keys).

### Key Code

The code assigned to the key name. Enter a value of 16 characters or less. The Key Code is what links your Key Name to the actual key. You can specify the key code either as a hexadecimal value or use the standard CitectSCADA label already associated with the key.

### Comment

Any useful comment. Enter a value of 48 characters or less.

**Note:** The following fields are implemented with extended forms (press F2).

### Echo

Determines if the key is echoed on the screen (when the key is used). Enter a value of 8 characters or less.

Echo                      TRUE

(Display (echo) the **Key Name** when the key is pressed. The key name is displayed on the graphics page in the keyboard entry line - AN1)

Echo                      FALSE

(Do not display (echo) the **Key Name** when the key is pressed)

This property is optional. If you do not specify Echo, Echo defaults to True.

### Keyboard Type (16 Chars.)

The type of keyboard. This field is only required if you have more than one type of keyboard on the same CitectSCADA system.

If you do have more than one type of keyboard, use Keyboard Type 1 for your first type of keyboard, use a separate number for each type (1, 2, 3, etc.), and define all keys for each keyboard. You can use the default (Type 0) for all common keys.

## Keyboards

You can use non-standard keyboards (as well as multiple keyboards) to control CitectSCADA. You can also define key names.

### Using non-standard keyboards

You can use many types of keyboards with your runtime system. The most common keyboards are IBM compatible keyboards; CitectSCADA uses this type

of keyboard as a default. Many industrial keyboards do not conform with this standard; if you use a non-standard keyboard, you must define each of the keyboards in the database.

### Using multiple keyboards

Sometimes you might need to use keyboards of different types; for example, an IBM compatible keyboard in your control room and a sealed-membrane keyboard on the plant floor. If you use more than one type of keyboard, you must define all keys for each keyboard and assign each keyboard type to the respective computer.

You must set the keyboard type for each CitectSCADA computer, with the [Keyboard]Type parameter.

**Note:** If you define commands that use mouse buttons, ensure that a double-click command cannot be mistaken for a single-click command. A double-click is an extension of a single click; a single click message is always received before a double-click message.

### Defining key names

You can refer to a keyboard key by a meaningful name, rather than by the key itself. For example, you can refer to the F1 key as the “Help” key and the F2 key as the “Login” key. When you use the key in a command, you can use the name you have defined.

You can assign a key name to any key on the keyboard (including the alphanumeric keys A - Z, a -z, and 0 - 9). However, after a key is defined as a command key it can only be used as a command key. If you do assign a definition to an alphanumeric key (for example the character A), that key can never be used as a data key. Each time it is pressed, CitectSCADA recognizes the definition for the key and not the keyboard character itself. Keyboard key definitions are usually only used with non-alphanumeric characters (e.g. the function keys).

See Also [Defining Key Sequences for Commands](#)  
[Defining Commands and Controls](#)

## Defining Key Sequences for Commands

To define a command, you must specify the key sequence that the operator types to issue the command. You can specify a single key for the key sequence, for example, the function key F2:

Key Sequence	F2
--------------	----

Alternatively, you can specify several keys that must be typed in sequence, for example, the function key F2 followed by the Enter key:

Key Sequence	F2 Enter
--------------	----------

**Note:** If you use more than one key for the sequence, you must separate each key with a space.

You must prevent the ambiguity in keyboard commands that can occur if you define separate commands that all use a common key. For example, if you define a key sequence for one command as F3, and the key sequence for a second command as F3 F4, then when F3 is pressed, the first command would execute immediately; the second command could never execute. To prevent this conflict, add a **delimiter** to common keyboard commands, for example:

Key Sequence	F3 Enter
Command	SP1 = 50;

Key Sequence	F3 F4 Enter
Command	SP1 = 100;

These commands do not execute until the operator types the delimiter (the Enter key).

See Also [Using a hot key](#)  
[Using variable data input](#)  
[Passing multiple arguments](#)  
[Passing keyboard arguments to functions](#)

## Using a hot key

A command defined with a 'hot' key executes immediately the key is pressed. You can only define a single key in the key sequence, and you should only use a 'hot' key to define commands that act on the current keyboard buffer (for example the Backspace and Delete keys). To define a 'hot' key, prefix the key sequence with an asterisk (\*) as in the following example:

Key Sequence	*Backspace
Command	KeyBs()

At run time, this command operates in exactly the same way as the Backspace key on a standard computer keyboard, to correct typing errors. (Each time the Backspace key is pressed, the last key in the command line is removed.)

**Note:** You can only define a 'hot' key command as a system (global) command.

See Also [Using variable data input](#)

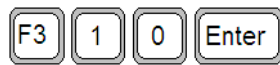
## Using variable data input

You can configure a keyboard command to accept variable data at run time. When the system is running, an operator can enter a value with the command, and the value is passed as an argument (or [arguments](#)) into the Cicode command. You can therefore define a single keyboard command that your

operators use with different values. For example, you could define the following command to set the variable SP1 at run time:

Key Sequence	F3 ### Enter
Command	SP1 = Arg1;

In this example, an operator can set the variable SP1 to a new value by first pressing the F3 function key, entering the new value, and pressing the Enter key, for example:



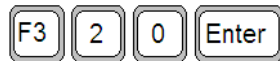
sets the value of SP1 to 10.

Each '#' character (in the Key Sequence) represents one keyboard character that an operator can enter in the command. In the above example, the operator can enter up to three keyboard characters when issuing the command. (The number of # characters determines the maximum number of characters that an operator can enter for the argument; if the operator enters more than three characters, an "Invalid Command" error message displays.)

The command in the above example could be issued as follows:



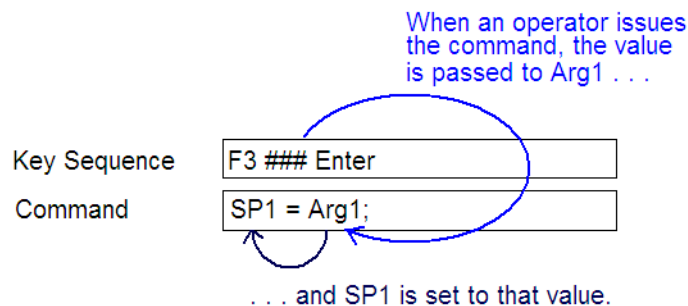
or



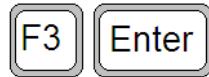
or



When the command is issued (the operator presses the Enter key), the value is passed to the command at Arg1.



**Note:** If an operator does not enter any data (i.e., the key sequence:



is used), the value of Arg1 is zero, and the variable is set to zero. To prevent this from happening, use the **ArgValue1** label, for example:

```
Command SP1 = ArgValue1;
```

The **ArgValue1** label checks for illegal input; if the input is invalid, the value of the variable is not changed. You can also use the StrToValue() function.

Note that the ArgValue1 label and the StrToValue() function halts the command. Any instructions following either the ArgValue1 label or the StrToValue() function do not execute.

See Also [Passing multiple arguments](#)

## Passing multiple arguments

You can pass multiple arguments into a Cicode command by separating each argument with a comma (.). Each argument is passed into the command in sequence, and is referred to in the command field as **Arg1**, **Arg2**, **Arg3**, and so on. For example:

Key Sequence	F3 ###, ### Enter
Command	SP1 = Arg1; SP2 = Arg2;

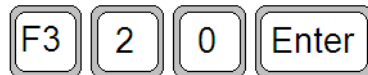
In this case, an operator can set two variables with the same command, for example:



sets the variables SP1 to 20 and SP2 to 35.

You can use up to eight arguments. However, avoid passing many arguments.

**Note:** There is no way to check if the input for each argument is valid. If an operator does not enter any data for one of the arguments (i.e. the key sequence:



is used), the value of Arg2 is zero, and the second variable is set to zero. Do not use multiple arguments in a command if invalid input causes problems.

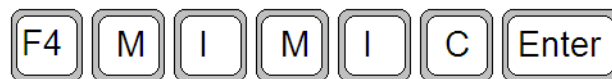
See Also [Passing keyboard arguments to functions](#)

## Passing keyboard arguments to functions

You can also pass arguments directly to functions at run time, as in the following example:

Key Sequence	F4 ##### Enter
Command	PageDisplay(Arg1);

In this example, an operator can select any graphics page (defined in the project) with a single command, for example:



selects the “Mimic” page.

**Note:** All keyboard arguments are passed as string values. If the command (or function) requires a numeric value, Cicode converts the string to a numeric value before it is used.

If you use variable data, the operator can only enter alphanumeric characters (A - Z, a-z, and 0 - 9) for the data. Do not use variable data input as the last item in a key sequence, as ambiguity could occur; always use a delimiter.



# Chapter 22: Configuring and Processing Alarms

---

Protecting your plant equipment is a key role of CitectSCADA. The CitectSCADA alarm facility constantly monitors equipment data and alerts operators of any equipment fault or alarm condition.

CitectSCADA supports two types of alarms:

- **Hardware alarms.** CitectSCADA continually runs diagnostic routines to check all peripheral equipment, such as I/O devices. All faults are reported automatically to the operator. This facility is fully integrated within CitectSCADA; no configuration is necessary.
- **Configured alarms.** Unlike hardware alarms, you must configure the alarms that report fault conditions in your plant (for example, when a tank level is too high or when a motor overheats).

See Also

[Configured alarms](#)  
[Using alarm delay](#)  
[Using custom alarm filters](#)  
[Alarm Categories](#)  
[Formatting an Alarm Display](#)  
[Using Alarm Properties as Tags](#)  
[Handling Alarms at Runtime](#)

## Configured alarms

You can use seven types of configured alarms:

- [Digital Alarms](#)
- [Multi-digital Alarms](#)
- [Time-stamped Alarms](#)
- [Analog Alarms](#)
- [Advanced Alarms](#)
- [Time-stamped Digital Alarms](#)
- [Time-stamped Analog Alarms](#)

You can process each alarm individually, or assign each of your alarms to separate categories, where they can be prioritized. You can then display, acknowledge, reset, and log all alarms in a particular category.

You can also customize the order in which alarms will be displayed on the alarm summary page using the SummarySort and SummarySortMode parameters. (This order will override the alarm category priority order.)

To help operators with alarms, you can create graphics pages that contain information about the alarms (such as the action an operator must perform to correct the situation). You can display these pages automatically when the alarm occurs, or only when an operator uses the Alarm Help [keyboard command](#).

Alarm properties can also be used anywhere a normal variable tag can be used. For example the status of an alarm could be used to change the color of a symbol on a [graphics page](#).

See Also [Using alarm delay](#)  
[Alarm Categories](#)

## Using alarm delay

The Alarm Delay property enables you to configure digital, analog, and advanced alarms so that they do not activate unless their triggering conditions remain true for a specified period.

For analog alarms, this means the analog variable must fall within a specified range for the duration of the alarm delay period before an alarm will activate. In the case of digital and advanced alarms, the triggering condition of the digital variable or Cicode [expression](#) must remain true for the delay period.

An example of where alarm delay could be useful is in monitoring temperature. By setting a delay period, you can filter out momentary alarms caused by the temperature moving in and out of different ranges.

See Also [Using custom alarm filters](#)  
[Alarm Categories](#)

## Using custom alarm filters

CitectSCADA allows you to define keywords for your alarm tags that you can then use to perform customized queries on your alarm data.

The Alarm Properties dialog allows you to define up to eight custom filters for each of the alarms in your system, allowing you to generate queries that filter your alarm data for specific information.

For example, you may want to pay attention to all alarms that represent a potential fire hazard. You could set Custom Filter 1 for each relevant tag to “fire hazard”. You could then call a Cicode function that requests all alarms with a “custom filter 1” field equal to “fire hazard”. The end result would be a list of alarms notifying an operator of any potentially flammable circumstances.

## Implementing queries that use custom alarm filters

You could also set aside Custom Filter 2 to define the type of equipment the alarm is associated with, and label each alarm accordingly (e.g. “pump”, “conveyor”, etc.). Queries could then be created to list all the alarms related to a particular type of machinery; for example, all alarms associated with pumps.

Implementing a query based on the custom alarm filters you have defined requires two steps:

- 1 Create a Cicode function that performs the query you want to implement. The format of the query function you need to create is:

```
INT FUNCTION Query (INT nRecordID, INT nVersion, STRING sInfo)
```

where:

- |           |  |
|-----------|--|
| nRecordID | The value to be used in calls to <a href="#">AlarmGetFieldRec</a> ; e.g. AlarmGetFieldRec(nRID, "CUSTOM1", nVer). (See the help for AlarmGetFieldRec for details on the fields available.) |
| nVersion  | The version of the record; this should increase each time a record is changed.   |
| sInfo     | A user defined string to be used to control the logic in the Query function.   |

CitectSCADA expects a 0 or 1 returned value, with 1 determining that the record will be displayed.

- 2 Set this query by calling it via the function [AlarmSetQuery](#). Make sure you’ve defined the custom filters appropriately for your alarm tags before proceeding.

### Example

You want to create a query that calls current alarms for the conveyors in your plant. This could be drawn from the alarm tags you have identified as being associated with a conveyor, by applying the keyword “conveyor” to the field **Custom Filter 1**.

The query function you would create would be as follows:

```
INT
FUNCTION CheckCustom1(INT nRid, INT nVer, STRING sValue)
STRING sCustom;
// Get the information in CUSTOM1
sCustom = AlarmGetFieldRec(nRid, "CUSTOM1", nVer);
IF sCustom = sValue THEN
// Lets display this
RETURN 1;
ELSE
// Skip over this
RETURN 0;
END
END
```

## Efficiency considerations

Say you then want to create a button on a graphics page that lists all current conveyor alarms. You would implement this by applying the [AlarmSetQuery](#) function to the button.

```
AlarmSetQuery(0, "CheckCustom1", "conveyor");
```

You could also create a reset button that clears the current displayed list by cancelling the query:

```
AlarmSetQuery(-1, "", "")
```

You have the choice of calling a specific Cicode function, for example, "Customfeld1", with the argument "pumpcontrol", or using a more generic approach with the function "Checkfield" with argument "CUSTOM1=pumpcontrol". In this case, the Cicode function parses the passed string and checks the field specified.

Cicode takes longer to implement than CitectSCADA's predefined filters. To avoid unnecessary processing of alarms that have already been checked, smart custom filters are enabled by default to minimize the processing required for large alarm record counts.

Smart custom filters mean that the first time a query is run, each alarm record will be checked by your function. Subsequent screen displays or record changes will then only call the function for changed or new records.

For most queries, this is how you would want your system to behave. However, there may be special queries where you wish to turn this behavior off and check each alarm record.

An example might be if you were to request all changed or new alarms in the last 10 seconds. In this case the "smart" filtering will fail because some records will not have changed, yet they may be included or excluded from your query.

Two mechanisms exist to control this:

- If you set the global ini parameter [ALARM]EnableSmartCustomFilters to 0, it will disable smart filters for all queries.
- On a per query basis, set an optional 4th argument to 1 or TRUE, for example:

```
AlarmSetQuery(0, "MyQueryTime", "10", TRUE);
```

This, when set to TRUE, forces the query to always run. The default value is 0 or FALSE (allow for smart queries).

See Also [Alarm Categories](#)

## Alarm Categories

Each alarm in your system can be assigned to a category, and each category can be processed as a group. For each category, you can set alarm display details (font and page type), logging details (printer or data file), and the action to be taken when an alarm in the category is triggered (e.g., activating an audible alarm).

Each category can have an associated priority. The alarm priorities can be used to order alarm displays, providing useful filtering for the operator.

You can also customize the order in which alarms will be displayed on the alarm summary page using the SummarySort and SummarySortMode parameters. (This order will override the alarm category priority order.)

You can configure up to 16376 alarm categories. If you do not specify a category for an alarm, the alarm has the same attributes as alarm category 0. If you do not define an alarm category 0, CitectSCADA uses a default for the category.

### To define an alarm category:

- 1 Choose **System | Alarm Categories**. The Alarm Categories dialog box appears.
  - 2 Enter the alarm category properties.
  - 3 Click **Add** to append a new record, or **Replace** to modify an existing record.
- Use the Alarm Categories dialog box to configure your alarms.

See Also [Alarm Category Properties](#)

## Alarm Category Properties

[Alarm Categories](#) have the following properties:

### Category Number

The alarm category (0–16375). Enter a category of 16 characters or less.

Category 254 is reserved for user-created alarm summary entries. Category 255 is reserved for hardware alarms.

### Priority

The priority which will apply to all alarms assigned to this alarm category (0–255). Alarm priority governs the order in which alarms are displayed, acknowledged, enabled and so on. Enter a priority of 16 characters or less.

Priority 1 is the highest priority, and priority 255 is the lowest. For example, if alarms with priorities 1 to 8 were displayed, all priority 1 alarms would be displayed first in their time/date order, then priority 2 alarms, then priority 3, and so on up to priority 8.

Priority 0 (zero) is the default priority and all categories have priority zero (as well as the value entered in this field). Priority 0 is used to reference all

priorities. For example, to change the display parameters of an alarm list, so that alarms of all priorities are displayed, **AlarmSetInfo** would be used with *Type* = 7, *Value* = 0.

**Note:** When priority 0 is used to display alarms of all priorities, priority 0 only alarms will display first, followed by priority 1 alarms, then priority 2 etc. You can also customize the order in which alarms will be displayed on the alarm summary page using the SummarySort and SummarySortMode parameters. (This order will override the alarm category priority order.)

#### **Display on Alarm Page**

Defines whether or not alarms of this category will be displayed on the Alarm Page. The default for this field is TRUE. Enter a value of 6 characters or less.

#### **Display on Summary Page**

Defines whether or not alarms of this category will be displayed on the Summary Page. The default for this field is TRUE. Enter a value of 6 characters or less.

#### **Unacknowledged: Alarm Off Font**

The font to display alarms that are no longer active (but have not been acknowledged). This property is optional. If you do not specify a font, the font defaults to 10 pt. BROWN. Enter a value of 16 characters or less.

#### **Acknowledged: Alarm Off Font**

The font used to display alarms that are no longer active and have been acknowledged. This property is optional. If you do not specify a font, the font defaults to 10 pt. WHITE. Enter a value of 16 characters or less.

#### **Unacknowledged: Alarm On Font**

The font used to display an [active alarm](#) that has not been acknowledged. This property is optional. If you do not specify a font, the font defaults to 10 pt. YELLOW. Enter a value of 16 characters or less.

#### **Acknowledged: Alarm On Font**

The font used to display active alarms that have been acknowledged. This property is optional. If you do not specify a font, the font defaults to 10 pt. CYAN. Enter a value of 16 characters or less.

#### **Disabled Font**

The font used to display disabled alarms. This property is optional. If you do not specify a font, the font defaults to 10 pt. WHITE. Enter a value of 16 characters or less.

### ON Action

A Cicode command that is executed when an alarm of this Category is triggered (maximum of 254 characters). For example:

Alarm ON Action                      STOP\_PROCESS = 1;

The digital variable STOP\_PROCESS is set to ON when an alarm in this category is triggered.

Do not put a [Cicode blocking function](#) in this field.

A special case of this command occurs when the Alarm ON Action is self-referring, with a form such as TAG1 = TAG1 + 1. This command will not work properly since CitectSCADA does not reread the tags before processing the Alarm On action (for performance reasons). This particular command will therefore initially set the value of TAG1 to 1 rather than incrementing it.

To correctly run a command of this type in the Alarm ON Action, use TaskNew() to run your own Cicode function to perform the tag command:

Alarm ON Action                      TaskNew("MyFunc","Data",5);

### OFF Action

A Cicode command that is executed when an alarm of this Category is reset (maximum of 254 characters). For example:

Alarm OFF Action                      ENABLE\_PROCESS = 1;

The digital variable ENABLE\_PROCESS is set to ON when an alarm in this category is reset.

**Note:** Do not put a [Cicode blocking function](#) in this field.

### ACK Action

A Cicode command that is executed when an alarm of this Category is acknowledged (maximum of 254 characters).

**Note:** Do not put a blocking function in this field.

### Alarm Format

The screen display format for all alarms in this category (maximum of 120 characters). The Alarm Display format specifies how the data (for all alarms in the category) are displayed on the alarms page (on the screen only). Each alarm displays on the alarms page in a single line, for example:

12:32:21RFP3    Raw Feed pump 3    Overload

The Display Format property is optional. If you do not specify a Alarm Display format, the format defaults to:

Format {Time,12} {Tag,10} {Name,20} {Desc,32}

See [Alarm display fields](#) for the formatting details for each field type.

You can change this default Alarm Display Format for all alarms by setting the [Alarm] DefDspFmt parameter.

**Note:** If an alarm value is longer than the field it is to be displayed in, it will be truncated or replaced with the #OVR (“overflow of format width”) error. When the alarm is logged to a device (i.e. printed or written to a file or database), the format specified for the logging device overrides the display format.

### Summary Format

The summary display format for all alarms in this category (maximum of 120 characters). The Summary Display format specifies how the alarm summary displays on the alarms summary page (on the screen only).

The display format is defined exactly as the Alarms Display Format. However, you can also use additional data fields.

This property is optional. If you do not specify a Summary Display format, the format defaults to:

Format {Name,20} {OnTime,8} {OffTime,8} {DeltaTime,8} {Comment,22}

See [Alarm summary fields](#) for formatting details for each field type.

You can change the default Summary Display Format for all alarms by setting the [Alarm] DefSumFmt parameter.

**Note:** When the alarm is logged to a summary device (i.e. printed or written to a file or database), the format specified for the logging device overrides the display format.

### Summary Device

The device where the alarm summary is sent (maximum of 16 characters).

An alarm is logged to the summary device when it has gone off, and has been displayed for a longer period than the time specified in the [Alarm] SummaryTimeout parameter. When the alarm is logged to the device, it is removed from the alarm summary page.

When the alarm is printed, or written to a file or device, the format specified in the device overrides the display format.

This property is optional. If you do not specify a Summary Device, alarm summaries are not logged.



### Log Device

The device where the alarm state changes are sent (maximum of 16 characters).

An alarm entry is made in the log device each time an alarm changes state (e.g., on, off, acknowledged, enabled, and disabled).

When the alarm is printed, or written to a file or device, the format specified in the device overrides the display format.

This property is optional. If you do not specify a log device, alarm state changes are not logged.

### Log Alarm Transitions: ON

Logs the alarm details when the alarm becomes active (maximum of 6 characters). The default for this field is TRUE.

### Log Alarm Transitions: OFF

Logs the alarm details when the alarm becomes inactive (maximum of 6 characters). The default for this field is TRUE.

### Log Alarm Transitions: ACK

Logs the alarm details when the alarm is acknowledged (maximum of 6 characters). The default for this field is TRUE.

### Comment

Any useful comment (maximum of 48 characters).

## Digital Alarms

You can configure digital alarms to activate based on the state of one or two digital variables. The alarm becomes active when the triggering condition spans the duration of a specified delay period.

CitectSCADA polls the variables configured for a digital alarm at the rate set by the Citect.ini parameter [\[Alarm\]ScanTime](#). If an alarm state changes, notification will occur the next time the variables are polled. Note that the time associated with the alarm state will represent the time the variable was polled, not the actual time the alarm condition occurred.

### To configure a digital alarm:

- 1 Choose **Alarm | Digital Alarms**. The Digital Alarms dialog box is displayed.
- 2 Complete the properties. See [Digital Alarm Properties](#).
- 3 Click the **Add** button to append a new record, or **Replace** if you have modified a record.

See Also [Digital Alarm Properties](#)

Digital Alarm Properties [Digital Alarms](#) have the following properties:

#### Alarm Tag

The name of the alarm (maximum of 79 characters). The name must be unique to the cluster. Alarm Tag names must adhere to the [Tag name syntax](#). If you have many tags, use a naming convention (see [Using structured tag names](#)). This makes it easier to find and debug your tags.

**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

#### Cluster Name

The name of the cluster that runs this alarm. If the Cluster Name is not set, then CitectSCADA considers this alarm to run on all defined clusters.

#### Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters). This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

#### Alarm Desc

The description of the alarm (maximum of 254 characters). This can include variable data. This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

#### Var Tag A/Var Tag B

The digital variables (tags) that trigger the alarm (maximum of 79 characters). You can configure digital alarms to activate based on the state of one or two digital variables.

If you only use one variable to trigger the alarm, use the **Var Tag A** field. For example:

Var Tag A                      RFP3\_TOL

When the state of the variable RFP3\_TOL changes to ON (1), the alarm is triggered.

Alternatively, you can define the alarm to trigger when the state of the variable changes to OFF (0), by preceding the digital address with the logical operator NOT, for example:

Var Tag A                      NOT RFP3\_TOL

In this case, the alarm is triggered when the state of the variable MCOL304 changes to OFF (0).

You can also configure digital alarms to activate based on the state of two digital variables, for example:

Var Tag A	RFP3_TOL
Var Tag B	NOT MCOL304

In this case, the alarm is triggered when the state of both variables changes to the active state: when the state of the variable RFP3\_TOL changes to ON (1), and when the state of the variable MCOL304 changes to OFF (0).

**Note:** If you leave the **Var Tag B** property blank, only Var Tag A triggers the alarm.

### Category

The alarm category number or label (maximum of 16 characters). This property is optional. If you do not specify a category, the alarm defaults to Category 0.

### Delay (hh:mm:ss)

The alarm delay period.

A digital alarm becomes active when the state of the triggering condition remains true for the duration of the delay period. The active alarm has an ON time of when the state became true.

This property is optional. If you do not specify a delay period, the alarm is active as soon as it is triggered by the digital tag(s).

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### Help

The name of the graphics page that displays when the AlarmHelp() function is called (maximum of 64 characters). This property is optional. If you do not specify a help page, no action occurs when the AlarmHelp() function is called.

You must define a command that calls the AlarmHelp() function.

### Comment

Any useful comment (maximum of 48 characters).

**Note:** The following fields are implemented with extended forms (press **F2**).

The screenshot shows a software window titled "Digital Alarms [ SampleProject ]". It contains the following fields and controls:

- Alarm Tag: Text input field.
- Cluster Name: Dropdown menu.
- Alarm Name: Text input field.
- Alarm Desc: Text input field.
- Variable Tag A: Dropdown menu.
- Variable Tag B: Dropdown menu.
- Category: Text input field.
- Help: Text input field.
- Delay: Dropdown menu.
- Comment: Text input field.
- Privilege: Text input field.
- Area: Text input field.
- Custom Filter 1 through Custom Filter 8: Eight text input fields.
- Buttons: Add, Replace, Delete, Help.
- Record: Indicator at the bottom left.

### Privilege

The privilege required by an operator to acknowledge or disable the alarm (maximum of 16 characters).

**Note:** If you assign an acknowledgment privilege to an alarm, do not assign a privilege to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator must have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.

### Area

The [area](#) to which the alarm belongs (maximum of 16 characters). If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter **Area 1** here, operators must have access to Area 1 (plus any required privileges) to acknowledge or disable this alarm.

**Note:** The area and privilege fields defined here must be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.

### Custom Filter1...Custom Filter8

A user-defined string for filtering active alarms (maximum 64 characters).

Used in a custom Cicode query function as search criteria, the custom [custom alarm filters](#) enable operators to identify and display a subset of active alarms.

**Notes:**

- Custom filters are visible only when the Digital Alarms form is open in Extended mode.
- The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '\_'.
- A custom filter cannot start with a digit.

See Also [Using custom alarm filters](#)

## Multi-digital Alarms

Multi-digital alarms use the output from three digital variables (for example: tags A, B, and C) to define eight states. The states represent all possible combinations of true/false values the variables can have.

The tag values in each state are represented in the order tag C, tag B, tag A. A true value is represented by the tag letter, and 0 (zero) represents false.

The eight states are as follows:

- **State 000** – All 3 tags are false.
- **State 00A** – Tags C and B are false and Tag A is true.
- **State 0B0** – Tags C and A are false and Tag B is true.
- **State 0BA** – Tag C is false and Tags B and A are true.
- **State C00** – Tag C is true and Tags B and A are false.
- **State C0A** – Tags C and A are true and Tag B is false.
- **State CB0** – Tags C and B are true and Tag A is false.
- **State CBA** – All 3 tags are true.

When configuring the multi-digital alarm properties, you can set which states should trigger an alarm, and specify Cicode functions to be called when alarms become active and inactive.

CitectSCADA polls the variables configured for a multi-digital alarm at the rate set by the Citect.ini parameter [\[Alarm\]ScanTime](#). If an alarm state changes, notification will occur the next time the variables are polled. The time associated with the alarm state will represent the time the variable was polled, not the actual time the alarm condition occurred.

### Examples

In the following example, tag C is left blank, and the variables BIT\_12 and BIT\_1 are specified for tags A and B. With state 0BA specified to activate an alarm, when tags A and B change to ON (1) the alarm will activate.

Var Tag A                      BIT\_12

Var Tag B                      BIT\_1

Var Tag C

In this example, variables are specified for all three tags. If state CBA is specified to activate an alarm, when all three variables change to ON (1) the alarm will activate.

Var Tag A                      RFP3\_TOL

Var Tag B                      BIT\_1

Var Tag C                      MCOL\_304

#### To configure a multi-digital alarm:

- 1 Choose **Alarm** | **Multi-digital Alarms**. The Multi-digital Alarms dialog box appears.
- 2 Enter the alarm properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Multi-digital Alarm Properties](#)

## Multi-digital Alarm Properties

[Multi-digital Alarms](#) have the following properties.

### Alarm Tag

The name of the alarm (maximum of 79 characters). The name must be unique to the cluster. Alarm Tag names must adhere to the [Tag name syntax](#). If you have many tags, use a naming convention (see [Using structured tag names](#)). This makes it easier to find and debug your tags.

**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

### Cluster Name

The name of the cluster that runs this alarm. If the Cluster Name is not set, then CitectSCADA considers this alarm to run on all defined clusters.

### Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters). This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

### Alarm Desc

The description of the alarm (maximum of 254 characters). This can include variable data. This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

#### **Var Tag A/Var Tag B/Var Tag C**

The digital variables used to define eight states (maximum of 79 characters). Each state represents a different combination of tag values.

In the following example, the digital variables RFP3\_TOL, BIT\_1, and MCOL304 are specified for Tags A, B, and C.

Var Tag A	RFP3_TOL
Var Tag B	BIT_1
Var Tag C	MCOL_304

#### **States and Descriptions**

The following eight states represent all possible tag value combinations. The tags are represented in the order Tag C, Tag B, Tag A.

- **State 000** – All 3 tags are false.
- **State 00A** – Tags C and B are false and Tag A is true.
- **State 0B0** – Tags C and A are false and Tag B is true.
- **State 0BA** – Tag C is false and Tags B and A are true.
- **State C00** – Tag C is true and Tags B and A are false.
- **State C0A** – Tags C and A are true and Tag B is false.
- **State CB0** – Tags C and B are true and Tag A is false.
- **State CBA** – All 3 tags are true.

For each state, there are two fields on the Multi-Digital Alarms dialog. In the first field you can enter a description (e.g. Healthy or Stopped), with a maximum of eight characters.

In the second field, you indicate whether the state should trigger an alarm. A value of 1 indicates an alarm state, 0 indicates no alarm will be triggered.

#### **Realarm (1 char.)**

Indicates what happens when there is a transition from one alarm state to another. A value of 1 in this field causes a new time and State Description {State\_Desc,n} to be recorded when the states change. With a value of 0, only the time and State Description of the first alarm state is recorded in the Alarm Summary.

#### **ON Function (254 Chars.)**

A Cicode function that is executed when a Multi-Digital Alarm becomes active, e.g.

ON Function                      STOP\_PROCESS = 1;

The digital variable STOP\_PROCESS is set to ON when the alarm is triggered.

**Note:** Do not put a blocking function in this field.

A special case of this command occurs when the Alarm ON Function is self-referring, with a form such as TAG1 = TAG1 + 1. This command will not work properly since CitectSCADA does not reread the tags before processing the Alarm On function (for performance reasons). This particular command will therefore initially set the value of TAG1 to 1 rather than incrementing it.

To correctly run a command of this type in the Alarm ON Function, use TaskNew() to run your own Cicode function to perform the tag command:

ON Function                      TaskNew("MyFunc","Data",5);

#### **OFF Function (254 Chars.)**

A Cicode function that is executed when a Multi-Digital Alarm becomes inactive (maximum of 254 characters). For example,

OFF Function                      ENABLE\_PROCESS = 1;

The digital variable ENABLE\_PROCESS is set to ON when an alarm in this category is reset.

**Note:** Do not put a blocking function in this field.

#### **Category**

The alarm category number or label (maximum of 16 characters). This property is optional. If you do not specify a category, the alarm defaults to Category 0.

#### **Help**

The name of the graphics page that displays when the AlarmHelp() function is called (maximum of 64 characters). This property is optional. If you don't specify a help page, no action occurs when the AlarmHelp() function is called.

You must define a command that calls the AlarmHelp() function.

#### **Privilege**

The privilege required by an operator to acknowledge or disable the alarm (maximum of 16 characters).



**Note:** If you assign an acknowledgment privilege to an alarm, do not assign a privilege to the command(s) that acknowledge the alarm. If you do assign a different privilege to the commands, then an operator must have both privileges to acknowledge the command.

#### Area

The area to which the alarm belongs (maximum of 16 characters). If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to acknowledge or disable this alarm.

#### Comment

Any useful comment (maximum of 48 characters).

#### Suppression

The number of the Suppression Group to which the alarm belongs. This *must* be an integer value between 0–65535. Alarms in the same group display the same value in this field.

This property is used in conjunction with Suppression Level.

**Note:** To assign a name to a Suppression Group, define the name as a label with an integer value.

#### Level (Suppression Level)

The level of an alarm within its Suppression Group (integer value). This is a value between 0 and 255, where a lower level represents a higher priority.

This property enables an [active alarm](#) to suppress lower priority alarms within the same Suppression Group. When this occurs, only the higher priority (lower level) alarms are displayed. Alarms with lower priorities (higher levels) will only activate and display when the higher priority (lower level) alarms become inactive.

If two alarms of different priorities in the same Suppression Group are triggered at the same time, both will display in the alarm list. This is because at the time they activated, the higher priority alarm was not already active and so could not suppress the lower priority alarm.

The only way to ensure that the higher priority alarms always activate before lower priority ones (and can therefore suppress them when appropriate), is to store the higher priority alarms closer to the beginning of the Alarms database. The database is scanned from beginning to end for triggered alarms, and if higher priority alarms are higher in the database, they will activate first and be able to suppress any lower priority alarms within the Suppression Group.

**Note:** The following fields are implemented with extended forms (press F2).

### Custom Filter1...Custom Filter8

A user-defined string for filtering active alarms (maximum 64 characters).

Used in a custom Cicode query function as search criteria, the custom [custom alarm filters](#) enable operators to identify and display a sub-set of active alarms.

#### Notes:

- The custom filters are visible only when the Digital Alarms form is open in Extended mode.
- The fields are not case sensitive, and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '\_'.
- A custom filter cannot start with a digit.

See Also [Using custom alarm filters](#)

## Time-stamped Alarms

Time-stamped alarms are similar to digital alarms, except that a counter is used to provide an accurate timestamp of when a triggering condition occurs, rather than just the time the variable was polled. Time-stamped alarms can only be associated with a single digital variable.

An alarm's variables are polled at the rate set by [\[Alarm\]ScanTime](#), however, the timer value is used to define the time associated with a change of state.

You can use one of three types of counter or timer to record the triggering of time-stamped alarms:

- **Continuous counter.** CitectSCADA reads a continuous counter in the unit to determine the sequence in which the alarms are triggered. CitectSCADA sorts the alarms based on the value of the counter when the alarm was triggered (the exact time is not recorded).
- **Millisecond counter.** If your unit supports a millisecond counter, you can program a counter (in the unit) to count (in milliseconds) for 24 hours, and then reset (at midnight). CitectSCADA reads the value of this timer variable (in the unit) to determine the exact time when the alarm was triggered.
- **LONGBCD timer.** Using a LONGBCD timer, you can log the exact time when a time-stamped alarm becomes active. CitectSCADA reads this variable, along with the alarm tag when the alarm activates.

**To configure a time-stamped alarm:**

- 1 Choose **Alarms | Time-stamped Alarms**. The Time-stamped Alarms dialog box appears.
- 2 Enter the alarm properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Time-stamped Alarm Properties](#)

### Time-stamped Alarm Properties

[Time-stamped Alarms](#) have the following properties:

#### Alarm Tag

The name of the alarm (maximum of 79 characters). The name must be unique to the cluster. Alarm Tag names must adhere to the [Tag name syntax](#). If you have many tags, use a naming convention (see [Using structured tag names](#)). This makes it easier to find and debug your tags.

**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

#### Cluster Name

The name of the cluster that runs this alarm. If the Cluster Name is not set, then CitectSCADA considers this alarm to run on all defined clusters.

#### Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters). This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

#### Alarm Desc

The description of the alarm (maximum of 254 characters). This can include variable data. This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

#### Variable Tag

The digital variable (tag) that triggers the alarm (maximum of 79 characters).

#### Timer

The variable tag or Cicode [expression](#) that represents the counter (or millisecond timer) configured in the I/O device (maximum of 254 characters). The counter must be configured and maintained by the program in the I/O device; it is read only when the alarm is triggered.

You can use one of three types of counter or timer to record the triggering of time-stamped alarms:

- **Continuous counter.** CitectSCADA reads a continuous counter in the unit to determine the sequence in which the alarms are triggered. CitectSCADA sorts the alarms based on the value of the counter when the alarm was triggered (the exact time is not recorded). You must program the counter (in the unit) to count continually to its limit, reset, and again count to its limit.
- **Millisecond counter.** If your unit supports a millisecond counter, you can program a counter (in the unit) to count (in milliseconds) for 24 hours, and then reset (at midnight). CitectSCADA reads the value of this timer variable (in the unit) to determine the exact time when the alarm was triggered.
- **LONGBCD timer.** Using a LONGBCD timer, you can log the exact time when a Time-stamped alarm becomes active. CitectSCADA reads this

variable, along with the alarm tag when the alarm activates. You must program the LONGBCD variable in the following format:

BYTE	MEANING	RANGE
1 <sup>st</sup>	Hours	00–24 (most significant byte)
2 <sup>nd</sup>	Minutes	00–60
3 <sup>rd</sup>	Seconds	00–60
4 <sup>th</sup>	100 <sup>th</sup> /sec	00–100 (most significant byte)

- This field can also be used to handshake with the PLC code: CitectSCADA informs the PLC that it has read the timer register and now the PLC can overwrite the last value. For example, with the following code saved in a Cicode file:

```
INT
FUNCTION
AlarmTimerReset(INT iTimer, STRING sTimerTrigger)
TagWrite(sTimerTrigger, 0); //Reset the trigger
RETURN iTimer; //Return the timer value to the alarm system
END
```

You could configure a time-stamped alarm as follows:

```
Variable Tag      AlmTrigger1
Timer             Timer(AlmTimer1, "AlmTrigger1")
```

where **AlmTimer1** is the PLC register that stores the alarm time, and **AlmTrigger1** is the alarm trigger bit.

When AlmTrigger1 is set to one (1), the alarm is triggered, and the Cicode function is called. On calling the function, CitectSCADA reads the AlmTimer1 register. The function resets the trigger bit (handshaking), and the value of AlmTimer1 is returned to the alarm system.

### Category

The alarm category number or label (maximum of 16 characters). This property is optional. If you do not specify a category, the alarm defaults to Category 0.

### Help

Name of the graphics page that appears when the AlarmHelp() function is called (maximum 64 characters). This property is optional. If you do not specify a help page, no action occurs when the AlarmHelp() function is called.

You must define a command that calls the AlarmHelp() function.

### Comment

Any useful comment (maximum 48 characters).

**Note:** The following fields are implemented with extended forms (press F2).

The screenshot shows a Windows-style dialog box titled "Time Stamped Alarms [ SampleProject ]". It contains the following fields and controls:

- Alarm Tag: Text input field.
- Cluster Name: Dropdown menu.
- Alarm Name: Text input field.
- Alarm Desc: Text input field.
- Variable Tag: Dropdown menu.
- Timer: Text input field.
- Category: Dropdown menu.
- Help: Text input field.
- Comment: Text input field.
- Privilege: Text input field.
- Area: Text input field.
- Custom Filter 1 through Custom Filter 8: Eight text input fields.
- Buttons: Add, Replace, Delete, and Help.
- Record: Dropdown menu at the bottom left.

### Privilege

The privilege required by an operator to acknowledge or disable the alarm (maximum of 16 characters).

**Note:** If you assign an acknowledgment privilege to an alarm, do not assign a privilege to the command(s) that acknowledge the alarm. If you do assign a different privilege to the commands, an operator must have both privileges to acknowledge the command.

### Area

Area to which the alarm belongs (maximum 16 characters). If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to acknowledge or disable this alarm.

### Custom Filter1...Custom Filter8

A user-defined string for filtering active alarms (maximum 64 characters).

Used in a custom Cicode query function as search criteria, the custom [custom alarm filters](#) enable operators to identify and display a subset of active alarms.

**Notes:**

- Custom filters are visible only when the Digital Alarms form is open in Extended mode.
- The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '\_'.
- A custom filter cannot start with a digit.

See Also [Using custom alarm filters](#)

## Analog Alarms

Analog alarms are triggered when an analog variable changes beyond one or more specific limits. Each alarm can be configured as any combination of the following types.

- **high** and **high high** alarms - where the value reaches an atypical high
- **low** and **low low** alarms - where the value reaches an atypical low
- **deviation** alarm - where the values moves away from a predefined set point
- **rate of change** alarm - where a dramatic value change occurs within a specified period of time

CitectSCADA polls the variables configured for an analog alarm at the rate set by the Citect.ini parameter [\[Alarm\]ScanTime](#). If an alarm state triggers, notification will occur the next time the variables are polled. Note that the time associated with the alarm state will represent the time the variable was polled, not the actual time the alarm condition occurred.

### To configure an analog alarm:

- 1 Choose **Alarms** | **Analog Alarms**. The Analog Alarms dialog box appears.
- 2 Enter the alarm properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Analog Alarm Properties](#)

## Analog Alarm Properties

[Analog Alarms](#) have the following properties:

### Alarm Tag

The name of the alarm (maximum of 79 characters). The name must be unique to the cluster. Alarm Tag names must adhere to the [Tag name syntax](#). If you have many tags, use a naming convention (see [Using structured tag names](#)). This makes it easier to find and debug your tags.

**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

**Cluster Name**

The name of the cluster that runs this alarm. If the Cluster Name is not set, then CitectSCADA considers this alarm to run on all defined clusters.

**Alarm Name**

The name of the physical device associated with the alarm (maximum of 79 characters). This is an optional property. CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

**Variable Tag**

The analog variable (tag) that triggers the alarm (maximum of 79 characters).

**Setpoint**

An analog variable tag or base value that determines if a [deviation alarm](#) is to be triggered (maximum of 79 characters). This property is optional. If you do not specify a setpoint, it will default to 0 (zero).

**High High**

The value used as the triggering condition for a high high alarm (maximum of 10 characters). The high high alarm becomes active when the value of the variable tag exceeds this value for the duration of the high high delay period. The active alarm has an ON time of when the tag exceeded the high high value.

Because a high alarm must precede a high high alarm, when the high high alarm is triggered it replaces the high alarm. If you want an analog alarm to display more than one state on the alarm page at the same time, configure a separate alarm for each state. (Each alarm would monitor the same tag.)

**High High Delay**

The delay period for High High Alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the high high alarm will be activated as soon as the tag exceeds the high high value.

**Note:** The delay period must be entered in the format HH:MM:SS (hours:minutes:seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

**High**

The value used as the triggering condition for a high alarm (maximum of 10 characters). The high alarm becomes active when the value of the variable tag exceeds this value for the duration of the high delay period. The active alarm has an ON time of when the tag exceeded the high value.



### High Delay

The delay period for high alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the high alarm will be activated as soon as the tag exceeds the high value.

**Note:** The delay period must be entered in the format HH:MM:SS (hours:minutes:seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

When a tag value increases from high to high high within the high delay period, the delay timer is reset. The high high alarm is only activated if the value remains in the high high range for the delay period.

When the value increases from high to high high after the high delay period has expired, a high alarm is activated and then the delay period for the high high alarm begins.

If the tag value exceeds the high high value and then falls below it before the high high delay period expires, at the time it falls, the high alarm is triggered immediately. It has an ON time of when the tag value exceeded the high high value.

These points also apply to tag values travelling between Low and Low Low ranges.

### Low

The value used as the triggering condition for a Low Alarm (maximum of 10 characters). A Low Alarm becomes active when the value of the Variable Tag drops below this value and remains there for the duration of the Low Delay period. The active alarm has an ON time of when the tag fell below the Low value.

### Low Delay

The delay period for Low Alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the Low Alarm is activated as soon as the tag drops below the Low value.

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### Low Low

The value used as the triggering condition for a Low Low Alarm (maximum of 10 characters). A Low Low Alarm becomes active when the value of the Variable Tag drops below this value and remains there for the duration of

the Low Low Delay period. The active alarm has an ON time of when the tag fell below the Low Low value.

Because a Low Alarm must precede a Low Low Alarm, when the Low Low Alarm is triggered it replaces the Low Alarm. If you want an analog alarm to display more than one state on the alarm page at the same time, configure a separate alarm for each state. (Each alarm would monitor the same tag.)

### Low Low Delay

The delay period for Low Low Alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the Low Low Alarm is activated as soon as the tag drops below the Low Low value.

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### Deviation

The value used as the triggering condition for a Deviation Alarm (maximum of 10 characters). A Deviation Alarm is activated when the value of the Variable Tag remains outside the deviation range (determined by the Setpoint) for the duration of the Deviation Delay period.

This property is optional. If you do not specify a deviation, no Deviation Alarm is activated.

### Deviation Delay

The delay period for Deviation Alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the Deviation Alarm is activated as soon as the Variable Tag falls outside the deviation range.

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### Rate

By dividing this value by the alarm period, CitectSCADA determines the “maximum rate” at which the value of the variable tag can change (maximum of 10 characters). At each Scan Time, CitectSCADA checks the value of the tag. If its rate of change is greater than the maximum rate, a Rate of Change Alarm is triggered.

For example, to ensure that a tank does not fill too quickly, you might configure a rate of change alarm, using a Rate of 300 liters, an **[Alarm]Period** of 60 seconds, and an **[Alarm]ScanTime** of 1 second. This means that the

maximum allowable rate of change for the tank level is 5 l/sec (300 liters / 60 seconds). CitectSCADA calculates the actual rate of change at each ScanTime. i.e. Every second, it checks the current level of the tank and compares it to the level recorded a second earlier. If the actual rate of change is, say, 8 l/sec, a Rate of Change Alarm is triggered immediately.

The variable tags deadband % must be less than the alarms rate divided by the engineering scale of the variable tag. Otherwise, the rate of change alarm will only go off when the change in the variable tag exceeds the deadband value.

This property is optional. If you do not specify a value, no Rate of Change Alarm is activated.

### **Deadband**

The value that **Variable Tag** must return to before the Deviation Alarm becomes inactive (maximum of 10 characters).

### **Format**

The display format of the value (of the variable) when it is displayed on a graphics page, written to a file or passed to a function (that expects a string) (maximum of 10 characters).

This property is optional. If you do not specify a format, the format defaults to the format specified for Variable tag.

### **Category**

The alarm category number or label (maximum of 10 characters). This property is optional. If you do not specify a category, the alarm defaults to Category 0.

### **Help**

The name of the graphics page that displays when the AlarmHelp() function is called (maximum of 64 characters). This property is optional. If you do not specify a help page, no action occurs when the AlarmHelp() function is called.

You must define a command that calls the AlarmHelp() function.

### **Comment**

Any useful comment (maximum of 48 characters).

**Note:** The following fields are implemented with extended forms (press **F2**).

**Analog Alarms [ SampleProject ]**

Alarm Tag:

Cluster Name:

Alarm Name:

Variable Tag:  Setpoint:

High High:  High:

High High Delay:  High Delay:

Low:  Low Low:

Low Delay:  Low Low Delay:

Deviation:  Rate:

Deviation Delay:

Deadband:  Format:

Category:  Help:

Comment:

Privilege:  Area:

Custom Filter 1:

Custom Filter 2:

Custom Filter 3:

Custom Filter 4:

Custom Filter 5:

Custom Filter 6:

Custom Filter 7:

Custom Filter 8:

Record:

### Privilege

The privilege required by an operator to acknowledge or disable the alarm (maximum of 16 characters).

**Note:** If you assign an acknowledgment privilege to an alarm, do not assign a privilege to the command(s) that acknowledge the alarm. If you do assign a different privilege to the commands, then an operator must have both privileges to acknowledge the command.

### Area

The area to which the alarm belongs (maximum of 16 characters). If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to acknowledge or disable this alarm.

### Custom Filter1...Custom Filter8

A user-defined string for filtering active alarms (maximum 64 characters).

Used in a custom Cicode query function as search criteria, the custom [custom alarm filters](#) enable operators to identify and display a sub-set of active alarms.

**Notes:**

- The custom filters are visible only when the Digital Alarms form is open in Extended mode.
- The fields are not case-sensitive and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '\_'.
- A custom filter cannot start with a digit.

See Also [Using custom alarm filters](#)

## Advanced Alarms

An Advanced Alarm becomes active when the result of the Cicode [expression](#) changes. CitectSCADA polls the expression at the rate set by the Citect.ini parameter [\[Alarm\]ScanTime](#) and tests for a change in outcome. If one occurs, an alarm state notification will occur.

**To configure an advanced alarm:**

- 1 From the **System | Advanced Alarms**. The Advanced Alarms dialog box appears.
- 2 Enter the alarm properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Advanced Alarm Properties](#)

### Advanced Alarm Properties

[Advanced Alarms](#) have the following properties.

**Alarm Tag**

The name of the alarm (maximum of 79 characters). The name must be unique to the cluster. Alarm Tag names must adhere to the [Tag name syntax](#). If you have many tags, use a naming convention (see [Using structured tag names](#)). This makes it easier to find and debug your tags.

**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

**Cluster Name**

The name of the cluster that runs this alarm. If the Cluster Name is not set, then CitectSCADA considers this alarm to run on all defined clusters.

**Alarm Name**

The name of the physical device associated with the alarm (maximum of 79 characters). This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

**Alarm Desc**

The description of the alarm (maximum of 254 characters). This can include variable data. This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

**Expression**

The Cicode expression that triggers the alarm (maximum of 254 characters). Whenever the result of the expression changes to TRUE, the alarm is triggered.

**Category**

The alarm category number or label (maximum of 16 characters).

This property is optional. If you do not specify a category, the alarm defaults to Category 0.

**Delay**

The delay period for the Advanced Alarm.

An Advanced Alarm becomes active when the result of the Cicode expression triggering the alarm remains TRUE for the duration of the delay period. The active alarm has an ON time of when the expression returned TRUE.

This property is optional. If you do not specify a value, the alarm becomes active as soon as the triggering expression becomes true.

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

**Help**

The name of the graphics page that displays when the AlarmHelp() function is called (maximum of 64 characters). This property is optional. If you do not specify a help page, no action occurs when the AlarmHelp() function is called.

You must define a command that calls the AlarmHelp() function.

**Comment**

Any useful comment (maximum of 48 characters).

**Note:** The following fields are implemented with extended forms (press F2).

### Privilege

The privilege required by an operator to acknowledge or disable the alarm (maximum of 16 characters).

**Note:** If you assign an acknowledgment privilege to an alarm, do not assign a privilege to the command(s) that acknowledge the alarm. If you do assign a different privilege to the commands, then an operator must have both privileges to acknowledge the command.

### Area

The area to which the alarm belongs (maximum of 16 characters). If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to acknowledge or disable this alarm.

### Custom Filter1...Custom Filter8

A user-defined string for filtering active alarms (maximum 64 characters).

Used in a custom Cicode query function as search criteria, the custom [custom alarm filters](#) enable operators to identify and display a sub-set of active alarms.

**Note:**

- The custom filters are visible only when the Digital Alarms form is open in Extended mode.
- The fields are not case sensitive, and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '\_'.
- A custom filter cannot start with a digit.

See Also [Using custom alarm filters](#)

## Time-stamped Digital Alarms

Time-stamped digital and time-stamped analog alarms differ to other alarm types, as they do not rely on the polling of variables to determine alarm conditions. They operate via a process where the alarm server is notified of any value changes to a specified variable using the Cicode function `[Alarm]NotifyVarChange`.

The alarm server uses this information to update alarms that monitor the variable. This process allows an accurate timestamp to be associated with an alarm condition.

This process updates the **Var Tag A** and **Var Tag B** properties for time-stamped digital alarms.

Events trends can be used in conjunction with time-stamped digital alarms to provide millisecond accuracy for both trend and alarm data. See [TrnEventSetTable](#) and [TrnEventSetTableMS](#).

### To configure a time-stamped digital alarm:

- 1 Choose **Alarm | Time-Stamped Digital Alarms**. The Time-Stamped Digital Alarms dialog box appears.
- 2 Complete the properties in the form that appears.
- 3 Click the **Add** button to append a new record, or **Replace** if you have modified a record.

**Note:** For time-stamped digital alarms to function correctly, you must configure the Cicode function [AlarmNotifyVarChange](#) to ensure the alarms server is notified of any value changes for the associated variable.

See Also [Time-stamped Digital Alarm Properties](#)

## Time-stamped Digital Alarm Properties

[Time-stamped Digital Alarms](#) have the following properties:

### Alarm Tag

The name of the alarm (maximum of 79 characters). The name must be unique to the cluster. Alarm Tag names must adhere to the [Tag name](#)



**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

The name of the cluster that runs this alarm. If the Cluster Name is not set, then CitectSCADA considers this alarm to run on all defined clusters.

The name of the physical device associated with the alarm (maximum of 79 characters). This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

The description of the alarm (maximum of 254 characters). This can include variable data. This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

The digital variables (tags) that trigger the alarm (maximum of 79 characters). You can configure time-stamped digital alarms to activate based on the state of one or two digital variables.

If you only use one variable to trigger the alarm, use the **Var Tag A** field. For example:

When the state of the variable RFP3\_TOL changes to ON (1), the alarm is triggered.

Alternatively, you can define the alarm to trigger when the state of the variable changes to OFF (0), by preceding the digital address with the logical operator NOT, for example:

In this case, the alarm is triggered when the state of the variable MCOL304 changes to OFF (0).

You can also configure digital alarms to activate based on the state of two digital variables, for example:

Var Tag A	RFP3_TOL
Var Tag B	NOT MCOL304

In this case, the alarm is triggered when the state of both variables changes to the active state: when the state of the variable RFP3\_TOL changes to ON (1), and when the state of the variable MCOL304 changes to OFF (0).

**Note:** If you leave the **Var Tag B** property blank, only Var Tag A triggers the alarm.

### Category

The alarm category number or label (maximum of 16 characters). This property is optional. If you do not specify a category, the alarm defaults to Category 0.

### Delay (hh:mm:ss)

The alarm delay period.

A time-stamped digital alarm becomes active when the state of the triggering condition remains true for the duration of the delay period. The active alarm has an ON time of when the state became true.

This property is optional. If you do not specify a delay period, the alarm is active as soon as it is triggered by the digital tag(s).

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### Help

The name of the graphics page that displays when the AlarmHelp() function is called (maximum of 64 characters). This property is optional. If you do not specify a help page, no action occurs when the AlarmHelp() function is called.

You must define a command that calls the AlarmHelp() function.

### Comment

Any useful comment (maximum of 48 characters).

**Note:** The following fields are implemented with extended forms (press F2).

### Privilege

The privilege required by an operator to acknowledge or disable the alarm (maximum of 16 characters).

**Note:** If you assign an acknowledgment privilege to an alarm, do not assign a privilege to the command(s) that acknowledge the alarm. If you assign a different privilege to the commands, an operator must have both privileges to acknowledge the command. More importantly, the area defined here may be ignored.

### Area

The area to which the alarm belongs (maximum of 16 characters). If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter **Area 1** here, operators must have access to Area 1 (plus any required privileges) to acknowledge or disable this alarm.

**Note:** The area and privilege fields defined here must be designed to work in conjunction. A privilege defined on a button (say) will ignore the alarm defined area.

### Custom Filter1...Custom Filter8

A user-defined string for filtering active alarms (maximum 64 characters).

Used in a custom Cicode query function as search criteria, the custom [custom alarm filters](#) enable operators to identify and display a sub-set of active alarms.

**Note:**

- The custom filters are visible only when the Digital Alarms form is open in Extended mode.
- The fields are not case sensitive, and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '\_'.
- A custom filter cannot start with a digit.

See Also [Using custom alarm filters](#)

## Time-stamped Analog Alarms

Time-stamped digital and time-stamped analog alarms differ to other alarm types, as they do not rely on the polling of variables to determine alarm conditions. They operate via a process where the alarm server is notified of any value changes to a specified variable using the Cicode function [Alarm]NotifyVarChange.

The alarm server will then use this information to update all the alarms that monitor the variable. This process allows for an accurate timestamp to be associated with an alarm condition.

This process is used to update the **Variable Tag** and **Setpoint** properties for time-stamped analog alarms.

Events trends can be used in conjunction with time-stamped analog alarms to provide millisecond accuracy for both trend and alarm data. See [TrnEventSetTable](#) and [TrnEventSetTableMS](#).

### To configure an analog time-stamped alarm:

- 1 Choose **Alarms | Analog Time-stamped Alarms**. The Analog Time-stamped Alarms dialog box appears.
- 2 Enter the alarm properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

**Note:** For time-stamped digital alarms to function correctly, you must configure the Cicode function [AlarmNotifyVarChange](#) to ensure the alarms server is notified of any value changes for the associated variable.

See Also [Time-stamped Analog Alarm Properties](#)

## Time-stamped Analog Alarm Properties

[Time-stamped Analog Alarms](#) have the following properties:

### Alarm Tag

The name of the alarm (maximum of 79 characters). The name must be unique to the cluster. Alarm Tag names must adhere to the [Tag name syntax](#). If you have many tags, use a naming convention (see [Using structured tag names](#)). This makes it easier to find and debug your tags.

**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

#### Cluster Name

The name of the cluster that runs this alarm. If the Cluster Name is not set, then CitectSCADA considers this alarm to run on all defined clusters.

#### Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters). This property is optional, CitectSCADA only uses it when details of the alarm are displayed on the screen or logged to a device.

#### Variable Tag

The analog variable (tag) that triggers the alarm (maximum of 79 characters).

#### Setpoint

An analog variable tag or base value that determines if a deviation alarm is to be triggered (maximum of 79 characters). This property is optional. If you do not specify a setpoint, it will default to 0 (zero).

#### High High

The value used as the triggering condition for a high high alarm (maximum of 10 characters). The high high alarm becomes active when the value of the variable tag exceeds this value for the duration of the high high delay period. The active alarm has an ON time of when the tag exceeded the high high value.

Because a high alarm must precede a high high alarm, when the high high alarm is triggered it replaces the high alarm. If you want an analog alarm to display more than one state on the alarm page at the same time, configure a separate alarm for each state. (Each alarm would monitor the same tag.)

#### High High Delay

The delay period for High High Alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the high high alarm will be activated as soon as the tag exceeds the high high value.

**Note:** The delay period must be entered in the format HH:MM:SS (hours:minutes:seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### High

The value used as the triggering condition for a high alarm (maximum of 10 characters). The high alarm becomes active when the value of the variable tag exceeds this value for the duration of the high delay period. The active alarm has an ON time of when the tag exceeded the high value.

### High Delay

The delay period for high alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the high alarm will be activated as soon as the tag exceeds the high value.

**Note:** The delay period must be entered in the format HH:MM:SS (hours:minutes:seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

When a tag value increases from high to high high within the high delay period, the delay timer is reset. The high high alarm is only activated if the value remains in the high high range for the delay period.

When the value increases from high to high high after the high delay period has expired, a high alarm is activated and then the delay period for the high high alarm begins.

If the tag value exceeds the high high value and then falls below it before the high high delay period expires, at the time it falls, the high alarm is triggered immediately. It has an ON time of when the tag value exceeded the high high value.

These points also apply to tag values travelling between Low and Low Low ranges.

### Low

The value used as the triggering condition for a Low Alarm (maximum of 10 characters). A Low Alarm becomes active when the value of the Variable Tag drops below this value and remains there for the duration of the Low Delay period. The active alarm has an ON time of when the tag fell below the Low value.

### Low Delay

The delay period for Low Alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the Low Alarm is activated as soon as the tag drops below the Low value.

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### **Low Low**

The value used as the triggering condition for a Low Low Alarm (maximum of 10 characters). A Low Low Alarm becomes active when the value of the Variable Tag drops below this value and remains there for the duration of the Low Low Delay period. The active alarm has an ON time of when the tag fell below the Low Low value.

Because a Low Alarm must precede a Low Low Alarm, when the Low Low Alarm is triggered it replaces the Low Alarm. If you want an analog alarm to display more than one state on the alarm page at the same time, configure a separate alarm for each state. (Each alarm would monitor the same tag.)

### **Low Low Delay**

The delay period for Low Low Alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the Low Low Alarm is activated as soon as the tag drops below the Low Low value.

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### **Deviation**

The value used as the triggering condition for a Deviation Alarm (maximum of 10 characters). A Deviation Alarm is activated when the value of the Variable Tag remains outside the deviation range (determined by the Setpoint) for the duration of the Deviation Delay period.

This property is optional. If you do not specify a deviation, no Deviation Alarm is activated.

### **Deviation Delay**

The delay period for Deviation Alarms. The alarm will only activate if its triggering condition is met for the duration of this period.

This property is optional. If you do not set a value, the Deviation Alarm is activated as soon as the Variable Tag falls outside the deviation range.

**Note:** The delay period must be entered in the format HH:MM:SS (Hours:Minutes:Seconds). The value must be between 0 seconds (00:00:00) and 24 hours (24:00:00).

### Rate

By dividing this value by the alarm period, CitectSCADA determines the “maximum rate” at which the value of the variable tag can change (maximum of 10 characters). At each Scan Time, CitectSCADA checks the value of the tag. If its rate of change is greater than the maximum rate, a Rate of Change Alarm is triggered.

For example, to ensure that a tank does not fill too quickly, you might configure a rate of change alarm, using a Rate of 300 liters, an [Alarm]Period of 60 seconds, and an [Alarm]ScanTime of 1 second. This means that the maximum allowable rate of change for the tank level is 5 l/sec (300 liters/60 seconds). CitectSCADA calculates the actual rate of change at each ScanTime; that is, every second it checks the current level of the tank and compares it to the level recorded a second earlier. If the actual rate of change is, say, 8 l/sec, a Rate of Change Alarm is triggered immediately.

This property is optional. If you do not specify a value, no Rate of Change Alarm is activated.

### Deadband

The value that **Variable Tag** must return to before the Deviation Alarm becomes inactive (maximum of 10 characters).

### Format

The display format of the value (of the variable) when it is displayed on a graphics page, written to a file or passed to a function (that expects a string) (maximum of 10 characters).

This property is optional. If you do not specify a format, the format defaults to the format specified for Variable tag.

### Category

The alarm category number or label (maximum of 10 characters). This property is optional. If you do not specify a category, the alarm defaults to Category 0.

### Help

The name of the graphics page that displays when the AlarmHelp() function is called (maximum 64 characters). This property is optional. If you do not specify a help page, no action occurs when the AlarmHelp() function is called.

You must define a command that calls the AlarmHelp() function.

### Comment

Any useful comment (maximum of 48 characters).



**Note:** The following fields are implemented with extended forms (press F2).

The screenshot shows a software window titled "Time Stamped Analog Alarms [ SampleProject ]". It contains the following fields and controls:

- Alarm Tag: Text input field.
- Cluster Name: Dropdown menu.
- Alarm Name: Text input field.
- Variable Tag: Dropdown menu.
- Setpoint: Text input field.
- High High: Text input field.
- High: Text input field.
- High High Delay: Dropdown menu.
- High Delay: Text input field.
- Low: Text input field.
- Low Low: Text input field.
- Low Delay: Dropdown menu.
- Low Low Delay: Text input field.
- Deviation: Text input field.
- Rate: Text input field.
- Deviation Delay: Dropdown menu.
- Deadband: Text input field.
- Format: Dropdown menu.
- Category: Text input field.
- Help: Dropdown menu.
- Comment: Text input field.
- Privilege: Text input field.
- Area: Text input field.
- Custom Filter 1 through Custom Filter 8: Eight text input fields.
- Buttons: Add, Replace, Delete, Help.
- Record: Dropdown menu.

### Privilege

The privilege required by an operator to acknowledge or disable the alarm (maximum of 16 characters).

**Note:** If you assign an acknowledgment privilege to an alarm, do not assign a privilege to the command(s) that acknowledge the alarm. If you do assign a different privilege to the commands, then an operator must have both privileges to acknowledge the command.

### Area

The area to which the alarm belongs (maximum of 16 characters). If an operator does not have access to an area, the alarm is not visible on the alarm display. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to acknowledge or disable this alarm.

### Custom Filter1...Custom Filter8

A user-defined string for filtering active alarms (maximum 64 characters).

Used in a custom Cicode query function as search criteria, the custom [custom alarm filters](#) enable operators to identify and display a sub-set of active alarms.

**Note:**

- The custom filters are visible only when the Digital Alarms form is open in Extended mode.
- The fields are not case sensitive, and can contain 'A'..'Z', 'a'..'z', '0'..'9', and the underscore '\_'.
- A custom filter cannot start with a digit.

See Also [Using custom alarm filters](#)

## Formatting an Alarm Display

The display format specifies how alarms are displayed on screen for the alarms and alarm summary pages. For details on how to change the order of alarms listed on the alarm summary page, see [Changing the Order of the Alarm Summary Display](#).

See Also [Including CitectSCADA data](#)  
[Including fixed text](#)  
[Displaying lists and tables](#)  
[Variable data in alarm messages](#)  
[Alarm display fields](#)  
[Alarm summary fields](#)

### Including CitectSCADA data

Include CitectSCADA data in an alarm display by specifying the field name and width for each field to display. You must enclose each field in braces {} and use the following syntax:

```
{<field name>, [width[, justification]]}
```

For example:

```
Format           {Tag,8} {Name,32}
```

In this case, data displays in two fields: Tag, with 8 characters; and Name, with 32 characters. The width specifier is optional; if you do not use it, the width of the field is determined by the number of characters between the braces.

```
Format           Name of Alarm:{Name  }
```

In this case, Name is followed by four spaces; the data {Name} displays with 8 characters.

**Note:** The screen resolution of your CitectSCADA computer determines the total number of characters (and therefore the number of fields) that can be displayed on the alarms page.

See Also [Including fixed text](#)

## Including fixed text

You can include fixed text by specifying the text exactly as it will display; for example:

Format                      Name of Alarm:

Any spaces that you use in a text string are also included in the display.

See Also [Displaying lists and tables](#)

## Displaying lists and tables

To set the justification of the text in each field, use a justification specifier. You can use three justification characters, L (Left), R (Right), and N (None); for example:

Format              Name of Alarm:{Name,32,R} {Tag,8,L}

The justification specifier is optional; if it is omitted, the field is left justified. If you use a justification specifier, you must also use the width specifier.

To display field text in columns, use the tab character (^t); for example:

Format              {Tag,8}^t{Name,32}^t{Desc,8}

This format aligns the tag, name and description fields of the alarm when using a proportional font to display the alarms.

See Also [Variable data in alarm messages](#)

## Variable data in alarm messages

The **Alarm Desc** field of digital, advanced and time-stamped alarms can be used to display variable data. An expression (variable tag, function etc.) can be embedded into the text of the **Alarm Desc** field. This expression is evaluated when the alarm is tripped, returning the value of any variable tags at the point in time when the alarm was generated.

Enclosing the expression in braces separates the variable data from the static text. For example:

Alarm Desc              Line Broken Alarm at Line Speed {LineSpeed1}

When *LineSpeed1* is a variable tag, this expression will produce the following output on the alarm display or alarm log:

**Line Broken Alarm at Line Speed 1234**

The following alarm entry uses an expression instead of a tag:

Alarm Desc              High Level at Total Capacity {Tank1+Tank2+Offset()}

When *Tank1* and *Tank2* are variable tags, and *Offset* is a Cicode function, this expression produces the following output:

**High Level at Total Capacity 4985 liters**

**Note:** The result is formatted according to the formatting specified for the first variable tag in the expression. Standard variable formatting specifiers can be used to define the format for the numeric variable, over-riding the default format specified in Variable Tags.

See Also [Alarm display fields](#)

Alarm display fields

You can use any of the following fields (or combination) to format an alarm display (see [Alarm Categories](#)) and an alarm log device (see [Formatting an Alarm Display](#)):

Field Name	Description
{Tag,n}	Alarm Tag
{TagEx,n}	Alarm Tag with Cluster Name prefix
{Cluster,n}	Cluster Name
{Name,n}	Alarm Name
{Native_Name,n}	Alarm Name in the <a href="#">native language</a>
{Desc,n}	Alarm Description
{Native_Desc,n}	Alarm Description in the native language
{Category,n}	Alarm Category
{Help,n}	Help Page
{Area,n}	Area
{Priv,n}	Privilege
{Type,n}	The type of alarm or condition: ACKNOWLEDGED CLEARED DISABLED UNACKNOWLEDGED
{Time,n}	The time at which the alarm changed state (hh:mm:ss). (Set the [Alarm]SetTimeOnAck parameter to use this field for the time the alarm is acknowledged.)
{Date,n}	The date on which the alarm changed state (dd:mm:yyyy). Note you can change the format used via the parameter [ALARM]ExtendedDate.
{DateExt,n}	The date on which the alarm changed state in extended format.

**Note:** If the **Tag** and **Name** fields are configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.

You can use the following fields for digital alarms only:

Field Name	Description
{State,n}	The current state of the alarm. This field may be used for Alarm Display Only. It is not applicable to Alarm Summary. ON OFF

You can use any of the following fields for time-stamped alarms only:

Field Name	Description
{Millisec,n}	Adds milliseconds to the {Time,n} field

You can use any of the following fields for analog alarms only:

Field Name	Description
{High,n}	High Alarm trigger value
{HighHigh,n}	High High Alarm trigger value
{Low,n}	Low Alarm trigger value
{LowLow,n}	Low Low Alarm trigger value
{Rate,n}	Rate of change trigger value
{Deviation,n}	Deviation Alarm trigger value
{Deadband,n}	Deadband
{Format,n}	Display format of the Variable Tag
{Value,n}	The current value of the analog variable
{State,n}	The current state of the alarm. This field may be used for Alarm Display Only. It is not applicable to Alarm Summary. DEVIATION RATE LOW LOWLOW HIGH HIGHHIGH CLEARED

You can use any of the following fields for **Hardware Alarms (Category 255) only**:

Field Name	Description
{ErrDesc,n}	Text string associated with a protocol (communication) error. This field is only associated with hardware errors and contains extra description associated with the error (e.g. if the error is associated with a device, the device name is returned; if the error is associated with a Cicode function, the function name is returned; if the error is associated with an I/O device, the error message is returned).
{ErrPage,n}	The page, device, etc. associated with the alarm.

You can use any of the following fields for alarm log devices only:

Field Name	Description
{LogState, <i>n</i> }	The last state that the alarm passed through. (This is useful when logging alarms to a device.)

You can use any of the following fields for multi-digital alarms only:

Field Name	Description
{State_desc, <i>n</i> }	The configured description (e.g. healthy or stopped) of a particular state. This description is entered when configuring the Multi-Digital Alarm Properties

Where *n* specifies the display field size.

Note the following points:

- If an alarm value is longer than the field it is to be displayed in (i.e. *n*), it will be truncated or replaced with the #OVR (“overflow of format width”) error.
- Only use the {State} field for display on the alarm pages. For summary pages use {SumState}. To log the state to a device, use {LogState}. State is the current state of the alarm, SumState is the state of the alarm when it occurred, and Log State is the state of the alarm at the transition.
- Use only the fields above to format an alarm display or alarm log device. Using alarm summary fields might produce unreliable results.

See Also [Alarm summary fields](#)

## Alarm summary fields

You can use any fields listed below (or a combination) to format an alarm summary display and an alarm summary device.

Format the alarm summary for an entire category of alarms by specifying field names in the **Summary Format** field of the Alarm Category Properties dialog box.

You can also use the [Alarm]DefSumFmt parameter to format the alarm summary, particularly if all of your alarm summary formats are to be the same.

Field Name	Description
{UserName, <i>n</i> }	The name of the user (User Name) who was logged on and performed some action on the alarm (e.g. acknowledging the alarm or disabling the alarm, etc.). Note that when the alarm is first activated, the user name is set to “system” (because the operator did not trip the alarm).
{FullName, <i>n</i> }	The full name of the user (Full Name) who was logged on and performed some action on the alarm (e.g. acknowledging the alarm or disabling the alarm, etc.). Note that when the alarm is first activated, the full name is set to “system” (because the operator did not trip the alarm).
{UserDesc, <i>n</i> }	The text related to the user event

Field Name	Description
{OnDate, <i>n</i> }	The date when alarm was activated
{OnDateExt, <i>n</i> }	The date (in extended format) when the alarm was activated (dd/mm/yyyy)
{OffDate, <i>n</i> }	The date when the alarm returned to its normal state
{OffDateExt, <i>n</i> }	The date (in extended format) when the alarm returned to its normal state (dd/mm/yyyy)
{OnTime, <i>n</i> }	The time when the alarm was activated
{OffTime, <i>n</i> }	The time when the alarm returned to its normal state
{DeltaTime, <i>n</i> }	The time difference between OnDate/OnTime and OffDate/OffTime, in seconds
{OnMilli, <i>n</i> }	Adds milliseconds to the time the alarm was activated.
{OffMilli, <i>n</i> }	Adds milliseconds to the time the alarm returned to its normal state.
{AckTime, <i>n</i> }	The time when the alarm was acknowledged
{AckDate, <i>n</i> }	The date when the alarm was acknowledged
{AckDateExt, <i>n</i> }	The date (in extended format) when the alarm was acknowledged (dd/mm/yyyy)
{SumState, <i>n</i> }	Describes the state of the alarm when it occurred
{SumDesc, <i>n</i> }	A description of the alarm summary
{Native_SumDesc, <i>n</i> }	A description of the alarm summary, in the native language
{AlmComment, <i>n</i> }	The text entered into the Comment field of the alarm properties dialog.
{Comment, <i>n</i> }	A comment the operator adds to an Alarm Summary entry during runtime. The comment is specified using the AlarmComment() function.
{Native_Comment, <i>n</i> }	Native language comments the operator adds to an Alarm Summary entry during runtime.

Where *n* specifies the display field size.

**Note:** You can also include in your Alarm Summary any alarm display field other than **State**. However, you cannot include any of the above Alarm Summary fields in an Alarm Display or Alarm Log Device, as this might produce unreliable results.

See Also [Changing the Order of the Alarm Summary Display](#)

## Changing the Order of the Alarm Summary Display

CitectSCADA allows you to customize the order in which alarms are displayed on the alarm summary page by using the `SummarySort` and `SummarySortMode` parameters. The `SummarySort` parameter allows you to display alarms according to `OnTime`, `OffTime`, and `AckTime`. `SummarySortMode` determines if the alarms will be arranged in ascending or descending order. (The order set using these parameters will override the alarm category priority order.)

See Also [Formatting an Alarm Display](#)

## Using Alarm Properties as Tags

Alarm properties can be used wherever variable tags can be used (except in alarm descriptions). For instance, you can provide the operator with a visual indication when the alarm **CV110\_FAULT** is active. When it is active, **CV110\_FAULT.On** will be TRUE; when it is inactive, **CV110\_FAULT.On** will be FALSE. For example, CV110\_FAULT.On could be entered as the fill color expression in a graphics object. When the conveyor has a fault, the graphics object will change color.

To use an alarm property as a tag, it must be formatted as follows: Alarm tag (e.g. **CV100\_STOP**) followed by a full stop (.) then the property (e.g. **Category**). The completed alarm property would then be **CV100\_STOP.Category**.

**Note:** If you intend to use time-stamped digital or time-stamped analog alarm properties as variable tags, you need to ensure they are configured correctly with the required data being pushed to the relevant variables via the Cicode function [AlarmNotifyVarChange](#).

See [Time-stamped Digital Alarms](#) and [Time-stamped Analog Alarms](#) for more details on how these alarms operate.

See Also [Supported alarm properties](#)  
[Writing to alarm properties](#)  
[Setting up alarms](#)

### Supported alarm properties

The following properties can be used for all alarm types. Remember, the return value relates to the description. For example, for a digital, if 1 is returned, that means the description is TRUE, whereas 0 (zero) means it is FALSE.

Property	Description	Return Type
.On*	Alarm active	Digital
.Ack	Alarm acknowledged	Digital
.Disabled	Alarm disabled (see note below)	Digital
.Time	32 bit value of the time the alarm was triggered	Long
.Tag	Alarm tag	String (80 bytes)
.Name	Alarm name	String (80 bytes)
.Category	Alarm category	Integer
.Priority	Alarm priority	Integer

\* The .On property for Analog alarms is true if any alarms associated with the alarm tag are active.

**Note:** Once an alarm is disabled, it cannot be re-enabled unless you use the function `AlarmEnable()` or `AlarmEnableRec()`



For digital alarms, time-stamped digital alarms, and advanced alarms, the following properties can also be used:

Property	Description	Return Type
.Desc	Alarm description	String (128 bytes)
.Delay	Alarm delay	Long

**Note:** Desc exists for all alarm types but will not return meaningful data for analog or time-stamped analog alarms. Desc returns an empty string that indicates whether the read succeeded; the write indicates that the tag was resolved and that the write request was sent.

For analog alarms and time-stamped analog alarms, the following properties can also be used:

Property	Description	Return Type
.Value	Alarm tag value	Real
.Setpoint	Setpoint	Real
.HighHigh	High High	Real
.High	High	Real
.LowLow	Low Low	Real
.Low	Low	Real
.DeadBand	Deadband	Real
.Rate	Rate	Real
.Deviation	Deviation	Real
.HHDelay	High High delay	Long
.HDelay	High delay	Long
.LDelay	Low delay	Long
.LLDelay	Low Low delay	Long
.DevDelay	Deviation delay	Long

For the digital properties below, only one can be true at any point in time for each alarm. They are arranged in order of priority, from lowest to highest.

.DVL	Deviation alarm triggered (Low)	Digital
.DVH	Deviation alarm triggered (High)	Digital
.R	Rate of Change alarm triggered	Digital
.L	Low alarm triggered	Digital
.H	High alarm triggered	Digital
.LL	Low Low alarm triggered	Digital
.HH	High High alarm triggered	Digital

**Note:** DVL and DVH are only evaluated if Deviation > 0. R is only evaluated if Rate > 0.

Some alarm properties return configuration data. If the user has not defined this information, the following defaults are provided:

Property	Default
.Setpoint	0
.HighHigh	3.4e+38
.High	3.4e+38
.LowLow	-3.4e+38
.Low	-3.4e+38
.Rate	0
.Deviation	0
.Deadband	0
.Category	0
.Priority	0

See Also [Writing to alarm properties](#)  
[Setting up alarms](#)

## Writing to alarm properties

If you have the required access rights, you can write to the following alarm properties. (Remember, the value you write to the property relates to the description. For example, if you set a digital alarm property to 1, you are making the description TRUE. If you set it to 0 (zero), you are making it FALSE.)

Property	Description	Input Type
.Ack	Alarm acknowledged (once acknowledged cannot be "unacknowledged")	Digital
.Deadband	Alarm Deadband	Real
.Deviation	Deviation from setpoint	Real
.Disabled	Alarm disabled	Digital
.HighHigh	High High	Real
.High	High	Real
.LowLow	Low Low	Real
.Low	Low	Real
.HHDelay	High High delay	Long
.HDelay	High delay	Long
.LDelay	Low delay	Long
.LLDelay	Low Low delay	Long
.DevDelay	Deviation delay	Long

Analog alarm thresholds can also be changed using the `AlarmSetThreshold()` function.

Note the following:

- The alarm tag must be unique.

- The alarms databases in all included projects on the alarms server and the CitectSCADA display [client](#) computer must be identical.

See Also [Supported alarm properties](#)  
[Setting up alarms](#)

## Setting up alarms

To use alarm properties, you must configure an alarm I/O device in your project. Configure the following fields in the I/O devices form:

Name	User supplied unique name for the I/O device
Number	Network wide I/O device number
Address	Leave this field blank
Protocol	ALARM
Port Name	ALARM

For example:

Name	Alarm_Device
Number	12
Address	
Protocol	ALARM
Port Name	ALARM

The alarm I/O device only works on a computer that is defined as an alarms server and an [I/O server](#). After you have configured the alarm I/O device, use the Computer Setup Wizard to set up the alarms server, defining it as an I/O server, even if it has no physical devices attached to it.

See Also [Supported alarm properties](#)  
[Writing to alarm properties](#)

## Handling Alarms at Runtime

When an alarm is triggered, it becomes active. The active state of a digital alarm is ON, while the active state of an analog alarm varies, depending on the type of alarm (for instance HIGH, LOW LOW, RATE, and so on). When an operator acknowledges the alarm, its state changes to ACKNOWLEDGED. When the alarm is reset (when the conditions that caused the alarm have been rectified), its state changes to OFF.

CitectSCADA displays alarms on the standard alarm display page. To acknowledge an alarm, an operator either selects the alarm with the mouse and clicks the left mouse button, or moves the cursor onto the alarm and presses the **Enter** key. Alternatively, the operator can acknowledge all alarms by clicking **Alarm Ack**. When an alarm is acknowledged, its display color (on screen)

changes. Acknowledged alarms remain on screen until their state changes to OFF.

If an alarm is faulty or unnecessary, an operator can disable it. CitectSCADA ignores disabled alarms until they are reenabled. (You must define a command that uses the `AlarmDisable()` function to disable alarms.)

To maintain a history of alarm activity, CitectSCADA keeps an event log of all alarms. This log stores the time when each alarm was activated, acknowledged, and reset. You can display all alarms from the event log (including disabled alarms) on the alarm summary page.

You must create a page called **Summary** based on the `AlarmSummary` template, so that the alarm summary button (on other pages such as the menu page) operates correctly. (The alarm summary button calls the `PageSummary()` function.).

Operators can add comments to any alarm in the summary log. (You must define a command that uses the `AlarmComment()` function to add comments to an alarm.)

**Note:** If you have many alarms on the alarm page or alarm summary page, use **Page Up** and **Page Down** to scroll through the list.

#### To create an alarm page:

- 1 In **Citect Explorer**, double-click **Create New Page** in the **Graphics|Pages** folder.
- or -
- 1 In the **Graphics Builder**, choose **File | New** and then click **Page**.
- 2 Select the alarm template you want to use. Use the **Alarm** template to create a page to display configurable alarms, the **Summary** template for summary alarms, the **Disabled** template for disabled alarms, and the **Hardware** template for hardware alarms.
- 3 Choose **File|Save**.
- 4 Specify a name in the page title field. The new page name should match the template name. For example, call the new hardware alarm page **Hardware**.
- 5 Click **OK**.

**Note:** You can also create your own (non-standard) alarm pages. The easiest way to do this is by copying and modifying the standard alarm templates.

#### To display an alarm page at runtime:

- 1 Create an alarm (or hardware alarm) page in your project if you have not already done so. The page should be called **Alarm** for a configurable alarm page, and **Hardware** for a hardware alarm page.

- 2 Create a new keyboard command or a button, to call the page at runtime. You can also add a touch command to an existing screen object.
- 3 In the command field, enter **PageAlarm()** to display the configurable alarms page, or **PageHardware()** to display the hardware alarms page.
- 4 Configure other properties as required.
- 5 Click **Add** to append a new record, or **Replace** to modify an existing record.

**Note:** If using the standard CitectSCADA page templates, you don't usually need to create a command to display the page: the commands are already built in.

To display a customized alarm page (with a non-standard name), use the `PageDisplay()` function to display the page, followed by the `AlarmSetInfo()` function as required.

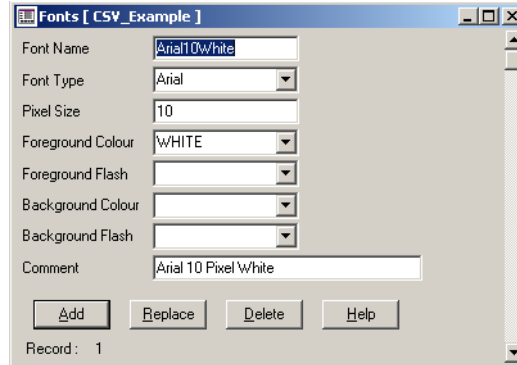
## Using CitectSCADA Fonts

Alarm categories, Cicode functions, and button objects allow you to use predefined fonts, known as CitectSCADA fonts, to display text. You can also configure your own CitectSCADA fonts; for details see [Configuring Custom Color Fonts](#).

**Note:** If an animation is associated with an I/O device that fails at startup or goes offline while CitectSCADA is running, the associated animation is grayed on the relevant graphics pages (because the values are invalid). You can disable this feature with the `[Page]ComBreak` parameter. See [Handling Communication Errors in Reports](#).

### To define a CitectSCADA font:

- 1 In the Project Editor or the Graphics Builder, choose **System | Fonts**. The Fonts dialog box appears.
- 2 Enter your font properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.



See Also [Fonts properties](#)

## Fonts properties

Use the Fonts dialog box to define your font properties. Fonts have the following properties:

### Font Name

The name of the font (16 characters maximum). Unlike background text, strings, and numbers (which can use any standard Windows font), you must define a CitectSCADA font for animated text.

### Font Type

Any text font supported by Windows (16 characters maximum). Choose a font type from the menu.

You can also specify bold, italic, and underlined text. To specify any of these options, append the appropriate specifier to the Font Type, for example:

Font Type      Courier,B

Specifies bold characters.

Font Type      Helv,I

Specifies italic characters.

Font Type      TmsRmn,U

Specifies underlined text.

You can also specify multiple options, for example:

Font Type      Courier,B,I,U

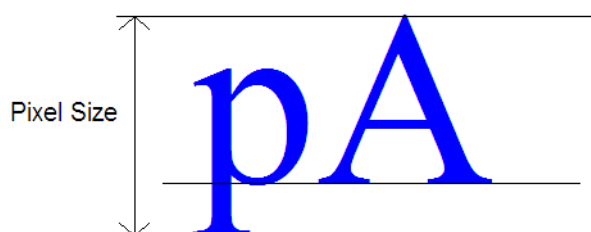
Specifies bold, italic, and underlined characters.

**Note:** To use a font that is not displayed in the menu, you must install the font on your computer before you can use it in your CitectSCADA system. To view, install, or remove Windows fonts, use the Windows Control Panel (Fonts

Option). Refer to your Windows documentation for details. (If you are using CitectSCADA on a network, all computers must have the font installed.)

### Pixel Size

The size of the displayed text (16 characters maximum). You can specify text fonts in pixels or points. The following figure shows how a font size defined in pixels relates to the displayed characters.



To specify a point size, enter a negative number in the **Pixel Size** field; for example:

Font Size        -10

specifies a ten-point font size. Note that you can only specify a point size as a whole number (integer).

If you have not installed the Font Type (or Pixel Size) on your system, a default font and/or size is used that most closely resembles the font and/or size you have specified.

If you use a point size, the size remains constant across all screen resolutions. On low-resolution screens, the font appears larger than on high-resolution screens, which might cause misalignment of animation. Only use a point size to display text on computer screens of the same resolution.

### Foreground Color

The foreground color of the displayed text (i.e., the color of the text characters). You can use a predefined color label (accessible via the menu), a user-defined custom label, or the RGB encoded number generated by the function `MakeCitectColour` (see the *Cicode Reference Guide*).

**Note:** Do not confuse predefined labels with the color [Name](#) feature associated with the Color Picker. You cannot use color names with this dialog; doing so generates a compile error.

### Foreground Flash

The secondary color applied to the font if using flashing color for your text characters. You can use a predefined color label (accessible via the menu), a

user-defined custom label, or the RGB encoded number generated by the function `MakeCitectColour` (see the *Cicode Reference Guide*).

If you do not specify a color, the text remains solid.

### Background Color

The background color of the displayed text. You can use a predefined color label (accessible via the menu), a user-defined custom label, or the RGB encoded number generated by the function `MakeCitectColour` (see the *Cicode Reference Guide*).

This property is optional. If you do not specify a background color, it defaults to transparent.

### Background Flash

The secondary color applied to the background if using flashing color. You can use a predefined color label (accessible via the menu), a user-defined custom label, or the RGB encoded number generated by the function `MakeCitectColour` (see the *Cicode Reference Guide*).

If you do not specify a color, the text remains solid.

### Comment

Any useful comment (48 characters maximum).

## Configuring Custom Color Fonts

From CitectSCADA version 6.0 onwards, colors are referenced by using hex codes for red, green and blue (RGB) values. You can view RGB values (in decimal) of a selected color by choosing **Tools | Edit Favorite Colors** in Graphics Builder.

If, for example, the color you want to use has the values R = 128, G = 170, B = 213, you can convert these to their hex equivalents (R = 0x80, G = 0xAA, B = 0xD5).

If you define a label for your color, you can use decimal or hex values. For example:

- **Label Name:** Plate\_Blue
- **Expression:** 0x80AAD5 (or 8432341)

Once you have defined your label, you can create fonts to use in your project for alarm categories, Cicode functions, and button objects. For example, use the Fonts dialog box to define a font with the following properties:

- **Font Name:** Plate\_Font
- **Font Type:** Arial
- **Pixel Size:** 12



---

■ **Foreground Color:** Plate\_Blue

For more information, refer to Knowledge base article Q4024.

See Also [Using CitectSCADA Fonts](#)



## Chapter 23: Configuring Events

---

You can use an event to trigger an action, such as a command or set of commands. For example, an operator can be notified when a process is complete, or a series of instructions can be executed when a process reaches a certain stage.

Events must be enabled for events to run. Use the *CitectSCADA Computer Setup Wizard* (Custom setup) to enable Events. If using CitectSCADA on a network, you can process events on any CitectSCADA computer (or all computers).

**Note:** Events do not provide a service with redundancy. If you want to run an event with redundancy, use reports.

**To define an event:**

- 1 Choose **System | Events**. The Events dialog box appears.
- 2 Enter the event properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record

**Note:** The event server must be enabled for events to work.

See Also [Events Properties](#)  
[Running Events](#)

### Events Properties

[Events](#) have the following properties:

#### Name

For a single computer system, specify **GLOBAL** for the event name:

Name	GLOBAL
------	--------

If you are using CitectSCADA on a network and want to run an event on all computers, specify **GLOBAL** for the event name. If you want to run an event only on specific computers, specify an event name and use the CitectSCADA Computer Setup Wizard (Custom setup) to specify which CitectSCADA computer(s) will run the event. The event name does not have to be unique: you can specify many events with the same name.

Enter a value of 16 characters max.

**Note:** Do not use spaces in the name of your event; use the underscore character (\_) instead.

#### Time

The time of day to synchronize the **Period** in hh:mm:ss (hours:minutes:seconds). If you do not specify a time, **Period** is synchronized at 00:00:00 (that is, midnight). Enter a value of 32 characters maximum.

### Period

The period to check for the event, in hh:mm:ss (hours:minutes:seconds). Enter a value of 32 characters maximum. Alternatively you can specify:

- A weekly period by entering the day of the week to check for the event, for example, Monday, Tuesday, Wednesday, etc.
- A monthly period by entering the day of the month to check for the event, for example, 1st, 2nd, 3rd, 4th, 5th, etc.
- A yearly period by entering the day and the month to check for the event, for example, 1st January, 25th February, and so on. The day and month must be separated by a space.

If you do not specify a period or time, the period defaults to one second. If you do not specify a period, but do specify the time, the period defaults to one day.

### Trigger

The Cicode [expression](#) (or Variable tag) which is used to determine whether the event Action is executed. This expression is checked every one second. Enter a value of 64 characters maximum.

### Action

The commands to execute. Enter a value of 64 characters maximum. These commands will execute in the following circumstances:

- When the specified **Time** and **Period** occurs, and the **Trigger** condition is TRUE or blank.
- When the **Trigger** becomes TRUE, and the **Time** and **Period** field are blank. The Trigger must become FALSE and TRUE again for the action to re-execute.

### Comment

Any useful comment. Enter a value of 64 characters maximum.

## Running Events

The event server must be enabled for events to work. You can run an event automatically:

- At a specified time and period.
- When a trigger condition becomes TRUE.
- When a trigger condition is TRUE at a specified time and period.

See Also [Specifying times and periods](#)  
[Using triggers](#)

## Specifying times and periods

The Period determines when the event is run. You can specify the period in hh:mm:ss (hours:minutes:seconds), for example:

Period	Comment
1:00:00	Run the event every hour
6:00:00	Run the event every six hours
72:00:00	Run the event every three days
Monday	Run the event each Monday
15th	Run the event on the 15th of each month
25th June	Run the event on the 25th of June

You can also specify the time of day to synchronize the event, for example:

Period	Comment
6:00:00	Synchronize the event at 6:00 am
12:00:00	Synchronize the event at 12:00 midday

The Time synchronizes the time of day to run the event and, with the Period, determines when the event is run; for example:

Time	6:00:00
Period	1:00:00

In this example, the event is run every hour, on the hour. If you start your runtime system at 7:25am, your event is run at 8:00am, and then every hour after that.

See Also [Using triggers](#)

## Using triggers

You can use any Cicode expression (or variable tag) as a trigger for an event. If the result of the expression (in the **Trigger** field) becomes TRUE, and if the **Time** and **Period** fields are blank, the event is run. For example:

Time	
Period	
Trigger	RCC1_SPEED<10 AND RCC1_MC

This event is only run when the expression (Trigger) becomes TRUE, that is, when the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10. The expression must become FALSE and then TRUE again before the event is run again.

If you use the Time and/or Period fields, the Trigger is checked at the Time and/or Period specified, for example:

Time	6:00:00
------	---------

---

Period	1:00:00
Trigger	RCC1_SPEED<10 AND RCC1_MC

This event is run each hour, but only if the expression (Trigger) is TRUE (that is, if the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10).

See Also [Running Events](#)

## Chapter 24: Using Accumulators

Accumulators track incremental runtime data, such as motor run hours, power consumption, and downtime. You set a trigger (e.g., motor on) to increment three counters:

- The number of times the accumulator is triggered (e.g. the number of starts for the motor).
- The run time, in steps of 1 second.
- A totalized value, by an increment you define (e.g. the current).

The accumulated data is stored as variable tags in an I/O device. Variable tags are read at CitectSCADA startup and updated regularly while the trigger is active. You can monitor and display accumulated data by animating, trending, or logging the variable tags.

**Note:** You can control (re-read or reset) any accumulator at runtime by using the `AccControl()` Cicode function.

**To configure an accumulator:**

- 1 Choose **System** | **Accumulators**. The Accumulators dialog box appears.
- 2 Enter the accumulator properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

You use the Accumulator Properties dialog box to configure your accumulators.

The screenshot shows the 'Accumulators [ CSV\_Example ]' dialog box. It contains the following fields and values:

- Name: PUMP\_A\_ACC
- Cluster Name: (empty)
- Trigger: PUMP\_A\_CMD
- Run Time: PUMP\_A\_ART
- No. of Starts: PUMP\_A\_AS
- Totaliser Inc: 10
- Totaliser: PUMP\_A\_TOT
- Comment: Pump\_A Accumulator

At the bottom of the dialog are four buttons: Add, Replace, Delete, and Help. Below the buttons, the status bar indicates 'Record : 1'.

See Also [Accumulator Properties](#)

## Accumulator Properties

Accumulators have the following properties:

### Name

The name of the accumulator. Enter a value of 16 characters or less.

### Cluster Name

The name of the cluster that this accumulator runs on. If the Cluster Name is not set, then CitectSCADA considers this accumulator to run on all defined clusters.

### Trigger

The Cicode [expression](#) (or variable tag) to trigger the accumulator. If the result of the expression (in this field) is TRUE, accumulation starts. If the result of the expression (in this field) becomes FALSE, accumulation stops. Enter a value of 64 characters or less.

The frequency with which CitectSCADA checks the trigger is controlled by the [Accumulator]WatchTime parameter.

### Run Time

The variable (tag) that contains the run time (in seconds). Enter a value of 32 characters or less. On startup, CitectSCADA reads this value. CitectSCADA then increments its local copy of this variable (while **Trigger** is TRUE) and writes the variable (back to the I/O device) at a frequency determined by the [Accumulator]UpdateTime parameter.

### No. of Starts

The variable (tag) that contains the number of starts (i.e. the number of times the trigger changes from FALSE to TRUE). Enter a value of 32 characters or less. On startup, CitectSCADA reads this value. CitectSCADA then increments its local copy of this variable and writes the variable (back to the I/O device) at a frequency determined by the [Accumulator]UpdateTime parameter.

### Totalizer Inc

Any Cicode expression (or variable tag) to add to (increment) the **Totalizer** variable while the **Trigger** condition is TRUE. Enter a value of 64 characters or less.

### Totalizer

The variable (tag) that contains the totalized value. Enter a value of 32 characters or less.



On startup, CitectSCADA reads this value. Each time CitectSCADA checks the trigger and the trigger is TRUE, CitectSCADA adds the value in the **Totalizer Inc** field to its local copy of this **Totalizer** variable. CitectSCADA writes the new **Totalizer** variable (back to the I/O device) at a frequency determined by the [Accumulator]UpdateTime parameter.

For example, if you configure an accumulator for a motor, and **Totalizer Inc** is the current (amperage) used by the motor, then **Run Time** will contain the time (in seconds) that the motor has run, **No. of Starts** will contain the number of times the motor was started, and **Totalizer** will contain the total current used by the motor. The average current used by the motor is **Totalizer/Run Time**.

### Comment

Any useful comment. Enter a value of 48 characters or less.

**Note:** The following fields are implemented with extended forms (press F2).

### Privilege

The privilege required by an operator to perform operations on the accumulator (by using accumulator functions). Enter a value of 16 characters or less.

### Area

The [area](#) to which the accumulator belongs. Only users with access to this area (and any required privileges) will be able to perform operations on the accumulator. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to perform operations on the accumulator. Enter a value of 16 characters or less.

Note the following:

- CitectSCADA stores the run time, number of starts, and totaled value in variables in your I/O device. This allows easy access to these variables and, because they are saved in the I/O device, their values are saved if CitectSCADA is shutdown. You can increase system performance by storing these variables in a [disk I/O device](#). (If you store these variables in your external I/O devices, CitectSCADA will consume communication bandwidth updating the variables.
- If using CitectSCADA on a network, you can use a redundant Disk I/O device to secure these variables.
- The accumulator server runs as part of the reports server. If you have a redundant reports server, you must use a primary/standby configuration to stop the accumulators running on both reports servers. Use the CitectSCADA Computer Setup Wizard to define the reports servers.
- You can control (re-read or reset) any accumulator at runtime by using the `AccControl()` Cicode function.

# Chapter 25: Logging and Trending Data

---

Since CitectSCADA version 6.0, the Process Analyst (an in-built trend visualization tool) has superseded the functionality of trend graphs. However, trend graphs are still supported. Note that if you plan to use SPC trends, you *must* use trend graphs, since SPC is not supported by the Process Analyst.

For details, see the [Process Analyst](#) Help.

**Note:** If you use the link above, you should select **Open this file from its current location** from the dialog that appears.

See Also [Trending Data](#)  
[Trend Graphs](#)  
[Printing Trend Data](#)  
[Exporting Trend Data](#)  
[Using Trend History Files](#)  
[Using Path Substitution](#)  
[Debugging Trending](#)

## Trending Data

The trend system can help you better understand your plant and equipment's performance. Trending can be used for dynamic visual analysis (trend and SPC graphs), production records, or for regularly recording the status of equipment for efficiency and preventive maintenance.

Using trend tags, you can specify the data you want to collect from your I/O device variables. This information can be logged at regular intervals ([periodic trend](#)), or only when an event occurs (event trend). Event trends are used for trending data that is not time-based, for example, for a product as it comes off an assembly line. Trend data is usually saved on disk for analysis or displayed on a trend graph.

The trend system is based on real-time samples. The trend system expects a return of one data point each time it samples the data. Although gaps in the data can be filled, ensure that your field device can return data values at the rate you specify (especially if you are using sample periods of less than 100 ms).

CitectSCADA can collect and store any amount of data. The only restriction on the amount of data that you can store is the size of the hard disk on your computer. (CitectSCADA uses an efficient data storage method - ensuring that space on your computer's hard disk is maximized.) For long term storage, you can archive the data to disk or tape (without disrupting your runtime system). For efficient storage, store trend files on a compressed volume.

**Note:** If you are trending data across a network ([distributed processing](#)), it is recommended that you enable time synchronization using the Computer Setup Wizard.

You might also consider staggering your trend sample requests using the [Trend]StaggerRequestSubgroups parameter.

See Also [Configuring trend tags](#)

Configuring trend tags

You use the Trend Tags dialog box to configure your trend tags.

To configure a trend tag:

- 1 Choose **Tags | Trend Tags**. The Trend Tags dialog box appears. (Press **F2** to view the extended Trend Tag form.)
- 2 Enter your trend tag properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Trend Tag Properties](#)

Trend Tag Properties

**Trend Tag Name**

The name assigned to the trend data (79 characters maximum). If the trend tag is logging a particular variable, you should use a 16-character name that resembles the 32-character name of the related variable tag. This will mean an association between the two is easily recognizable. The name must be unique to the cluster. Trend Tag names must adhere to the [Tag name syntax](#). If you have many tags, use a naming convention (see [Using structured tag names](#)). This makes it easier to find and debug your tags.

**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

The first 8 characters of your trend tag names must not be the same as the first 8 characters of your SPC tag names within the cluster(s) that run this trend.

**Cluster Name**

The name of the cluster that runs this trend. If the Cluster Name is not set, then CitectSCADA considers this trend to run on all defined clusters.

**Expression**

The logged value of the trend tag (64 characters maximum). You can log individual variables by using a variable tag. For example:

Expression	LT131
Comment	Logs the Variable Tag LT131

The value of the process variable LT131 is logged. Variable LT131 must be defined as a variable tag.

You can also log any Cicode expression or function, for example:

Expression	LT131/COUNTER
Comment	Logs Variable Tag LT131 divided by the Variable Tag COUNTER

**Note:** When a variable tag is used in the expression field of a trend tag property, the **Eng Zero Scale** and **Eng Full Scale** fields of that variable tag must be set appropriately, or data will be lost because the trend logs negative values as invalid.

### Trigger

The Cicode expression (or variable tag) that triggers data logging (64 characters maximum). For example:

Trigger	LT131<50
---------	----------

In this example, logging occurs when the value of the variable tag (LT131) falls below 50.

For a periodic trend, data is logged only while the value of the trigger is TRUE. (The trend graph will still scroll, but will display <GATED> where the trigger is FALSE.) In the above example, data is logged continuously while the value of LT131 remains less than 50. Logging ceases when the value rises to (or above) 50. Logging does not occur again until the value of LT131 falls below 50.

You do not have to specify a trigger for a periodic trend. If you do not specify a trigger for a periodic trend, logging occurs continuously.

For an event trend, data is logged once when the value of the trigger changes from FALSE to TRUE. In the above example, one sample is logged when the value of LT131 first becomes less than 50. Another sample is not logged until the value of LT131 rises to (or above 50) and again falls below 50.

### Sample Period

The sampling period of the data (16 characters maximum). You can either enter a period of your own, or choose one from the menu.

Sampling periods of greater than one second should be entered in hh:mm:ss (hours:minutes:seconds) format. If you enter a single digit, without the colon (:), it will be considered a second. For example, if you enter **2**, it will be interpreted as 2 seconds.

Sampling periods of less than one second must be entered as decimals. For example, to enter a period of 200 milliseconds, you would enter **0.2**.

If the sample period is less than one second, then one second must be divisible by the period (to give an integer). For example, a sample period of 0.05 is valid, because  $1/0.05 = 20$ , whereas a sample period of 0.3 is not valid because  $1/0.3 = 3.333...$ .

Note the following:

- Your I/O device must be capable of providing data at the specified rate, otherwise gaps will appear in the trend data and/or the hardware alarm **Trend has missed samples** will be evoked. You can fill gaps in the file and graph using the [Trend]GapFillTime parameter. Gaps in the graph only can be filled using the TrnSetDisplayMode() function.
- If trends with a sample period of less than a second are shared by several clients across a network (distributed processing), you should enable time synchronization using the Computer Setup Wizard.

CitectSCADA checks the **Trigger** each sample period. If the Trigger is TRUE (or has just changed from FALSE to TRUE, in the case of event trends), CitectSCADA logs the value of the **Expression**.

#### Examples

Sample Period	Comment
30	Logs data every 30 seconds
10:00	Logs data every 10 minutes
10:00:00	Logs data every 10 hours
2:30:00	Logs data every 2 and a half hours

The sampling period of the fastest trend on the page is taken as the default value for the [display period](#) of the page.

This property is optional. If you do not specify a sample period, the sampling period defaults to 10 seconds.

**Note:** If you edit this property in an existing project, you must delete the associated trend files before running the new runtime system. (For location of the trend files, see [File Name](#).)

#### Type

The type of trend (32 characters maximum):

- **TRN\_PERIODIC** - A trend that is sampled continuously at a specified period.
- **TRN\_EVENT** - A trend that is sampled when a particular event is triggered in the plant.
- **TRN\_PERIODIC\_EVENT** - A trend that is sampled on a continuous basis while a trigger value is TRUE. The trigger can be a Cicode expression or variable tag.

### Comment

Any useful comment (48 characters maximum).

**Note:** The following fields are implemented with extended forms (press F2).

### File Name

The file where the data is to be stored (64 characters maximum). Specify the full path or use path substitution.

When CitectSCADA collects data from your plant floor, it stores the data in a file on the hard disk of your computer. When CitectSCADA subsequently uses the data to display a trend or SPC graph, it reads the data from this file. (CitectSCADA uses a separate file for each trend tag.)

By default, CitectSCADA stores the file in the C:\Documents and Settings\All Users\Application Data\Citect\CitectSCADA\data directory on the hard disk where CitectSCADA is installed. The default name of the file is the first eight characters of the trend tag name. However, you can specify an alternate file name. If you do specify a file name, you can specify the full path, for example:

File Name                      C:\DATA\TRENDS\TANK131

or use the path substitution string:

File Name                      [DATA]:TANK131

where [DATA] specifies the disk and path for the data. Use path substitution to make your project more portable.

**Notes:**

- With CitectSCADA Versions 5.xx, you can't store trend files in the bin, runtime, backup or user directories or any subdirectories of these. If you have existing Version 3.xx or 4.xx projects that use these directories to store trend files, the path for these will have to be changed to the Data directory.
- The trend system will buffer the acquired data before saving it to a file. The [Trend]CacheSize parameters determine the buffer sizes for returned data.

The File Name property is optional. If you do not specify a file name, the file name defaults to C:\Documents and Settings\All Users\Application Data\Citect\CitectSCADA\data\*<name>* on the hard disk where you installed CitectSCADA. *<name>* is the trend tag name. If you use this property, ensure that no other trend tags have the same name, otherwise data might be lost.

**Notes:**

- Do not use a file extension when specifying a file name. If you edit this property (change the file name or path) in an existing project, all existing SPC data is ignored.
- This file name must be different to your SPC tag file names.

**Storage Method**

Select **Scaled** or **Floating Point**. Scaled is a 2-byte data storage method; floating point uses 8 bytes.

Floating point storage has a dramatically expanded data range in comparison to scaled storage, allowing values to have far greater resolution. However, you need to consider that it also uses a lot more disk space. Scaled should be used where compatibility with pre-V5.31 trend history files is required.

If you do not specify a storage method, it is set to **Scaled** by default.

**Note:** If you edit this property in an existing project, you must delete the associated trend files - before you run the new runtime system. (For location of the trend files, see the [File Name](#).)

**Privilege**

The privilege required by an operator to display the trend data on a trend (16 characters maximum).

**Area**

The [area](#) to which the trend data belongs (16 characters maximum).

**Eng Units**



The engineering units of the variable/expression being logged (8 characters maximum). The engineering units are used by the trend scales and trend cursor displays.

#### Format

The format of the variable/expression being logged (10 characters maximum). The format is used by the trend scales and trend cursor displays.

This property is optional. If you do not specify a format, the format defaults to #####.

#### No. Files

The number of history files stored on your hard disk (for this tag) (4 characters maximum).

If you do not specify the number of files, 2 history files are stored on your hard disk. The maximum number of files you can specify per trend tag is 270.

**Note:** If you edit this property in an existing project, you must delete the associated trend files - before you run the new runtime system. (For location of the trend files, see the [File Name.](#))

#### Time

The time of day to synchronize the beginning of the history file, in hh:mm:ss (32 characters maximum). If you do not specify a time, the file is synchronized at 0:00:00 (i.e. midnight).

**Note:** If you edit this property in an existing project, you must delete the associated trend files - before you run the new runtime system. (For location of the trend files, see the [File Name.](#))

#### Period

The period of the history file, in hh:mm:ss (32 characters maximum). Alternatively, you can:

- Specify a weekly period by entering the day of the week on which to start the history file, e.g. Monday, Tuesday, Wednesday, etc.
- Specify a monthly period by entering the day of the month on which to start the history file, e.g. 1st, 2nd, 3rd, 4th, 5th, etc.
- Specify a yearly period by entering the day and the month on which to start the history file, e.g. 1st January, 25th February, etc. The day and month must be separated by a space.

If you do not specify a period, the period defaults to Sunday (weekly).

When deciding on a period setting, note that the performance of a trend viewer (be it the existing CitectSCADA [client](#) or Process Analyst) may be

impacted by the size of a trend file. This is particularly true when displaying event-based trend data.

**Note:** If you edit this property in an existing project, you must delete the associated trend files before you run the new runtime system. (For location of the trend files, see [File Name.](#))

## Trend Graphs

Trend graphs visually represent past and current activity of plant-floor data, building a picture over time of how a variable (such as product output, level, temperature, and so on) changes or how a device or process is performing. You can monitor current activity as it happens and scroll back through time to view trend history.

As the values of variables change over time or as events happen, the graph moves across the page. The latest values are always displayed. You can scroll back through historical data to display past values of the variable (or process).

You can trend any single variable or Cicode expression. You can display any number of trends on the screen simultaneously, even if they have different sample periods. You can also display up to eight trend tags (pens) in any trend window.

A trend graph can only communicate with one cluster, therefore you cannot mix trends from multiple clusters on a single trend graph. To graph trends from multiple clusters you will need to use multiple trend graphs, or, use the Process Analyst which has no such restrictions.

Historical data collection continues even when the display is not active. You can switch between pages without affecting trend graphs. Trend data acquisition and storage of data (in trend history files) continues even when the display is not active.

You can use the following standard trends:

- A single full page trend, where one trend window displays on a [graphics page](#).
- A double full page trend, where two trend windows display on a graphics page.
- A zoom trend with two trend windows and added functionality for zooming.
- A pop-up trend that you can 'pop up' anywhere (in a separate window) on your computer screen.
- User-defined trends that you can position anywhere on any graphics page.

**Note:** Variable tags can also be visually trended using an SPC Control Chart. Statistical Process Control (SPC) is a facility that enables you to control the quality of materials, manufactured products, services, etc. This quality control is achieved by collecting, arranging, analyzing, and testing sampled data in a manner that detects lack of uniformity or quality.

See Also [Creating trend pages](#)

## Creating trend pages

You can use any of the predefined trend templates for your trend pages, or use a predefined template to produce your own trend templates. You can draw a trend background (such as gridlines) on your trends.

**To configure a trend page:**

- 1 Click **New Page**, or choose **File | New**.
- 2 Select **Type: Page**.
- 3 Choose the **Resolution** (size) of the trend page.
- 4 Choose a trend **Template** for the trend page:
  - **Singletrend** - One trend on the page
  - **Doubletrend** - Two trends on the page
  - **Eventtrend** - One event trend on the page
  - **Zoomtrend** - Two trends on the page (one window for zooming)
  - **Poptrend** - A single trend on the page (for display in a pop-up window)
- 5 Click **OK**.

To create multiple trend pages, you can:

- Create a trend page for each set of trends to display in the runtime system.
- Create a single trend page and use the `PageTrend()` function to display trends as required. With this function, you can display all the trends in the system with a single trend page
- Create the trend page with the Graphics Builder, and set all the pen names to blank. You then display that page by calling this function and passing the required trend tags (up to 8). Call the `PageTrend()` function from a menu of trend pages.

See Also [Trend interpolation](#)

## Trend interpolation

Trend interpolation is used to define the appearance of a trend graph when the incoming samples fall out of synchronization with the [display period](#) or when samples are missed.

For example, a particular trend might be sampled five times between each update of the trend graph. As only one value can be displayed for each update, a

single value must be used that best represents the five samples; and that could be the highest value, the lowest value, or an average.

To define how CitectSCADA calculates the value to use, you set a particular trend interpolator display method.

The following table shows the available interpolator display methods, grouped into *condense methods* (where the [display period](#) is longer than the sample period) and *stretch methods* (where the display period is less than or equal to the sample period).

Condense methods	Stretch methods
<b>Average</b> ( <i>default</i> ) - this displays the average of the samples within the previous display period	<b>Step</b> ( <i>default</i> ) - This method simply displays the value of the most recent sample.
<b>Minimum</b> - This displays the lowest value that occurred during the previous display period.	<b>Ratio</b> - This method uses the ratio of sample times and values immediately before and after the requested time to interpolate a "straight line" value.
<b>Maximum</b> - This displays the highest value that occurred during the previous display period.	<b>Raw Data</b> - This method displays the actual raw values.

The interpolation display method is set via TrnSetDisplayMode() function. You can also use the [Trend]GapFillMode parameter, but it will interpolate values within the actual trend file as well as on the trend graph.

## Printing Trend Data

You can print trend data using the following functions:

Function	Purpose
TrnPrint	Prints a trend that is displayed on the screen.
TrnPlot	Prints a plot of one or more trend tags.
TrnComparePlot	Prints two trends (one overlaid on the other), each of up to four trend tags.
WinPrint	Prints the active window

The standard trend templates have buttons that call these functions to print data. When you print using the TrnPrint function, the Plot Setup dialog box appears. Use this dialog box to:

- Specify the title of the trend.
- Add a comment which is displayed beneath the title.
- Specify whether the trend is going to print in black and white, or in color. The selection that you make here will become the setting for the [General]PrinterColorMode parameter.
- Define your printer setup. The printer that you select here will be set as the default printer at the [General]TrnPrinter parameter.

- Specify whether or not the form displays the next time the function is used. This check box sets the [General]DisablePlotSetupForm parameter.

See Also [Exporting Trend Data](#)

## Exporting Trend Data

You can export trend data to reports and databases with the following functions:

Function	Purpose
TrnGetTable	Retrieves trend information and stores it in a Cicode array
TrnExportClip	Copies trend data to the clipboard
TrnExportCSV	Copies trend data to a CSV file
TrnExportDBF	Copies trend data to a DBF file

The standard trend templates have buttons that call these functions to export data.

**Note:** You can also select part of your trend graph (click and drag) and copy the underlying values to the Windows clipboard. You can then paste them into an Excel spreadsheet. (If you are pasting millisecond values, you will need to create a custom format for the TIME column to display these values correctly. To do this, select the column and select **Format | Cells**. In the **Number** tab, select **Custom** for Category, and type **h:mm:ss.000 AM/PM**.)

See Also [Using Trend History Files](#)

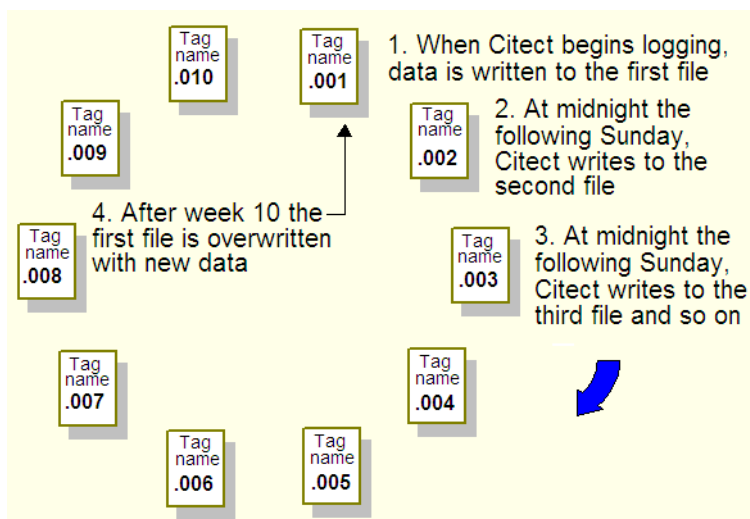
## Using Trend History Files

When CitectSCADA starts up for the first time, it creates all the trend files required by each trend tag in the runtime system. (You can change this default using the [Trend]AllFiles parameter.)

CitectSCADA uses a system of rotational history files to store the trend data. Data is stored in several files rather than in a single large file.

By default, CitectSCADA uses 2 files (for each trend tag). You can change the default by specifying the number of files to use, for example:

No. Files                      10  
 Comment                      Use ten files for the data, as in the following diagram:



The maximum number of files you can specify per trend tag is 270.

You can also specify the period between files, i.e., when a new history file is used, for example:

Period                      1:00:00  
 Comment                      Use a new file each hour

Period                      6:00:00  
 Comment                      Use a new file every six hours

Period                      72:00:00  
 Comment                      Use a new file every three days

Period                      Monday  
 Comment                      Use a new file each week beginning on Monday

Period                      15<sup>th</sup>  
 Comment                      Use a new file every month beginning on the 15th of each month

Period                      25th June  
 Comment                      Use a new file every year beginning on the 25th of June

You can also specify the time of day to synchronize the start of the history file; for example:

Time	6:00:00
Comment	Synchronize the file at 6:00 am
Time	12:00:00
Comment	Synchronize the file at 12:00 midday
Time	18:30:00
Comment	Synchronize the file at 6:30 pm

See Also [Storage method](#)

## Storage method

CitectSCADA allows you to select the storage method to use for trend tags and SPC tags. You are given a choice of either **Scaled** or **Floating Point**. Scaled represents a 2-byte data storage method; floating point uses 8 bytes.

Floating point storage has a dramatically expanded data range in comparison to Scaled storage, allowing values to be more precise, but it also uses more disk space. Scaled should be used where compatibility with pre-V5.31 trend history files is required.

You can set the required storage method via the Trend Tag or SPC Tag properties form (press the **F2** key to view the extended form). The storage method is set to Scaled by default.

See Also [Calculating disk storage](#)

## Calculating disk storage

The following equations allow you to calculate the total disk space required to store a trend across a specified period of time.

Note that the storage method used for a trend (**Scaled** or **Floating Point**) affects the number of bytes required for each sample, so it is important to base your calculations on the appropriate formula.

To find out which storage method a particular trend is using, refer to the extended Trend Tag Properties dialog. (By default, the **Scaled** storage method is used.)

### Scaled

Each data sample requires two bytes of storage. You can therefore calculate the total disk storage required for each trend by using the following formula:

$$\text{Bytes reqd for each trend} = 464 \times \text{No. Files} + 176 + \left( \frac{\text{File Period (secs)} \times (\text{No. Files}) \times 2}{\text{Sample Period (secs)}} \right)$$

For example, if a trend record produces one sample every ten seconds for one week, and you are using five data files (five weeks), the number of bytes required is:

$$\begin{aligned}\text{Bytes required} &= 464 \times 5 + 176 + \left( \frac{(7 \times 24 \times 60 \times 60) \times 5 \times 2}{10} \right) \\ &= 607296 \text{ bytes}\end{aligned}$$

#### Floating point

Each data sample requires eight bytes of storage. This alters the equation to:

$$\begin{aligned}\text{Bytes reqd} &= 704 \times \text{No. Files} + 160 + \left( \frac{\text{Period (secs)} \times (\text{No. Files}) \times 8}{\text{Sample Period}} \right) \\ \text{for each} & \\ \text{trend} &\end{aligned}$$

The number of bytes required then becomes:

$$\begin{aligned}\text{Bytes required} &= 704 \times 5 + 160 + \left( \frac{(7 \times 24 \times 60 \times 60) \times 5 \times 8}{10} \right) \\ &= 2422976 \text{ bytes}\end{aligned}$$

Note that the calculations above do not take into account the space required to store the history file for each trend. This is because these files remain at a set size and therefore do not significantly impact the amount of disk space required.

**Note:** For efficient trends storage, use Windows NT file compression. By using this method you often can reduce your files to 10% of their original size; the actual amount of compression varies depending on the rate of change of the data.

See Also [Reconfiguring history files](#)

## Reconfiguring history files

If you change the configuration of your trend history files (in an existing project), or you change the configuration of a trend tag that affects the number, time, or period of the trend files, you must delete all the existing trend files - before you run the new system.

If you change the path of your trend history files (in an existing project), all existing trend data is ignored.

**Note:** You must not delete history files (that CitectSCADA creates) from your hard disk **while your system is running**.

## Using Path Substitution

Instead of specifying the full path to data files in your system, you can use path substitution.



With path substitution, you define a name that is a substitution for the full directory path. You can then use the substitution name in the following format:

File Name                      [SUBSTITUTION]:<filename>

For example, if you decide to store a trend data file called MYFILE in a directory called C:\CITECT\DATA\MYTRENDS, you can specify the full path to the file, for example:

File Name                      C:\CITECT\DATA\MYTRENDS\MYFILE

or define a path substitution (for example MYDATA) and specify the path as:

File Name                      [MYDATA]:MYFILE

Path substitution provides greater control of data storage. You can change the location of all data files by changing the definition of the data path - instead of locating and changing each occurrence of the data path.

See Also [Default path definitions](#)

## Default path definitions

CitectSCADA has the following predefined path substitutions:

Path Name	Default Directory
Bin	\CITECT\BIN
Data	\CITECT\DATA
User	\CITECT\USER
Run	The current project directory
Copy	The current copy project directory
Back	The current backup project directory

## Debugging Trending

CitectSCADA provides the TrendDebug citect.ini parameter to help you while logging and trending data.

### Example:

```
[Trend]
TrendDebug=n
```

where *n* can be the combination of the following debug options:

- 1 - Logs the client, server, and redundancy message types and also the samples being written in the trend server from normal acquisition.
- 2 - Logs detailed information about the currently active backfill process, including the redundant samples written to the archive.
- 4 - Logs detailed information for the TrendSetTable functions.
- 7 - Logs all trend activities.

- 8 - Logs the summary information only for the currently active backfill process.

These settings can be added together to have combinations of logging levels. For example

```
[Trend]  
TrendDebug=6
```

logs the detailed backfill process and `TrendSetTable` functions.

These settings are read dynamically, meaning that you can change these settings while CitectSCADA is running and the changes will take effect from that point onwards.

# Chapter 26: Understanding Statistical Process Control

---

Statistical quality control (SQC) helps track and improve product or service quality. Statistical process control (SPC) is the primary tool of SQC and encompasses the collection, arrangement, and interpretation of process variables associated with a product, in regard to uniformity of quality.

Every process is subject to variation and can never be perfect. To respond to this, a continuous improvement strategy should be adopted. By repeating the following cycle, a strategy of prevention can be implemented. This is the key to using SPC effectively. Consider the following steps:

- Analyze the process
- What do we know about the variability of this process?
- Is this process statistically controllable (predictable)?
- Is the process capable of meeting the set requirements?
- What problems are the most critical?
- Maintain (control) the process
- Improve the process

SPC encompasses the concepts of variation, statistical control, and process capability, and typically uses the tools XRS control chart, capability chart, and the Pareto chart.

See Also

[Process Variation](#)

[Statistical Control](#)

[Process Capability](#)

[XRS Control Charts](#)

[Capability Charts](#)

[Configuring capability charts](#)

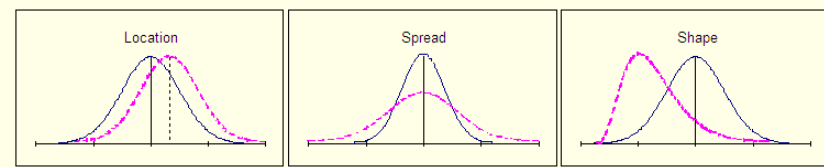
[Pareto Charts](#)

[Using Statistical Process Control \(SPC\) with CitectSCADA](#)

## Process Variation

To use SPC effectively, you should understand the concept of *variation*. When a product characteristic is measured repeatedly, each measurement is likely to differ from the last. This is because the process contains sources of variability.

When the data is grouped into a frequency histogram, it will tend to form a pattern. The pattern is referred to as a probability distribution and is characterized in three ways:



**Note:** Most SPC techniques assume that the collected data has a [normal distribution](#).

Variation is generally categorized into one of two types:

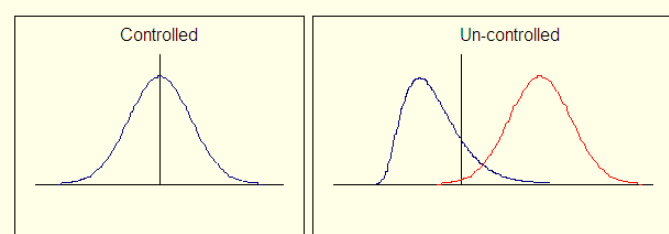
- **Common:** refers to variation that is predictable and repeatable over time. The distribution characteristics will be stable. Common variation could be due to consistent process inaccuracy or similar.  
  
Statistics indicate that common variations account for about 85% of a process problems. Usually these problems require solution at the management level.
- **Special:** refers to variation that is not always present. When special variation occurs it will tend to change the distribution characteristics. The distribution is not stable over time.

Statistics indicate that special variations account for about 15% of process problems. Typically these problems require local action (specific equipment fixing and so on) for solution.

See Also [Process Variation](#)

## Statistical Control

A process is said to be in statistical control when the only sources of variation are from common causes. A statistically controllable process is desirable because it is predictable, while a statistically uncontrollable process will yield unpredictable distributions.



Even though a process might not be statistically controllable, it might still meet requirements. Conversely, a process which is controllable might not meet requirements. This issue is clarified by considering Process Capability.

See Also [Process Variation](#)

## Process Capability

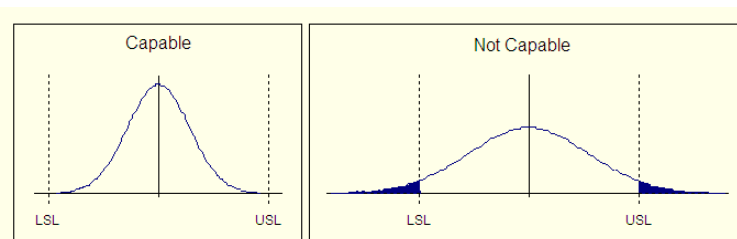
A process is said to be capable when the percentage of samples outside the specification limits is less than a predefined value. The following assumptions must also be true:

- The process is statistically stable (only common causes of variation exist)
- The individual measurements conform to a [normal distribution](#)
- Measurement variation (due to the measuring instrument) is small

The specification limits are a reflection of the customers requirements and are selectable. The percentage of samples that must lie within the specification limits is calculated from the standard deviation (sigma)—“3-sigmas” on either side of the mean.

**Note:** The “3-sigma” term refers to the boundaries which are located  $3 \times$  standard deviation (s) on either side of the center. For a normal distribution 99.74% of the samples are expected to fall within this boundary.

Ultimately capability determines whether the process is statistically able to meet the specification or not. Reducing the effects of common variation will make your process more capable, as shown here:

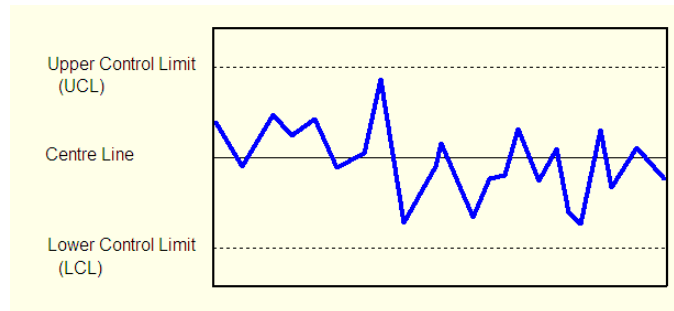


See Also [XRS Control Charts](#)

## XRS Control Charts

XRS charts are the primary tool of SPC and convey information about variation and controllability. The charts are trend graphs that individually show mean (X

bar), range (R), and standard deviation (s or sigma). Each point on the graph represents sub-grouped data, not individual samples.



### CL, UCL and LCL

Each graph has center, upper control limit (UCL), and lower control limit (LCL) lines drawn. The control limits are estimates of where the “3-sigma” limits should lie for an approximately normal distribution. In practise the limits are calculated from the measured  $\bar{X}$ ,  $\bar{R}$ , and  $\bar{s}$ , and table values which are based on the size of the subgroup used.

These lines are used as a guide for analysis as they are based on the natural variability of the process. These limits are not specification limits.

### Interpreting the chart

Control charts can provide important information about process variation. By watching for particular patterns and events, conclusions can be drawn as to how the process is controlling.

A (normal) process that is statistically under control will tend to distribute data according to a normal distribution. The data should appear randomly, but centered around the center line. Data that appears near the control limits should be infrequent but expected. Presence of data that appears outside of the control limits indicates the process is statistically uncontrolled and action should be taken to address the cause. Similarly, runs of constant data or patterns indicate instability.

The following list of patterns and events are considered to be cause for alarm:

- Points beyond the control limits
- Several consecutive points on only one side of the average
- Several consecutive points that are consistently increasing or decreasing
- Substantially more than 2/3 of plotted points lie close the average
- Substantially less than 2/3 of plotted points lie close to the average

The presence of these types of patterns and events has different meanings, depending on which type of chart is being analyzed. By using all three control charts ( $\bar{X}$ , R and s) together, a better understanding of readings is gained.

See Also [Capability Charts](#)

## Capability Charts

The process capability chart is the frequency histogram and distribution of all the process mean data and is used to analyze process capability. This chart is always drawn with center line, lower specification limit and upper specification limit indicators in place. Interpreting capability charts is typically made with the  $C_p$ ,  $C_{pk}$ , [kurtosis](#) and skewness indices.

### USL and LSL

Upper specification limit (USL) and lower specification limit (LSL) specify the requirements of the product, and so are customer driven. The target value should be the midpoint between USL and LSL.

### Cp index

The inherent process capability ( $C_p$ ) is the ratio of tolerance to 6-sigma. Essentially this index indicates whether the distribution would fit inside the USL and LSL limits. Its meaning is defined as follows:

- $C_p > 1.0$  - Indicates the process variation will fit within the specified limits (USL and LSL) and therefore, is capable.
- $C_p < 1.0$  - Indicates the process is not capable.

### Cpk Index

The process capability based on the worst case ( $C_{pk}$ ) is similar  $C_p$ . This index, however, indicates where the mean lies in relation to the USL and LSL limits. It is used to mathematically clarify  $C_p$ . Its meaning is defined as follows:

- $C_{pk} < 0$  - Indicates the process mean is outside the specified limits (USL and LSL)
- $C_{pk} = 0$  - Indicates the process mean is equal to one of the specified limits.
- $C_{pk} > 0$  - Indicates the process mean is within the specified limits.
- $C_{pk} = 1.0$  - Indicates that one side of the 6-sigma limits falls on a specification limit.
- $C_{pk} > 1.0$  - Indicates that the 6-sigma limits fall completely within the specified limits.

See Also [Pareto Charts](#)

## Pareto Charts

A Pareto chart is a tool which is used to analyze relative failure or defect frequency. The Pareto chart is a frequency histogram which is ordered from highest to lowest. Unlike control and capability charts, this chart uses no statistical guides.

Pareto charts emphasize Pareto's empirical law that any assortment of events consists of a few major and many minor elements. In the context of SQC, it is important to select the few vital major opportunities for improvement from the many trivial minor ones. The Pareto chart is particularly useful for Cost-to-fix versus Defect-frequency analysis.

In addition to the histogram, typically a cumulative percentage is also given. From top to bottom, the percentage represents the ratio of the sum of all values to this point, to the sum of all values in the chart.

## Using Statistical Process Control (SPC) with CitectSCADA

CitectSCADA displays Statistical Process Control information on three types of chart; XRS Control Chart, Process Capability Chart, and Pareto Chart.

**To configure an SPC chart:**

- 1 Click **New Page** or choose **File | New**.
- 2 Select **Type: Page**.
- 3 Choose the **Resolution** (size) of the SPC page.
- 4 Choose an SPC **Template** for the SPC page:
  - **SPCXRSChart** - An XRS control chart
  - **SPCCpk** - A capability (Cpk) chart combined with an Mean chart

**Note:** Pareto charts are configured slightly differently and hence, are not included here.
- 5 Click **OK**.
- 6 Double-click the graph display.
- 7 Enter the variable tag in the [Genie](#) pop-up.
- 8 Click **OK**.
- 9 Save the page.

See Also

[SPC Tags](#)  
[SPC Control Charts](#)

## SPC Tags

SPC tags specify data that is to be collected for use in SPC operations. Once data is defined it can be dynamically analyzed (as SPC graphs and alarms) at run-time. SPC tags are similar to trend tags.

Like trends, CitectSCADA can collect and store any amount of SPC data. The only restriction on the amount of data that you can store is the size of the hard disk on your computer. (CitectSCADA uses an efficient data storage method -



ensuring that space on your computer's hard disk is maximized.) For long-term storage, you can archive the data to disk or tape (without disrupting your runtime system). You can also log data at regular intervals ([periodic trend](#)), or only when an event occurs (event trend), in the same manner as Trend Tags.

**Note:** SPC does not support Periodic-Event trends, which is a combination of the properties of Periodic and Event trends. In addition, there may be display refresh problems using event-based SPC tags.

When you define an SPC tag you must be sure to fill in the upper specification limit (USL) and lower specification limit (LSL) if you intend to analyze capability. These values should accurately represent the users requirements, and the target value should lie midway between the two. If these fields are left blank the capability analysis will be meaningless.

#### To configure an SPC tag:

- 1 Choose **Tags | SPC Tags**. The SPC Tags dialog box appears.
- 2 Enter the SPC tags properties.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [SPC tag properties](#)

## SPC tag properties

Use the SPC Tags dialog box to configure the SPC tag properties. Statistical Process Control (SPC) Tags have the following properties:

### SPC Tag Name

The name assigned to the SPC data. If you are logging a variable, you should use the same name for the SPC tag that you used for the variable tag. Enter a value of 16 characters or less.

**Note:** The first eight (8) characters of your SPC tag names must not be the same as the first 8 characters of your Trend tag names.

### Cluster Name

The name of the cluster the SPC tag runs in.

If the Cluster Name is not set, then CitectSCADA considers this SPC tag to run on all defined clusters.

### Expression

The logged value of the SPC tag. Enter a value of 64 characters or less. You can log individual variables by using a Variable Tag, for example:

Expression	PT104
Comment	Logs the Variable Tag PT104

The value of the process variable PT104 is logged. Variable PT104 must be defined as a variable tag. You can also log any Cicode expression or function, for example:

Expression	PT104/COUNTER
Comment	Logs Variable Tag PT104 divided by the Variable Tag COUNTER

### Trigger

The Cicode expression (or variable tag) that triggers data logging. Enter a value of 64 characters or less. For example:

Trigger	PT104<500
---------	-----------

In this example, logging occurs when the value of the variable tag (PT104) falls below 500.

For a periodic SPC trend, data is logged only while the value of the trigger is TRUE. In the above example, data is logged continuously while the value of PT104 remains less than 500. Logging ceases when the value rises to (or above) 500. Logging does not occur again until the value of PT104 falls below 500.

You do not have to specify a trigger for a periodic SPC trend. If you do not specify a trigger for a periodic SPC trend, then logging will occur continuously.

For an event SPC trend, data is logged once when the value of the trigger changes from FALSE to TRUE. In the above example, one sample is logged when the value of PT104 first becomes less than 500. Another sample is not logged until the value of PT104 rises to (or above 500) and again falls below 500.

### Sample Period (16 Chars.)

The sampling period of the data, in hh:mm:ss (hours:minutes:seconds). CitectSCADA checks the **Trigger** each sample period. If the Trigger is TRUE (or has just changed from FALSE to TRUE, in the case of event SPC trends), CitectSCADA will log the value of the **Expression**.

#### Examples

Sample Period	30
Comment	Logs data every 30 seconds
Sample Period	10:00
Comment	Logs data every 10 minutes
Sample Period	10:00:00
Comment	Logs data every 10 hours

Sample Period            2:30:00  
 Comment                 Logs data every 2 and a half hours

This property is optional. If you do not specify a sample period, the sampling period will default to 10 seconds.

**Note:** If you edit this property in an existing project, you must delete the associated trend files before you run the new runtime system.

#### **Type (32 Chars.)**

The type of SPC trend:

- 1    TRN\_PERIODIC
- 2    TRN\_EVENT

**Note:** SPC does not support Periodic-Event trends, which is a combination of the properties of Periodic and Event trends. In addition, there may be display refresh problems using event-based SPC tags.

#### **Lower Spec Limit (16 Chars.)**

The Lower Specification Limit (LSL). This value is used as the lower limit to determine process capability. When used in conjunction with the USL it provides a tolerance for your process.

If you are unfamiliar with process capability and capability indices, ask for expert opinion. Rather than leave this blank you should (at least) attempt an estimate. Enter a value that you think is the lowest acceptable value of this tag. If you leave this field blank only your capability analysis will be affected.

#### **Upper Spec Limit (16 Chars.)**

The Upper Specification Limit (USL). This value is used as the upper limit to determine process capability. When used in conjunction with the LSL it provides a tolerance for your process.

If you are unfamiliar with Process Capability and capability indices, ask for expert opinion. Rather than leave this blank you should (at least) attempt an estimate - Enter a value that you think is the highest acceptable value of this tag. If you do leave this field blank only your capability analysis will be affected.

#### **Comment (48 Chars.)**

Any useful comment.

**Note:** The following fields are implemented with extended forms (press **F2**).

**File Name (231 Chars.)**

The file where the data is to be stored. You must specify the full path or use path substitution.

When CitectSCADA collects data from your plant floor, it stores the data in a file on the hard disk of your computer. When CitectSCADA subsequently uses the data to display an SPC trend, it reads the data from this file. (CitectSCADA uses a separate file for each SPC tag.)

By default, CitectSCADA stores the file in the \CITECT\DATA directory on the hard disk where you installed CitectSCADA. The default name of the file is the first eight characters of the SPC tag name. However, you can specify an alternate file name. If you do specify a file name, you can specify the full path, for example:

File Name                      C:\DATA\SPCS\TANK131

or use the path substitution string:

File Name                      [DATA]:TANK131

where [DATA] specifies the disk and path for the data. Use path substitution to make your project more 'portable'.

The File Name property is optional. If you do not specify a file name, the file name defaults to \CITECT\DATA\<Name> on the hard disk where you installed CitectSCADA. <Name> is the first eight characters of the SPC Tag Name. If you use this property, ensure that no other SPC tag names have the same first eight characters, otherwise the data might be lost.

**Note:** Do not use a file extension when specifying a file name. If you edit this property (change the file name or path) in an existing project, all existing SPC data is ignored. This file name must be different to your Trend tag file names.

**Storage Method**

Select either **Scaled** or **Floating Point** as the storage method for the SPC data. The key difference between these two options is that Scaled is a two-byte data storage method, whereas Floating Point uses eight bytes.

Floating Point storage has a dramatically expanded data range in comparison to Scaled storage, allowing values to have far greater resolution. However, you need to consider that it also uses a lot more disk space. Scaled should be used where compatibility with pre-V5.31 trend history files is required.

If you do not specify a storage method, it is set to **Scaled** by default.

**Note:** If you edit this property in an existing project, you must delete the associated trend files - before you run the new runtime system.

**Privilege (16 Chars.)**

The privilege required by an operator to display the SPC data on an SPC page.

**Area (16 Chars.)**

The [area](#) to which the SPC data belongs. Only users with access to this area (and any required privileges) will be able to display the SPC data on an SPC page. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to display the SPC data.

**Eng Units (8 Chars.)**

The engineering units of the variable/expression being logged. The engineering units are used by the SPC trend scales and SPC trend cursor displays.

**Format (10 Chars.)**

The format of the variable/expression being logged. The format is used by the SPC trend scales and SPC trend cursor displays.

This property is optional. If you do not specify a format, the format defaults to ####.#.

**No. Files (4 Chars.)**

The number of history files stored on your hard disk (for this tag).

If you do not specify the number of files, 2 history files are stored on your hard disk. The maximum number of files you can specify is 270.

**Note:** If you edit this property in an existing project, you must delete the associated SPC trend files - before you run the new runtime system.

**Subgroup Size (8 Chars.)**

The size of each subgroup. The default value for this value is 5. Valid values are 1 - 25 inclusive.

**Time (32 Chars.)**

The time of day to synchronize the beginning of the history file, in hh:mm:ss (hours:minutes:seconds). If you do not specify a time, the file is synchronized at 0:00:00 (i.e. midnight).

**Note:** If you edit this property in an existing project, you must delete the associated SPC trend files before you run the new runtime system.

**Period (32 Chars.)**

The period of the history file, in hh:mm:ss (hours:minutes:seconds). Alternatively, you can:

- Specify a weekly period by entering the day of the week on which to start the history file, e.g. Monday, Tuesday, Wednesday, etc.
- Specify a monthly period by entering the day of the month on which to start the history file, e.g. 1st, 2nd, 3rd, 4th, 5th, etc.
- Specify a yearly period by entering the day and the month on which to start the history file, e.g. 1st January, 25th February, etc. The day and month must be separated by a space.

If you do not specify a period, the period defaults to Sunday (weekly).

**Note:** If you edit this property in an existing project, you must delete the associated SPC trend files before you run the new runtime system.

#### **Process Mean (16 Chars.)**

The calculation override for process mean ( $\bar{X}$ ). If a value is specified here it will be used in all SPC calculations, instead of the value calculated by CitectSCADA. This will affect the calculation of control limits which are normally a function of the collected samples of data.

Do not use this field unless you are experienced in SPC.

#### **Standard Deviation (16 Chars.)**

The calculation override for process standard deviation ( $s$ ). If a value is specified here it will be used in all SPC calculations, instead of the value calculated by CitectSCADA. This will affect the calculation of control limits which are normally a function of the collected samples of data.

Do not use this field unless you are experienced in SPC.

#### **Range (16 Chars.)**

The calculation override for process range ( $R$ ). If a value is specified here it will be used in all SPC calculations, instead of the value calculated by CitectSCADA. This will affect the calculation of control limits which are normally a function of the collected samples of data.

Do not use this field unless you are experienced in SPC.

## **SPC Control Charts**

CitectSCADA uses the following types of control charts:

- [XRS control chart](#)
- [Capability charts](#)
- [Pareto Charts](#)

See Also [Control Chart Line Constants](#)

## XRS control chart

The XRS charts display trends of subgroup means ( $\bar{X}$ ), ranges (R) and standard deviations (s). The XRS chart operates similarly to a standard trend, but with additional SPC extra features. Each subgroup displays as a single node on the graph and consecutive nodes are linked by a line.

Each control chart has a central line and two control limits-upper and lower (UCL and LCL). CitectSCADA automatically calculates these SPC values at run-time. If you want to override the UCL and LCL you can do so by entering the Process Mean, Range, and Standard Deviation fields in SPC Tags.

See Also [Configuring XRS charts](#)

## Configuring XRS charts

Genies simplify the task of adding a new SPC page. To create a new chart:

- Define the SPC Tags.
- Create the page using an XRS template.

**Note:** If you want to develop your own XRS template, the method is to copy and modify an existing template.

## Capability charts

The process capability chart is a frequency histogram and distribution of all the sample data currently displayed (on the Mean chart). CitectSCADA automatically takes the data being trended, builds a distribution, adds the LSL and USL. CitectSCADA also calculates the Cp, Cpk, [kurtosis](#), and skewness indices.

The process capability is defined in relation to the upper and lower specification limits (USL and LSL) for a given SPC Tag. These values are defined in SPC Tags and should accurately represent the users requirements.

See Also [Configuring capability charts](#)

## Configuring capability charts

Genies simplify the task of adding a new SPC page. To create a new chart:

- 1 Define the SPC Tags and specify the LSL and USL.
- 2 Create the page using a Capability (Cpk) template.

## Pareto Charts

Pareto analysis is a technique used to identify the relative importance of problems and conditions. The Pareto chart is a frequency histogram ordered from highest to lowest – CitectSCADA automatically orders the bars at run-time. The data for each bar in the histogram represents one CitectSCADA variable - as defined in Variable Tags. Do not use SPC tags.

**Note:** Typically the frequency in a histogram is of integer type, though you can use floating point types if you want. Negative values are not valid.

See Also [Configuring Pareto charts](#)

## Configuring Pareto charts

Genies simplify the task of adding a new Pareto chart.

### To create a new chart:

- 1 Define the Variable Tags (Pareto charts do not use SPC Tags).
- 2 Create the page using a Pareto template.

### To configure a Pareto chart:

- 1 Click New Page, or choose **File | New**.
- 2 Select **Type: Page**.
- 3 Choose the **Resolution** (size) of the SPC page.
- 4 Choose the **SPCPareto Template** for the SPC chart.
- 5 Click **OK**.
- 6 Double-click the display (where prompted by the template).
- 7 Enter the variable tags in the **Tag Name** fields. (Use Variable tags here, not necessarily SPC tags.)
- 8 Enter the variable descriptions in the **Tag Description** boxes.
- 9 Click **OK**.
- 10 Save the page.

## SPC Alarms

CitectSCADA automatically monitors several special kinds of conditions that are specific to SPC data. When specific patterns or events occur to an SPC tag, CitectSCADA will set the appropriate alarm. Typically these alarms are related to, and used in conjunction with, the XRS control charts.

SPC alarms are configured differently to standard [digital alarms](#) to provide for this extra functionality. SPC alarms must be configured using the Advanced Alarms form. You use the [SPCAlarms](#) Cicode function to check for the condition of the alarms:

**Note:** An SPC alarm can only be defined on an alarm server in the same cluster as the trend server that contains the SPC tag (though the variable tag referenced in the SPC tag can be on a different IOserver cluster).

Complete the Advanced Alarm form as shown here:

### Advanced Alarms

Alarm Tag	Feed_Above_UCL
Alarm Desc	Un-controlled variation
Expression	SPCAlarms("Feed_SPC", XAboveUCL)
Comment	Several samples are above UCL



The SPC (trend) server checks for any specified alarm conditions. When one is detected, it informs the alarms server that an alarm has occurred. Be aware of the number of subgroups displayed on your SPC charts, and the number used in SPC alarm calculations (as set by the [SPC]AlarmBufferSizeparameter). If these two values differ, SPC alarms might not correlate with your SPC charts.

The following list shows the alarms types that are valid:

Name	Description
XFreak	Single point greatly differs ( $\pm 2$ sigma) from the center line.
XOutsideCL	Process mean outside either of the control limits (UCL or LCL).
XAboveUCL	Process mean above the upper control limit (UCL).
XBelowLCL	Process mean below the lower control limit (LCL).
XOutsideWL	Process mean outside the warning limits which are 67% of the UCL and LCL.
XGradualUp	Process mean is gradually drifting up to a new level indicated by several consecutive points above the mean.
XGradualDown	Process mean is gradually drifting down to a new level indicated by several consecutive points below the mean.
XUpTrend	Several points continuously increasing in value.
XDownTrend	Several points continuously decreasing in value.
XErratic	Large fluctuations that are greater than the control limits.
XStratification	Artificial constancy. Several consecutive points are close to (within $\pm 1$ sigma of) the center line.
XMixture	Several consecutive points are far from (outside $\pm 1$ sigma of) the center line.
ROutsideCL	Process range outside either of the control limits (UCL or LCL).
RAboveUCL	Process range above the upper control limit (UCL).
RBelowLCL	Process range below the lower control limit (LCL).

**Note:** The above alarms rely on *n* number of consecutive points to generate the alarm. The value of *n* can be set for each type of alarm through SPC parameters.

See Also [SPC Formulas and Constants](#)

## SPC Formulas and Constants

The SPC calculations are based on the samples collected in subgroups. Each subgroup will have the same number of samples, typically 4. The subgroup size for each SPC tag is set at the **SPC Tags** properties form.

The number of samples in each subgroup can range from 1 to 25 inclusive.

When the number of samples in each subgroup is 1:

**Subgroup Mean ( $\bar{X}$ ):**

Is the value of the single sample in the group, and is defined by:

$$\bar{X}_i = X_i$$

Where  $X_i$  is the single sample value in the subgroup.

**Moving Range (MR):**

Is the difference between successive sample values, and is defined by:

$$MR_i = X_i - X_{i-1} \quad i \geq 2$$

Where  $X_i$  is the current sample value and  $X_{i-1}$  is the previous sample value. The number of moving ranges in the process is always one less than the number of subgroups.

**Subgroup Standard Deviation (s):**

Is a measure of absolute variation or dispersion. It describes how much the sample values differ from their mean, and is estimated by:

$$s_i = \frac{MR_i}{D_2} \quad i \geq 2$$

The number of subgroup standard deviations in the process is always one less than the number of subgroups.

**Process Average ( $\bar{\bar{X}}$ ):**

$$\bar{\bar{X}} = \frac{\bar{X}_1 + \bar{X}_2 + \dots + \bar{X}_m}{m}$$

Where  $\bar{X}_1$ ,  $\bar{X}_2$ , and  $\bar{X}_m$  are the subgroup means, and m is the total number of subgroups in the process.

**Process Range ( $\bar{R}$ ):**

$$\bar{R} = \frac{MR_2 + MR_3 + \dots + MR_m}{m - 1}$$

Where  $MR_2$ ,  $MR_3$ , and  $MR_m$  are the subgroup moving ranges, and  $m$  is the total number of subgroups in the process.

**Process Standard Deviation ( $\bar{s}$ ):**

$$\bar{s} = \frac{\bar{R}}{D_2}$$

**When the number of samples in each subgroup is greater than 1 :**

**Subgroup Mean ( $\bar{X}$ ):**

Is the average (not median or centre) of the samples in the group, and is defined by:

$$\bar{X} = \frac{X_1 + X_2 + \dots + X_n}{n}$$

Where  $X_1$ ,  $X_2$ , and  $X_n$  are the sample values in the subgroup, and  $n$  is the total number of samples in the subgroup.

**Subgroup Range (R):**

Is the difference between the highest and lowest samples in the group, and is defined by:

$$R = X_{\max} - X_{\min}$$

Where  $X_{\max}$  is the maximum sample value and  $X_{\min}$  is the minimum sample value in the group.

**Subgroup Standard Deviation ( $s$ ):**

Is a measure of absolute variation or dispersion. It describes how much the sample values differ from their mean, and is defined by:

$$s = \sqrt{\frac{\sum (X_i - \bar{X})^2}{n - 1}}$$

Where  $X$ 's are the sample values in the group,  $\bar{X}$  is the group average, and  $n$  is the number of samples in the group.

**Process Average ( $\bar{\bar{X}}$ ):**

$$\bar{\bar{X}} = \frac{\bar{X}_1 + \bar{X}_2 + \dots + \bar{X}_m}{m}$$

Where  $\bar{X}_1$ ,  $\bar{X}_2$ , and  $\bar{X}_m$  are the subgroup averages, and  $m$  is the total number of subgroups in the process.

**Process Range ( $\bar{R}$ ):**

$$\bar{R} = \frac{R_1 + R_2 + \dots + R_m}{m}$$

Where  $R_1$ ,  $R_2$ , and  $R_m$  are the subgroup ranges, and  $m$  is the total number of subgroups in the process.

**Process Standard Deviation ( $\bar{s}$ ):**

$$\bar{s} = \frac{s_1 + s_2 + \dots + s_m}{m}$$

Where  $s_1$ ,  $s_2$ , and  $s_m$  are the group standard deviations, and  $m$  is the total number of groups in the process.

**Average Control Limits (UCL<sub>x</sub> and LCL<sub>x</sub>):**

Specify the approximated 3-sigma boundaries. For a normal distribution 99.74% of the samples will fall within this boundary.

$$UCL_x = \bar{\bar{X}} + A_2 * \bar{R}$$

$$LCL_x = \bar{\bar{X}} - A_2 * \bar{R}$$

Where  $\bar{R}$  is the Process Range and A2 is a constant (given in the Control Chart Line Constants table).

**Range Control Limits (UCL<sub>R</sub> and LCL<sub>R</sub>):**

Specify the approximated 3-sigma boundaries. For a normal distribution 99.74% of the samples will fall within this boundary.

$$UCL_R = D_4 * \bar{R}$$

$$LCL_R = D_3 * \bar{R}$$

Where  $\bar{R}$  is the Process Range and D3 and D4 are constants (given in the Control Chart Line Constants table).

**Standard Deviation Control Limits (UCL<sub>s</sub> and LCL<sub>s</sub>):**

Specify the approximated 3-sigma boundaries. For a normal distribution 99.74% of the samples will fall within this boundary.

$$UCL_s = B_4 * \bar{s}$$

$$LCL_s = B_3 * \bar{s}$$

Where  $\bar{s}$  is the Process Standard Deviation and B3 and B4 are constants given in the Control Chart Line Constants table).

**Process Capability (C<sub>p</sub>):**

Is the capability of a process to meet a specific tolerance. A process is considered capable when the percentage of samples of a variable for that process that fall within the upper and lower specification limits is greater than a specified value.

The inherent process capability is defined as:

$$C_p = \frac{(USL - LSL)}{6\bar{s}}$$

- $C_p > 1.0$  - Indicates the process variation is within the specified limits (USL and LSL) and therefore, is capable.
- $C_p < 1.0$  - Indicates the process is not capable.

The process capability based on worst case data is defined as:

$$C_{pk} = \frac{\min((USL - \bar{\bar{X}}), (\bar{\bar{X}} - LSL))}{3\bar{s}}$$

- $C_{pk} < 0$  - Indicates the process mean is outside the specified limits (USL and LSL)
- $C_{pk} = 0$  - Indicates the process mean is equal to one of the specified limits.
- $C_{pk} > 0$  - Indicates the process mean is within the specified limits.
- $C_{pk} = 1.0$  - Indicates that one side of the 6-sigma limits falls on a specification limit.
- $C_{pk} > 1.0$  - Indicates that the 6-sigma limits fall completely within the specified limits.

#### Skewness (Sk):

Is the degree of asymmetry of a frequency distribution (usually in relation to a normal distribution).

$$Sk = \frac{\sum (X_i - \bar{\bar{X}})^3}{N\bar{s}^3}$$

where N is the number of samples for the entire process (i.e. Subgroup Size \* number of Subgroups).

- Skewness  $> 0$  - Indicates that the histogram's mean (and tail) is pushed to the right.
- Skewness  $< 0$  - Indicates that the histogram's mean (and tail) is pushed to the left.

#### Kurtosis (Ku):

Is the degree of peakedness of a frequency distribution (usually in relation to a normal distribution).

$$Ku = \frac{\sum (X_i - \bar{\bar{X}})^4}{N\bar{s}^4}$$

where N is the number of samples for the entire process (i.e. Subgroup Size \* number of Subgroups).

- Kurtosis < 3 - Indicates a thin distribution with a relatively high peak.
- Kurtosis > 3 - Indicates a distribution that is wide and flat topped.

## Control Chart Line Constants

The table below shows the control chart line constants:

Samples in	Averages	Ranges		Standard Deviations			
Group	A2	D2*	D3	D4	B3	B4	
1	2.660	1.128	0	3.267	0	3.267	
2	1.880		0	3.267	0	3.267	
3	1.023		0	2.574	0	2.568	
4	0.729		0	2.282	0	2.266	
5	0.577		0	2.114	0	2.089	
6	0.483		0	2.004	0.030	1.970	
7	0.419		0.076	1.924	0.118	1.882	
8	0.373		0.136	1.864	0.185	1.815	
9	0.337		0.184	1.816	0.239	1.761	
10	0.308		0.223	1.777	0.284	1.716	
10	0.308		0.223	1.777	0.284	1.716	
11	0.285		0.256	1.744	0.321	1.679	
12	0.266		0.283	1.717	0.354	1.646	
13	0.249		0.307	1.693	0.382	1.618	
14	0.235		0.328	1.672	0.406	1.594	
15	0.223		0.347	1.653	0.428	1.572	
16	0.212		0.363	1.637	0.448	1.552	
17	0.203		0.378	1.622	0.466	1.534	
18	0.194		0.391	1.608	0.482	1.518	
19	0.187		0.403	1.597	0.497	1.503	
20	0.180		0.415	1.585	0.510	1.490	
21	0.173		0.425	1.575	0.523	1.477	
22	0.167		0.434	1.566	0.534	1.466	
23	0.162		0.443	1.557	0.545	1.455	
24	0.157		0.451	1.548	0.555	1.445	
25	0.153		0.459	1.541	0.565	1.435	

\* D2 is only used for estimating standard deviation when there is one sample per subgroup.

Reference ANSI Z1.1-1985, Z1.2-1985 & Z1.3-1985: American National Standard, *Guide for Quality Control Charts, Control Chart Method of Analyzing Data, Control Chart Method of Controlling Quality During Production*.

**Hints**

Double-click the chart area on an SPC page to display the SPC Genie and change the SPC variables.

The tools and menu items in these procedures automatically open the CitectSCADA form for you. Move the cursor till it changes to a hand to find these “hot” tools or options.



## Chapter 27: Reporting Information

---

You can request regular reports on the status of your plant, and reports that provide information about special conditions in your plant. Reports can be run on a request basis, at specified times, or when certain events occur (such as a change of state in a bit address). Output from a report is controlled by a device. A report can be printed when it runs or saved to disk for printing later. You can use a text editor or word processor to view, edit, or print the report, or you can display it in CitectSCADA as part of a page.

Reports can also include Cicode statements that execute when the report runs.

Reports are configured in two stages:

- Report properties
- Report format file

If report data is associated with an I/O device that fails at startup or goes offline while CitectSCADA is running, the associated data is not written to the report (because the values would be invalid). An error code is written instead.

See Also

[Configuring reports](#)  
[Running Reports](#)  
[Report Format File](#)  
[Handling Communication Errors in Reports](#)

### Configuring reports

To design, configure and use a report:

- 1 Configure a device for output of the report (e.g., if you want to save a report to a file when it is run, set up an ASCII\_DEV device).
- 2 Configure the report properties.
- 3 Edit the [report format file](#). Remember that for an RTF report, the report format file must be saved in RTF format (i.e., with an .RTF file extension).
- 4 Define your PC as a [reports server](#) using the **Computer Setup Wizard**.

To configure report properties:

- 1 Choose **System | Reports**. The Reports dialog box appears.
- 2 Enter the reports properties. Click **Edit** to edit the report format file.

Use the Reports dialog box to configure your reports.

The screenshot shows a Windows-style dialog box titled "Reports [ CSV\_Example ]". It contains several input fields and buttons. The "Report Name" field is filled with "Example". The "Cluster Name" is a dropdown menu. The "Time" is a dropdown menu, and the "Period" is set to "00:00:30". The "Trigger" is a dropdown menu. The "Report Format File" is "example.rtf", and the "Output Device" is "relog". The "Comment" field contains "Example Report". At the bottom, there are five buttons: "Add", "Replace", "Delete", "Edit", and "Help". Below the buttons, it says "Record : 1".

See Also [Reports dialog box](#)

## Reports dialog box

The Reports form has the following properties:

### Name

The name of the report. The name can be a maximum of 64 characters, or 253 characters including the path. It can consist of any character other than the semi-colon (;) or single quote ('). The name must be unique to the cluster.

**Note:** Where Cluster Name is left blank, the name must be unique to all defined clusters.

### Cluster Name

The name of the cluster that runs this report. If the Cluster Name is not set, then CitectSCADA considers this report to run on all defined clusters.

### Time

The time of day to synchronize the report, in hh:mm:ss (hours:minutes:seconds). If you do not specify a time, the report is synchronized at 0:00:00 (i.e., midnight). Enter a value of 32 characters or less.

### Period

The period of the report, in hh:mm:ss (hours:minutes:seconds). Enter a value of 32 characters or less. Alternatively you can:

- Specify a weekly period by entering the day of the week when the report is to start, e.g., Monday, Tuesday, Wednesday, etc.
- Specify a monthly period by entering the day of the month when the report is to start, e.g., 1st, 2nd, 3rd, 4th, 5th, etc.

- Specify a yearly period by entering the day and the month when the report is to start, e.g., 1st January, 25th February, etc. The day and month must be separated by a space.

If you do not specify a period, the report is run daily.

### Trigger

Any Cicode [expression](#) (or Variable tag) to trigger the report. Enter a value of 64 characters or less. If the result of the expression (in this field) is TRUE, and the **Time** and **Period** fields are blank, the report is run. The report is only run when the expression becomes TRUE, and it must become FALSE then TRUE again before the report is re-run.

### Report Format File

The name of the report format file. Enter a value of 253 characters or less. If you do not specify a file extension, it defaults to .RPT. Any valid file name can be used; however, you cannot use a Path Substitution in this field. If you specify a filename without a path, the file saves into the directory predefined as Run. The report is assumed (by the CitectSCADA compiler) to be ASCII unless an RTF extension is used.

**Note:** The file name of your report format file can be up to 64 characters long, or 253 characters including the path. It can consist of any characters other than the single quote ('), and the semi-colon (;).

### Output Device

The device where the report will be sent. Enter a value of 10 characters or less.

For RTF reports that are to be saved as a file, select a device of type ASCII\_DEV here. Due to the differing natures of their content; however, it is NOT recommended that the same ASCII device be used for logging both RTF and non-RTF reports.

**Note:** If two or more reports are running at the same time and are sending their output to the same printer, the output of each report can become mixed. You must use semaphores to control the access to the printer in each report. If the report only contains Cicode statements (and has no output data), this property is optional.

### Comment

Any useful comment. Enter a value of 48 characters or less.

**Note:** The following fields are implemented with extended forms (press F2).

### Privilege

The privilege required by an operator to run this report if the report is a command-driven report. Enter a value of 16 characters or less.

If the report is time-driven or event-driven, this property is ignored.

**Note:** If you assign an acknowledgment privilege to a report, do not assign a privilege to the command(s) that run the report. If you do assign a different privilege to the commands, an operator must have both privileges to run the report.

### Area

The [area](#) to which this report belongs. Enter a value of 16 characters or less. Only users with access to this area (and any required privileges) will be able to run this report. For example, if you enter Area 1 here, operators must have access to Area 1 (plus any required privileges) to run this report.

## Running Reports

You can run a report by the following methods:

- Automatically when CitectSCADA starts up
- Automatically at a specified time and period
- Automatically when an event is triggered
- By using a command
- A combination of the above

See Also [Running a report on startup](#)  
[Specifying times and periods](#)  
[Using triggers](#)  
[Using commands](#)

## Running a report on startup

You can run a report on startup. CitectSCADA searches for a report called "Startup" when it starts up, and if CitectSCADA locates this report, it is run automatically. You can change the name of the default report with the *Computer Setup Wizard*.

See Also [Specifying times and periods](#)

## Specifying times and periods

The period determines when the report is run. You can specify the period in hh:mm:ss (hours:minutes:seconds), for example:

Period	1:00:00
Comment	Run the report every hour
Period	6:00:00
Comment	Run the report every six hours
Period	72:00:00
Comment	Run the report every three days
Period	Monday
Comment	Run the report each Monday
Period	15 <sup>th</sup>
Comment	Run the report on the 15th of each month
Period	25th June
Comment	Run the report on the 25th of June

You can also specify the time of day to synchronize the report, for example:

Time	6:00:00
Comment	Synchronize the report at 6:00 am
Time	12:00:00
Comment	Synchronize the report at 12:00 midday

The time synchronizes the time of day to run the report and, with the Period, determines when the report is run, for example:

Time	6:00:00
Period	1:00:00

In this example, the report is run every hour, on the hour. If you start your runtime system at 7:25am, your report is run at 8:00am, and then every hour after that.

See Also [Using triggers](#)

## Using triggers

You can use any Cicode [expression](#) (or variable tag) as a trigger for a report. If the result of the expression (in the **Trigger** field) becomes TRUE, and if the **Time** and **Period** fields are blank, the report is run. For example:

Time	
Period	
Trigger	RCC1_SPEED<10 AND RCC1_MC

This report is only run when the expression (Trigger) becomes TRUE, i.e., when the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10. The expression must become FALSE and then TRUE again before the report is run again.

If you use the **Time** and/or **Period** fields, the trigger is checked at the time and/or period specified, for example:

Time	6:00:00
Period	1:00:00
Trigger	RCC1_SPEED<10 AND RCC1_MC

This report is run each hour, but only if the expression (Trigger) is TRUE (i.e., if the digital tag RCC1\_MC is ON and the analog tag RCC1\_SPEED is less than 10).

See Also [Using commands](#)

## Using commands

If the **Time**, **Period**, and **Trigger** fields are blank, the report can only be run by a command that calls the Cicode `Report ( )` function.

## Report Format File

The report format file specifies how data is formatted in a report. You can use fixed text, Cicode expressions, and database variables in any report.

You use a text editor that is supported by Windows to create (and modify) the report format file. If your report format file is in RTF (Rich Text Format), you should use Microsoft Wordpad.

### Including fixed text

You can include fixed text in the report, specifying the text exactly as you want it to appear (e.g., Name of Report, Description, and so on).

### Including OLE (RTF files only)

Objects can be linked to or embedded within an RTF report format file; however, such objects will not be displayed or printed from CitectSCADA.

### Using fonts (ASCII format only)

If your format file is in ASCII format, you can use any text font supported by Windows in the report. To specify a font, use the `PrintFont ( )` function. RTF

format files do not require this function, as they use the formatting features of the host word processor.

### Including Cicode expressions and variables

You can include Cicode expressions and variables by enclosing them (and optional format specifications) in braces {} - for example:

```
{TIME(1) }
{PV12:####.##}
{PV12:4.2}
```

The size of each field (number of characters) is determined by either the format specification, or by the number of characters between the braces. In the above example, the variable PV12 is formatted with four characters before the decimal point and two characters after.

You do not have to include the format, for example:

```
{PV12}
```

Here, the variable is formatted using only four characters (the number of characters between the braces).

The following rules apply when logging a report to a **database device**:

- The format (for the report field) must not specify a field size greater than the size of the relevant field specified in the device.
- No spaces are allowed between each field specification, e.g.:

```
{TIME(1) } {PV12:####.##} {PV12:4.2}
```

### Including blocks of Cicode

You can include a block of Cicode, using the following format:

```
{CICODE}
Statements;
{END}
```

The block of Cicode is delimited by the commands {CICODE} and {END}. After the {END} command, the report switches back into WYSIWYG mode. If the entire report is Cicode or the last section is Cicode, the {END} command is not required.

A block of Cicode does not send any output to the device unless you use either the Print() or PrintLn() functions. If you use one of these functions, the argument is printed to the device.

### Cicode variables

You can also declare variables for use within your Cicode block. You must declare all variables at the beginning of the file (i.e. before any report format or Cicode). Add a {CICODE} block first; for example:

```
{CICODE}  
INT nVar1;  
STRING sVar2;  
Statements;  
{END}
```

Remainder of report

**Including comments**

You can include comments by using the comment character ‘!’ enclosed in braces - for example:

```
{!This is a Comment}
```

A comment in the body of a report differs from a comment in a Cicode block: a comment in a Cicode block does not require braces.

**Including other report elements**

The following table describes other elements you can include in reports.

To...	Do this...
Issue a form feed	Use a form feed specifier: {FF}
Include a plot	Use the Plot functions.
Include trend data	Use the TrnGetTable() function.
Include trend graphs	Use the TrnPlot() function.

See Also [Report example](#)

**Report example**

The following is an example of a report format file (for a printer or ASCII file device):

```
-----  
SHIFT REPORT  
-----  
{Time(1) } {Date(2) }  
Shift Production {Shift_Prod:###.##} tonnes  
Total Production {Total_Prod} tonnes  
{! The following Cicode displays "Shift Report Complete" on the  
screen}  
{CICODE}  
Print("End of Report")  
Shift_Prod = 0; ! Reset the Shift production tonnage  
  
Prompt("Shift Report Complete");  
{END}  
{FF}
```

This report produces the following output to the device and displays “Shift Report Complete” on the [graphics page](#).



```
-----
SHIFT REPORT
-----
```

```
6:00am      12/3/92
Shift Production 352.45 tonnes
Total Production 15728 tonnes
End of Report
```

#### To edit a report format file:

- 1 From the Reports properties form, select the relevant report, and click **Edit**. Alternatively, click **Edit Report**.
- 2 Select the report to edit
- 3 Click **Edit**.

**Note:** If the report format file exists, it is loaded into the editor for you to edit. If the file does not exist, CitectSCADA creates a new file.

#### To change the report format file editor:

- 1 Click the **Options** tool, or choose **File | Options**.
- 2 Enter the name of the new **Editor**.
- 3 Click **OK**.

## Handling Communication Errors in Reports

You can handle errors in communication with I/O devices (for example, an I/O device fails at startup or goes off line while CitectSCADA is running) in two ways:

- You can write communication errors and invalid data to the report as error codes.
- You can disable the running of reports that are triggered from an I/O device, if the I/O device has a communication failure.

See Also [Reporting errors in I/O device data](#)  
[Suppressing reports](#)

### Reporting errors in I/O device data

If a communication error occurs (with an I/O device) or if the data is invalid, one of the following errors is written to the report (instead of the value):

Error	Meaning
#ASS	The value is incorrectly associated (with a substitution string or Genie).
#COM	Communication with the I/O device has failed
#DIV0	An attempt was made to divide a number by 0 (zero)
#ERR	An uncommon error has occurred. (Use the IsError function to find the error.)
#MEM	Out of memory or more than 64K bytes of memory requested.

Error	Meaning
#PEND	Data from this device is pending an initial update to display a value.
#RANGE	The value returned is out of range
#STACK	The value has caused a stack overflow
#WAIT	Data from this scheduled device is not available as it has not reached its scheduled interval and has no cache value.

For example:

**Report Format:**

{PV\_1} {SP\_1} {OP\_1}

If the above report is run when the value of PV\_1 is out of range (e.g. 101.5), SP\_1 is 42.35 and OP\_1 is 60.0, the output of the report is:

**Report Output:**

#RANGE 42.35 60.0

When reports are written to a database device, you might sometimes want to disable the error messages and write the values to the report (even if the values are invalid). Use the ERR\_FORMAT\_OFF command to disable all error messages and write all data as values.

For example:

**Report Format:**

{ERR\_FORMAT\_OFF}  
{PV\_1} {SP\_1} {OP\_1}

If the above report is run when the value of PV\_1 is out of range (e.g. 101.5), SP\_1 is 42.35 and OP\_1 is 60.0, the output of the report is:

**Report Output:**

42.35 60.0

To re-enable the error messages, use the ERR\_FORMAT\_ON specifier.

**Note:** If an I/O device goes offline and you have disabled communication errors, the value printed into the report is either 0 (zero) or the last value read from the I/O device when the report was last run. In either case, the value is invalid.

See Also [Suppressing reports](#)

**Suppressing reports**

You can suppress reports that are triggered from I/O devices if a communication error occurs by using the [Report]ComBreak parameter. For example, you might configure a report to be run every hour when a bit is on. The I/O device associated with that bit goes offline. If the [Report]ComBreak parameter is 0, the report does not run. If the parameter is 1, and if the latest valid value that was read from that bit was 1, the report is run.

---

This parameter only applies to the trigger of the report, not to the data in the report.



## Chapter 28: Using Security

---

For large applications, or applications where access to certain processes or machinery must be restricted, you can build security into your system. You can then restrict access to commands that should not be available to all your operators; for example, commands that operate specialized machinery, acknowledge critical alarms, or print sensitive reports.

You can assign a separate password to each of your operators (or class of operators), that must be entered before the operator can use the system.

See Also [Maintaining User Records](#)  
[Defining User Privileges](#)  
[Defining Areas](#)

### Maintaining User Records

You can add login records for some (or all) users of your runtime system. User records enforce an orderly login and restrict access to your system. Each operator for whom you add a user record must enter their user name and password to gain access to your runtime system.

You can add a user record for each of your users when you configure your project, or add a single record for each class (or type) of user (for example, Operators, Managers, Supervisors, and so on). When your system is running, you can add new users (based on a defined class) as required. Each class of users shares common attributes, such as privileges.

#### User records and project restoration

If you restore a project from a backup, or install a new project from a compiled offline master, the user records are reset to match those originally configured in the project. If the runtime user creation, password change ability, or password expiry functions are used, the runtime details might be thrown out of synchronization with master offline projects.

Here, you must have procedures in place to use the current `Users.dbf` file (which is running live in the plant) when offline project compilations are performed. This minimizes the likelihood of either losing users created at runtime, or of having expired user records locked when a new system is deployed and run up.

**Note:** Online changes arising from user creations and modifications are reflected only in the local `_Users.rdb` and `Users.dbf` files. To ensure that user records remain synchronized across a distributed network, the user administration

should only be performed on a central node. All other nodes will use the Copy= functionality in CitectSCADA or custom engineered database replication.

See Also [Adding user records](#)

## Adding user records

You must add user records for those people you want to be able to use your system.

To add a user record:

- 1 Choose **System** | **Users**. The Users dialog box appears.
- 2 Complete the Users dialog box.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [User properties](#)

## User properties

Use the Users dialog box to define properties for your users.

### User Name

The user's name. Enter a value of 16 characters or less. You can assign a user record for a single user, for example:

User Name	J Smith
User Name	John Smith

Each operator must enter the **User Name** and **Password** to use the system.

### Full Name

The full name of the user or class of user. Enter a value of 32 characters or less. This name is used as a comment and for display in alarm logs and command logs.

### Password

The user's password. Enter a value of 16 characters or less. When you enter the password, an asterisk (\*) will display for each character entered. When you save

the user record, the password will be encrypted before it is saved to the Users.dbf.

Each operator must enter the **User Name** and **Password** to use the system.

Use the [General>PasswordExpiry parameter to specify when the password will expire.

### **Confirm Password**

Re-enter the user's password to confirm the text entered in the **Password** field. Enter a value of 16 characters or less. If the contents of the **Password** and **Confirm Password** fields are different when the record is saved, a message will be displayed that indicates a mismatch and invites you to try again.

### **Global Privilege**

The privilege classes assigned globally to the user. Enter a value of 8 characters or less.

As you configure your system, you can assign privileges to the various elements, such as graphics objects, alarms, accumulators, commands, and so on. For example, a user with a Global Privilege of 3 will be able to issue any command that is assigned a privilege of 3, or action any alarm with a privilege of 3, or click any button that is assigned a privilege of 3, etc. Unless you are using areas, if you do not specify a global privilege, the user cannot access any command with a privilege assigned.

You can make your security more flexible by dividing your system into areas, and assigning users privileges or view-only rights to specific areas (see below).

**Note:** Global privileges will override the Viewable Areas settings you have applied for a user.

### **Type**

The generic type of user. Enter a value of 16 characters or less. For example:

Type	Operator
Type	Supervisor
Type	Manager

Only use this property to define a user class from which individual users (of that class) are to be created at runtime with the UserCreate() function.

### **Comment**

Any useful comment. Enter a value of 48 characters or less.

**Note:** The following fields are implemented with extended forms (press **F2**).

### Viewable Areas

The areas the user is permitted to view. Enter a value of 16 characters or less. Remember, however, you must still assign privileges to the elements in these areas, such as graphics objects, alarms, accumulators, commands, etc. If you do not, the user will have full access to them. For example, if you do not assign a privilege to, say, a command in one of these areas, the user will be able to issue it.

To make an element (such as a button on a [graphics page](#)) view only for a particular user, assign it an [area](#) and a privilege. Add the area to the user's list of Viewable Areas, but don't give the user the required privileges in that area (or the required global privilege).

Multiple areas can be defined using groups.

If you do not specify viewable areas, the user has access to the default area only (area 0).

### Areas for Priv 1 . . . Priv 8

The privileges (by area) assigned to the user. Enter a value of 16 characters or less. Using this combination of areas and privileges, you can assign a user different privileges for different areas. For example, users with privilege class 6 in areas 29 and 30 only have access to commands in those areas that require privilege class 6. This does not affect the Global Privileges (see above) assigned to the user. A user who has global privilege classes 1 and 2 can still access commands in all viewable areas that have privilege classes 1 and 2.

If you do not specify areas with associated privileges, access is defined by Viewable Areas and Global Privileges alone.

**Note:** The privileges entered in these fields will only apply if the relevant areas are listed in the Viewable Areas field above.

### Entry Command

A Cicode command that is executed when the user logs in. You can use any Cicode command or function. Enter a value of 64 characters or less.

### Exit Command

A Cicode command that is executed when the user logs out. You can use any Cicode command or function. Enter a value of 64 characters or less.

**Note:** To login a user, you must use the `Login( )` or `LoginForm( )` Cicode functions.

## Defining User Privileges

To restrict access to a particular system element (command, object, report, alarm, etc.), you assign it a **privilege** requirement, then allocate that privilege to the



users who will use it. CitectSCADA provides eight privileges, numbered 1 to 8. You can, for example, allocate different privileges to different types of operation, as in the following table:

Privilege	Command
1	Operate the conveyors
2	Operate the ovens
3	Operate the canners
4	Acknowledge alarms
5	Print reports

To allow a user to operate the conveyors, you assign privilege 1 to the user's login record, for example:

Global Privilege                      1

To allow a user to acknowledge alarms, you assign privilege 4 to the user's login record, for example:

Global Privilege                      4

To allow a user to acknowledge alarms and operate the conveyors, you assign both privilege 1 and privilege 4 to the user's login record:

Global Privilege                      1, 4

Privilege classifications must be separated by commas (,).

To allow a user access to all commands in your system, allocate all privileges in the user record, for example:

Global Privilege                      1, 2, 3, 4, 5

After you have allocated privileges, you can define the privilege requirements of your system elements (commands, reports, objects, alarms, etc.):

Command	CONVEYOR = 1;
Privilege	1
Comment	An Operator with Privilege classification 1 can operate the conveyor
Command	Report("Shift");
Privilege	5
Comment	An Operator with Privilege classification 5 can print the report

Not all system elements need a privilege classification. At least one command must be issued by all users, a command to log in to the system:

Command	LoginForm();
Privilege	
Comment	A blank Privilege (or Privilege 0) means that the command has no classification - it is available to all users

See Also [Using hierarchical privilege](#)

## Using hierarchical privilege

By default, privileges are non-hierarchical (i.e. users with privilege 3 only have access to commands with classification 3). Non-hierarchical privileges add flexibility to your system, especially when used with the area facility.

When privileges are set to hierarchical, privilege 1 is the lowest and 8 is the highest (i.e. users with privilege 3 have access to commands with privilege classification 3, 2, and 1). To allocate all privileges, you would only need to specify privilege 8.

Global Privilege	8
------------------	---

Using the privilege facility, you can easily develop a secure CitectSCADA system. You should, however, carefully plan your security method before you set up your system. You need to decide which commands you can group into a class, the privilege for each class of commands, and the privileges to assign to each operator.

**Note:** If your plant can be divided into several discrete sections (or areas), you can add an extra level of system security by using the CitectSCADA area facility.

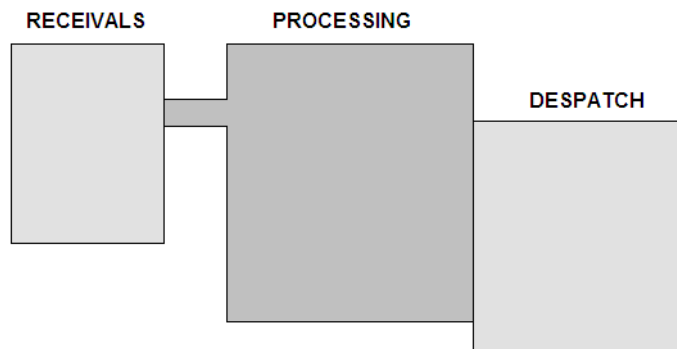
## Defining Areas

When implementing CitectSCADA for a large application, you would usually visualize the plant as a series of discrete sections or **areas**. You can define these areas geographically (especially where parts of the plant are separated by vast distances or physical barriers) or logically (as discrete processes or individual tasks).

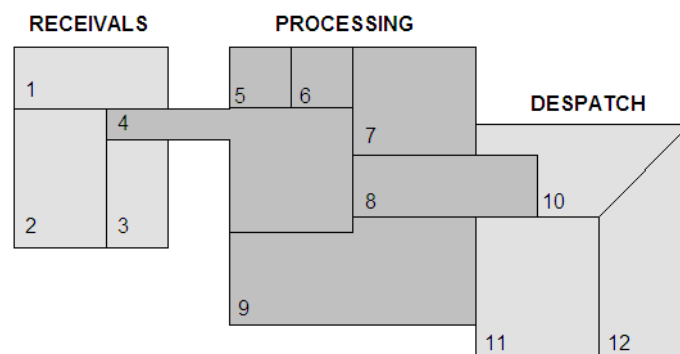
**Note:** The area facility is implemented with extended properties.

By thinking about your plant in terms of areas, you can add flexibility to your system security. Without areas, you can only assign global privileges to users. A user with a global privilege can access any part of the system with a matching privilege. Areas, on the other hand, allow you to add an extra level of control. Instead of assigning a global privilege, you can assign a user different privileges for different areas. You can then assign each of your system elements (objects, alarms, reports, accumulators, etc.) a privilege requirement, and allocate each to a specific area. This means that a user has full control only when he or she has access to the required area and possesses the required privileges for that area.

Some plants can be divided into just three areas - raw product arrives in the receives area, is transported to an area for processing, and is then transported to a packaging or despatch area.

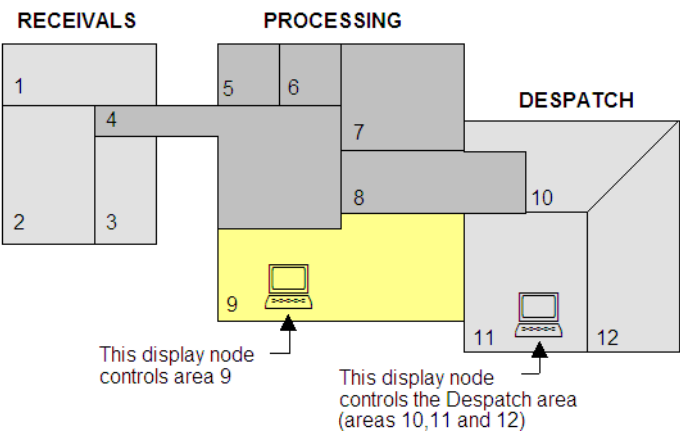


However, with larger or more complex plants you might need to define several areas, like this:



When defining an area, you would usually encompass a section of the plant that is controlled by one operator (or controlled from one CitectSCADA display [client](#)).

You can also define smaller areas that are collectively controlled by an operator or [display client](#). This method can increase flexibility, but can introduce a higher level of complexity to your system.



You can define up to 255 separate areas. You can then refer to these areas by number (1 to 255) or you can use a label to assign a meaningful name for the area (e.g. receivals, pre\_process, conveying, etc.).

- See Also
- [Using areas for security](#)
  - [Using labels to name areas](#)
  - [Using groups of areas](#)
  - [Using areas with privileges](#)
  - [Specifying security requirements](#)
  - [Viewing areas of the plant](#)

**Using areas for security** After you have defined your areas, you can configure the commands, objects, alarms, reports, etc. your operators will use in those areas.

For example:

```
Command      CONVEYOR = 1;
Area          10
Comment      This command belongs to Area 10
```

In this example, an operator without access to Area 10 will not be able to issue the command.

- See Also
- [Using labels to name areas](#)

**Using labels to name areas** It might be easier to remember an area by a meaningful label (name) rather than a number. For example:

```
Label Name    DespatchAccum
Expression    10
```

Comment	Label Area 10 as "DespatchAccum"
---------	----------------------------------

In this case, "DespatchAccum" could be used whenever area 10 is referred to, for example:

Command	CONVEYOR = 1;
Area	DespatchAccum
Comment	This command belongs to Area 10 (DespatchAccum)

**Note:** If you leave the Area field blank on a form, the command does not belong to any particular area - it is assigned to all areas of the plant.

#### To label an area:

- 1 Choose **System** | **Labels**.
- 2 Enter a **Name** for the label.
- 3 Enter an expression to be substituted for the label.
- 4 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Using groups of areas](#)

### Using groups of areas

You can group several areas and define a name for the group.

Group Name	Despatch
Association 1	DespatchAccum
Association 2	11
Association 3	12
Comment	Areas 10, 11, 12 = "Despatch"

In the above example, areas 10, 11, and 12 are associated with the name "Despatch". Any command assigned to "Despatch" belongs to areas 10, 11, and 12.

Command	CONVEYOR = 1;
Area	Despatch
Comment	This command belongs to Areas 10, 11 and 12

You can also define a group that includes other groups.

Group Name	Plantwide
Association 1	Receivals
Association 2	Process
Association 3	Despatch
Comment	Associate all areas with "Plantwide"

In this example, the name "Plantwide" refers to all areas defined in the "Receivals", "Process", and "Despatch" groups.

To define a group of areas:

- 1 Choose **System** | **Groups**. The Groups dialog box appears.
- 2 Complete the Groups dialog box.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Groups properties](#)

## Groups properties

Use the Groups dialog box to configure properties of groups:

Groups have the following properties:

### Group Name

The name of the group. You can use this facility, for example, to define multiple areas or multiple devices. Enter a value of 16 characters or less.

After you have defined a group, it can be used anywhere that an individual entity can be used. You can also specify complex groups by defining a group of groups.

### Association 1 . . . Association 10

A list of the entities associated with the Group Name. Enter a value of 16 characters or less. An Association can be a number, a name, or another group. You can also specify a range of numbers in the format <n1..n2> for example:

Association 1            4..10

Specifies numbers 4,5,6,7,8,9,10.

You can also define a group of devices to be accessed with a single name, for example:

Association 1            AlarmPrint  
Association 2            AlarmLog

Association 3 AlarmDBF

In this case, when the group name is used as a device, the information is sent to all three devices - AlarmPrint, AlarmLog, and AlarmDBF.

### Comment

Any useful comment. Enter a value of 48 characters or less.

## Using areas with privileges

By combining area and privilege restrictions, you can select what control an operator has within a specific area. You can still assign privileges to each of your operators without using areas - to allow them access to the entire plant (global privileges), but by combining Areas and Privileges, you add an extra level of flexibility.

User Name	J Smith
Global Privilege	2, 3
Viewable Areas	9, 10, 11, 12
Areas for Priv 1	Despatch
Areas for Priv 2	
Areas for Priv 3	
Areas for Priv 4	DespatchAccum
Areas for Priv 5	DespatchAccum, 11
Areas for Priv 6	
Areas for Priv 7	
Areas for Priv 8	
Comment	Login for John

Here, John Smith has global privileges 2 and 3; he can use commands with privilege classification 2 or 3 in any viewable area of the plant. He has privilege 1 in the "Despatch" areas (10, 11, and 12), privilege 4 in the "DespatchAccum" area (10) and privilege 5 in areas 10 and 11. This means he can control system elements (alarms, reports, accumulators, objects, etc.):

- Located in area 9, with privilege requirement 2 or 3.
- Located in area 10, with privilege requirement 1, 2, 3, 4 or 5.
- Located in area 11, with privilege requirement 1, 2, 3 or 5.
- Located in area 12, with privilege requirement 1, 2 or 3.

Also, in this example, Groups and Labels have been used to make the security configuration intuitive.

See Also [Specifying security requirements](#)

## Specifying security requirements

Each of your system elements (objects, alarms, reports, accumulators, etc.) can be assigned a privilege requirement and allocated to a specific area. For a user to

be able to acknowledge an alarm, for example, he or she must have access to the correct area, with the required privileges for that area.

For example:

Command	CONVEYOR = 1;
Privilege	1
Area	10
Comment	This command belongs to Area 10, and requires privilege 1

In this example, an operator without privilege 1 in Area 10 will not be able to issue the command.

### Privilege - area combinations

It is important to know how the various privilege - area combinations will affect your security.

Privilege specified?	Area specified?	Resulting Security
Yes	Yes	Operator must have the required privileges for the area specified.
Yes	No	Security is determined by the user's Global Privileges alone.
No	Yes	Operators only need view-access to the area specified.
No	No	All operators have full control.

See Also [Viewing areas of the plant](#)

### Viewing areas of the plant

You might need to provide an operator with access to information from other areas of the plant - without providing control of the process in those areas. For example, the processes in one area might directly affect another area.

In the following example, John Smith has control of:

- Any system element with a privilege requirement of **2** or **3**;
- System elements located in **Despatch**, with a privilege requirement of **1**; and
- System elements located in **DespatchAccum**, with a privilege requirement of **4**.

Everything else in the plant is view-only.

User Name	J Smith
Global Privilege	2, 3
Viewable Areas	Plantwide
Areas for Priv 1	Despatch
Areas for Priv 2	
Areas for Priv 3	
Areas for Priv 4	DespatchAccum



---

Areas for Priv 5

Areas for Priv 6

Areas for Priv 7

Areas for Priv 8

Comment

Login for John

Alternatively, you could restrict an operator to a group of areas (e.g., "Receivals") or to a single area (e.g., 12).



## Chapter 29: Using Labels

Labels allow you to use a series of commands (or expressions) in your system without having to repeat them each time they are used. When you compile the project, the commands (or expressions) defined in the label are substituted in every occurrence of the label.

Labels are similar to macros used in programming languages such as Basic or 'C'.

You often use the same combination of statements in different commands. For example, when an operator acknowledges an alarm, you might want to log the details to a file. Such a command would require several statements:

Command            `FileWrite(AlarmFile, Time()); FileWrite(AlarmFile, Date()); . . .`

Instead of entering the same statements when required in a command, you can define a label, and then use the label instead of the statements. When you compile your project, each occurrence of the label is resolved; that is, the expression in the label is substituted for the label name. For example:

Label	
Label Name	<input type="text" value="_Log_Alarms"/>
Expression	<input type="text" value="FileWrite(AlarmFile, Time());&lt;br/&gt;FileWrite(AlarmFile, Date()); . . ."/>

Notice the use of an underscore to identify the label.

Once defined, a label can be used as a statement in a command, for example:

System Keyboard	
Key Sequence	<input type="text" value="F5 Enter"/>
Command	<input type="text" value="_Log_Alarms;"/>
Privilege	<input type="text" value="1"/>

When an operator issues this command, the expression defined in the label is substituted in the command.

Label	
Label Name	<input type="text" value="_Log_Alarms"/>
Expression	<input type="text" value="FileWrite(AlrmFile, Time());&lt;br/&gt;FileWrite(AlrmFile, Date()); ..."/>

When an operator issues the command, the statements in the expression execute

System Keyboard	
Key Sequence	<input type="text" value="F5 Enter"/>
Command	<input type="text" value="_Log_Alarms;"/>
Privilege	<input type="text" value="1"/>

You can also use the label in combination with other statements, for example:

Label	
Label Name	<input type="text" value="_Log_Alarms"/>
Expression	<input type="text" value="FileWrite(AlrmFile, Time());&lt;br/&gt;FileWrite(AlrmFile, Date()); ..."/>

The label expression is substituted in each occurrence of the label

System Keyboard	
Key Sequence	<input type="text" value="F5 Enter"/>
Command	<input type="text" value="AlarmAck(0,0);_Log_Alarms;"/>
Privilege	<input type="text" value="1"/>

Users	
User Name	<input type="text" value="Supervisor"/>
Entry Command	<input type="text" value="PageAlarm();_Log_Alarms;"/>
Privilege	<input type="text" value="1"/>

The main advantage of a label is that it is a global definition, recognized throughout the CitectSCADA system. If you want to change something (in the above example you might change the file name or the way the data is logged),

you only need to change it in one place - in the label definition. All other occurrences of the label name will reflect the changes.

See Also [Using Arguments in Labels](#)  
[Converting Values into Strings](#)  
[Substituting Strings](#)  
[Defining Labels](#)

## Using Arguments in Labels

You can define labels that accept arguments enclosed in parentheses (). The following example shows a label that increments a variable by a specific value:

Label Name	Inc(X, STEP)
Expression	X = X + STEP

Here, "X" is the variable to be incremented and "STEP" determines the amount of the increment. You can then use this label in a command, as in the following example:

Key Sequence	FastInc
Command	Inc(SP12, 10);

An operator can use this command to increment the value of SP12 by 10.

### Specifying default values

You can specify a default value for an argument when you define a label, for example:

Label Name	Inc(X, STEP = 10)
Expression	X = X + STEP

When you subsequently use this label without any arguments in a command, the default value is used, for example:

Key Sequence	FastInc
Command	Inc(SP12);

See Also [Converting Values into Strings](#)

## Converting Values into Strings

Sometimes, you must convert a value into a string before it can be used. In the following example, the value of a tag is converted before it is used in the DspStr() function.

Label Name	ShowVariable(TAG)
Expression	DspStr(25, "BigFont", #TAG + "=" + TAG:##.##);

In the above example, only one argument (TAG) is passed to a function that actually requires three arguments (AN, font and message). When you use this label in a command, the function always uses AN 25 and the message always displays in "BigFont". Only the third argument (the actual message) varies.

The third argument passed to the function is:

```
... #TAG+ "=" +TAG:##. #
```

#TAG indicates that the name of the tag (and not its value) is displayed.

TAG:##. # indicates that the value of TAG is converted to a string and displayed. It is formatted with two numbers before the decimal point and one number following the decimal point.

You can use the above label in a command such as:

```
Command          ShowVariable(SP12);
```

When you use this command in your runtime system, the command displays "SP12=<value>", where **value** is the actual value of SP12 at the time (e.g. **SP12=42.0**).

See Also [Substituting Strings](#)

## Substituting Strings

You can pass a string substitution as an argument in a label, for when several variables have part of the variable name in common; for example:

```
Label Name      SPDev(TAG)
Expression      Prompt("Deviation=" + "IntToStr(CP##TAG## -
                  SP##TAG##));
```

In the above example, TAG is the common portion of the variable name, and is substituted at each occurrence in the expression. To display the difference between two variables CP123 and SP123, you would specify SPDev(123) in a command, for example:

```
Command          SPDev(123);
```

You cannot use a substitution within a string. In the following example, the DESC Parameter (a text description) will not be substituted as it is between quotation marks:

```
Prompt("Deviation for ##DESC##=" + "IntToStr(CP##TAG## -
      SP##TAG##) )
```

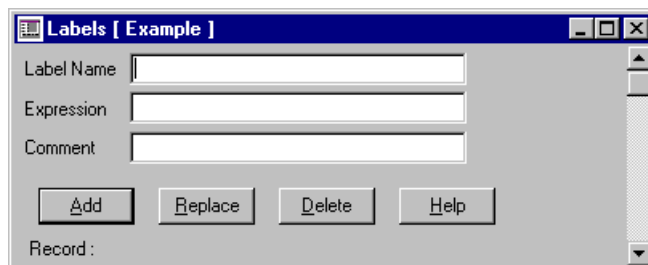
See Also [Defining Labels](#)

## Defining Labels

You can define labels to use in your system.

**To define a label:**

- 1 Choose **System | Labels**. The Labels dialog box appears.



The screenshot shows a Windows-style dialog box titled "Labels [ Example ]". It has a standard title bar with minimize, maximize, and close buttons. Inside the dialog, there are three text input fields stacked vertically, labeled "Label Name", "Expression", and "Comment". Below these fields are four buttons: "Add", "Replace", "Delete", and "Help". At the bottom left of the dialog, there is a label "Record:" followed by a small downward-pointing arrow, indicating a dropdown menu.

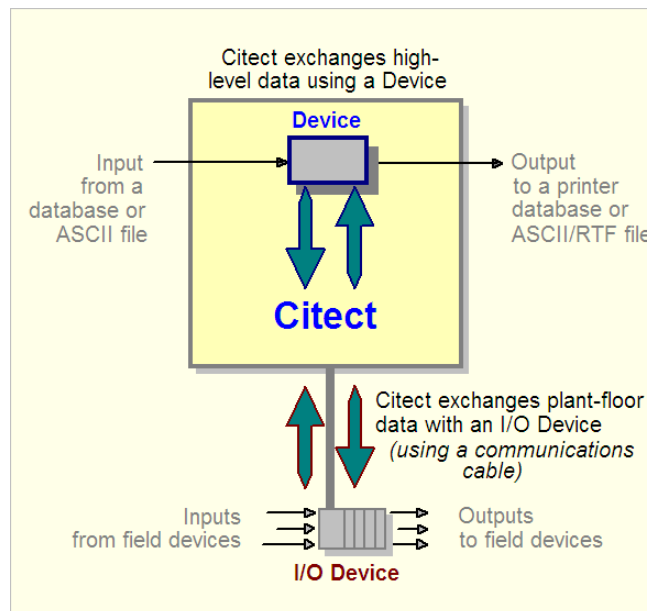
- 2 Enter a **Label Name** of 64 characters or less. Whenever this name is used (i.e., in Cicode or a field), CitectSCADA automatically substitutes the expression below.
- 3 Enter an **Expression** to be substituted for the label (maximum length is 254 characters). You can use a label to substitute a name for an entity or Cicode expression; for instance, when you use the entity (or Cicode expression) in several database records.
- 4 Add a **Comment** (of 48 characters or less).
- 5 Click **Add** to append a new record, or **Replace** to modify an existing record.



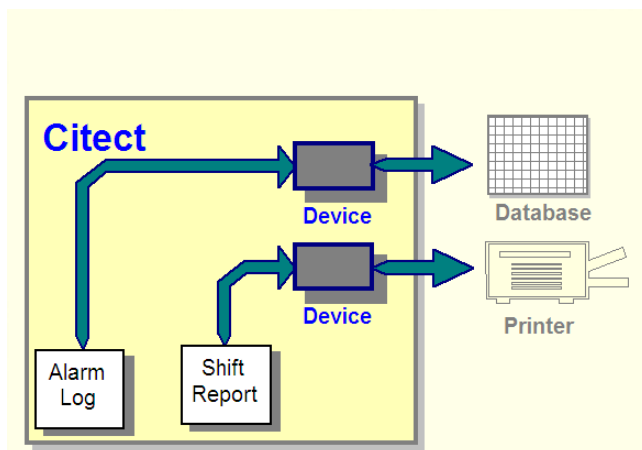


## Chapter 30: Using Devices

A device transfers high-level data (such as a report, command log or alarm log) between CitectSCADA and other elements (such as a printer, database, RTF file, or ASCII file) in your CitectSCADA system. Devices are similar to I/O devices in that they both allow CitectSCADA to exchange data with other components in your control and monitoring system.



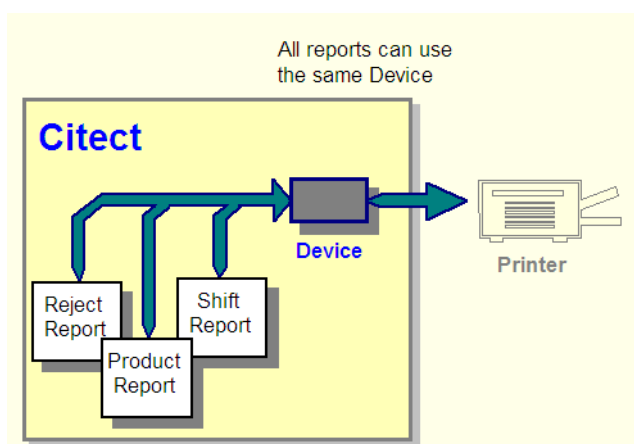
You can use devices for various purposes; for example, to send the output of a report to a printer, or write data to a database.



Using a device you can write data to:

- RTF files
- ASCII files
- dBASE databases
- SQL databases (through ODBC-compliant drivers)
- Printers (connected to your CitectSCADA computer or network)

You can configure any number of devices; however, a device is a common resource. You can, for example, configure a single device that sends the output of all your CitectSCADA reports to a printer (when they are requested).

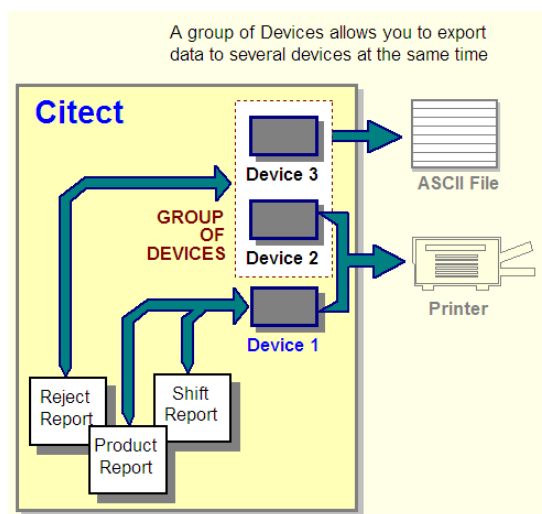


See Also [Using groups of devices](#)

[Configuring Devices](#)  
[Formatting Data in the Device](#)  
[Using Device History Files](#)

## Using groups of devices

You can add flexibility to your system by using a group of devices. A group of devices allows you to export the same data to two (or more) locations.



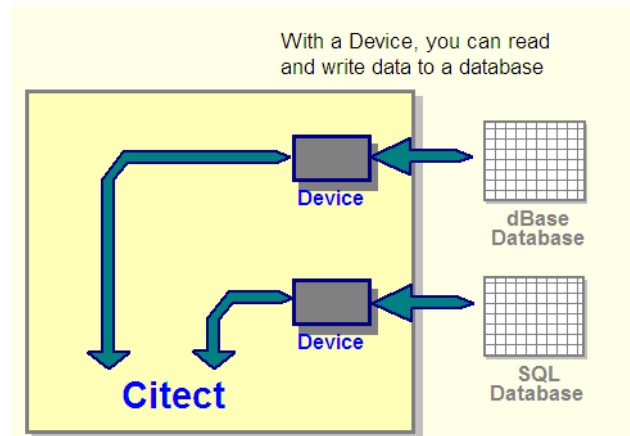
See Also [Using devices to read data](#)

## Using devices to read data

Using a device (and Cicode functions), you can also read data from:

- ASCII files
- dBASE databases

## ■ SQL databases



**Note:** When you read from a group of devices, data is only read from the first device in the group.

See Also [Configuring Devices](#)

## Configuring Devices

You must configure your devices before you can use them with your CitectSCADA system.

**To configure a device:**

- 1 Choose **System | Devices**. The Devices dialog box appears.
- 2 Complete the Devices dialog box using the description of the text boxes below.

- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Devices Properties](#)

## Devices Properties

[Devices](#) have the following properties:

### Name

The name of the device. The device name can be the name of a group of devices, or a label for a device. Enter a value of 16 characters or less.

See Also [Using groups of devices](#)  
[Using Labels](#)  
[Predefined Devices](#)

### Format

Specifies how the data is formatted in the device. The format is determined by the type of Device, and the data that is sent to the device. Enter a value of 120 characters or less.

If you are logging alarms or command messages, you must specify a format, or no data is written to the device.

**Note:** The log device for a command is specified wherever the command is defined. The log device for an alarm is specified at the Alarm Categories form.

When producing reports, the format is ignored. (The format defined for the report is used to write the report to the device.)

See Also [Formatting Data in the Device](#)  
[Alarm display fields](#)  
[Alarm summary fields](#)  
[Using Command Fields](#)

**Header**

Additional information for the device. Enter a value of 120 characters or less.

**Printer devices**

The header is printed on each page. A new page is created each time the form length is reached. The [Device]FormLength parameter is used to set the form length.

**ASCII file devices**

Do not use this property.

**dBASE database devices**

Contains the field name used to index the database, for example:

Header	{Name}
--------	--------

**Note:** Index Key fields must not exceed 100 characters.

**SQL database devices**

The connection string for the particular database type.

**Note:** CitectSCADA database devices only support STRING data types. If you use another database editor to modify your database, you must ensure that all fields are in string format.

**File Name**

The file name of the device. Enter a value of 64 characters or less.

**Printer devices**

The printer port or UNC name, for example:

File Name	LPT1:
File Name	COM2:
File Name	\\PrintServer\BubbleJet1

When you specify a printer port, you must include the colon character (:), otherwise CitectSCADA tries to write to a file (device) with a name similar to the printer port (i.e. LPT1 or COM2).

**Note:** When using a UNC name in Windows 95, the printer must be in the Printers section of the Control Panel.

**ASCII file devices and dBASE database devices**

The name of the active file, for example:

File Name	ALARMLOG.TXT
File Name	[DATA]:ALARMLOG.TXT

This property is optional. If you do not specify a file name, **File Name** defaults to \CitectSCADA 7\bin<Name> on the hard disk where you installed CitectSCADA. <Name> is the first eight characters of the device name. If you use this property, ensure that no other devices have the same first eight characters in the device name.

### SQL database devices

The database table, for example:

File Name	LOGFILE
File Name	REPTBL

### Type

The type of device. Enter a value of 16 characters or less.

Device Type	Device Description
ASCII_DEV	ASCII file*
PRINTER_DEV	Printer
dBASE_DEV	dBASE file
SQL_DEV	SQL database

\* When defining RTF report properties, an ASCII device would be selected if the report was to be saved as a file.

This property is optional. If you do not specify a type, the device **Type** is ASCII\_DEV unless:

The file name is a printer device (LPT1: to LPT4: or COM1: to COM4: or a UNC name), where **Type** is PRINTER\_DEV, or

The file name extension is .DBF, where **Type** is dBASE\_DEV.

See Also [About Print Management](#)

### No. Files

The number of history files. Enter a value of 4 characters or less.

By default, CitectSCADA creates a single data file for each device. (This data file is called <filename.TXT> or <filename.DBF>, depending whether the device is an ASCII device or database device.) The number of history files you specify here are in addition to the data file.

**Note:** If you do not want history files created, you must enter 0 (zero) here, and set the [Device]CreateHistoryFiles parameter to 0; otherwise, 10 history files will be created as a default. You must also ensure that the data file is of a fixed size. (If the data accumulates, the file eventually fills the hard disk.)

If you specify -1 the data is appended to the end of one file.

If you are logging alarm, keyboard commands, or reports to the device, specify the number of files to be created, and the time of each file.

See Also [Using Device History Files](#)

#### **Time**

The time of day to synchronize the beginning of the history file, in hh:mm:ss (hours:minutes:seconds). Enter a value of 32 characters or less. If you accepted the default number of history files above, and you specify a time and period, 10 history files will be created. If you do not specify a time, the file is synchronized at 0:00:00 (i.e. midnight).

If you omit both the time and the period, additional history files will still be created (with the default time and period). If you don't want history files to be created, you must set the [Device]CreateHistoryFiles parameter to 0 (zero).

#### **Period**

The period of the history file, in hh:mm:ss (hours:minutes:seconds). Enter a value of 32 characters or less. Alternatively you can:

- Specify a weekly period by entering the day of the week on which to start the history file, e.g. Monday, Tuesday, Wednesday, etc.
- Specify a monthly period by entering the day of the month on which to start the history file, e.g. 1st, 2nd, 3rd, 4th, 5th, etc.
- Specify a yearly period by entering the day and month on which to start the history file, e.g. 1st January, 25th February, etc. The day and month must be separated by a space.

If you accepted the default number of history files above, and you specify a time and period, 10 history files will be created.

If you do not specify a period, the period defaults to Sunday (weekly).

If you omit both the time and the period, additional history files will still be created (with the default time and period). If you don't want history files to be created, you must set the [Device]CreateHistoryFiles parameter to 0 (zero).

#### **Comment**

Any useful comment. Enter a value of 48 characters or less.

## **Formatting Data in the Device**

The device format specifies how to format the data in the device. The format is determined by the type of device, and the data that is sent to the device.

- [Printer and ASCII devices format](#)
- [dBASE and SQL database devices format](#)



See Also [Using a database device](#)

## Printer and ASCII devices format

The format specifies how each line of data is printed on the printer or written to the ASCII file, for example:

```
RFP3   Raw Feed pump 3      Overload    12:32:21
RFP9   Secondary Feed      Overtemp     13:02:45
```

When producing reports, the device format is ignored. The format defined for the report (i.e. the report format file) is used to write the report to the device.

To include CitectSCADA data you must specify the field name and (optionally a width for each field to be printed or written to the file. The format has the following syntax:

```
{<field name>, [width[, justification]]}
```

You must enclose each field in braces {}, for example:

```
Format           {Tag,8}{Name,32}
```

In this case, two fields are printed or written to the file - Tag, with 8 characters, and Name, with 32 characters. The width specifier is optional - if you do not specify a width, the width of the field is determined by the number of characters between the braces, for example:

```
Format           {Name }
```

In this case, Name is followed by four spaces - the field is printed or written to the file with 8 characters.

### Creating lists and tables

To set the justification of the text in each field, use a justification specifier. You can use three justification characters, L (Left), R (Right), and N (None) - for example:

```
Format           {Tag,8,L} {Name,32,R}
```

The justification specifier is optional - if it is omitted, the field is left justified. If you use a justification specifier, you must also use the width specifier.

To display field text in columns, use the tab character (^t) - for example:

```
Format           {Tag,8}^t{Name,32}^t{Desc,32} {Time,8,R}
```

### Including fixed text

You can include fixed text by specifying the text exactly as it is to be printed or written to the file - for example:

```
Format           Name of Alarm:
```

Any spaces that you use in a text string are also included in the string.

See Also [Formatting Data in the Device](#)

The format specifies the structure (field names and field widths) of the database. The format has the following syntax:

You must use braces ( { } ) to enclose each field, for example:

Format	{Tag,8}{Name,32}
--------	------------------

You can define your own fields (as well as the standard CitectSCADA fields) for the database device, for example:

```
Format {Name,16}{Water,8}{Sugar,8}{Flour,8}
       {Salt,8}{Yeast,8}{Milk,8}
```

[illegible]

In this example, the database is created with seven database fields. To access the above dBASE device, use a Cicode function similar to the following:

```
hDev = DevOpen("Recipe");
DevFind(hDev, "Name", "Bread");
PLC_Water = DevGetField(hDev, "Water");
PLC_Sugar = DevGetField(hDev, "Sugar");
. . .
. . .
DevClose(hDev);
```

If the database does not exist, it is created with the specified format. If you do not specify a format, and if the file name specifies an existing dBASE file, CitectSCADA uses the existing fields in the database.

### SQL devices

You must create the SQL database by an external application before it can be used.

**Note:** If you edit a dBASE or SQL device record (in an existing project), the associated physical device is not edited. For example, if the device is a dBASE type device and you add an extra field in the device, the extra field is not added to existing database files (when you run CitectSCADA). New files are created with the edited fields. If you want to keep the existing device database data, you must manually copy the data. (Use dBASE, Excel or some other database tool.) If you don't need to keep the existing data, delete the existing database files. The next time CitectSCADA tries to open the device, it creates the database with the required changes.

See Also [Formatting Data in the Device](#)  
[Using a database device](#)

### Using a database device

Before you can use a database device, you must open it. You can open several devices at the same time. The `DevOpen()` function returns an integer handle to identify each device, as in the following example:

```
INT hRecipe;
hRecipe = DevOpen("Recipe");
```

### Writing dBASE records using a CitectSCADA database device

To write data to a database device, first append a record to the end of the device, then add data to the fields of the record. For a dBASE database, the `DevAppend()` function appends the record, and the `DevSetField()` function writes data to a field. The following function writes a recipe record to a device:

```
FUNCTION
WriteRecipeData(INT hDevice, STRING sName, INT Water, INT Sugar,
INT Flour, INT Salt, INT Yeast, INT Milk)

DevAppend(hDevice);
DevSetField(hDevice, "NAME", sName);
DevSetField(hDevice, "WATER", IntToStr(Water));
DevSetField(hDevice, "SUGAR", IntToStr(Sugar));
DevSetField(hDevice, "FLOUR", IntToStr(Flour));
DevSetField(hDevice, "SALT", IntToStr(Salt));
DevSetField(hDevice, "YEAST", IntToStr(Yeast));
DevSetField(hDevice, "MILK", IntToStr(Milk));
END
```

### Writing SQL records using a CitectSCADA database device

To use an SQL device in CitectSCADA, you cannot use all the Cicode Device functions; you can only use the following functions:

```
DevOpen()      DevClose()  DevGetField() DevFind()   DevWrite()
DevNext()      DevSeek()   DevAppend()   DevWrite()  DevZap()
DevControl()
```

To write CitectSCADA data to an SQL database, use the DevWrite() function, but you must add a new record and write to all fields in the new record. No data is written to the database if you do not write to all fields.

For example:

```
FUNCTION
WriteRecipeData(INT hDevice, STRING sName, INT Water, INT Sugar, INT Flour, INT Salt, INT
Yeast, INT Milk)

DevWrite(hDevice, sName);
DevWrite(hDevice, Water);
DevWrite(hDevice, Sugar);
DevWrite(hDevice, Flour);
DevWrite(hDevice, Salt);
DevWrite(hDevice, Yeast);
DevWrite(hDevice, Milk);
END
```

The following functions are not compatible with the SQL devices and should not be used with a SQL devices:

```
DevFlush()      DevPrev()      DevSize()      DevRecNo()
DevDelete()     DevRead()      DevSetField()
```

### Locating and reading database records using a CitectSCADA database device

To read data from a dBASE or SQL database device, use the DevFind() function to locate the record, and then the DevGetField() function to read each field:

```
FUNCTION
GetRecipe(STRING sName)

INT hDev;

hDev = DevOpen("Recipe");
IF hDev >= 0 THEN
    IF DevFind(hDev, sName, "NAME") = 0 THEN
        PLC_Water = DevGetField(hDev, "WATER");
        PLC_Sugar = DevGetField(hDev, "SUGAR");
        PLC_Flour = DevGetField(hDev, "FLOUR");
        PLC_Salt = DevGetField(hDev, "SALT");
        PLC_Yeast = DevGetField(hDev, "YEAST");
        PLC_Milk = DevGetField(hDev, "MILK");
```

```

        ELSE
            DspError("Cannot Find Recipe " + sName);
        END
        DevClose(hDev);
    ELSE
        DspError("Cannot open recipe database");
    END
END

```

#### Deleting records using a CitectSCADA database device

You can delete dBASE records with the `DevDelete()` function. The following Cicode function deletes all records from a dBASE device:

```

FUNCTION DeleteRecords(INT hDev)

    WHILE NOT DevEOF(hDev) DO
        DevDelete(hDev);
        DevNext(hDev);
    END
END

```

To delete all records from a dBASE database, use the `DevZap()` function:

```

FUNCTION DeleteRecords(INT hDev)

    DevZap(hDev);
END

```

To delete records from an SQL database, use the Cicode SQL functions.

#### Closing a CitectSCADA database device

When finished with a device, close it to free Cicode system resources by using the `DevClose()` function:

```
DevClose(hRecipe);
```

#### To define a group of devices:

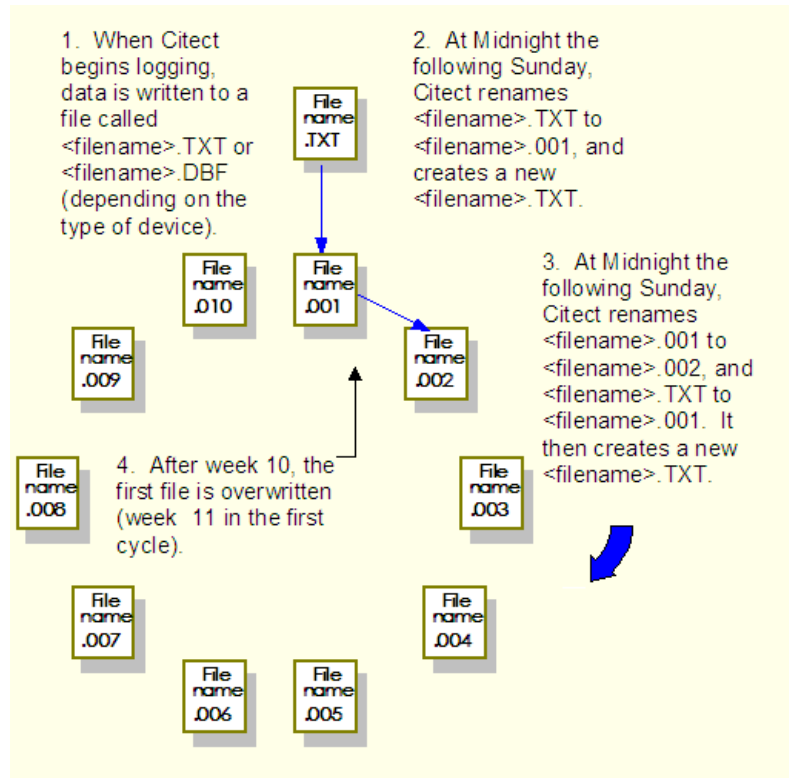
- 1 Choose **System** | **Groups**. The Groups dialog box appears.
- 2 Complete the Groups form.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Using Device History Files](#)

## Using Device History Files

To makes the long-term storage of logged data easier to organize and more accessible, CitectSCADA uses a system of rotational history files to store historical data. To use this system, you must specify how many device history

files you want to keep. For example, if you want to keep 10 history files, they would be saved rotationally as illustrated below:



Note the 10 history files are in addition to the default data file that is saved for all devices.

By default, CitectSCADA uses 10 files (if history files are specified). You can change the default by specifying the number of files to use, for example:

No. Files	20
Comment	CitectSCADA uses twenty files for the data

The maximum number of files you can specify is 999.

You can also specify the period between files, i.e., when a new history file is used, for example:

Period	1:00:00
Comment	Use a new file each hour
Period	6:00:00
Comment	Use a new file every six hours

Period	72:00:00
Comment	Use a new file every three days
Period	Monday
Comment	Use a new file each week beginning on Monday
Period	15th
Comment	Use a new file every month beginning on the 15th of each month
Period	25th June
Comment	Use a new file every year beginning on the 25th of June

For best system performance, specify a period of at least one week.

You can also specify the time of day to synchronize the beginning of the history file, for example:

Time	6:00:00
Comment	Synchronize the file at 6:00 am
Time	12:00:00
Comment	Synchronize the file at 12:00 midday
Time	18:30:00
Comment	Synchronize the file at 6:30 pm

The first file does not actually begin at this time: the first file begins when you start your runtime system. The time and period together determine when new history files are created, for example:

Time	6:00:00
Period	Monday

In the above example, CitectSCADA creates a new file each Monday at 6:00am. If you start your runtime system at 7:30am on Sunday, your first file only contains 22.5 hours of data. If you leave your system running, subsequent files start each Monday at 6:00am, and contain one full week of data.

#### Archiving data

To archive your data for long-term storage, back up your history files before they are overwritten. Use the Windows File Manager from the Main program group to check file creation dates (of the history files) and to back up files. Refer to your Windows documentation for a description of the File Manager.

See Also [Using Command Fields](#)

## Using Command Fields

You use the following fields (or combination) to format a command logging device:

Field Name	Description
{UserName,n}	The name of the user (User Name) who was logged on when the command was issued.
{FullName,n}	The full name of the user (Full Name) who was logged on when the command was issued.
{Time,n}	The time (in short format) when the command was issued (hh:mm).
{TimeLong,n}	The time (in long format) when the command was issued (hh:mm:ss).
{Date,n}	The date (in short format) when the command was issued (dd:mm:yy).
{DateLong,n}	The date (in long format) when the command was issued (day month year).
{DateExt,n}	The date (in extended format) when the command was issued (dd:mm:yyyy).
{Page,n}	The page that was displayed when the command was issued.
{MsgLog,n}	The message sent as the <i>Message Log</i> property (of the command record).

You can use the following fields (in the command field) for **Keyboard commands only**:

{Arg1,n}	The first keyboard command argument (if any).
{Arg2,n}	The second keyboard command argument (if any).
...	
{Arg8,n}	The eighth keyboard command argument (if any).
{Native_MsgLog,n}	The <a href="#">native language</a> version of the message sent as the <i>Message Log</i> property (of the command record).

Where **n** specifies the display field size.

For example, you could have a device configured as follows:

Name	KeyLog
Format	{Date,9} {MsgLog,27} {Arg1,3} by {FullName,11}

Then a [keyboard command](#) (object, page, or system) could be created with the following configuration:

Log Device	KeyLog
Key Sequence	### ENTER
Log Message	Density setpoint changed to

Resulting in an output of the following kind: "01/01/99 Density setpoint changed to 123 by Timothy Lee".

See Also [About Print Management](#)

## About Print Management

The Windows printer management has been designed for page-based printers: laser printers and shared network printers. The printer driver does not print



---

anything on the printer until the entire page is complete; it then prints the page. This is the preferred printing method (when printers are shared on a network), because it prevents conflict of data when more than one operator uses the print facility at the same time.

However, this method is inappropriate when logging alarms or keyboard commands. If you send alarm logging to this type of printer, CitectSCADA flushes the data to the printer when the current page is full, or when the [DEVICE]FlushTime parameter has been exceeded (it defaults to 10 seconds). If, for example, you have one alarm occurring each minute, each alarm is printed on a new page (because the default flush time is less than the alarm frequency).

You can bypass the Windows print management by writing the output to a file. Set the device type to ASCII\_DEV and specify the file name as lpt1.dos, lpt2.dos or lpt3.dos (depending on the port to which your printer is connected). The printer must be either on a local port, or a captured network printer. When you log to this device, the data is printed immediately on the printer with no extra form feeds.

For correct logging operation, reserve one printer to be your logging printer. This printer should be a local printer, not on the network server. You can then send any other non-logging printouts, (e.g., reports) to a shared network or local printer.

See Also [Using Devices](#)



# Chapter 31: Exchanging Data with Other Applications

---

You can transfer data between CitectSCADA and other software for storage, analysis, and post processing, or to control and tune your CitectSCADA system.

CitectSCADA uses the following methods to exchange data:

- [dynamic data exchange \(DDE\)](#), where CitectSCADA can act as a:
  - **DDE server** providing tag values to requesting clients
  - **DDE client** to request data from other applications.
- [open database connectivity \(ODBC\)](#), where CitectSCADA functions as an ODBC server, allowing other applications to read CitectSCADA variables directly.
- By using a common **external database**, where CitectSCADA and other applications use the same database to store and share information.

CitectSCADA also supports importing and linking variable tag data from external databases. See [Linking, Importing, and Exporting Tags](#).

## Using DDE (Dynamic Data Exchange)

Microsoft Windows DDE allows the continuous and automatic exchange of data between different Windows applications on the same machine without the need for any user intervention. For example, your company's Production group might use a spreadsheet application to graphically represent plant-floor data (product output). This could be dynamically updated with the latest live data using DDE to read values directly from CitectSCADA.

Windows DDE uses the DDE [protocol](#) to send messages between applications that share data.

Dynamic Data Exchange occurs between a DDE client application (which requests the data or service) and a DDE server application (which provides the data or service). The DDE Client starts the exchange by establishing a conversation with the DDE server, and requesting data or services. The DDE server responds to these requests by providing the data or services to the DDE Client. The DDE Client terminates the conversation when it no longer needs the DDE server's data or services.

**Note:** As the DDE protocol is not designed for high-speed [data transfer](#), the use of DDE is only appropriate when data communication speed is not critical.

- For information about DDE conversations, see [DDE conversations and client syntax](#).
- To establish a DDE conversation between applications on the same computer, see [Setting up DDE conversations](#).
- To establish DDE conversations between applications running on different computers over the same network, see [Network DDE](#).
- To establish DDE Conversations with the CitectSCADA tag database directly, see [Connecting to the CitectSCADA tag database using DDE](#).

**Note:** When reading or writing to CitectSCADA tags using DDE, you might unknowingly add to your CitectSCADA License point count. Once this tally reaches a certain limit, CitectSCADA will no longer function correctly. Therefore when accessing tags via DDE, it's important to remain aware of how many points you have used. For details, see Citect license point count in the Installation and Configuration Guide.

## DDE conversations and client syntax

Two applications participating in Dynamic Data Exchange are said to be engaged in a **DDE conversation**. The application that initiates the conversation is the **DDE Client**, and the application that responds to the DDE Client is the DDE server.

An application can have several DDE conversations running at the same time. The application can be the DDE Client in some conversations (requesting data or services), and the DDE server (the data/service provider) in others. Each request or response in a DDE conversation specifies the data or service to be sent or received.

**Note:** A DDE **conversation** is sometimes referred to as a **channel** or a **link**.

The syntax sent by the DDE Client when it tries to establish a DDE conversation with the DDE server, consists of three parts:

- The name of the application to retrieve the data from.
- The file or topic name which contains the data to be retrieved.
- The cell range, value, field, or data item that's being requested.

These are combined in the format:

```
<DDE server application name>|<DDE Topic name>!<DDE Data item name>
```

where:

- **<DDE server application name>** identifies the DDE server application.
- **|** (pipe character) separates the DDE server application name from the DDE Topic Name with no spaces between them.

- **<DDE Topic name>** identifies the context of the data. For DDE Servers that operate on file-based documents, DDE topic names are typically file names. For other DDE Servers, they are other DDE application-specific strings.
- **!** (exclamation character) separates the DDE Topic Name from the DDE Data item name with no spaces.
- **<DDE Data item name>** is a string that identifies the data item that a DDE server can pass to a DDE Client during a DDE transaction. In some instances, the DDE Data item name is optional. Refer to the DDE application documentation for particulars.

**Note:** In the DDE Client syntax structure example above, every placeholder shown inside arrow brackets ( <placeholder> ) should be replaced with the actual name of the item that it describes. The arrow brackets and the placeholder words they contain should not be included in the statement, and are shown here only for your information.

As the DDE protocol was designed in an era before long file names, DDE only supports the use of short (8 character) file names. To overcome this limitation, enclose the three parts of the DDE syntax within single quotes respectively. For example:

```
Citect|Variable!'Process Variable 1'
```

This instructs DDE to treat the characters within the quotes as strings, thus permitting them to contain long file names, the space character ( ), the pipe character (|), the exclamation or bang character (!), or any other non alphanumeric character.

See Also [Setting up DDE conversations](#)

## Setting up DDE conversations

The DDE protocol itself does not support the launch of applications, so both the DDE Client application and the DDE server application must already be running before any DDE conversations can occur (unless the calling application is coded to detect and launch the DDE server application when required).

At the beginning of a DDE conversation, a DDE Client requests the services of a DDE server using DDE Client syntax (which contains the DDE server application name, topic or file name, and the data item name in the request). For DDE Client syntax details, see [DDE conversations and client syntax](#).

To set up an application as a DDE Client, that is, to request data from a DDE server application, you need to use appropriate values in the DDE Client syntax as follows:

### DDE server application name

The DDE server name is usually the DDE server application name, e.g. the DDE server name for CitectSCADA is "**Citect**", the DDE server name for Microsoft Excel is "**Excel**", the DDE server name for Microsoft Word is "**WinWord**", and the

DDE server name for Microsoft Access is "**MSAccess**". Most DDE Servers respond to only one name.

### DDE Topic name

The DDE Topic name for CitectSCADA is either "**Data**" (if you use the Cicode DDE functions) or "**Variable**" (if you use CitectSCADA as the DDE server and want to access the variable tag database directly). The DDE Topic name for Microsoft Excel is the name of the worksheet (which may also include the workbook name enclosed in square brackets). The DDE Topic name for Microsoft Word is the document name. The DDE Topic name for Microsoft Access is the Database name and Table name, Query name or an SQL string as detailed in the following note:

**Note:** The proper DDE Client syntax of the DDE Topic name section for accessing a Microsoft Access database is constructed like this:

```
"<DataBaseName>; TABLE <TableName>".
```

The **<DataBaseName>** placeholder is for the name of the Access database file followed by a semicolon ( ; ). You might have to include the file path; however this might not be the case (i.e. if it is known that Access will be running with the target file open). The .MDB suffix is optional (as .MDB is the default suffix for Access databases), unless the full path was included.

The **TABLE <TableName>** is the command string to instruct Access which table data you intend to converse with. DDE also supports the use of **QUERY <QueryName>** or **SQL <SQLString>** in place of **TABLE <TableName>**.

The use of the semi-colon ( ; ) after the '<DataBaseName>' placeholder, and the use of UPPERCASE for the 'TABLE', 'QUERY', and 'SQL' commands in the DDE string syntax are required. The whole section must be enclosed in quotes ( " ).

### DDE Data item name

The DDE Data item name for CitectSCADA depends upon the DDE Topic name being used. When using 'Variable' as the DDE Topic name to access the variable tag database directly, the DDE Data item name is the CitectSCADA variable tag name. When using 'Data' as the DDE Topic name to access a value posted using the Cicode DDEPost() function, the DDE Data item name is the posted name.

The DDE Data item name for Microsoft Excel is the cell range in Row number Column number format (e.g. R1C1). The DDE Data item name for Microsoft Word is a bookmark name. The DDE Data item name for Microsoft Access is dependant upon which topic name was used. Refer to the Microsoft Access Help for details.

**Note:** These CitectSCADA DDE help topics and examples were originally written for Windows 3.xx and subsequently updated for Office 95 on Windows 95. Microsoft has since introduced security measures with Office 2000 and later

versions which, by default, block Office applications from being DDE Clients. For security details, see [Using DDE with Microsoft Office applications](#).

CitectSCADA can perform as both a DDE server and as a DDE Client as required. To set up CitectSCADA to use DDE, see [Exchanging CitectSCADA data via DDE](#).

CitectSCADA comes with an Excel workbook file which contains macros to help you insert correct DDE Client syntax links from your CitectSCADA runtime project tag database directly into an Excel worksheet.

## CitectSCADA DDE function types

There are two classes of DDE functions in Cicode, the original **DDE** functions and the later **DDEh** functions.

### DDE functions

The original Cicode DDE functions do not return a DDE Channel Number and were designed to insulate the user from the need to manage DDE Channels. The DDERead(), DDEPost(), DDEWrite(), and DDEExec() functions each perform a single exchange of data. Each of these functions starts a DDE conversation with the external application, sends or receives the data (or command), and ends the conversation - all in one operation.

### DDEh functions

The Cicode DDEh functions were introduced to afford more control over DDE communications, especially for Network DDE and for circumstances where it is necessary to explicitly terminate and re-initiate a DDE Channel (after deleting rows from a table for example).

The DDE handle (*DDEh...*) functions return a handle to the conversation - a DDE channel number.

Use the DDEh handle functions for Network DDE, and for Access DDE.

See Also [Exchanging CitectSCADA data via DDE](#)

## Exchanging CitectSCADA data via DDE

CitectSCADA runtime can exchange data as a DDE server or a DDE Client.

CitectSCADA behaves as a DDE server when providing other applications with access to its data. When acting as a DDE server, CitectSCADA runtime can:

- Provide DDE access to the complete variable tag database **automatically** with no further setup required
- Provide access to selected variable values by posting select CitectSCADA data using DDE

CitectSCADA behaves as a DDE client when requesting other applications to provide access to their data. When acting as a DDE Client, CitectSCADA runtime can:

- Read data directly from another application
- Write data directly to another application

**Note:** You can also execute commands in another application from CitectSCADA with the DDEExec() function. Similarly, you can run Cicode functions from another application by passing the functions through that application's DDE Execute command.

See Also [Connecting to the CitectSCADA tag database using DDE](#)

### Connecting to the CitectSCADA tag database using DDE

CitectSCADA runtime behaves as a DDE server and automatically provides DDE access to the complete variable tag database with no further setup required.

To create DDE links to the CitectSCADA variable tags, use the DDE Client syntax. For syntax details, see [DDE conversations and client syntax](#).

In the DDE Client call, the DDE Application name must be "Citect", the DDE Topic name must be "Variable", and the DDE Data item name must be the CitectSCADA tag name.

For instance, the PV1 tag value can be accessed from a cell in Excel containing the following formula:

```
=Citect|Variable!PV1
```

If the CitectSCADA variable tag name contains spaces or non alphanumeric characters, the DDE data item section of the DDE Client call syntax must be enclosed within single quotes. For example:

```
=Citect|Variable!'Process Variable 1'
```

CitectSCADA runtime and the DDE Client application (e.g. Excel) must both be running on the same computer. For information about DDE conversations, see [DDE conversations and client syntax](#).

To establish a DDE conversation between applications on the same computer, see [Setting up DDE conversations](#).

To establish DDE conversations between applications running on different computers over the same network, see [Network DDE](#).

### Posting select CitectSCADA data using DDE

You might have a tag naming convention which is not DDE compatible, or inappropriate for use in a DDE call. CitectSCADA provides the ability to publish specific tags under different names in DDE. This involves using the DDEpost function.

To make selected CitectSCADA variable values or the results of calculations available to external DDE Client applications currently running on the same computer, use the Cicode DDEPost() function to have CitectSCADA runtime behave as a DDE server.



This conversation is one-way, which allows an external DDE Client application (like Excel, Word, etc.) to read the value from CitectSCADA using DDE. The Client application cannot change this value in CitectSCADA.

You can use this function to present data under a different name than its source. For instance, the following example presents the value of variable tag PV1 as "Process1".

The following Cicode example posts the value of variable PV1 using DDE:

```
DDEPost("Process1", PV1)
```

Once posted, this value can be accessed, for example, from a cell in Excel containing the following formula:

```
=Citect|Data!Process1
```

or from a field in Microsoft Word containing the following function:

```
{DDEAuto Citect Data Process1 }
```

#### Notes

- The name of the posted value (e.g. [Process1](#)) must not exceed 8 characters or contain spaces if you want to link to it via a Microsoft Word DDE field. Microsoft Excel, however, accepts long data item names if enclosed within single quotes (e.g. ['Process Variable 1'](#)).
- You must use Data as the DDE Topic name when accessing posted values. See [Setting up DDE conversations](#).

This method of DDE connection requires that both CitectSCADA runtime and the DDE Client application (e.g. Excel or Word) are running on the same computer at the same time. Once the DDE connection has been made, the DDE Client would normally display the updated value of the CitectSCADA DDE posting as soon as it becomes available, usually within milliseconds of the post.

Unfortunately, this posting method of DDE linking is subject to breakage whenever CitectSCADA runtime is closed, even if CitectSCADA runtime is subsequently restarted. The DDE Client application might not detect the break, as updates to the data in the DDE Client (e.g. Excel) only occur after each post by the DDE server (CitectSCADA runtime). If no further posts occur, and in this scenario none will (as the DDE link is broken), the DDE Client application receives no update, and subsequently might display data which could be out of date. This broken state will remain until the DDE link in the DDE Client application is refreshed or the DDE Client application is restarted.

See Also [Writing values to a DDE application](#)

#### Writing values to a DDE application

To write a CitectSCADA variable value directly to an external DDE server application currently running on the same computer as CitectSCADA, use the Cicode DDEWrite() function.

For example, writing data from CitectSCADA to a Microsoft Office Application (using CitectSCADA as the DDE Client and the Office application as the DDE server), could be done using the following Cicode DDEWrite() function examples:

```
! Write PV1 to Excel
! Assumes the existence of Sheet1
DDEWrite("Excel", "Sheet1", "R1C1", PV1);

! Write PV1 to Word
! Assumes the existence of TestDDE.doc already loaded in Word
! containing the bookmark named TagPV1
DDEWrite("Word", "TestDDE", "TagPV1", PV1);
! Note that Access does not support direct DDE writes.
```

This DDE function is one-way, so to update the tag value in the remote DDE server application, this function will need to be called every time the value of this CitectSCADA variable changes.

Writing directly to a DDE server assumes a prior knowledge of the DDE server. In the above example, the spreadsheet or document must exist and Excel or Word (as appropriate) must be running for the DDE communication to be successful.

#### Notes

- Instead of using the once only DDEWrite(), you could use the multiple use DDE handle function DDEhPoke().
- Ensure that remote requests are enabled in the other application. For example, in Excel, you must ensure the **Ignore other applications** check box is cleared (the default setting). In Excel 2003, use **Tools | Options | General** to adjust this setting.
- The High security setting (if selected) on Microsoft Office 2000 and later versions does not appear to affect the use of remote DDE Client requests with those Office applications. See [Using DDE with Microsoft Office applications](#).

See Also [Reading values from a DDE application](#)

#### Reading values from a DDE application

To read a value into a CitectSCADA variable directly from an external DDE server application currently running on the same computer as CitectSCADA, use the Cicode DDERead() function. For example:

```
PV1 = DDERead("Excel", "[Book1]Sheet1", R1C1);
```

The data from Row 1, Column 1 on Sheet 1 of Book 1 in Excel is read and stored in the CitectSCADA variable "PV1". This DDE function is one-way, and will need to be called whenever the value needs to be updated in CitectSCADA.

## Using DDE with Microsoft Office applications

Reading from a DDE server assumes a prior knowledge of the DDE server. In the above example, Excel must be running and the appropriately named spreadsheet must exist. The CitectSCADA variable PV1 must also have been previously declared.

**Note:** Instead of using the once only `DDERead()`, you could use the multiple use DDE handle function `DDEhRequest()`.

Ensure that remote requests are enabled in the other application. For example, in Excel, you must ensure the **Ignore other applications** check box is cleared (the default setting). In Excel 2003, use **Tools | Options | General** to adjust this setting.

The High security setting (if selected) on Microsoft Office 2000 and later versions does not appear to affect the use of remote DDE Client requests with those Office applications. See [Using DDE with Microsoft Office applications](#).

Microsoft has introduced security measures with Office 2000 and later versions which, by default, block Office applications from being DDE Clients. See [Microsoft Office security](#).

Microsoft Office applications appear to support varying degrees of long file names with DDE. See [Long file names in DDE](#).

To enable DDE remote requests in Microsoft Excel, you must ensure the **Ignore other applications** check box is cleared (the default setting). In Excel 2003, use **Tools | Options | General** to adjust this setting.

### Long file names in DDE

According to MSDN Knowledge Base article Q109397, DDE does not support long file names, so DOS alias names must be used for directory and file names longer than eight characters (i.e. `C:\mydocu~1\file.mdb`).

Different Microsoft Office applications differ in their support for the use of long file names when used as DDE Clients. For instance, Microsoft Word does not appear to support the use of DDE data item names which exceed 8 characters, whilst Microsoft Excel however, accepts long data item names if enclosed within single quotes. See [Posting select CitectSCADA data using DDE](#) for an example.

### Microsoft Office security

In the interests of data security, Microsoft Office 2000 and later versions have their security settings set to high by default. To view or change your security level in Excel or Word, choose **Tools | Macro**, click **Security**, and click the **Security Level** tab.

- If you have your security level set to **High** (the default setting), then communication with external DDE Servers will not be available unless they are digitally signed and trusted. All you see in Excel cells that use the DDE function is #N/A, and with no additional explanation as to why the DDE

functions aren't working. The High security setting (if selected) does not appear to affect the use of remote DDE Client requests with those Office applications as DDE Servers.

- If you set your security level to **Medium**, you are asked if you want to run any DDE Servers that are not digitally signed and trusted and that are referenced by DDE functions.
- If you set your security level to **Low**, all external DDE Servers are run regardless of whether they are digitally signed and trusted, or not.

**Note:** If you need to manipulate another application's objects from Microsoft Office, consider using OLE Automation.

## Network DDE

Network DDE is a version of the DDE protocol for use across a network. For information about DDE, see [Using DDE \(Dynamic Data Exchange\)](#). For details about setting up CitectSCADA to use DDE, see [Exchanging CitectSCADA data via DDE](#).

Just like the establishment of a DDE conversation between applications running on the same computer, a network DDE conversation uses the same DDE protocol to share data between applications running on separate computers connected to a common network.

Network DDE is a Windows Service used to initiate and maintain the network connections, security, and shared file space needed for using DDE over a network, and must be running on both computers at the same time. For startup details, see [Starting network DDE services](#).

A network DDE trusted share must be manually created for the Network DDE server application on the Network DDE server application machine. This instructs the Network DDE Service (on the DDE server application machine), as to which application and topic to connect with. It is this share name which the Network DDE Client application can subsequently communicate with. For details, see [Setting up network DDE shares](#).

The Network DDE Client starts the Network DDE conversation by connecting to the Network DDE Share on the Network DDE server computer. For details, see [Using network DDE](#).

### Starting network DDE services

For Network DDE to function, NetDDE.EXE must be installed and running on both machines before attempting to conduct a Network DDE conversation. NetDDE.exe is a Windows Service system file that is used to communicate the shared dynamic data exchange used by Network DDE. It has no graphical user interface (it runs as a background Windows service).

It is necessary to initiate the automatic activation of Network DDE Services, or manually run NetDDE.EXE on both machines before attempting connection.

**To manually start Network DDE services:**

- On the Windows Start menu, click **Start | Run**, type in "netdde" (without the quotes) and press the **Enter** key. Do so on both machines.

**To automatically start the Network DDE Services on machine startup:**

- With Windows 2000 and later, use the Windows Services Manager (select **Start | Control Panel | Administrative Tools | Services**) to set the **Network DDE** service from **Manual** to **Automatic**. To do so, right-click the service and select **Properties** from the pop-up menu. On the **General** tab select **Automatic** from the drop-down list of the **Startup type** field. Click **OK**. Close all windows and restart the machine.

**To verify that the NetDDE Services are running:**

- The Windows Task Manager lists NetDDE.exe on the **Processes** tab when running. To view the Windows Task Manager, press **CTRL+ALT+DEL**.
- The Service Administrative Tools also lists the status of Network DDE and Network DDE DSDM. To view the Windows Services Manager, select **Start | Settings | Control Panel | Administrative Tools | Services**.

**Note:** If you are using only the Microsoft Client Service for NetWare Networks, the NW IPX/SPX/NetBIOS compatible protocol must be enabled for NetDDE.exe to load.

**To test that Network DDE is operational between two machines on the same network**

Microsoft Windows ships with a network DDE application called Chat. It is installed in the `system32` folder.

- 1 On the Windows Start menu, click **Start | Run**, type in "winchat" (without the quotes) and press the **Enter** key. Do so on both machines.
- 2 On one machine, select the Chat menu **Conversation | Dial** or click the dial button. The **Select Computer** dialog will display.
- 3 Select the other computer from the list, and click **OK**. Chat will attempt to establish a network DDE conversation between the computers.

**Note:** If Chat is not already running on the other computer, it times-out and states that the other computer didn't answer. If however, the other computer is already running Chat, it will keep dialling until answered.

- 4 On the other machine, (whilst it is being dialled), select the Chat menu **Conversation | Answer** or click the answer button. Type in a message and it will display in the Chat window on the other machine. The conversation will continue until either machine hangs up.

## Setting up network DDE shares

Once a Chat conversation is established, it proves that both machines are properly set-up and capable of handling network DDE conversations. You can view the share properties for Chat\$ by using the DDEShares.exe application as described in [Setting up network DDE shares](#).

You don't have to run Chat to use Network DDE with CitectSCADA. This Network DDE test only uses Chat as an example to confirm Network DDE functionality between two machines. Once you have established that Network DDE is functional, close the Chat windows, create the **Network DDE Trusted Share** for your Network DDE server application, and connect to the share using Network DDE. See [Connecting to a network DDE shared application](#).

To be able to create a DDE link over a network, the computer serving as the Network DDE server must be setup to provide a **Network DDE Share** to establish a network DDE Channel.

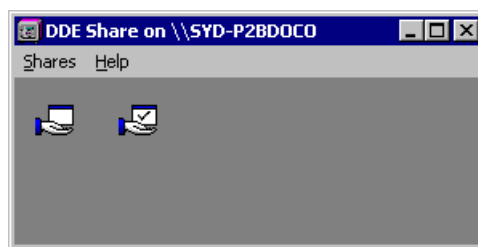
**Note:** You only have to create a DDE Share if you intend to use DDE between two separate applications running on different machines, and you only have to create the DDE Share on the machine that contains the application which will be the DDE server.

The Windows DDESHARE.EXE utility enables users to manage DDE shares. The 32-bit version is shipped with all Microsoft operating systems and is located in the Windows\System32 sub-directory.

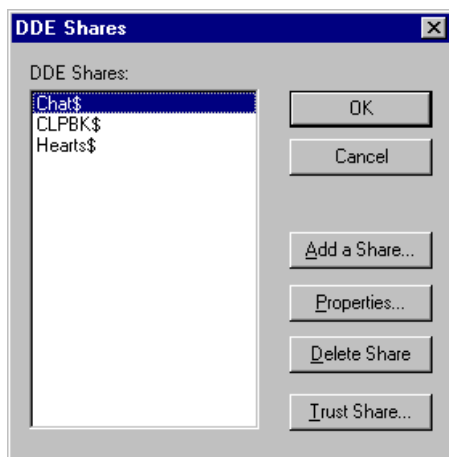
### To manually launch the DDE Share utility:

- On the Windows Start menu, click **Start | Run**, type in "ddeshare" (without the quotes) and press the **Enter** key.

When DDEShare.EXE is running, it displays the DDE Share utility window containing two icons which launch the DDE Shares dialog, and the DDE Trusted Shares dialog:



In the DDE Share utility, double-click the left icon (without the check mark) to launch the DDE Shares dialog:



The DDE Shares dialog is used to create, manage, and delete global DDE shares on your computer, and to view the DDE shares of any computer on the network.

**Note:** You can use this dialog to confirm the names of shares available on any machine on the same network. From the DDE Shares menu, select **Shares** | **Select Computer** and choose the computer name you're interested in from the list.

## DDE Shares

There are three types of DDE shares: old style, new style, and static. CitectSCADA only supports the static type. The names of static shares follow the convention

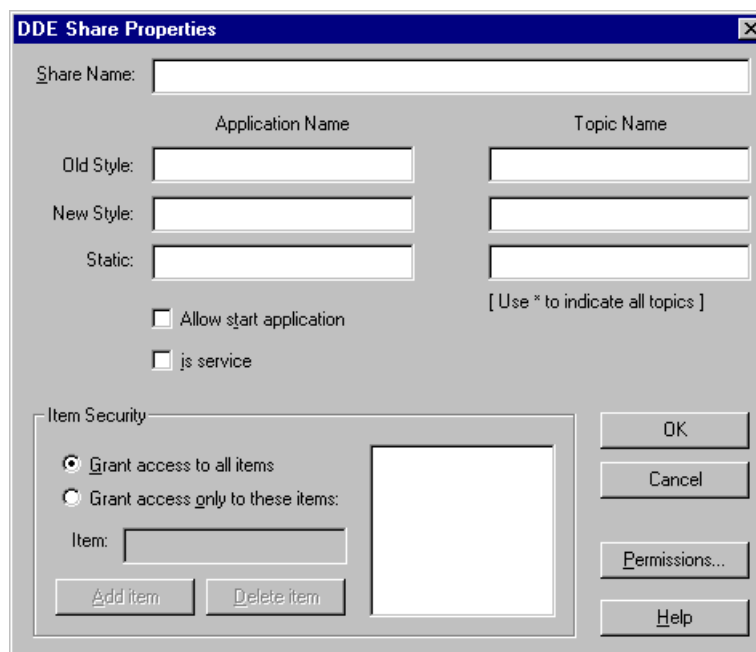
<ShareName>\$

so to set up a CitectSCADA server computer as a Network DDE share, use the name "**Citect\$**" as the sharename on that computer. To expose the CitectSCADA runtime variable tag database for suitable DDE linking, use the word "Variable" as the DDE Share Topic name.

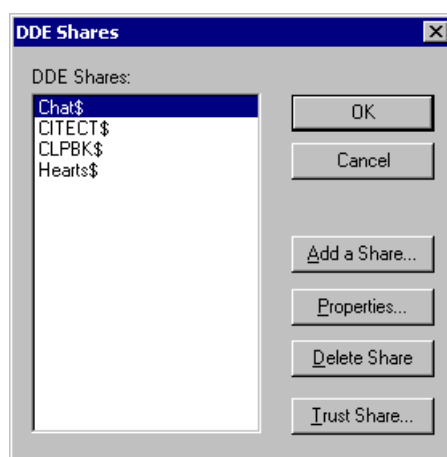
Note that the trailing dollar sign (\$) is required as part of the DDE share name syntax.

### To create a DDE share:

- 1 In the DDE Shares dialog, click **Add a Share**. The **DDE Shares Properties** dialog appears. Complete the fields exactly as shown here:



- 2 Click **Permissions**. The DDE Share Name Permissions dialog appears.
- 3 **Read and Link** is the default permission setting. If you want to write data to the DDE Share application, change the permission to **Full Control**.
- 4 Click **OK**.
- 5 Click **OK** to save the Share, and return to the **DDE Shares** dialog.



See Also [Using DDE Trusted Shares](#)



## Using DDE Trusted Shares

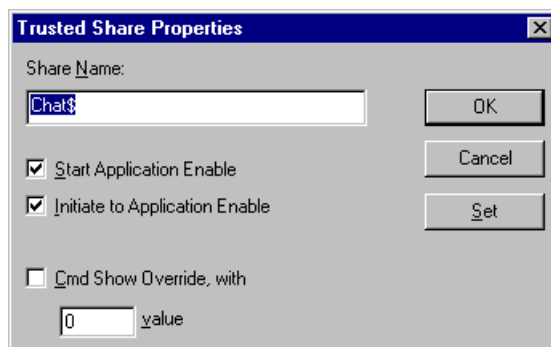
When a network DDE Client user connects to a network DDE Share from a remote computer, Network DDE accepts the request only if both:

- The user who created the share has granted trusted status to the share.
- The user who created the share is currently logged on to the server computer.

To link to the CitectSCADA tag database, and permit write actions from an external application using Network DDE, the DDE Client computer must be granted appropriate Trusted status.

**To create a trusted share:**

- 1 On the DDE Shares dialog, highlight the new 'Citect\$' share entry, and click **Trust Share**. The Trusted Share Properties dialog box appears.



- 2 Check **Initiate to Application Enable** to allow new connections to the DDE share.
- 3 Click **OK**.

**To view the trusted shares:**

- 1 In the DDESHARE utility double-click the right icon (with the check mark) to display the DDE Trusted Shares dialog.
- 2 The DDE Trusted Shares dialog lists the DDE shares that are trusted in the current user's context. You can view and modify trusted share properties and remove DDE shares from the list of trusted shares.
- 3 Once setup is completed, close the DDE Share utility dialog box.

## Using network DDE

Microsoft Network DDE Service must be running on both computers to communicate using Network DDE. For startup details, see [Starting network DDE services](#).

## Connecting to a network DDE shared application

Before a Network DDE Client can establish a DDE conversation with a Network DDE server application, the Network DDE server application computer must already have setup a **Network DDE Share**. For details, see [Setting up network DDE shares](#).

**Note:** You cannot connect using Network DDE to a shared application on the same machine. You can only connect using Network DDE to a shared application on another machine (which must also be on the same network).

To connect to a Network DDE shared application, you use an altered version of the DDE syntax, which replaces the "<ApplicationName>" with "<ComputerName>\NDDE\$" and replaces the "<TopicName>" with the Network DDE server Share "<ShareName>", and continues to use the "<DataItemName>" as normal.

At first glance, there appears to be no way to specify the DDE Application or Topic names in the Network DDE syntax call, and indeed, that is the case. However, the DDE Application and Topic names are defined in the DDE server Share settings. So, when the Network DDE server machine receives the call (from the Network DDE Client) containing the Share name, it knows which application and topic to connect with. See [Connecting to a network DDE shared application](#).

The network DDE Client specifies the remote DDE server share in the normal DDE Client syntax by replacing the DDE Application name and DDE Topic name with the DDE server computer name and DDE server share name in the call. For DDE client syntax details, see [DDE conversations and client syntax](#).

With Network DDE Client syntax, the DDE Application name is replaced with the following string enclosed in single quotes:

```
'\\<ComputerName>\NDDE$'
```

where "<ComputerName>" is the name of the computer running the DDE server application, and "NDDE\$" notifies Windows on the remote computer that the calling DDE Client wishes to establish a Network DDE channel. You cannot omit the NDDE\$ string, or it won't work.

The DDE Topic name is replaced with the following string also enclosed in single quotes:

```
'<ShareName>'
```

where "<ShareName>" is the name of the DDE Trusted Share previously set-up on the DDE server computer. The DDE Share on the DDE server machine contains the details of which application and topic to create the Network DDE link with. Most often, DDE server share names end with a \$ character.

**Note:** You must use a separate DDE share name on the remote computer for each combination of DDE application name and DDE topic name you want to share. You can not declare the topic as a wild card (\*).

For example, to create a Network DDE link with the following criteria:

- CitectSCADA variable tag name: "PV1"
- CitectSCADA server computer name: "PlantSvr"
- Remote DDE Share name: "Citect\$"

you would construct a Network DDE Client call containing:

```
'\\PlantSvr\NDDE$' | 'Citect$' !PV1
```

In Excel, the following formula could be placed directly into a worksheet cell:

```
= '\\PlantSvr\NDDE$' | 'Citect$' !PV1
```

If prompted for a username and password, use one that has appropriate permissions on the DDE server computer.

**Note:** You cannot omit the DDE syntax pipe character (!) or exclamation character (!), nor can you enclose those characters within quotes (").

CitectSCADA comes with an Excel workbook file which contains macros to help you insert correct DDE Client syntax links from your CitectSCADA runtime project tag database directly into an Excel worksheet.

## Using the Citect Tags Excel macros

CitectSCADA provides the Citect Tags Excel macros, which permit you to display the value of CitectSCADA variables directly in an Excel worksheet cell (so that you do not need to use Cicode DDE functions). The macros are contained in a workbook named `ddeformu.xls` (in the `\CitectSCADA 7\bin` directory), which was updated to support Network DDE with CitectSCADA version 5.41 and later.

**Note:** Microsoft Excel version 8 which shipped with Microsoft Office 97 provides macro virus protection to prevent potentially malicious macros from running. To enable macros, and be able to use the features provided with `ddeformu.xls`, in Excel 8, from the main menu, choose **Tools | Options** and on the General tab clear the **Macro Virus Protection** option.

Microsoft Excel version 9, which shipped with Microsoft Office 2000, provides security levels which silently disables macros by default. To enable macros, and be able to use the features provided with `ddeformu.xls`, in Excel 9 or later, from the main menu, select **Tools | Macro | Security** and select **Medium** or **Low**.

If your Excel security settings are enabled, when you attempt to open the `ddeformu.xls`, Excel warns you that the file contains macros. To enable the Citect Tags features, select **Enable Macros**.

When started, the Citect Tags macros expect that CitectSCADA runtime is operating on the same machine. If not, Excel displays a dialog requesting permission to start `citect.exe` (which will fail as `citect.exe` does not exist

unless you're running Citect version 3 or earlier). If you further select **Yes**, it will search the system 'path' for the non-existent program and subsequently fail. If you select **No**, and if previous values were saved with the worksheet, those are the values that will display initially, and be replaced with '#REF!' when updated. In any case, no valid values will be displayed in the example worksheets until CitectSCADA runtime is started and Excel is subsequently refreshed or restarted.

The Citect Tags macros also expect to find `citect.ini` at either the `C:\WINDOWS\` or `C:\WIN95\` folder locations on the local machine. If not, it will display the 'ERROR Reading Citect.INI' dialog requesting the proper location. Enter the full path including the file name, and clear **Restore Defaults on Start Up** to prevent the same thing happening next time the macro is started. If you are using an alternative INI file, enter it instead.

Once running, the right-click menu in Excel contains four additional menu items, permitting you to perform two new workbook related commands, and two new CitectSCADA-related commands to the cell beneath the mouse pointer location when you perform the right-click event. The new menu items provided with `ddeformu.xls` are:

- **Citect Settings** - Workbook command
- **Citect Get Tags** - Workbook command
- **Citect Select Tags** - CitectSCADA command
- **Citect Select Trends** - CitectSCADA command

**Note:** This feature is only compatible with Excel version 5.00 (or later).

## Using External Databases

You can store and update runtime data from your plant floor in a database external to CitectSCADA. You can also use CitectSCADA to send information from the database (such as a recipe) to I/O devices in your plant.

By using an external application to read and write the same database records, you can effectively interface to an external application through, for example, a relational database.

CitectSCADA supports two types of databases:

- [dBASE databases](#)
- [SQL databases](#)

### dBASE databases

The dBASE file format has become an industry standard for data storage, and CitectSCADA supports the standard dBASE format. You can use any database editor (that supports dBASE III files) to create a database, and to read and write database records.

To connect to a database, you must define a CitectSCADA Device. Devices specify the format and location of the database, for example:

Name	Recipe
Format	{Name,16}{Water,8}{Sugar,8}{Flour,8} {Salt,8}{Yeast,8}{Milk,8}
Header	Name
File Name	[DATA]:RECIPE.DBF
Type	dBASE_DEV
Comment	Recipe Device (dBASE file)

## SQL databases

SQL (Structured Query Language) also has become an industry standard. SQL is a command language that allows you to define, manipulate, and control data in SQL databases - and in other relational databases. Most database servers support SQL. SQL gives you direct access to database servers on other platforms, such as computers, mini-computers, and mainframe computers.

You can use the CitectSCADA Device functions to set up the format and locations of each of your SQL databases. Alternatively, use the SQL functions for direct control over SQL transactions.

You must define a CitectSCADA Device to specify the format and location of the database, for example:

Name	Recipe
------	--------

In the Format field, specify the field names in the SQL database, for example:

Format	{Name,16}{Water,8}{Sugar,8}{Flour,8} {Salt,8}{Yeast,8}{Milk,8}
--------	--

The Header is the database connection string for ODBC connection, for example:

Header	DSN = ORACLEDATABASE
--------	----------------------

Enter the database table name in the File Name field:

File Name	RECIPE
Type	SQL_DEV
Comment	Recipe Device (SQL)

See Also [Using Structured Query Language](#)

## Using Structured Query Language

You can use Structured Query Language (SQL) functions for direct access to an SQL database, instead of accessing the database as a Device. Using direct database access can provide greater flexibility. The SQL functions provide access to SQL databases through any ODBC-compatible database driver, e.g. MS Access, FoxPro, Paradox, etc.

See Also [Connecting to an SQL database](#)

[Executing SQL commands](#)  
[Using a transaction](#)  
[Expressing dates and times in SQL](#)

## Connecting to an SQL database

Before you can use SQL commands, you must connect to the SQL database system. The SQLConnect function provides this access. You must call this function before any other SQL functions. It has the format:

```
SQLConnect(sConnect);
```

where sConnect is the connection string, for example:

```
INT hSQL;
hSQL = SQLConnect("DSN=DBASE_FILES;DB=C:\ODBC\EMP;LCK=NONE;CS=ANSI");
! Connect to a dBASE Compatible Database File.
INT hSQL;
hSQL = SQLConnect("DSN=EXCEL_FILE;DB=C:\ODBC\EMP;FS=10");
! Connect to an Excel File.

INT hSQL;
hSQL = SQLConnect("DSN=ORACLE_TABLES;SRVR=X:ACCTS;UID=SCOTT;PWD=TIGER");
! Connect to an Oracle Database.
```

Refer to the documentation that accompanied your SQL server for details about connecting to an SQL database.

See Also [Executing SQL commands](#)

## Executing SQL commands

SQL allows you to manipulate data in a non-procedural manner; you specify an operation in terms of what is to be done, not how to do it. SQL commands allow you to:

- Create tables in the database.
- Store information in tables.
- Select exactly the information you need from your database.
- Make changes to your data and to the structure of a table.
- Combine and calculate data.

The SQLExec() function executes any SQL command that your SQL server supports. For example, to create a database table, you would execute the SQL "CREATE TABLE " command:

```
SQLExec(hSQL, "CREATE TABLE recipe ('Name' CHAR(16), 'Water' CHAR(8), 'Sugar' CHAR(8), 'Flour' CHAR(8), 'Salt' CHAR(8), 'Yeast' CHAR(8), 'Milk' CHAR(8))");
```

To add records into the database table, use the "INSERT INTO" command. The command has the following syntax:

```
INSERT INTO <filename> [(<col_name>, . . .)] VALUES (<expr>, . . .)
```

This command adds the values for each field in the table, for example:

```
SQLExec(hSQL, "INSERT INTO recipe VALUES ('Bread', '10', '5', '7', '1', '1', '2');");
```

Column names are optional; however, if you omit column (field) names, the values are inserted into the fields in the same order as the values.

To read data from an SQL database, use the SQL "SELECT" command. You can use the "SELECT" command to read an entire set of records, or a row, from the table. You can then use the SQLGetField() function to read the data in each field, for example:

```
SQLExec(hSQL, "SELECT * FROM recipe WHERE NAME = 'Bread'");
If SQLNext(hSQL) = 0 Then
PLC_Water = SQLGetField(hSQL, "WATER");
PLC_Sugar = SQLGetField(hSQL, "SUGAR");
PLC_Flour = SQLGetField(hSQL, "FLOUR");
PLC_Salt = SQLGetField(hSQL, "SALT");
PLC_Yeast = SQLGetField(hSQL, "YEAST");
PLC_Milk = SQLGetField(hSQL, "MILK");
END
```

To delete database records, use the SQL "DELETE" command. The command has the following syntax:

```
DELETE FROM <filename> [WHERE <conditions>]
```

This command deletes values from the table, for example:

```
SQLExec(hSQL, "DELETE FROM recipe WHERE NAME = 'Bread'");
```

See Also [Using a transaction](#)

## Using a transaction

You can use a database transaction for more sophisticated database operations. A database transaction allows you to execute a series of SQL commands and then either commit the changes to the database, or 'roll back' (cancel) the changes, for example:

```
SQLBeginTran(hSQL); ! Begin the transaction
SQLExec(hSQL, "UPDATE recipe SET water = '12' WHERE NAME = 'Bread'");
SQLExec(hSQL, "UPDATE recipe SET milk = '1' WHERE NAME = 'Bread'");
IF . . . THEN
SQLCommit(hSQL); ! Commit the transaction
ELSE
SQLRollBack(hSQL);! Cancel the transaction
END
```

Check the ODBC-compatibility level of your database driver if you cannot use transactions.

See Also [Expressing dates and times in SQL](#)

## Expressing dates and times in SQL

The way in which SQL dates are expressed depends upon the particular database system. With dBase, you normally specify a date in braces, for example {02/18/95}. For Oracle, use the format: to\_date('02/18/95', 'MM/DD/YY'). Other ODBC drivers might require another format - a common ODBC format is: 'YYYY-MM-DD'.

**Note:** Date references in an external database should be based on the Gregorian Calendar, or the database tables must be exported to text files before use in CitectSCADA. Dates in Microsoft Access database tables exported as text files are stored as Gregorian values.

### Database independent date-time syntax

For database independence, you can use the following syntax for dates and times:

```
[ <format> 'YYYY-MM-DD HH:MM:SS.FFFFFFFF' ]
```

where:

- **<format>** (the first character after the opening square bracket) must be one of the following:
- d - date
- t - time
- dt - date and time.

Whether you are specifying a date, time, or date and time, you must provide the full 26 character string, for example:

```
[ d '1995-02-18 00:00:00.000000' ]
```

Refer to the documentation that accompanied your SQL server for further information about SQL commands.

## Using ODBC drivers

CitectSCADA supports the [open database connectivity \(ODBC\)](#) standard. Many manufacturers of database packages now also supply an ODBC database driver for their software. As well as these there are independent parties manufacturing ODBC database drivers for various databases, such as Intersolv Q+E with their DataDirect ODBC Pack. Drivers from this package give full backward compatibility to the drivers used in CitectSCADA v2.0. Usually, however, any ODBC driver for your database will work.

See Also

[Installing the ODBC driver](#)  
[About the ODBC driver](#)  
[Setting up ODBC](#)  
[Getting the correct syntax with ODBC](#)  
[Programming style with ODBC](#)



### Using CitectSCADA as an ODBC server

#### Installing the ODBC driver

You must install and setup up your ODBC driver from the Windows Control Panel. To do this:

- 1 Open the Windows Control Panel.
- 2 Click the **ODBC** icon to start the ODBC setup utility (if you do not already have an ODBC icon in your control panel, you might need to install the icon. See the documentation provided with your ODBC driver).
- 3 Click **Add** to set up a DataSource.  
**Note:** If your ODBC driver is not included in the list of Installed ODBC Drivers, return to the ODBC setup utility and install your driver (select the **Drivers...** button).
- 4 From the **Add Data Source** dialog box, select your ODBC driver. The Setup dialog is displayed.
- 5 Enter the Data Source Name of the driver that you used previously. For example, if you had used SQL in CitectSCADA v2.0 with a dBaseIII database, then your connection string would have been "DRV=QEDBF". To avoid changing the connection strings throughout your project, use a Data Source Name of **QEDBF**.
- 6 Run your CitectSCADA project.

Some Cicode SQL functions will not work if your ODBC driver has limited functionality. This problem is rare, and in most cases affects only the ability to use transactions with the SQLBeginTran(), SQLCommit(), and SQLRollBack() functions. If you are using the Intersolv Q+E ODBC drivers, do not have any problems: these drivers are fully backward-compatible with drivers used with CitectSCADA Version 2.0.

You might need to change the data source in the Control Panel each time you switch from using one ODBC-compatible driver to another, e.g. from a dBASE file to an Access database. Click the ODBC icon and select from the list of available data sources. (Refer to the documentation supplied with your driver for more information.)

#### Notes

- For full compatibility with the Cicode SQL functions, the ODBC driver should provide a minimum of functions. For example, if your driver does not support the ODBC function SQLTransact, you cannot use the Cicode functions SQLBeginTran(), SQLCommit(), and SQLRollback().
- CitectSCADA used Q+E drivers in versions 2.xx and earlier. Any functions you might have created in these early versions are fully backward-compatible. Q+E drivers are now ODBC-compliant, so you need to upgrade your old Q+E database driver to a Q+E ODBC database driver.

See Also [About the ODBC driver](#)

## About the ODBC driver

CitectSCADA connects directly to the Microsoft Access ODBC driver, which allows applications to access information stored in MDB (Microsoft Access Database) files without actually running Microsoft Access. (Microsoft Access uses the "Jet Engine" DLL to access information stored in MDB files.)

ODBC normally implies heavy use of SQL statements to manipulate data. SQL statements can become quite complex and verbose. To implement them in Cicode they often have to be broken into chunks so that the maximum string length for Cicode variables is not exceeded.

With Access, it is possible to call queries that have been defined in Access so that the SQL statements become quite simple and straight forward. The Access tables & queries can be used to implement **RELATIONSHIPS** and **JOINS**, to **SORT & SELECT** only those rows (records) and return only those columns (fields) of particular interest at the time.

Developing queries in Access also has an advantage that the resulting Recordsets can be viewed in Access to make sure they contain the data that is expected. The queries can incorporate SQL Functions (such as BETWEEN & AND).

The Jet Engine can also call upon the VBA Expression Service. This means that many non ANSI functions can also be used (both in SQL statements and Access Query Definitions) provided there is no need to migrate to a non Access system at a later date. Refer to VBA Functions Reference in the Access or Excel help system (only those functions with (VBA) after them and are appropriate to an SQL environment, are likely to work in an SQL statement).

See Also [Setting up ODBC](#)

## Setting up ODBC

To use ODBC, the Access ODBC Driver must be installed. This can be obtained from Microsoft and is included with Microsoft Office. It is important to use the the 32 bit drivers for Windows 95/Windows NT CitectSCADA 4.x. The installation programme (eg for Microsoft Office) will copy the necessary drivers and the Jet Engine DLL into the appropriate Windows directories when the appropriate Data Access/ODBC options are selected.

With the Driver installed on the PC the ODBC Icon can be selected from the Control Panel and a Data Service Name set up for the desired MDB. This is used in the DSN= part of the connect string.

The Jet Engine DLL is quite large (1 MB) and a problem can arise if the Windows Virtual Memory Manager (VMM) swaps it out of memory. The next time an SQL is executed there will be short delay while the DLL is loaded back into memory. To force the VMM to keep the DLL in memory, design a simple dummy table with only one record and one field and set up a Cicode task that frequently (say

every 10-15 seconds) calls a SELECT query based only on the dummy table. This has no significant effect on CPU load and keeps the DLL in memory.

See Also [Getting the correct syntax with ODBC](#)

## Getting the correct syntax with ODBC

The ODBC syntax for SQLs varies from the Access syntax in some ways. A good way to get the syntax correct and view the resulting Recordset is to use the query designer in **Microsoft Query** then copy the SQL text from it into Cicode. Because MS Query uses ODBC, any syntax that works in it will work when called via ODBC from Cicode. MS Query can also be used to confirm that the DSN is correct.

MS Query tends to create SQL text that is possibly more complex than absolutely necessary. In particular it always includes the path with the file name which is not necessary because the path is already defined in the DSN entry. It is considered bad practice to hard code file paths. MS Query also tends to prefix all column (field) names with the table names to avoid any chance of ambiguity. Again this is not always necessary and it is desirable to keep the SQL text as brief as possible in your code.

The SQL statement text generated by the query designer can be pasted into Execute SQL window (under the File menu of MS Query), any surplus text removed and the SQL statement tested until the simplest syntax that works can be found. There is provision to save the SQL text if required. The final version of the SQL statement can be used with confidence in Cicode.

See Also [Programming style with ODBC](#)

## Programming style with ODBC

Most of the sample code in the following topics do not include error checking and reporting:

- Reading data from an access table with ODBC
- Writing data to an access table with ODBC
- Deleting rows from an Access table with ODBC
- Calling action queries with ODBC
- Parameter queries using ODBC

This has been done to keep the examples as simple as possible. Error checking is (however) essential for ODBC code.

Consideration should be given to implementing most of the complexity of queries in Access Query Definitions where they are easier to design and the results are easily viewed. A WHERE clause can be used when calling the query to select only the desired rows at run time. Where tables have many columns (fields), the Access Query Definitions can be used to restrict any particular call to view only the fields of interest.

It is helpful to build the SQL test up into strings. Firstly the ODBC function calls become simpler. Secondly the strings can be passed to TraceMsg() to make debugging simpler.

Remember that the Jet Engine runs on the same PC as CitectSCADA and that complex queries returning large Recordsets can have an adverse impact on CPU and memory resources. Potential problems can be avoided by careful table, query and relationship design.

If there is a need to execute the queries on a **Remote Computer**, the code can set up on a report server or an event server. This is especially relevant if the code is to be triggered by an event in a PLC. If the code is to be triggered by a User at a Display Station, and the query is considered too CPU intensive, the Display Station can be used to set the PLC bit that calls to code or call the Report using the Cicode Report() function. Another possibility is to use the Cicode MsgRPC() function to call a Cicode function (with parameters, if necessary) on a remote computer. All of these alternatives require CitectSCADA to be running on the remote computer.

See Also [Comparing DDE with ODBC](#)

## Comparing DDE with ODBC

Each has advantages and disadvantages. In general DDE is suitable for simple requirements but ODBC should be given serious thought if the limitations of DDE become too restrictive.

### DDE Advantages

- No need to set up a Data Service Name (DSN); however, a DDEShareName is required for Network DDE.
- Can call Access Macros & Functions.

### DDE Disadvantages

- Record sets with rows that exceed the maximum Cicode string length cannot be read directly.
- Rows (records) are returned to string variable with TAB characters between columns. The user must parse the string in Cicode to obtain the column (field) values.
- SQLs cannot perform Actions (such as INSERT, UPDATE or DELETE).
- DDE Client and server applications must both be running at the same time.

### ODBC Advantages

- MS Access does not have to be running. ODBC uses the JET Engine DLL on the same PC. This an advantage in many ways but can consume excessive PC resources if not managed properly.
- Large SQL statements can be broken into chunks.

- SQLGetField makes easier to get data from fields (columns). There is no need to parse the data in Cicode.
- Can handle large numbers of fields (columns) in the Recordset.
- SQLs can perform Actions (such as INSERT, UPDATE or DELETE).

#### ODBC Disadvantages

- Requires that a Data Service Name (DSN) be set up.

The JET Engine DLL cannot be directly called on a remote PC, (Reports or MsgRPC() can be used however, to run SQL statements on a Remote Computer which must be running CitectSCADA).

See Also [ODBC compatibility](#)

## ODBC compatibility

This section describes the required and optional ODBC functions that your database driver should support:

- [Essential functions](#)
- [Optional functions](#)

#### Essential functions

The CitectSCADA SQL devices and Cicode functions only work if the database driver supports the following ODBC functions:

Level 0 Compliance	Level 1 Compliance
SQLAllocConnect	SQLColumns
SQLAllocEnv	SQLDriverConnect
SQLAllocStmt	SQLGetData
SQLBindCol	SQLGetFunctions
SQLColAttributes	SQLGetInfo
SQLDescribeCol	SQLGetTypeInfo
SQLDisconnect	SQLParamData
SQLError	SQLPutData
SQLExecDirect	SQLSetConnectOption
SQLExecute	SQLSetStmtOption
SQLFetch	
SQLFreeStmt	
SQLGetCursorName	
SQLNumResultCols	
SQLPrepare	
SQLRowCount	
SQLSetParam	

Optional functions

CitectSCADA SQL devices and Cicode functions also use the following ODBC functions, but are not essential for operation. If these functions are absent in a driver, you get a loss of functionality in the operation of SQL devices and Cicode functions.

Level 0 Compliance

SQLTransact	If the driver does not support this function, the Cicode functions SQLBeginTran(), SQLCommit(), and SQLRollBack() are not supported.
-------------	--

Level 1 Compliance

SQLSpecial Columns	CitectSCADA uses this function if it is available. There is no loss of functionality otherwise.
SQLTables	(no effect on CitectSCADA)

Level 2 Compliance

SQLData Sources	The driver does not need to support this function. It is provided by ODBC.
SQLExtended Fetch	Without this function, CitectSCADA cannot take advantage of the native database's ability to fetch records at random.
SQLSetScroll Options	Without this function, CitectSCADA cannot take advantage of the native database's ability to fetch records at random.
SQLMore Results	(no effect on CitectSCADA)
SQLNativeSql	(no effect on CitectSCADA)
SQLProcedure Columns	(no effect on CitectSCADA)

See Also [Using CitectSCADA as an ODBC server](#)

Using CitectSCADA as an ODBC server

The ODBC server support allows CitectSCADA to function as an SQL database server. This will allow third-party applications that support ODBC to access data directly from CitectSCADA. This means that users can have direct access to data in CitectSCADA without having to develop Cicode or reports to export the data.

Currently, the CitectSCADA ODBC server allows variable tags to be accessed. The table for the variable tags is named **"TAGS"** and the format is as follows.

NAME	Variable tag name	read only
VALUE	The current runtime value	read/write

CitectSCADA can only function as a database server at runtime. Using tags through ODBC at runtime can still add to your CitectSCADA License point count. Once this tally reaches a certain limit, CitectSCADA will no longer function correctly. Therefore when accessing tags via the ODBC server, it's important to keep aware of how many points you have used. For details see Citect license point count in the Installation and Configuration Guide.

### Setting up the CitectSCADA ODBC server:

You must have TCP/IP installed on your computer first.

- 1 Choose **Start | Settings | Control Panel**.
- 2 Double-click the ODBC icon.
- 3 Click **Add** on the **User DSN** tab.

**Note:** If you select other tabs you will see that the CitectSCADA ODBC driver has been automatically installed.

- 4 Select the **Citect Driver** from the list and click **Finish**.
- 5 Enter "**Citect**" in the **Data Source** field. If you do not want to use this name, make sure the name you use is one word.
- 6 Enter the Computer Name in the **Host** field. The Computer Name is specified in the Network section of the Control Panel.
- 7 Click **OK**.

### Accessing the CitectSCADA ODBC server using MS Query (V2.00):

All ODBC capable applications use different ways to construct queries for accessing CitectSCADA tags. The example instructions for using MS Query, given here, show a simple implementation. MS Excel and MS Access use the same method.

- 1 Ensure that you have MS Query installed on your computer.
- 2 Set up the CitectSCADA ODBC server for Windows NT or Windows 95.
- 3 Run CitectSCADA.
- 4 Run MS Query.
- 5 From the **File** menu (in MS Query) select **New Query**.
- 6 Select the CitectSCADA Data Source Name (DSN) from the Available Data Sources list. Click the **Use** button.
- 7 Select the **Tags** table. Click the **Add** button and then the **Close** button.
- 8 You can now run a query to extract the Tag data from CitectSCADA. The simplest way to see this is by double-clicking **Names** and **Tags**.

### Accessing the CitectSCADA ODBC server using MS Query (V8.00):

Unlike Version 2.00, User DSNs are not used by Version 8.00. Instead it uses File DSNs which by default are stored in Program Files\Common Files\ODBC\Data Source folder. File DSNs are not stored in the Windows registry, they are text files given the .DSN extension. When you connect to an existing data source, only the available File DSNs that are stored on that PC are displayed. MS Query V8.00 does not display User or System DSNs. The simplest solution is to create a File DSN that points to a User DSN.

### To create a file DSN that points to a user DSN:

- 1 Use a text editor (Notepad for example) and create a file containing the following two lines:

```
[ ODBC ]
DSN=<MyUsrDSN>
```

where <MyUsrDSN> is the name of an existing user DSN that you have created via the ODBC icon in the Control Panel.

- 2 Click **Save As** on the **File** menu and type a name that includes a .DSN file extension. For example, "**Citect\_File.dsn**" is a valid name. Include the quotation marks to ensure that the .DSN file name extension is added correctly. Save it to the default File DSN directory listed above, then it will appear in the DSN list box.
- 3 Open the ODBC Manager from the Control Panel and ensure you can see your newly created File.DSN.
- 4 Open the ODBC Manager from the Control Panel and ensure you have created a User DSN called <MyUsrDSN>. For example:
  - 1 Select Citect Driver and click **Finish**.
  - 2 Type "**Citect**" in the **Data Source** field (i.e., <MyUsrDSN>).
  - 3 Enter **Computer Name** in the **Host** field.

When you run MS Query, you can now select your File DSN from the list.

See Also [Reading data from an access table with ODBC](#)

### Reading data from an access table with ODBC

You can use a SELECT query to read data from an Access table or to call an Access query.

A query is preferred over a table if there are many more columns in the table than are needed at the time, if the data needs to be sorted, or if you need to relate or join several tables. The Cicode required is as follows:

```
Function SQLTest
  INT hSQL, iResult;

  hSQL = SQLConnect("DSN=ODBCTest;UID=YourUID_C;PWD=YourPWD");
  IF hSQL <> -1 Then
    iResult = SQLExec(hSQL, "SELECT * FROM qryRecipes WHERE Recipe Between '3000' And '6000'");
    IF iResult = 0 Then
      WHILE SQLNext(hSQL) = 0 DO
        TraceMsg(">" + SQLGetField(hSQL, "Recipe") + "<>" + SQLGetField(hSQL, "Flour") + "<>" + SQLGetField(hSQL, "Water") + "<>" + SQLGetField(hSQL, "Cocoa") + "<");
      END
      SQLDisconnect(hSQL);
    ELSE

```



```

        Message("SQL Error", SQLErrMsg, 48);
    END
ELSE
    Message("SQL Error", SQLErrMsg, 48);
END
END

```

See Also [Appending data with ODBC](#)

## Appending data with ODBC

To append data to an Access table using ODBC, you can use an SQL INSERT statement.

Function SQLInsert

```

INT hSQL, iResult;

```

```

    hSQL = SQLConnect("DSN=ODBCTest;UID=YourUID;PWD=YourPWD");
    IF hSQL <> -1 Then
        iResult = SQLExec(hSQL, "INSERT INTO tblRecipes (Recipe, Flour, Water, Cocoa) VALUES
('X1234', 2, 3, 4)" );
        SQLDisconnect(hSQL);
    END
END

```

To avoid having to deal with SQL statements, you can use the standard Cicode device functions to append records to an Access table. First configure an SQL device. If the table has many fields that do not need to be written to, define only those fields that are required in the device definition (this keeps the device definition simple and reduces the number of DevWrite instructions). DevOpen, DevWrite and DevClose can then be used to add records to the table.

CitectSCADA accepts successive DevWrites until they equal the number of fields in the device definition at which time it constructs an SQL INSERT statement. The DevWrites must contain data for fields in the same order as the device definition. You should do a DevOpen followed immediately by successive DevWrites for as many records as are required then a DevClose to avoid data being out of context.

See Also [Editing data with ODBC](#)

## Editing data with ODBC

To edit data in an Access table there must be a unique (usually primary) key to identify the row (record) to be changed. This is to be able to provide a WHERE clause that will apply only to that row in an SQL UPDATE.

To edit data, read the data in the normal way, keeping track of the unique key. Any changed values can later be written to the same row using an UPDATE query with a WHERE clause.

### Function SQLUpdate

```

    INT hSQL, iResult;

    hSQL = SQLConnect("DSN=ODBCTest;UID=YourUID;PWD=YourPWD");
    IF hSQL <> -1 Then
        iResult = SQLExec(hSQL, "UPDATE tblRecipes SET Flour = 20, Water = 30,Cocoa = 40 WHERE
Recipe = 'X1234'");
        SQLDisconnect(hSQL);
    END
END

```

Note that the ODBC/SQL environment does not let you edit the "Current Record" (there is no current record). Consequently you cannot use DevAppend or DevSetField to add or modify records.

See Also [Deleting rows from an Access table](#)

### Deleting rows from an Access table

The DELETE keyword is used with a WHERE clause to delete the required row or rows. If the WHERE clause is not based on a primary key, more than one record may be deleted.

### Function SQLDelete

```

    INT hSQL, iResult;

    hSQL = SQLConnect("DSN=ODBCTest;UID=YourUID_C;PWD=YourPWD");
    IF hSQL <> -1 Then
        iResult = SQLExec(hSQL, "DELETE FROM tblRecipes WHERE Recipe = 'X1234'");
        SQLDisconnect(hSQL);
    END
END

```

See Also [Calling action queries with ODBC](#)

### Calling action queries with ODBC

Access ACTION queries cannot be called in a SELECT query such as:

```
"SELECT * FROM qdeDeleteRecipe"
```

To call an Access ACTION via ODBC, use the Call statement in SQLExec:

```
"{Call qdeDeleteRecipe}"
```

The statement must be enclosed in {curly} braces.

The Call statement can be used to Call SELECT queries, the resulting Recordset being accessible in the normal way.

See Also [Parameter queries](#)

### Parameter queries

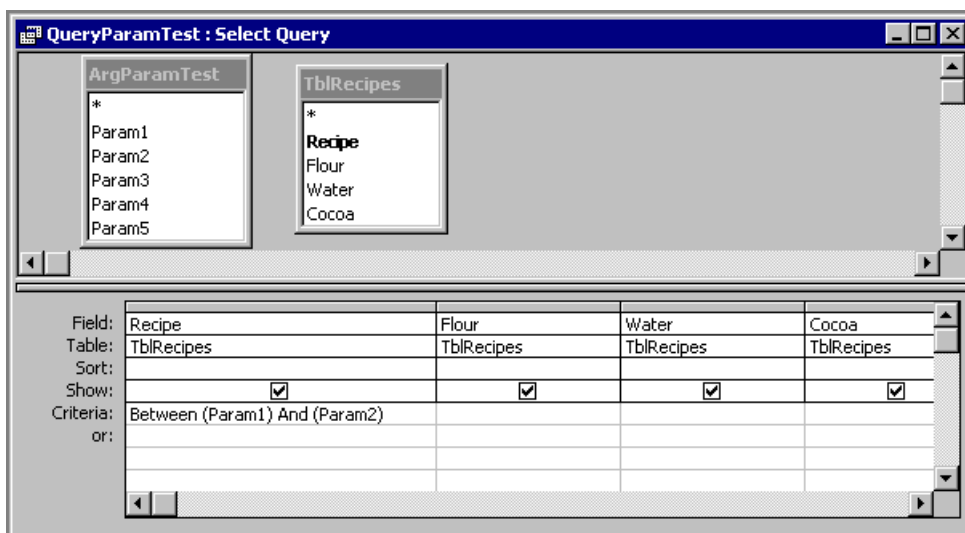
Many ODBC Servers will accept PARAMETERS in a Call statement so that PARAMETERS can be defined in a Query Definition on the server and their values supplied by ODBC Clients at run time. Unfortunately, the Access Jet

Engine uses Parameter Markers which are not supported in the standard Call statement. The method outlined here can be used as a work-around.

For each query that requires PARAMETERS, design a arguments table with the same name as the query but with a different prefix. For example, if the query is qryParamTest, the TABLE could be called argParamTest.

The TABLE should have, say, five fields called Param1, Param2, Param3, Param4, Param5.

This table should be added to the Access Query Definition for qryParamTest. Then the field names can be used as PARAMETERS anywhere in the Query Definition.



A simple Cicode function can be written to which the query name (without the prefix) and PARAMETERS are passed. The function inserts "arg" in front of the query name and executes a DELETE from the TABLE (to be sure that it is empty) and then performs an INSERT to leave the table containing ONE RECORD with the desired PARAMETERS in the appropriate fields. The function then prefixes the query Name with "qry" and Calls the query.

```
Function SQLCall(INT hSQL, STRING sQueryName, STRING sArg1 = " ", STRING sArg2 = " ",
STRING sArg3 = " ", STRING sArg4 = " ", STRING sArg5 = " ")
```

```
    STRING sTable, sQuery;
```

```
    sTable = "arg" + sQueryName;
    sQuery = "qry" + sQueryName;
```

```
    SQLExec(hSQL, "DELETE FROM " + sTable);
    SQLExec(hSQL, "INSERT INTO " + sTable + " (Param1, Param2, Param3, Param4, Param5)
```

```
VALUES ('" + sArg1 + "', '" + sArg2 + "', '" + sArg3 + "', '" + sArg4 + "', '" + sArg5 +
'')");
    SQLExec(hSQL, "{Call " + sQuery + "}");

END
```

Calling the Function from Cicode is then as simple as:

```
SQLCall(hSQL, "ParamTest", "2000", "4000");
```

The default parameters for SQLCall must be SPACES if "Allow Zero Length" is "No" in the Access table Definitions for fields Param1, Param2 etc.

The function can be used to call many different PARAMETER queries.

An advantage of this work-around is that, even after CitectSCADA has been shut down, the query can be called from Access and, because the PARAMETERS are still stored in the arguments table, the resulting Recordset can be viewed in Access.

Another method is to design queries that perform any required joins, sorting and field selection the call them using a WHERE clause to select the desired rows (records).

See Also [Access and Cicode date/time conversions](#)

## Access and Cicode date/time conversions

Access and Cicode have different Date/Time variables data types. All date references in an external database should be based on the Gregorian Calendar.

You can convert between date and time in three ways:

- **Convert to real numbers** - In both Cicode and Access you can equate real numbers to data/time variables, like this:

```
AccessTime = 25568.66667 + (CicodeTime/86400);
CitectTime = 86400*(AccessTime - 25568.66667);
```

- **Convert to strings** - The Data and/or Time are converted to and from text strings using the standard conversion functions available in each environment.
- **Use the #Date/Time# SQL syntax** - The Jet Engine will convert and Dates and Time strings enclosed in # markers. This date could be useful in a WHERE clause:

```
SELECT * FROM qryMyQuery WHERE 'Date' BETWEEN #3/20/96# AND #3/27/96#
```

The American Date format is always used in this case, the Jet Engine DLL ignores the local Date and Time settings as set in Windows Control Panel.

## Chapter 32: Using Genies and Super Genies

---

Usually each graphical object on a [graphics page](#) is configured individually. With a Genie, you can combine several related objects into a group, and store the group in a Genie library (similar to a symbol library). The Genie can then be used as a single object (pasted, moved, resized, etc.), and the elements configured collectively.

All types of graphic objects, and their configuration data, can be stored with the Genie. For example, you can define a Genie for a start/stop controller (with a start button, a stop button, and an indication lamp), and use the same Genie for all equipment (pumps, conveyors, etc.) that use that type of controller. When you use the Genie, you only need to specify the information that is unique to that pump or conveyor (i.e. the variable tag).

CitectSCADA has two types of Genies:

- [Genies](#) - collections of associated objects, which you add to your graphics pages when you configure your system. You can add any number of Genies to a graphics page (for example, multiple pumps on the same page).
- [Super Genies](#) - dynamic pages (usually pop-ups), to which you can pass information when the page displays in the runtime system. You can use Super Genies for pop-up type controllers (to control a process, or a single piece of plant floor equipment).

You can also use a combination of Super Genies and Genies to use the features of both. Most implementations of Super Genies are *attached* to a Genie.

CitectSCADA has included libraries of Genies and Super Genies that you can use in your CitectSCADA system, and you can easily define your own. You can construct a single Genie (or Super Genie) for complex entities such as loop controllers, custom controls and indication combinations.

Note the following:

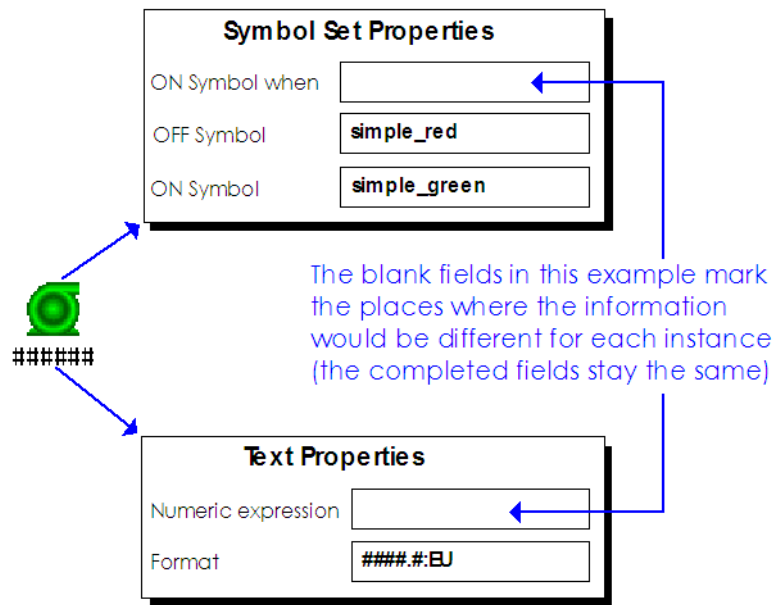
- If you modify a Genie or Super Genie after you have used it in your project, all occurrences of the Genie or Super Genie are automatically updated throughout the project (with the exception of Super Genie Environment Variables).
- If you modify a Genie when the project is running in the background, you must perform an Update Pages to see the changes in the runtime project. If a runtime page containing the Genie is displayed when the change is made, it will not be updated until you exit then re-display it.

See Also [Understanding Genies](#)

## Using Super Genies

### Understanding Genies

Genies work by substituting common information into each related object (in a group of objects). For example, a typical configuration that displays a pump and its speed, uses two objects: (1) a text object that shows the speed, and (2) a symbol object that indicates the state of the pump (by displaying different symbols):



To implement the above arrangement without the use of Genies, you would have to configure the Text and Symbol separately, for each instance on the page. This demonstrates that some common combinations of objects have *mostly* the same configuration in each instance. The concept of a Genie allows this partial configuration to be done, with provision for insertion of the specific information where required.

The power of a Genie is that objects are defined only once. Every time you place the Genie onto a page, you will only have to specify the substitution information.

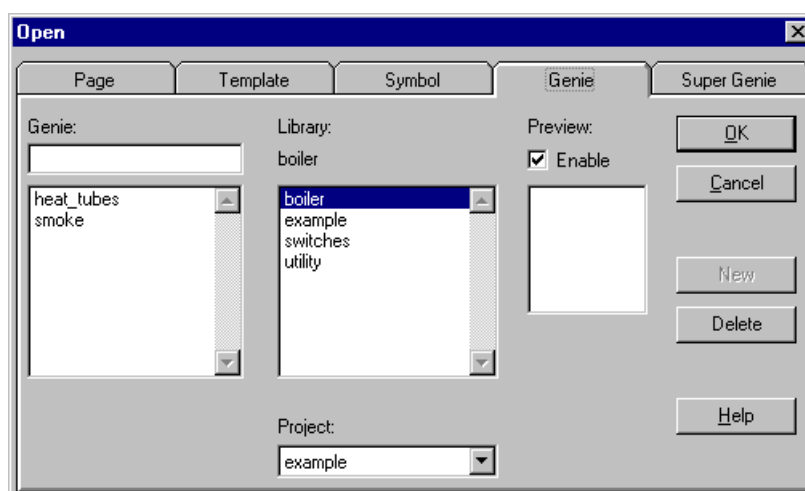
See Also [Creating Genies](#)  
[Opening a Genie](#)  
[Saving a Genie](#)  
[Defining Substitutions for Genies](#)  
[Using Genies](#)

## Creating Genies

Creating a new Genie is similar to creating a page, with graphical objects, but with no background. Typically you would create a **new Genie** using the Graphics Builder, add the objects, **defining** the Genie substitutions, and **save** the Genie in a Genie library.

**To create a new Genie:**

- 1 From the **File** menu select **New**.
- 2 Click the **Genie** button.
- 3 Now you can create your Genie objects (defining your substitution strings).



See Also [Opening a Genie](#)

## Opening a Genie

You can open an existing genie to work with it.

**To open an existing Genie:**

- 1 Click the **Open** tool or choose **File | Open**.
- 2 Select the **Genie** tab.
- 3 Select the **Project** and **Library** in which the Genie is stored.
- 4 Select the **Genie**.
- 5 Click **OK**.

To delete a Genie from the project, select the Genie name, and click **Delete**.

If you modify a Genie or Super Genie after you have used it in your project, all occurrences of the Genie or Super Genie are automatically updated throughout the project (with the exception of Super Genie Environment Variables).

If you modify a Genie when the project is running in the background, you must perform an Update Pages to see the changes in the runtime project. If a runtime

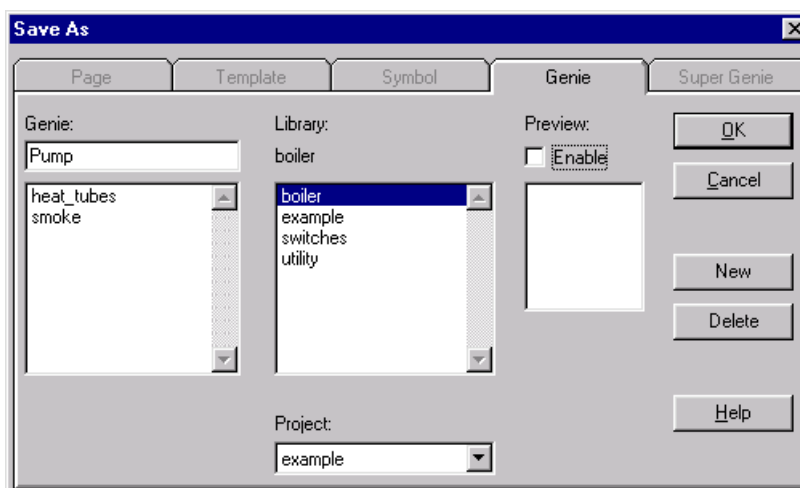
page containing the Genie is displayed when the change is made, it will not be updated until you exit then re-display it.

See Also [Saving a Genie](#)

## Saving a Genie

To save the current Genie:

- 1 Click the **Save** tool, or choose **File | Save**.
- 2 Select the **Project** and **Library** in which to store the Genie.
- 3 Enter a name for the Genie in **Genie**.
- 4 Click **OK**. (To create a new library for the Genie, click **New**.)



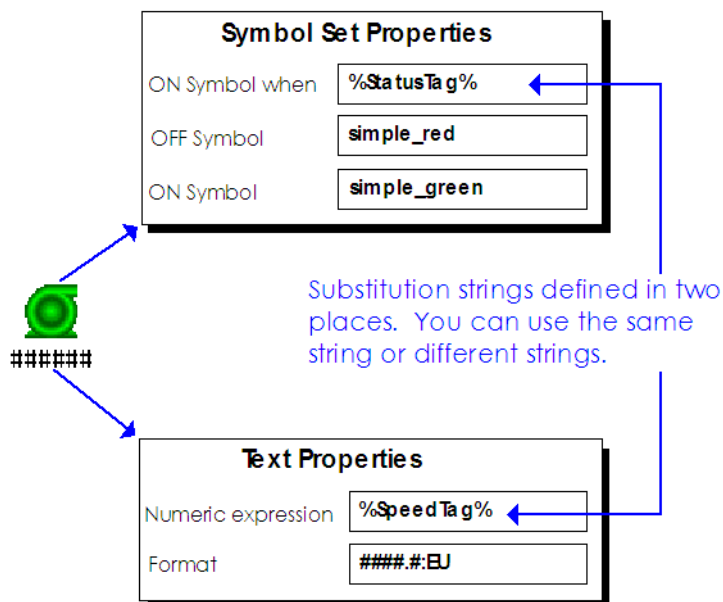
See Also [Defining Substitutions for Genies](#)

## Defining Substitutions for Genies

To define a Genie, you use substitution strings for the properties of the objects that will be specific to each instance. You can use substitution strings for any text property in any object (in the group of objects). To specify a piece of text as a substitution string, enclose the string between percentage (%) characters.



For example, to create a standard Genie, you can use two substitution strings - one substitution string for the status variable tag, one for the speed variable tag:



**Note:** You are not restricted to using only variable tags as substitution strings. Any [expression](#) can be substituted, such as constants or labels. Only fields that accept text can have Genie tag substitutions. You can also define substitutions to variables that aren't in the current project by using the IFDEF function.

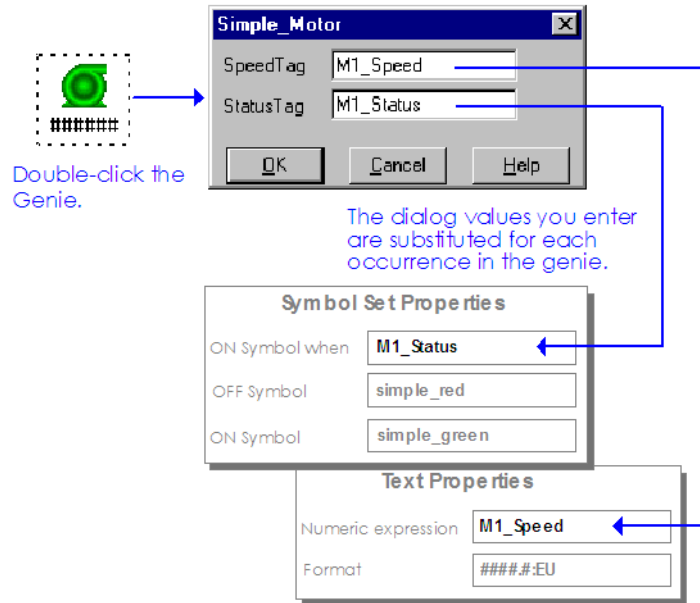
See Also [Using Genies](#)  
[Using Genie Substitutions in Templates](#)

## Using Genies

Once you have created (defined and saved) your Genie, you can use it on any graphics page. To use a Genie, paste it onto a page using the Paste Genie tool. Once the Genie is pasted, configure it by double-clicking the image.

For example, each time you use the above Genie, you only have to enter two values in a single dialog - one for the speed variable tag (%**SpeedTag**%) and one

for the status variable tag (%**StatusTag**%) - instead of properties for each object in the group.



Double-clicking a pasted Genie displays the Genie Properties. To display the properties of the individual objects in the Genie, hold the **Control** (CTRL) key down and double-click the specific object. If, however, a link to the Genie has been retained, most of these properties will be read-only.

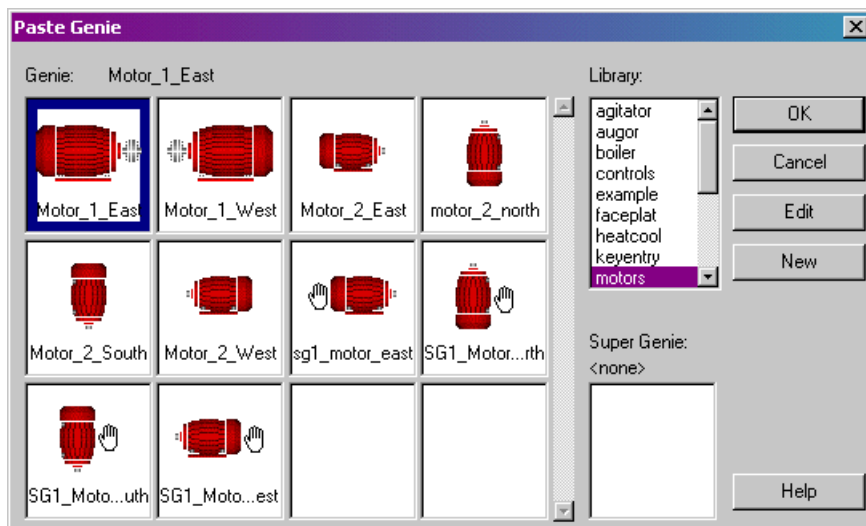
The above example is a simple use of a Genie - it only contains two objects and two substitution strings. You can define Genies that use many objects, with substitution strings for any **text** property (or properties) of an object.

**Note:** If you use structured tags, you can use substitution strings within a tag name to construct more sophisticated Genies. See [Using Structured Tag Names with Genies and Super Genies](#).

To paste a Genie onto a graphics page:

- 1 Click the Paste Genie tool (in the toolbox), or choose **Edit | Paste Genie**.
- 2 Select the library (from the Library list) that contains the Genie.
- 3 Select a Genie thumbnail from the Genie list in the Paste Genie dialog.

- 4 Double-click the thumbnail or click **OK**.



See Also [Paste Genie dialog box](#)

## Paste Genie dialog box

Use the Paste Genie dialog box to add a genie to your graphics page (or template).

### Genie

A table of Genies in the project, showing attached Super Genies.

To add a Genie, use the scroll bar to locate the thumbnail image of the Genie, then select the Genie and click **OK** (or double-click the thumbnail image).

**Note:** To edit the Genie, select it and click **Edit**. To create a new Genie, click **New**.

### Library

The library where the Genie is stored.

### Super Genie

If the selected Genie is attached to a Super Genie, a thumbnail image of the Super Genie is displayed; otherwise this field is blank.

## Genies properties

The Genie dialog box displays the substitution strings that you have entered for the Genie. The substitution tags you see on the form are defined in the Genie. The values you enter next to the tags will be substituted into the Genie (and possibly Super Genie, if one is attached).

**Note:** To display the properties of the individual objects in a Genie (instead of the Genie Properties), hold the **Control** (CTRL) key down and double-click the

## Using Genie Substitutions in Templates

object. If, however, a link to the Genie has been retained, most of these properties will be read-only.

See Also [Understanding Genies](#)

You can create custom page templates with the same characteristics as Genies by adding objects to a template, and using substitution strings (%) for the relevant properties of each object. (If you have default values for any property, you can add the default values to the native objects.)

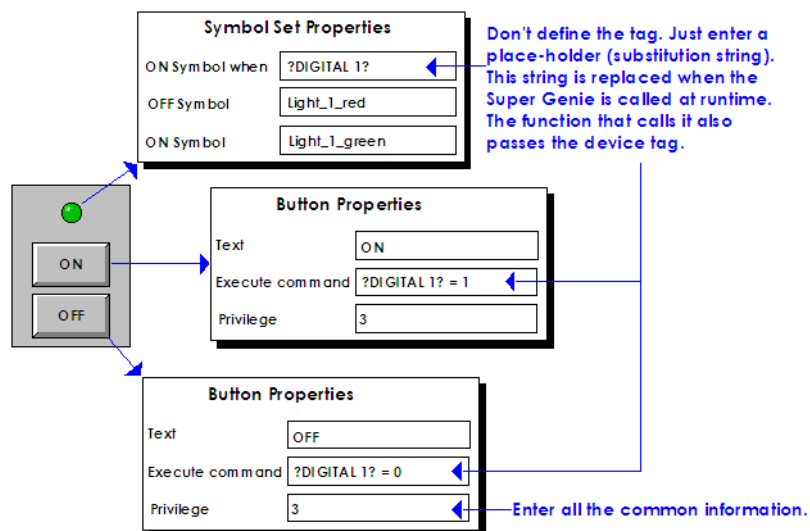
When you subsequently create a new page based on the template, a single dialog prompts you for values for all substitution strings used in the template. This is how the templates for trending and SPC were created.

## Using Super Genies

Individual pages (popup controllers, loop tune pages, etc.) are often used to control and monitor devices. Super Genies are ideal when there are many devices of the same type, because you can re-use them many times without re-configuring them for each device. Configure the common information once; the device-specific information is passed to the Super Genie at runtime.

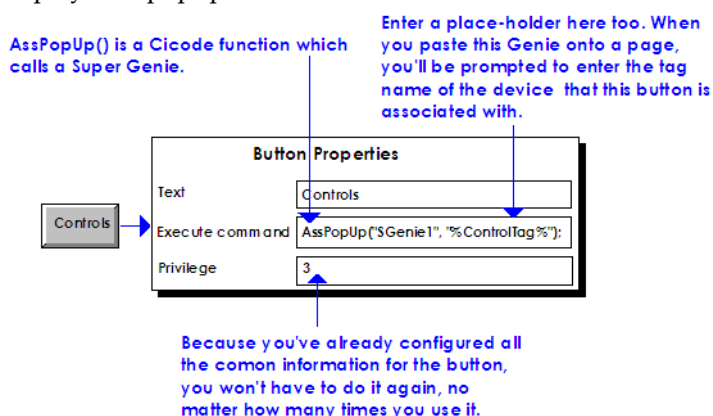
For instance, you might use a Super Genie to configure a single popup page for controlling all electric pumps that have the same functionality. The best way to configure this controller is:

In the Graphics Builder, select **File | New Super Genie**, draw your controller and fill out the associated properties forms as follows:



Save it in a Super Genie library using an exclamation mark (!) prefix. This keeps the pages hidden in the configuration environment (they're visible only if attached to a [Genie controller](#)).

Select **File | New Genie**, and draw the button that the user will click at runtime to display the popup controller. This button is called a Genie controller. It will call a Super Genie Cicode function, which performs the substitutions and displays the popup.



Because the Super Genie function call is made from the Genie Controller, you only have to configure it once.

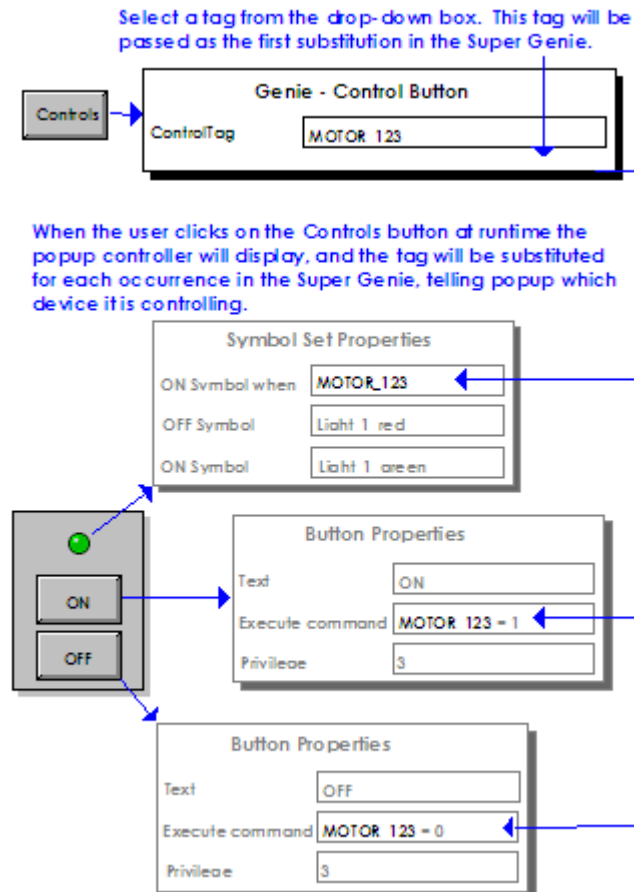
Save your Genie to a Genie Library. Like Genie libraries, Super Genies libraries are global and can be used between CitectSCADA projects.

With the Genie open, select **Edit | Attach Super Genie**, and select the Super Genie you just created. From now on, pasting this Genie will always call the new Super Genie.

By attaching each of your Super Genies to a Genie, you ensure that your Super Genies are stored in an orderly way in Genie libraries. This makes them easy to maintain and easy to paste into your projects. A Super Genie can be attached to more than one Genie controller.

Paste the Genie wherever you want the user to be able to use the popup controller. Select **Edit | Paste Genie**, browse for the Genie you just created, and

select it. A new page in your project will automatically be created for the Super Genie.



To implement the above situation without Genies and Super Genies, you would have to manually configure a separate page for each pump in your application, and a separate button to call each page. Using a Super Genie, you only have to configure one page manually. The rest are created automatically.

**Note:** All variable tags used in a Super Genie must be defined in the variable tags database. Alarm tags can also be used (allowing you use alarm tag properties).

Because of the overhead required for Super Genies, you should restrict the number of Super Genie variables. Arrays do not suffer the same limitation and perform well even with hundreds of variables.

Using tags through Super Genies at runtime increases your dynamic license point count. Super Genies called after you have reached your [point limit](#) will

return #COM. For more information see Citect license point count in the Installation and Configuration Guide.

You don't have to implement Super Genies using a Genie Controller. See [Using Super Genies without Genies](#).

#### See Also

[Defining Substitutions for Super Genies](#)  
[Using Super Genies without Genies](#)  
[Using Constants and Arrays with Super Genies](#)  
[Nesting Super Genies](#)  
[Super Genie areas](#)  
[Super Genie environment variables](#)  
[Using structured tags with Genies](#)

## Defining Substitutions for Super Genies

Super Genie substitution is more rigid and complex than that of Genies. Most importantly, you can only use Super Genie substitution in the properties of an object that accept tags, commands and expressions. (You can also use Super Genie substitution in log messages for object touch and keyboard commands, tool tips, page keyboard commands, or as part of the comment for Trend objects, and Color Floods.) You cannot use the Super Genie syntax in a report, alarm, trend, or background Cicode function.

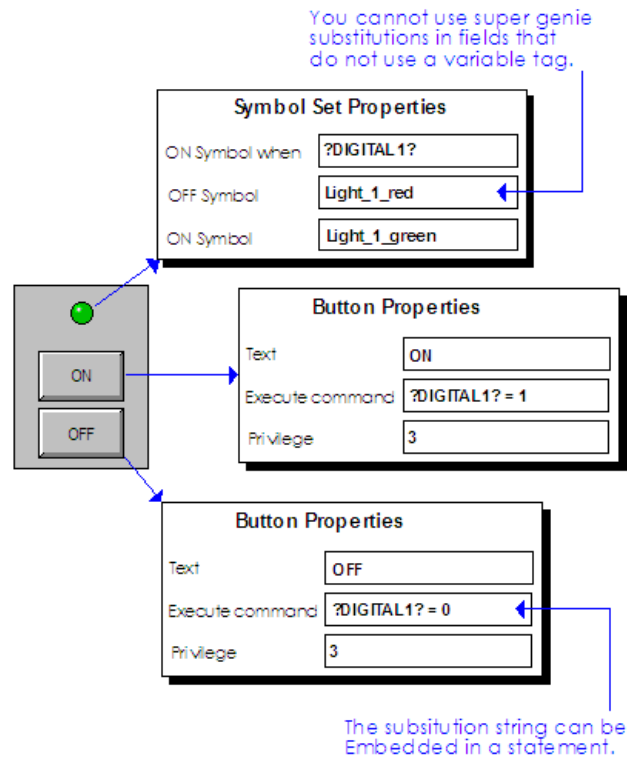
To mark a tag as a substitution string, enclose the tag between question mark (?) characters, in the following format:

`?<Data Type> <Substitution String Number>?`

where:

- **Data Type** is optional and can be any data type supported by CitectSCADA. In practice, only explicitly specify type when the type is STRING.
- **Substitution String Number** determines which variable tag (1 to 256) will be substituted when the Super Genie is displayed (using the Super Genie functions). If you use more than one substitution string in your Super Genie, your numbers should be sequential. This will make the Super Genie functions easier to use.

For example, to define substitutions for the pop-up controller, use a substitution string for the variable tag, as follows:



**Note:** This Super Genie should be saved as a page - called **SGenie1** - as opposed to a Super Genie, so that the Super Genie can be used without a [Genie controller](#).

If you do not specify a data type, it will default to TYPELESS. Typeless substitution allows you to pass tags of BYTE, BCD, DIGITAL, INT, UINT, LONG, LONGBCD, or REAL types, but not STRING. When you make a typeless substitution, CitectSCADA will automatically try to convert the substituted 'data' to the correct type at runtime.

For example, the above diagram uses **?Digital 1?** as the substitution string. At runtime you would get a hardware error if you passed a variable declared as INT. If instead, you used **? 1?**, at runtime you could pass a variable of any type but STRING.

**Note:** You might want to use typeless substitutions because they offer more flexibility, but you should be aware that errors can be harder to find.

See Also [Using Super Genies without Genies](#)



## Using Super Genies without Genies

You do not have to implement Super Genies as attachments to Genies. Instead, you can save an unattached Super Genie as a normal CitectSCADA page. This method has the advantage that you do not have to define a controlling Genie, but the disadvantage that you can't use the Paste Genie tool to place it.

If you configure a Super Genie in this way and name the page with an ! prefix to hide it, you must select **List System Pages** from the Graphics Builder **Options** menu to edit the page. At all times, the first eight characters of the Super Genie name must be unique for each Super Genie.

All Super Genies supplied with CitectSCADA are attached to Genies (as controls for the Super Genie).

### To create a new Super Genie:

- 1 Click the **New** tool, or choose **File | New**.
- 2 Click **Super Genie**.
- 3 Now you can create your Super Genie **page** (defining your substitution strings).

**Note:** For the Super Genie to display in the Paste Genie dialog, create a Genie to use as the Genie controller and attach the Super Genie to it. The first eight characters of the Super Genie name must be unique for each Super Genie.

### To open an existing Super Genie:

- 1 Click the **Open** tool or choose **File | Open**.
- 2 Select the **Super Genie** tab.
- 3 Select the **Project** and **Library** in which the Super Genie is stored.
- 4 Select the **Super Genie** and then click **OK**.

**Note:** To delete a Super Genie from the project, select the Super Genie name and click the **Delete** button.

### To save the current Super Genie:

- 1 Click the **Save** tool or choose **File | Save**.
- 2 Select the **Project** and **Library** in which to store the Super Genie
- 3 Enter a name for the Super Genie in the **Super Genie** field (you should limit the name of the Super Genie to eight (8) characters) and then click **OK**.

See Also

[Using Constants and Arrays with Super Genies](#)

## Using Constants and Arrays with Super Genies

You can use both [Constants](#) and [Arrays](#) with Super Genies.

### Constants

The ability to pass constants into Super Genies is restricted in that, the constant association can only be where you can enter a normal Cicode tag - keyboard

command, symbol address field etc. All types of constants are supported: STRING, INTEGER, DIGITAL, REAL, and LONG.

To pass a constant you need to format the argument in the Ass function to include a single quote on either side. For example, to pass the constant data **1.2345** into a Super Genie, you would call the Ass function like this:

```
Ass(hWin, nArg, "'1.2345'");
```

To pass a variable tag, you don't need the single quotes. For example, to pass variable tag **TAG1** into a Super Genie, you would call the Ass function as follows;

```
Ass(hWin, nArg, "TAG1");
```

### Arrays

Super Genies can accept array elements or entire arrays as substitution. Passing an element of an array is straight forward, and is done by reference to the element, as shown here:

```
AssPopUp("MyPopUp", "DigArray[42]");
```

To pass an entire array to a Super Genie, only the array name is used. For example:

```
AssPopUp("MyPopUp", "DigArray");
```

When passing an entire array, the Super Genie must be configured to accept an array - instead of a single value. The following syntax must be used for the Super Genie substitution string:

```
?<Data Type>[<array size>] <Substitution String Number>?
[<element>]
```

Only arrays of data type DIGITAL, INT, REAL, and LONG are supported.

**Note:** The <array size> is optional and if not defined then will default to 2048 digital, 128 integer or 64 real elements. You would only use it to check the range of the array - so that if an array smaller than expected is passed into the Super Genie, out of range values will default to 0 (or a null string) rather than generate a Cicode error.

For example, to display element [3] in the first substitution tag (which is a digital array), the following syntax could be used:

```
Expression      ?DIGITAL[] 1? [3]
```

Alternatively, the following syntax could be used to ensure that an array of the expected size is being passed into the Super Genie:

```
Expression      ?DIGITAL[4] 1? [3]
```

See Also [Creating a Genie controller](#)

## Creating a Genie controller

### To create a Genie controller:

- 1 Save the Super Genie (you should limit the name of the Super Genie to eight characters).
- 2 Create a Genie that uses a Super Genie function to display the Super Genie.
- 3 Choose **Edit | Attach Super Genies**.
- 4 Click **Add**. The Select Super Genie dialog is displayed.
- 5 Select the Super Genie that you saved in step 1 to add your Super Genie to the list for this Genie, and then click **OK**.
- 6 Save the Genie. The Super Genie appears in the Paste Genie dialog.

### To paste a Super Genie (controller) from the Genie library to the page:

- 1 Click the **Paste Genie** tool, or choose **Edit | Paste Genie**.
- 2 Select a library from the **Library** list in the Paste Genie dialog.
- 3 Select a Genie thumbnail from the **Genie** list. A thumbnail of the attached Super Genie appears in the **Super Genie** box.
- 4 Double-click the thumbnail or click **OK**.

**Note:** This procedure adds a Genie (to the page) to which a Super Genie is attached. The Genie is a controller for the Super Genie. When an operator selects the Genie in the runtime system, the Super Genie is displayed.

See Also [Attach Super Genie dialog box](#)

## Attach Super Genie dialog box

You use the Attach Super Genie dialog box to attach a Super Genie to the current Genie.

### Attached Super Genies

A list of Super Genies attached to the current Genie.

### To attach a new Super Genie:

- 1 Click **Add**.
- 2 Use the Select Super Genie dialog box to select the Super Genie to attach.
- 3 Click **OK** to save the changes, or click **Cancel**.

### To detach a Super Genie:

- 1 Click **Remove**. You will not be asked to confirm if you want the attachment removed.
- 2 Click **OK** to save the changes, or click **Cancel**.

See Also [Select Super Genie dialog box](#)

## Select Super Genie dialog box

The Select Super Genie dialog box lets you select a Super Genie to attach to the current Genie.

### Super Genie

A table of Super Genies in the project.

To select a Super Genie, use the scroll bar to locate the thumbnail image of the Super Genie, then select the Super Genie and click **OK** (or double-click the thumbnail image).

**Note:** To edit the Super Genie, select it and click **Edit**. To create a new Super Genie, click **New**.

### Library

The library where the Super Genie is stored.

## Nesting Super Genies

CitectSCADA allows you to nest Super Genies. Nesting refers to where one Super Genie is embedded in another. For this to work, the embedded Genie controller (for the embedded Super Genie) must use `AssChain` functions instead of `Ass` functions.

See Also [Super Genie areas](#)

## Super Genie areas

When you display a Super Genie, the area of the Super Genie is inherited from its parent. For example, if the parent page is in [area 1](#), when you display a Super Genie it will also be area 1. This allows you to call the same Super Genie from different pages in different areas.

The inherited area may be avoided by defining the Super Genie to have a specific area. Then, every instance of the Super Genie will have the same area, no matter which area its parent is from. Super Genies will only inherit areas if their area is blank.

See Also [Super Genie environment variables](#)

## Super Genie environment variables

When you define a Super Genie, you are actually creating a Super Genie template, similar to a page template. When a [Genie controller](#) calls the Super Genie, this template is used to create a new Super Genie page. At this point, any environment variables saved with the template are copied across to the Super Genie page. However, if subsequent changes are made to the environment variables of the template, the environment variables of the Super Genie page is unchanged.

To update the Super Genie page environment variables with changes made to the template, you must find and delete the Super Genie page (remember it may be prefixed with an exclamation mark (!)) and then use the Genie controller to call the Super Genie again. This will create a new Super Genie page that has the updated environment variables.

## Using Structured Tag Names with Genies and Super Genies

Using structured tag names provides more power when using Genies and Super Genies, so use a structured tagging convention. Most Genies refer to the same physical device, and therefore using similar tag names for each element in the device reduces project configuration.

See Also [Using structured tags with Genies](#)  
[Using structured tags with Super Genies](#)

### Using structured tags with Genies

When you define a Genie, you can add a prefix or suffix to a Genie property to generate the complete tag when the Genie is used. For example, if you define a Genie property as %tag%\_PV, and then use DEV1 for the tag, the Genie will generate the complete tag DEV1\_PV.

You can add extra information at the beginning (prefix), or on the end (suffix) of the Genie property, or use both a prefix and suffix in the same Genie property. For example, if you have defined a loop controller with three bar graphs (created using the fill property in a rectangle) to display the tags DEV1\_PV, DEV1\_SP and DEV1\_OP, you can configure a Genie as follows:

Each rectangle has a separate Genie tag:

Level expression	%PV_Tag%
Level expression	%SP_Tag%
Level expression	%OP_Tag%

When you configure the Genie (with the Genie dialog), you have to enter three separate tags: DEV1\_PV, DEV1\_SP and DEV1\_OP. However, if you use structured tags, you can configure the rectangles as follows:

Level expression	%Tag%_PV
Level expression	%Tag%_SP
Level expression	%Tag%_OP

In this case, you only have to enter one tag (DEV1) to generate six objects. The Genie automatically concatenates DEV1 with either \_PV, \_SP, or \_OP, depending on where the tag is substituted. As well as a reduction in configuration time, this Genie is easier to maintain.

**Note:** The above example illustrates the power of Genies. The more complex and the greater number of objects in a Genie, the greater the advantage of using structured tags. You can also make complex Genies by using multiple variables for a Genie property. For example, %Area%\_TIC\_%Occ%\_PV or any combination of prefix, suffix and number of Genie variables.

See Also [Using structured tags with Super Genies](#)

## Using structured tags with Super Genies

Super Genies do not support direct concatenation of the Super Genie tag with other information (as do Genies). For example, ?INT 1?\_PV is not valid and will generate a compiler error. However, you can concatenate the tag using a Cicode expression. You must use a unique Super Genie variable for each real tag, and concatenate the tag with the Ass Cicode function. For example, if you have defined a loop controller with three bar graphs (created using the fill property in a rectangle) to display the tags DEV1\_PV, DEV1\_SP and DEV1\_OP, you can configure a Super Genie as follows:

Each rectangle has a separate Genie tag:

```
Level expression      ?INT 1?
Level expression      ?INT 2?
Level expression      ?INT 3?
```

If you do not use structured tags, you can call the Ass function for the above Genie as follows:

```
AssPage( "PageName" , "DEV1_PV" , "DEV1_SP" , "DEV1_OP" );
```

To concatenate information for the Genie, you could also write your own Cicode function, as follows:

```
FUNCTION
AssMine(STRING sPage, STRING sTag)
AssPage(sPage, sTag + "_PV", sTag + "_SP", sTag + "_OP");
END
```

With this function, you can call your AssMine() function (for example, from a command button), and pass a single tag (DEV1), as follows:

```
AssMine( "PageName" , "DEV1" );
```

Writing your own Cicode function to call a Genie provides extra flexibility; however, you can also use a Genie (for example, from a button command) to call the Ass function, as follows:

```
Execute command      AssPage("%Page%", "%tag%_PV", "%tag%_SP", "%tag%_OP");
```

When you use the above Genie, you only enter the page name and tag once.

You must pass the tag name (by enclosing it in quotation marks) to the Super Genie functions. You cannot pass the tag values. For example, if you pass %tag%\_SP (no quotes), the value of the variable and not the tag name is passed to the Genie, and the association will fail.

See Also [Using structured tags with Genies](#)

## Hiding Graphics Objects

You can configure a graphics object so that if the variable tag specified for the object is not defined in the tag database at compile time, the object does not display on a graphics page.

The [expression](#) entered in the **Hidden When** field of an object's property is used to determine if the object will display. The expression evaluates to either TRUE or FALSE and the object is hidden when the expression is TRUE.

You define the variable tag and conditions under which the object is hidden by entering an [IFDEF](#) statement into the **Hidden When** field when you configure the object. The IFDEF statement is evaluated by the compiler and the value of the resulting expression or variable tag will determine whether or not the object is hidden.

This can significantly reduce the number of required genies, as the configuration engineer does not need to generate several smaller genies to cater for operations driven by a slightly different range of tags.

See [IFDEF macro](#) for more information on using the IFDEF macro for hiding graphics objects. For more information on the IFDEF macro in general, see [IFDEF](#).

### IFDEF macro

The [IFDEF](#) statement consists of three arguments:

```
IFDEF (<"Tag name">, <Hidden When value if tag defined>, <Hidden When value if tag undefined>)
```

The first includes a variable tag name. If the variable tag is defined in the tag database at project compilation, the IFDEF statement is replaced in the **Hidden When** field by the second argument. If the variable tag is undefined, the **Hidden When** field will contain the third argument.

#### Example 1

```
IFDEF("Bit_1", 0, 1)
```

In the above example, if Bit\_1 is defined in the tag database, the value in the **Hidden When** field will be 0. If Bit\_1 is undefined, the value will be 1. Since the object is hidden when the value is TRUE, the object will be hidden when BIT\_1 is undefined (i.e. when the **Hidden When** field contains 1).

#### Example 2

```
IFDEF("Bit_2", , "1")
```

If the second argument is omitted, as in Example 2, the variable tag specified in the first argument is used. If Bit\_2 is defined, therefore, the **Hidden When** field will contain Bit\_2. The value of the variable tag Bit\_2 is then used to determine if the object is hidden. A non-zero value will equate to TRUE, causing the object to be hidden.

If Bit\_2 is undefined, the Hidden When expression evaluates to 1 (TRUE) and the object is hidden.

**To enter an IFDEF statement in the Hidden When Field:**

- 1 Double-click the graphics object for which you want to edit the field.
- 2 Select the **Appearance** tab.
- 3 Click the **Hidden When** field and enter the IFDEF statement.
- 4 Click **OK**.



# Chapter 33: Working with Multi-Language Projects

---

CitectSCADA's language switching facility allows you to use one language to configure a project, and another for runtime text items such as alarm descriptions, button text, keyboard/alarm logs, graphic text, Cicode strings, and so on. You can also dynamically change languages during runtime.

For example, if your native language is English, you could enter an English alarm description when configuring the project, but specify to display it in the French or German (or any other language) equivalent at runtime. You can specify the language you want before running the project, or change it dynamically at runtime (using the `SetLanguage()` function) without affecting any of the project's normal operations.

CitectSCADA distinguishes between the *native* language (that is, the language of the developer), and the *local* language (the language of the end user). Language changes are achieved by using a [language database](#), which has a field for native text, and a field for the translated local text. When the project is run, native text is replaced with the equivalent local text.

Alarm and keyboard logs can be processed in both the native and the local language, enabling native and local users to read historical logs. The data can be logged to the same device, or to separate devices.

See Also [Changing Languages](#)

## Changing Languages

This section describes how to change languages used by CitectSCADA.

See Also [Marking text for language change](#)  
[Language databases](#)  
[Multiple languages](#)  
[Multiple projects](#)  
[Changing languages at runtime](#)  
[Logging data in different languages](#)  
[ASCII and ANSI character sets](#)  
[OEM character sets](#)

### Marking text for language change

During project development, you must mark any text you want change to another language at runtime with a language change indicator, like this:

```
@( Native Text [,Width [,Justify]])
```

where `Native Text` is the identifying text to be displayed when configuring. This text will be replaced by the local equivalent at runtime. Note that the brackets are required as they specify the extent of the native language text; `Width` and `Justify` are optional (indicated by the square brackets).

For example, if English is the native language, you could enter the following alarm description:

```
Alarm Desc      @(Motor Failure)
```

This indicator serves two purposes: It flags the text as native, and tells CitectSCADA to change the text from native to local at runtime.

By default, the text that you enter here can be in any combination of upper- and lowercase. In other words, *Motor Failure* will be considered the same string as *motor failure* or *MOTOR Failure*, and they will all have the same [local language](#) translation. Case-sensitivity can be introduced by setting the `[Language]CaseSensitive` parameter to 1.

`Width` can be assigned any value from 0 to 254. If the local text is longer than specified, it is truncated and left-justified. If a width is not specified, the field is the length of the local text and the text left-justified.

`Justify` specifies the text justification and can only be used with `Width`. `Justify` can be one of the following values:

- `l` or `L` - Left
- `r` or `R` - Right
- `c` or `C` - Center
- `n` or `N` - None

For example, to limit the local text in the previous case to 20 characters with right justification:

```
Alarm Desc      @(Motor Failure, 20, R)
```

Characters that are normally part of the formatting - `@`, `(` - can also be used within the native text. To do this, place a caret (^) character before them. For example, to include a comma without ruining the format:

```
Alarm Desc      @(Motor Failure^, thermal overload, 20, R)
```

**Note:** The caret (^) character does appear at runtime or in the language database.

See Also [Language databases](#)

## Language databases

When the project is compiled, CitectSCADA creates a language database (dBASE III format), consisting of two fields: **NATIVE** and **LOCAL**. Text marked with a language change indicator is automatically entered in the **NATIVE** field. You can then open the database and enter the translated text in the **LOCAL** field.

For example:

NATIVE	LOCAL
Line Broken Alarm at Line Speed {LineSpeed1}	<Translation of <i>Line Broken Alarm at Line Speed {LineSpeed1}</i> >
Main Menu page	<Translation of <i>Main Menu page</i> >
Conveyor Belt Trip	<Translation of <i>Conveyor Belt Trip</i> >

When the project is run, the translation of *Line Broken Alarm at Line Speed {LineSpeed1}* will display in place of the English text, the translation of *Main Menu page* will display in place of the English text etc.

For the language change to occur automatically when the project is run, you must specify the language database to use *before* running the project by using the [Language]LocalLanguage parameter. Otherwise, you can change the language yourself at runtime using the SetLanguage( ) function.

If you do not enter a local equivalent of the native text string, the native text is displayed by default. You can specify to display "#MESS", instead of the native text by setting the [Language]DisplayError parameter to 1 (one); the default is 0 (zero).

For single-byte languages (such as French), the database can be edited using Microsoft Excel; for double-byte languages (such as Chinese), you should use Visual FoxPro.

By default, the language database created by the compile is called **English.dbf** (this can be changed using the [Language]LocalLanguage parameter). It is saved to the project directory. Once the database is created, it is updated when you compile. Text marked since the last compile is appended to the database; the rest of the database is unchanged.

See Also [Multiple languages](#)

## Multiple languages

**Note:** To use characters for Baltic, Central European, Cyrillic, Greek, Turkish, and Asian languages, or right-to-left languages (Arabic, Hebrew, Farsi, and Urdu) the operating system must have the corresponding language version of Windows, or have installed system support for that language.

Each [local language](#) must have its own language database, so that it can be displayed in place of a specified native language at runtime. Also, it must be set as the local language using the [Language]LocalLanguage parameter. With this parameter set *before* you compile, CitectSCADA automatically creates/updates the relevant language database.

For example, to display text in French at runtime, set the [Language]LocalLanguage parameter to French, flag all necessary native text in the project with @(), and compile. After compiling, look in the project directory for **French.dbf**, open it, enter the required French translations in the

**Local** field, and save the database. When the project is run, all marked native text will be replaced by the appropriate French text.

Because you can have any number of databases, you can use as many different languages as you like.

When you compile, all text marked with a language change indicator is entered in the **Native** field of whatever database is set as the local language using the `[Language]LocalLanguage` parameter. Therefore, you should know what database is set *before* you compile.

Also, if you have several language databases with the same native language, remember that newly marked text is only appended to the *current* local language database (as specified by the `[Language]LocalLanguage` parameter). To add this text to other databases with the same native language, change the `[Language]LocalLanguage` parameter, update pages, and recompile for each database. Remember that for each database, only relevant changes made since the last compile are added.

See Also [Multiple projects](#)

## Multiple projects

A language database can contain entries which are not actually included in a project. This means that a single language database can be developed that is applicable to many projects.

See Also [Changing languages at runtime](#)

## Changing languages at runtime

The language of runtime display items such as alarm descriptions, button text, keyboard/alarm logs, graphic text, Cicode strings and so on can be changed dynamically at runtime using the `SetLanguage()` function. All normal operations of the project continue unaffected.

**Note:** Forms do not automatically update when the language is changed using the `SetLanguage()` function. They must be closed and reopened for the change to take place.

Local translations that are missing from the specified language database are replaced by the native equivalent. You can specify to display "#MESS", instead of the native text, by setting the `[Language]DisplayError` parameter to 1 (one); the default is 0 (zero).

See Also [Logging data in different languages](#)

## Logging data in different languages

Alarm and keyboard logs can be processed in both the native and the [local language](#). This means that both native and local users can read the historical logs. The logs can employ the same device, or separate devices.

Logs in the local language are produced using the standard field names. For example, if {NAME} {DESC} {COMMENT} is entered in the format field of an alarm

category, the alarm name, description and comment of all alarms in that category will be logged in the local language.

All fields which support the automatic language change facility can also be logged in the native language. To do so, just precede the field name with **NATIVE**. For example, to log the name, description and comment of a category of alarms, enter {NATIVE\_NAME} {NATIVE\_DESCRIPTION} {NATIVE\_COMMENT} in the format field for that category.

To log both native and local to the same device, just enter the standard fields, and the native fields together in the format field. To log them to different devices, use a Group of two devices, and enter the local fields as the format for one, and the native fields as the format for the other.

See Also [ASCII and ANSI character sets](#)

## ASCII and ANSI character sets

Each screen character is defined by a code (number). Operating systems and applications need to know these codes to attach meaning to individual characters. A character set provides a code for every character. For your operating system/application to interpret a character correctly, you must use the correct character set.

**Note:** Character sets are distinct from fonts. A font defines the visual/appearance properties of a character, not its meaning.

ASCII (American Standard Code for Information Interchange) is a widely adopted 7-bit code specifying the basic alphanumeric character set of the English language. For example, the character capital "A" has the ASCII value of 65, the character lowercase "a" a value of 97.

The ASCII character set contains 96 characters and is commonly used as a standard for protocols and files.

Much of CitectSCADA uses ANSI (American National Standards Institute) character sets. ANSI character sets are language-based, with each different language version of Windows (French, Korean and so on) requiring a specific ANSI character set. Codes 32 to 127 always contain standard ASCII characters.

Windows uses Unicode, but still supports ANSI character sets. Several of CitectSCADA's utilities have been created for Unicode, i.e. Process Analyst and CitectSCADA Pocket. Unicode avoids the problem of multiple character sets by having one 16 bit (worldwide) character encoding standard.

See Also [OEM character sets](#)

## OEM character sets

OEM character sets are those which are used by MS-DOS or Console applications (they are operating system dependent). Most OEM character sets do not match the ANSI character sets. For example, line drawing characters commonly used in MS-DOS character sets were replaced with language characters in ANSI character sets.

Problems can arise when building multiple language projects if inadequate consideration is given to the role that ANSI and OEM character sets play, in the way the language strings are stored and interpreted.

Language configuration information is stored in dBase files (a database standard defined primarily for MS-DOS applications) where string information is customarily stored as OEM characters. When using a Windows application (such as Excel) to edit dBase files, the characters on screen are in the ANSI character set. When you save this information to the dBase file, Excel converts it to an OEM equivalent. For this conversion to work correctly the OEM character set must be compatible with the ANSI character set used in Excel. For example, if you have prepared strings for a project in Russian (using Excel), the OEM character set must support the Russian (Cyrillic) character set. The OEM character set used by Windows is primarily determined by your system setup and cannot easily be changed. This presents a problem for multi-language projects.

For example, consider a project to support Russian, French, and English. Excel is used to prepare the language dBase files. When saving information from Excel, it is translated from the respective ANSI character sets to OEM. To display this information, CitectSCADA will need to convert it from OEM back to ANSI. However, Russian requires a Cyrillic OEM character set and French and English requires a Latin OEM character set. This causes a problem, since Windows can have only one OEM character set at any given time (which cannot be changed dynamically). In this situation only one language can be correctly supported - not all three at the same time.

The only way to support multiple languages with differing character sets within one CitectSCADA project, is to ensure that the language information you store in dBase files is stored as ANSI (not OEM). The `[CtEdit]ANSItoOEM` parameter must be set to 0 (zero) to ensure no conversion takes place. The difficulty for the developer in preparing the project is in saving this information in the first place, because most applications store the language information in OEM format.

**Note:** A multi-language project is included in the samples directory on your installation CD. This project allows you to enter information into the language dBase files in ANSI format.

# Chapter 34: Using OPC Server DA2.0

---

CitectSCADA OPC Server allows you to access all data available in the CitectSCADA runtime environment through any OPC Client application (v1.0 or v2.0).

This section contains:

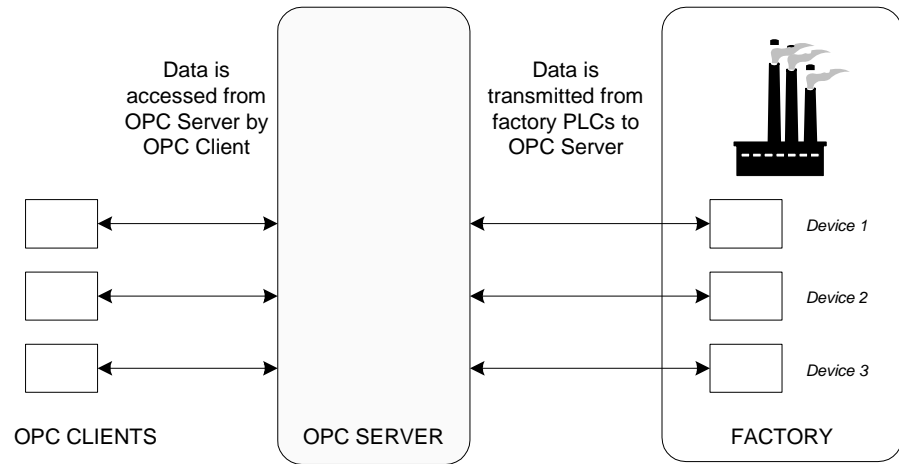
- [OPC Overview](#)
- [CitectSCADA OPC Server](#)
- [CitectSCADA OPC Server Installation](#)
- [Configuring Remote Access to the CitectSCADA OPC Server](#)
- [Troubleshooting](#)

## OPC Overview

OPC (OLE for Process Control) is a set of open standards communication specifications intended for the industrial automation industry. OPC provides a common way for applications to access data from sources such as PLCs and databases.

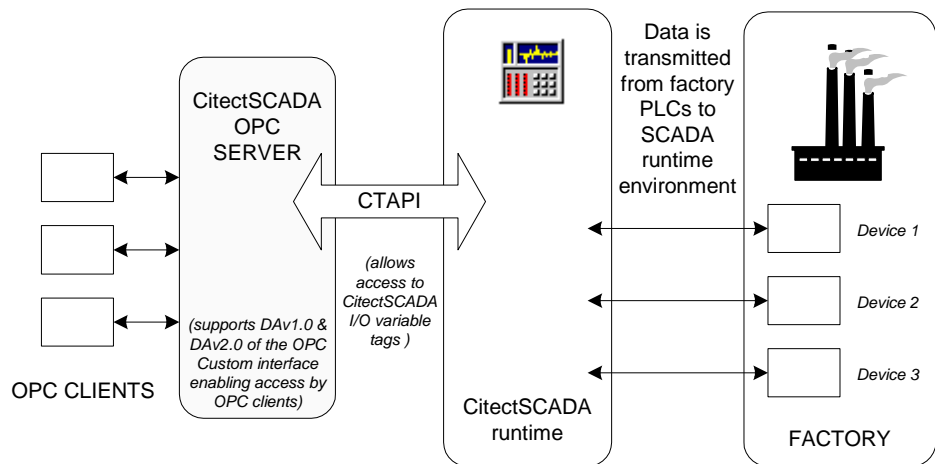
The specifications are maintained by the OPC Foundation and are reviewed constantly to ensure that they meet the needs of industry. The OPC Foundation's goal is to ensure interoperability (open connectivity) through the use of open standards. CitectSCADA runtime implements vDA2.05 of the OPC specifications. For details on the OPC specifications, navigate to [www.opcfoundation.org](http://www.opcfoundation.org).

When an OPC Server is used to collect data from physical equipment in a plant, any OPC Client can access data directly from the OPC Server, as shown below:



### CitectSCADA OPC Server

When CitectSCADA is used to monitor and control a plant, data from PLCs is collected and displayed in the CitectSCADA runtime environment. OPC Clients can access device and tag information from CitectSCADA through the OPC interface to the OPC Server, which in turn interacts with the CtAPI interface to the CitectSCADA Runtime, as shown here:



The CitectSCADA OPC Server can be invoked from an OPC Client on the same machine or on a different machine. When invoking the OPC Server on the same machine as the client, connect to the Citect.OPC option, when invoking an OPC Server remotely from the client, connect to the Citect.OPCRemote option.



The OPC Server that OPC Clients access starts automatically when you start the CitectSCADA runtime environment and does not require configuring.

## CitectSCADA OPC Server Installation

During CitectSCADA installation:

- The CitectSCADA OPC Server is installed.
- The OPC Core components are installed.
- The registry settings for the workstation are configured to support the OPC Server. Set out in the tables below are the registry settings for:
  - The In-Process OPC Server - Citect.OPC
  - The Local/Remote OPC Server - Citect.OPCRemote

### In-Process OPC Server

In-Proc OPC Server	
Application ID ( <i>AppID</i> )	Citect.OPC
Class ID ( <i>CLSID</i> )	{BA198B61-32E3-11d1-A1B5-00805F35623C}
Binary file	CtOpc32.dll
Interfaces	<i>DA 1.0 Custom Interface:</i> OPCServer: IOPCServer IOPCBrowseServerAddressSpace (optional) OPCGroup: IOPCItemMgt IOPCGroupStateMgt IOPCSynchIO IOPCASynchIO IDataObject  <i>DA 2.0 Custom Interface:</i> OPCServer: IConnectionPointContainer IOPCItemProperties OPCGroup: IOPCASynchIO2 IConnectionPointContainer
System Requirements	2000, XP and Windows Server 2003
Server Organisation	Flat (OPC_NS_FLAT)
Implemented Categories	<i>OPC DA Servers v1.0</i> {63D5F430-CFE4-11D1-B2C8-0060083BA1FB} <i>OPC DA Servers v2.0</i> {63D5F432-CFE4-11D1-B2C8-0060083BA1FB}

### Local/Remote OPC Server

Local / Remote OPC Server	
Application ID ( <i>AppID</i> )	Citect.OPCRemote
Class ID ( <i>CLSID</i> )	{BA198B62-32E3-11d1-A1B5-00805F35623C}
Binary file	CtOpc32.exe

### Local / Remote OPC Server

Interfaces	<i>DA 1.0 Custom Interface:</i>	
	OPCServer:	IOPCServer IOPCBrowseServerAddressSpace (optional)
	OPCGroup:	IOPCItemMgt IOPCGroupStateMgt IOPCSynchIO IOPCASynchIO IDataObject
	<i>DA 2.0 Custom Interface:</i>	
	OPCServer:	IOPCCommon IConnectionPointContainer IOPCItemProperties
	OPCGroup:	IOPCASynchIO2 IConnectionPointContainer
System Requirements	2000, XP and Windows Server 2003	
Server Organisation	Flat (OPC_NS_FLAT)	

**Note:** In the OPC architecture, each variety of server is allocated a unique identifier, known as a Class ID. OPC Clients use these 128-bit numbers, frequently displayed in the form 6B29FC40-CA47-1067-B31D-00DD010662DA, to access the particular vendor's OPC Server. To make it easier for users, these numbers are often referred to by a more manageable string identifier called a *ProgID (Program Identifier)*. These identifiers usually take the form **Vendor.Application** with an optional version number appended; for example, the CitectSCADA OPC Server ProgID is **Citect.OPC**.

Once successfully installed, the CitectSCADA OPC Server is ready for use. The OPC interface that OPC Clients access starts automatically when you start the CitectSCADA runtime environment. Further configuration is only required where the CitectSCADA OPC Server will be accessed by a remote OPC Client. See [Configuring Remote Access to the CitectSCADA OPC Server](#) for more information.

## Configuring Remote Access to the CitectSCADA OPC Server

You can use any OPC Client to access CitectSCADA OPC Server. While each OPC Client has its own interface and therefore operates differently, the same steps must be performed in order to access data exposed through the OPC Server:

- [Configure the OPC Server](#)
- [Configure the OPC Client](#)
- [Create a data group](#)
- [Add data items to the group](#)

## Configure the OPC Server

For details on how to operate your OPC Client, refer to the product's documentation.

**Note:** No configuration is needed if the client and server are on the same machine.

OPC operates using the Microsoft Distributed COM (DCOM) architecture. This means that if an OPC Client is connecting to a remote OPC Server, you must configure DCOM security settings on both the client and server machines.

**Note:** Failure to give users sufficient privileges results in communications failure and the OPC Client will not connect to the OPC Server.

The process of configuring DCOM security settings differs depending on which platform you're using:

- [Windows XP](#)
- [Windows XP SP2](#)
- [Windows 2000](#)
- [Windows Server 2003](#)

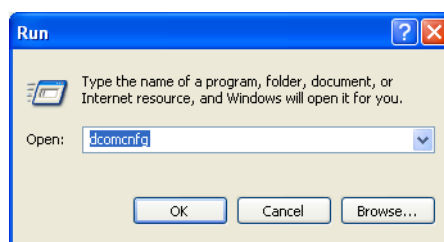
### Windows XP

This section describes how to configure the OPC Server on Windows XP.

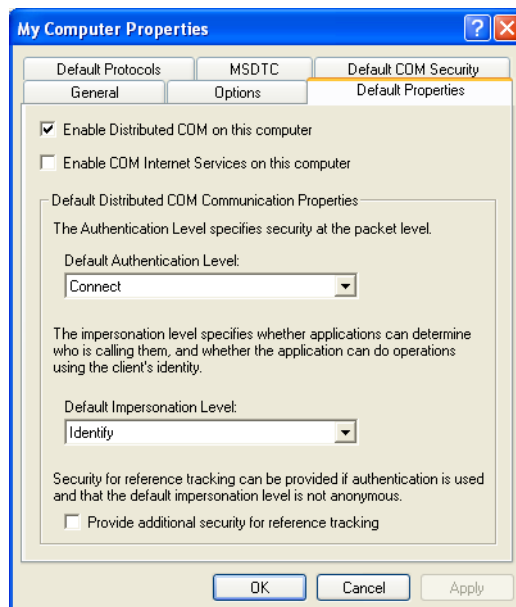
**Note:** After configuring your OPC Server, you must restart the server so that your settings can take effect.

#### To configure the OPC Server:

- 1 Choose **Start | Run**. In the Run dialog, type **dcomcnfg** and click **OK**.

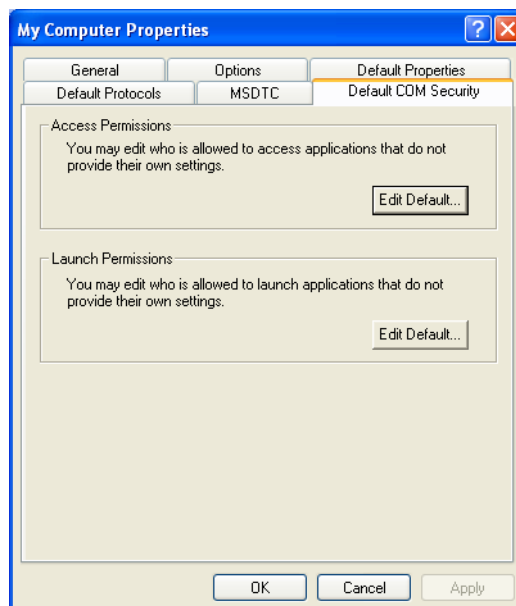


- 2 In the tree pane, click **Component Services** under the **Console Root**.
- 3 In the tree pane, click **Computers** under **Component Services**.
- 4 Right-click **My Computer** in the pane on the right and choose **Properties** from the context menu.
- 5 Select the **Default Properties** tab and make sure the settings appear as below. Click **OK** to save your settings.

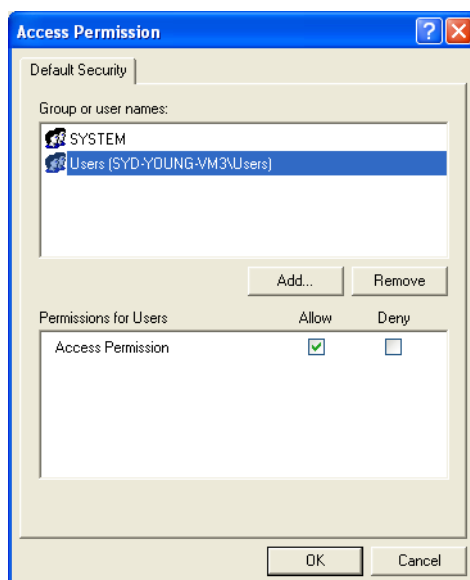


You must edit the **Access Permissions** and **Launch and Activation Permissions** settings.

- 6 Select the **Default COM Security** tab, and then click **Edit Default** in the **Access Permissions** section.



- 7 In the **Access Permission** dialog, select the user you want to edit permissions for, and then select the **Allow Access Permission** check box, as shown here.

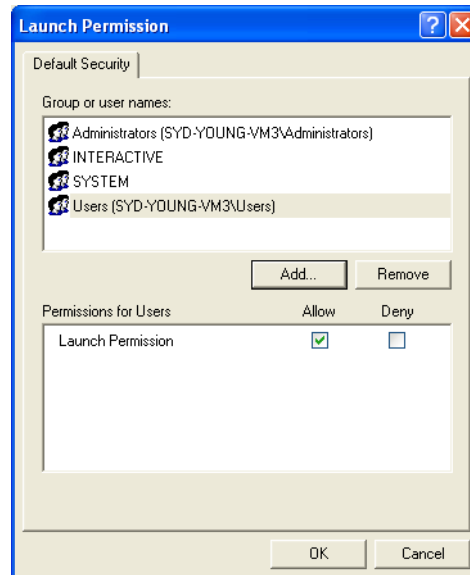


This setting is needed for OPCEnum.exe to function, and for some OPC Servers and clients that set their DCOM authentication level to None in order to allow anonymous connections. If you don't use OPCEnum.exe you might not need to enable remote access for anonymous users.

Click **OK** to save your changes.

Now you must edit the default launch permissions.

- 8 Click **Edit Default** in the **Launch Permissions** section. Make sure the settings are as shown here.



Click **OK** to save your changes.

You have now configured your OPC Server.

**Note:** You *must* restart this machine so that your settings can take effect.

You are now ready to [configure your OPC Client](#).

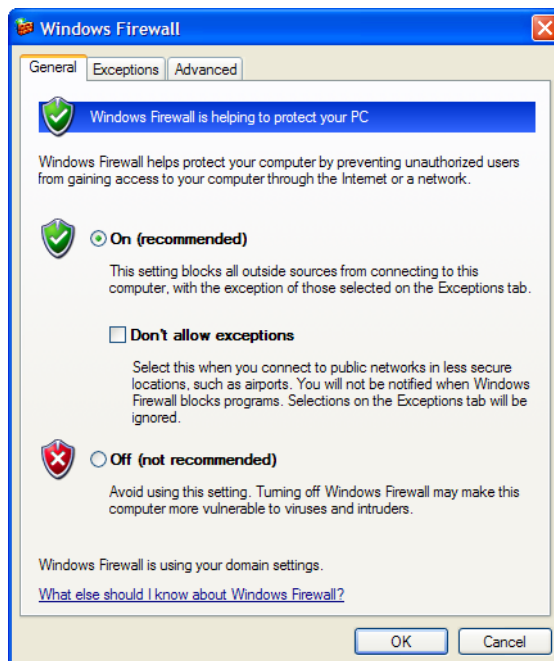
#### Windows XP SP2

This section describes how to configure the OPC Server on Windows XP Service Pack 2.

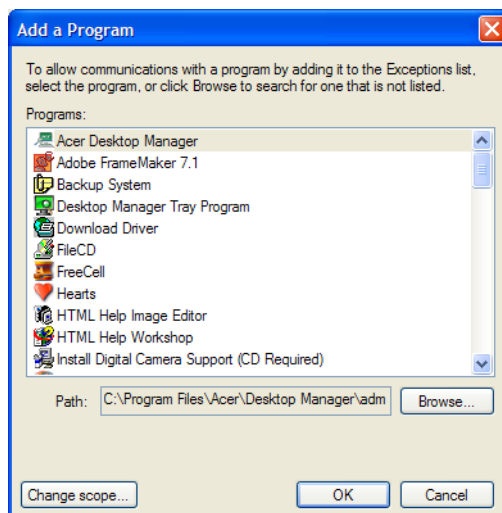
**Note:** After configuring your OPC Server, you must restart the server so that your settings can take effect.

#### To configure the OPC Server:

- 1 In Windows Control Panel, double-click **Security Center**, and then click **Windows FireWall**.



- 2 Select the **Exceptions** tab, and then click **Add Program**.



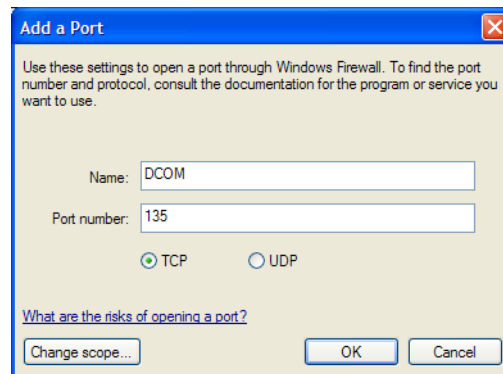
- 3 Locate the OPC Servers and clients you want to add as exceptions. You must add the **Microsoft Management Console** (used by the DCOM configuration utility, below) and the OPC utility **OPCEnum.exe** in the `Windows/System32` folder.

You might have to **Browse** for other executables installed on the computer to add them as exceptions. Note that only .exe files are added to the exceptions list. For in-process OPC Servers and clients (DLLs and OCXs) you must add the .exe applications that call them to the list instead.

- 4 Click **OK** to save your exceptions.

You must now add TCP port 135 to initiate DCOM communications and allow incoming echo requests.

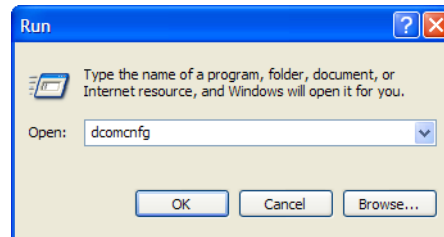
- 5 Click **Add Port**.



- 6 In the **Name** text box type **DCOM**.
- 7 In the **Port number** text box type **135**, select the **TCP** option, and then click **OK** to save your changes.

Now you have defined your exceptions. You can restart your firewall. You are now ready to configure DCOM for your launch, activation, and access permissions.

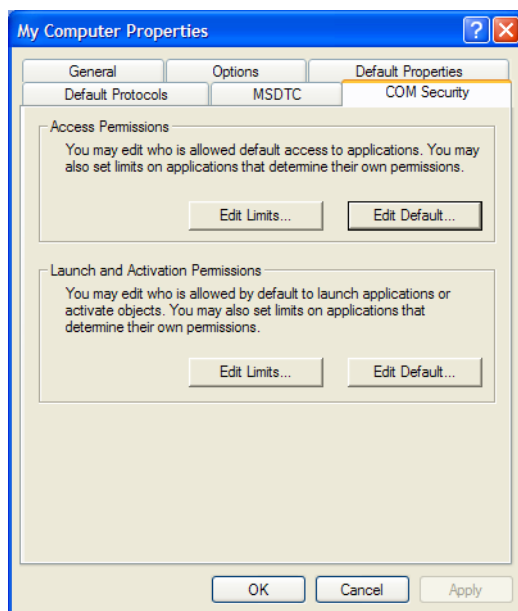
- 8 Choose **Start | Run**. In the Run dialog, type **dcomcnfg** and click **OK**.



- 9 In the tree pane, click **Component Services** under the **Console Root**.
- 10 In the tree pane, click **Computers** under **Component Services**.
- 11 Right-click **My Computer** in the pane on the right and choose **Properties** from the context menu.

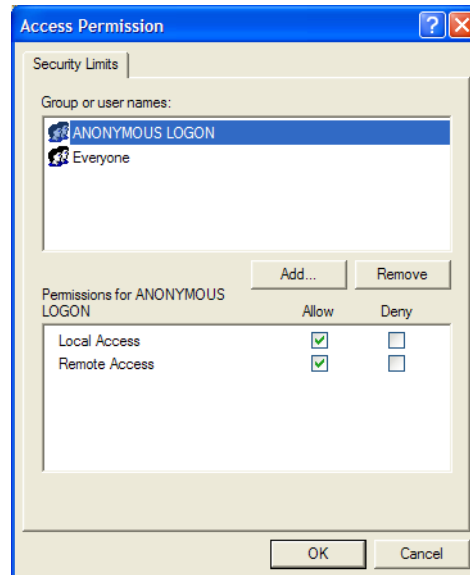


- 12 Select the **COM Security** tab.



You must edit the **Access Permissions** and **Launch and Activation Permissions** settings.

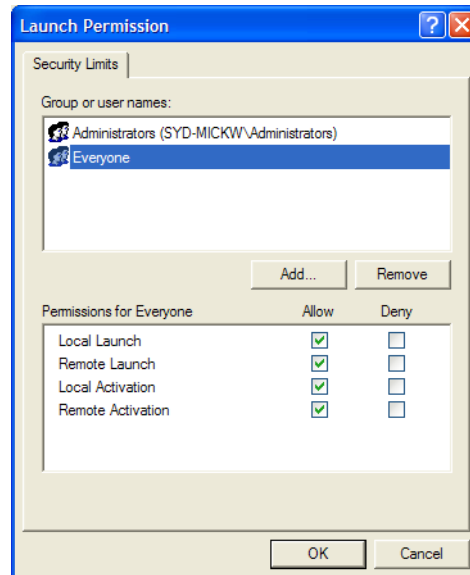
- 13 Click **Edit Limits** in the **Access Permissions** section. In the **Group or user names** list, select **ANONYMOUS LOGON** and then select the **Remote Access Allow** check box, as shown here.



This setting is needed for `OPCEnum.exe` to function, and for some OPC Servers and clients that set their DCOM authentication level to None in order to allow anonymous connections. If you don't use `OPCEnum.exe` you might not need to enable remote access for anonymous users.

Click **OK** to save your changes.

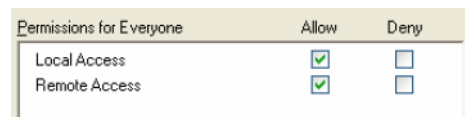
- 14 Click **Edit Limits** in the **Launch and Activation Permissions** section. In the **Group or user names** list, select **Everyone** and then select all the check boxes in the **Allow** column, as shown here.



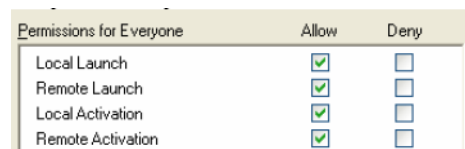
Now you're ready to edit the default permissions for **Access** and **Launch**. For each user (or group) that participates in OPC communication (for example, "OPC Users"), make sure that both the **Local Allow** and **Remote Allow** check boxes are selected.

Click **OK** to save your changes.

- 15 In the **Access Permissions** section, click **Edit Default** and configure the settings as shown here. Click **OK** to save your changes.



- 16 In the **Launch and Activation Permissions** section, click **Edit Default** and configure the settings as shown here. Click **OK** to save your changes.



You have now configured your OPC Server.

**Note:** You must restart this machine so that your settings can take effect.

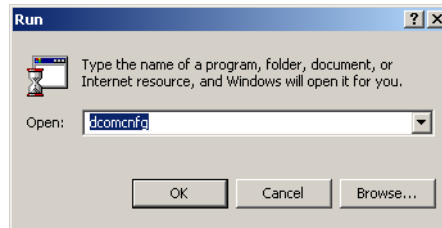
You are now ready to [configure your OPC Client](#).

## Windows 2000

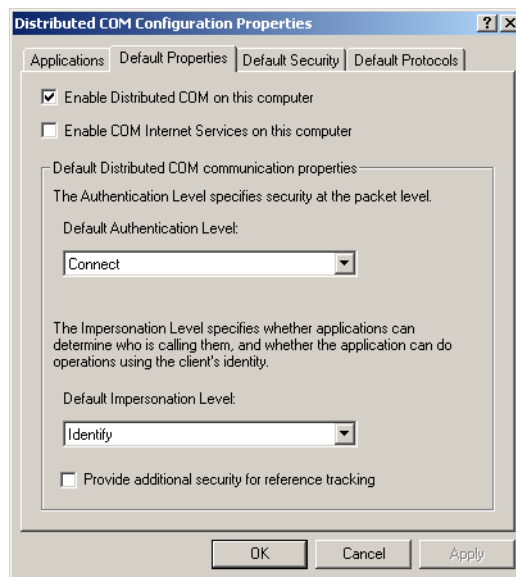
This section describes how to configure the OPC Server on Windows 2000.

### To configure the OPC Server:

- 1 Choose **Start | Run**. In the Run dialog, type **dcomcnfg** and click **OK**.

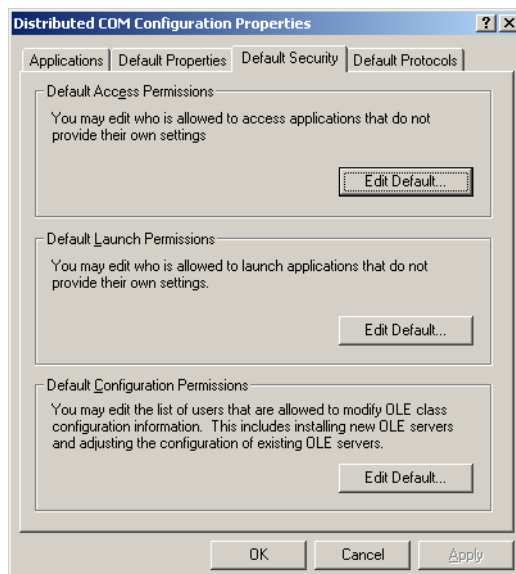


- 2 In the tree pane, click **Component Services** under the **Console Root**.
- 3 In the tree pane, click **Computers** under **Component Services**.
- 4 Right-click **My Computer** in the pane on the right and choose **Properties** from the context menu.
- 5 Select the **Default Properties** tab and make sure the settings appear as below. Click **OK** to save your settings.

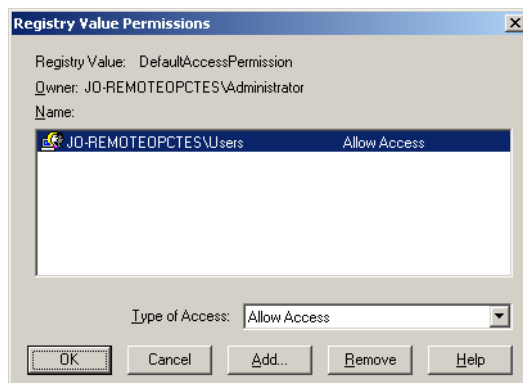


You must edit the **Access Permissions** and **Launch and Activation Permissions** settings.

- 6 Select the **Default Security** tab, and then click **Edit Default** in the **Default Access Permissions** section.



- 7 In the **Registry Value Permissions** dialog, select the user you want to edit permissions for and then choose **Allow Access** from the **Type of Access** menu, as shown here.

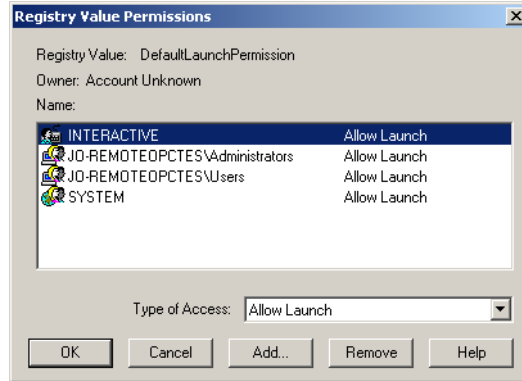


This setting is needed for `OPCEnum.exe` to function, and for some OPC Servers and clients that set their DCOM authentication level to None in order to allow anonymous connections. If you don't use `OPCEnum.exe` you might not need to enable remote access for anonymous users.

Click **OK** to save your changes.

Now you must edit the default launch permissions.

- 8 Click **Edit Default** in the **Default Launch Permissions** section. Make sure the settings are as shown here.



Click **OK** to save your changes.

You have now configured your OPC Server.

**Note:** You *must* restart this machine so that your settings can take effect.

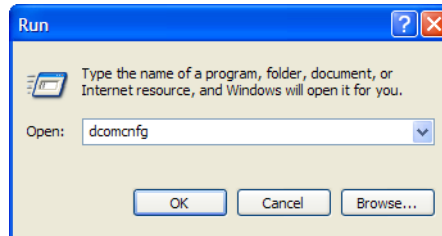
You are now ready to [configure your OPC Client](#).

#### Windows Server 2003

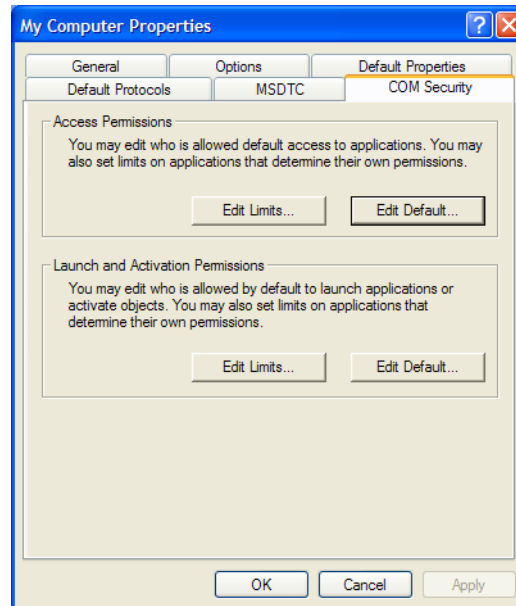
This section describes how to configure the OPC Server on Windows Server 2003.

#### To configure the OPC Server:

- 1 Choose **Start | Run**. In the Run dialog, type **DCOMCnfg** and click **OK**.

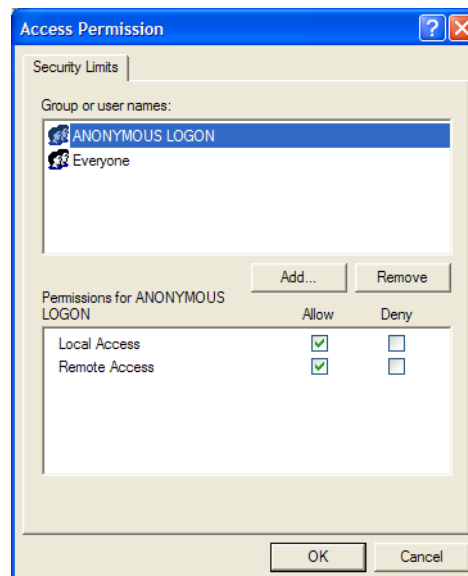


- 2 Click **Component Services** under the Console Root to expand it.
- 3 Click **Computers** under Component Services.
- 4 Right-click **My Computer** in the pane on the right and choose **Properties** from the menu.
- 5 Select the **COM Security** tab.



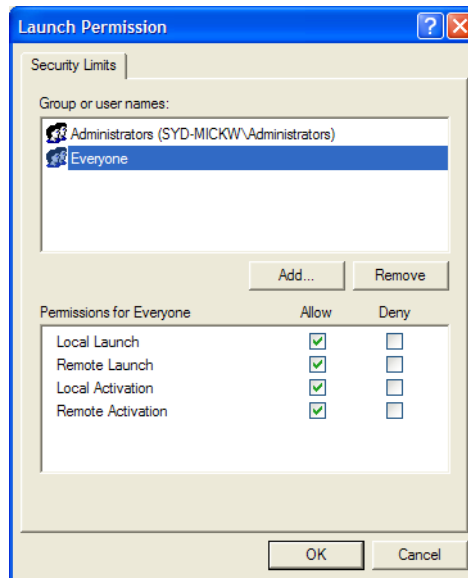
You must edit the **Access Permissions** and **Launch and Activation Permissions** settings.

- 6 Click **Edit Limits** in the **Access Permissions** section. In the **Group or user names** list, select **ANONYMOUS LOGON** and then select the **Remote Access Allow** check box, as shown here.



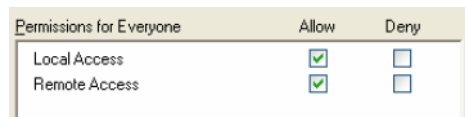
This setting is needed for OPCEnum.exe to function, and for some OPC Servers and clients that set their DCOM authentication level to None in order to allow anonymous connections. If you don't use OPCEnum.exe you might not need to enable remote access for anonymous users.

- 7 Click **Edit Limits** in the **Launch and Activation Permissions** section. In the **Group or user names** list, select **Everyone** and then select all check boxes in the **Allow** column, as shown here.



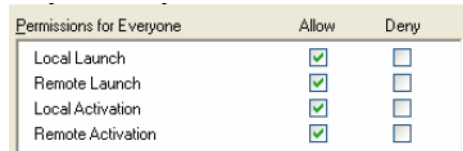
Now you're ready to edit the default permissions for **Access** and **Launch**. For each user (or group) that participates in OPC communication (for example, "OPC Users"), make sure that both the **Local Allow** and **Remote Allow** check boxes are selected.

- 8 In the **Access Permissions** section, click **Edit Default** and configure the settings as shown here:



- 9 In the **Launch and Activation Permissions** section, click **Edit Default** and configure the settings as shown here:





You have now configured your OPC Server.

**Note:** Now you *must* restart this machine so that your settings can take effect.

You are now ready to [configure your OPC Client](#).

## Configure the OPC Client

You must configure the OPC Client before it can communicate with the OPC Server. Configuration is different depending on which platform you're using:

- [Windows XP](#)
- [Windows XP SP2](#)
- [Windows 2000](#)
- [Windows Server 2003](#)

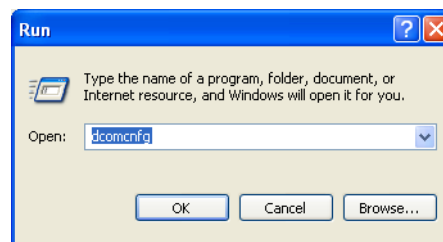
### Windows XP

This section describes how to configure the OPC Client on Windows XP.

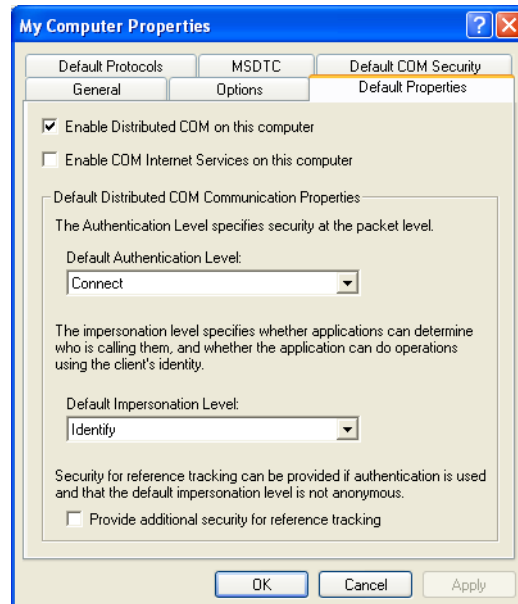
**Note:** After configuring your OPC Client, you must restart the client so that your settings can take effect.

#### To configure the OPC Client:

- 1 Choose **Start | Run**. In the Run dialog, type **dcomcnfg** and click **OK**.

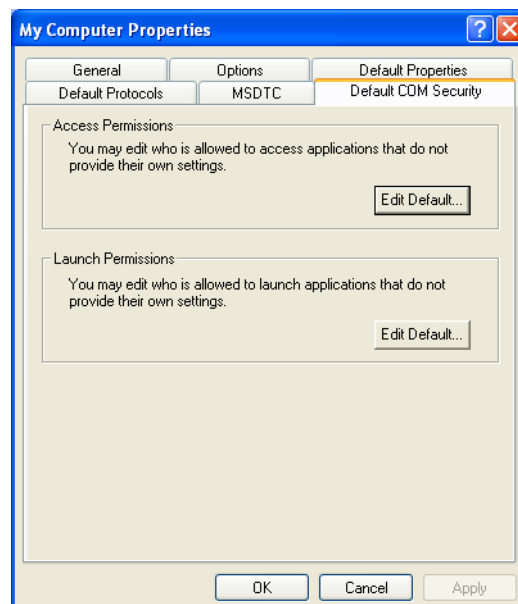


- 2 In the tree pane, click **Component Services** under the **Console Root**.
- 3 In the tree pane, click **Computers** under **Component Services**.
- 4 Right-click **My Computer** in the pane on the right and choose **Properties** from the context menu.
- 5 Select the **Default Properties** tab and make sure the settings appear as below. Click **OK** to save your settings.

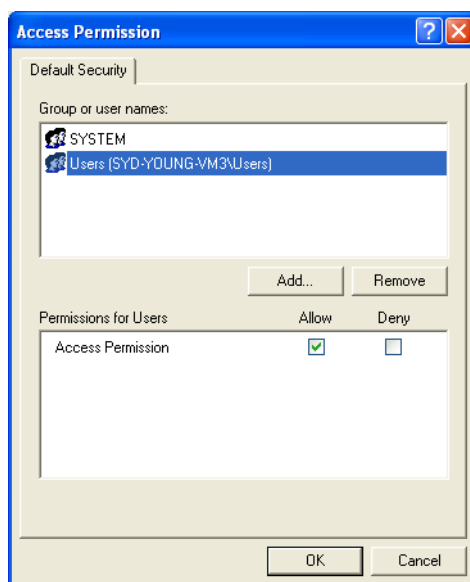


You must edit the **Access Permissions** and **Launch and Activation Permissions** settings.

- 6 Select the **Default COM Security** tab, and then click **Edit Default** in the **Access Permissions** section.



- 7 In the **Access Permission** dialog, select the user you want to edit permissions for and then select the **Allow Access Permission** check box, as shown here.

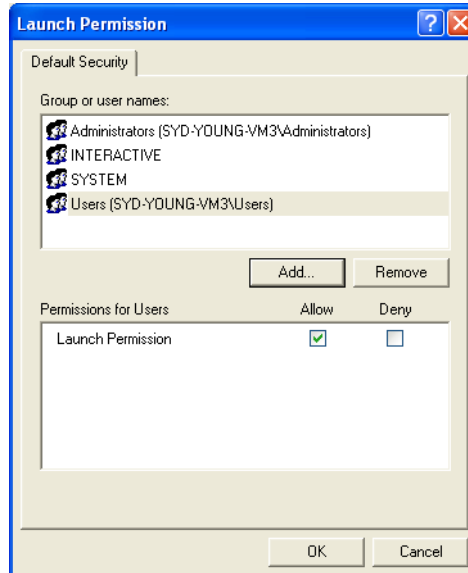


This setting is needed for `OPCEnum.exe` to function, and for some OPC Servers and clients that set their DCOM authentication level to None in order to allow anonymous connections. If you don't use `OPCEnum.exe` you might not need to enable remote access for anonymous users.

Click **OK** to save your changes.

Now you must edit the default launch permissions.

- 8 Click **Edit Default** in the **Launch Permissions** section. Make sure the settings are as shown here.



Click **OK** to save your changes.

You have now configured your OPC Client.

Some older OPC Clients cannot locate CitectSCADA OPC Server on a client machine. If this is the case, copy the `ctopc.reg` file located in the `bin` directory to the client machine. Double-click the file to add OPC Server-specific data to the registry.

**Note:** After configuring your OPC Client, you must restart this machine for your changes to take effect.

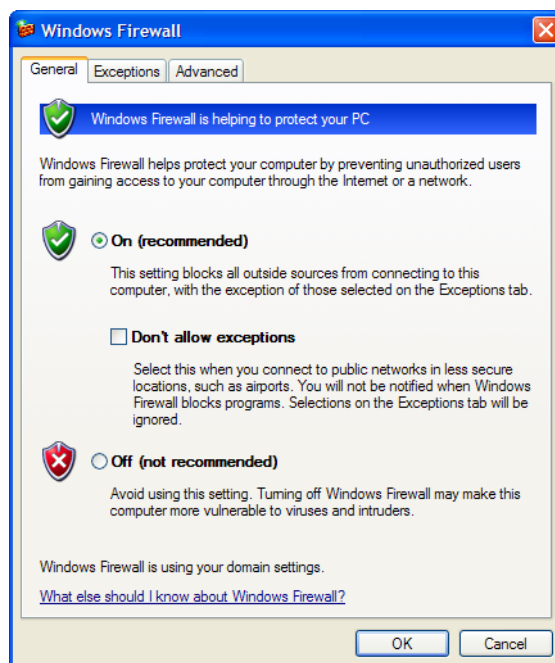
#### Windows XP SP2

This section describes how to configure the OPC Client on Windows XP SP2.

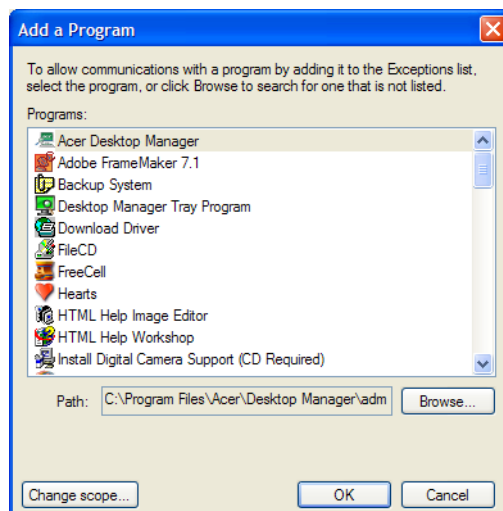
**Note:** After configuring your OPC Client, you must restart the machine so that your settings can take effect.

#### To configure the OPC Client:

- 1 In Control Panel, double-click **Security Center**, and then click **Windows FireWall**.



- 2 Select the **Exceptions** tab, and then click **Add Program**.



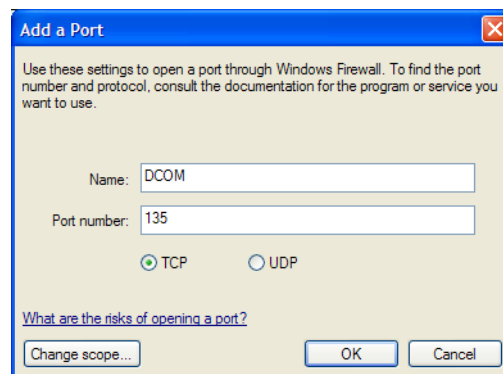
- 3 Locate to the OPC Clients and servers you want to add as an exception. You must add the **Microsoft Management Console** (used by the DCOM configuration utility, below) and the OPC utility **OPCEnum.exe** in the Windows/System32 folder.

You might have to **Browse** for other executables installed on the computer to add them as exceptions. Note that only .exe files are added to the exceptions list. For in-process OPC Servers and clients (DLLs and OCXs) you must add the .exe applications that call them to the list instead.

- 4 Click **OK** to save your exceptions.

You must now add TCP port 135 to initiate DCOM communications and allow incoming echo requests.

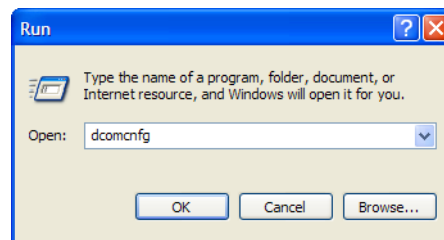
- 5 Click **Add Port**.



- 6 In the **Name** text box type **DCOM**.
- 7 In the **Port number** text box type **135**, select the **TCP** option, and then click **OK** to save your changes.

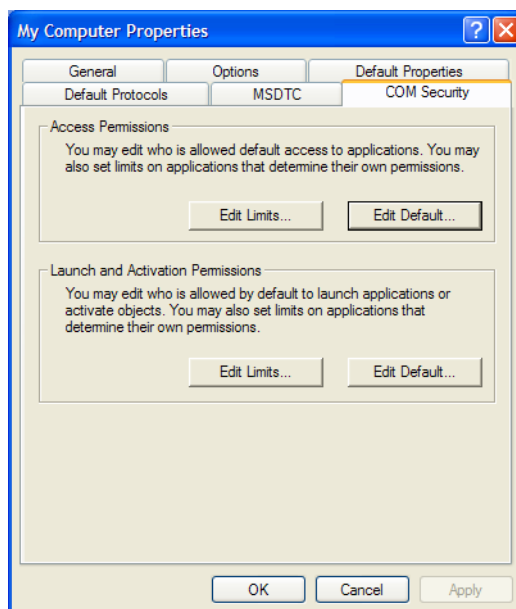
Now you have defined your exceptions. You can restart your firewall. You are now ready to configure DCOM for your launch, activation, and access permissions.

- 8 Choose **Start | Run**. In the Run dialog, type **DCOMCnfg** and click **OK**.



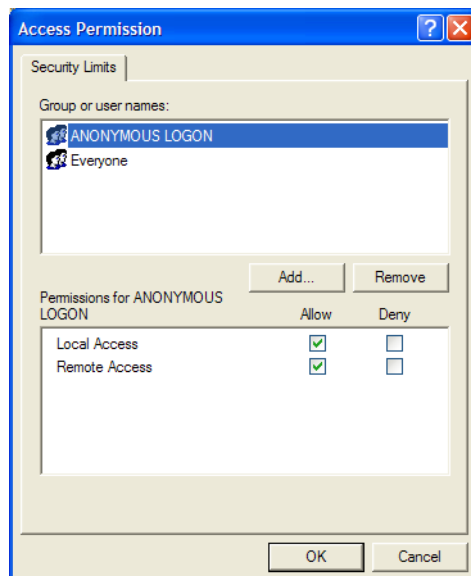
- 9 Click **Component Services** under the Console Root to expand it.
- 10 Click **Computers** under Component Services.
- 11 Right-click **My Computer** in the pane on the right and choose **Properties** from the menu.

- 12 Select the **COM Security** tab.



You must edit the **Access Permissions** and **Launch and Activation Permissions** settings.

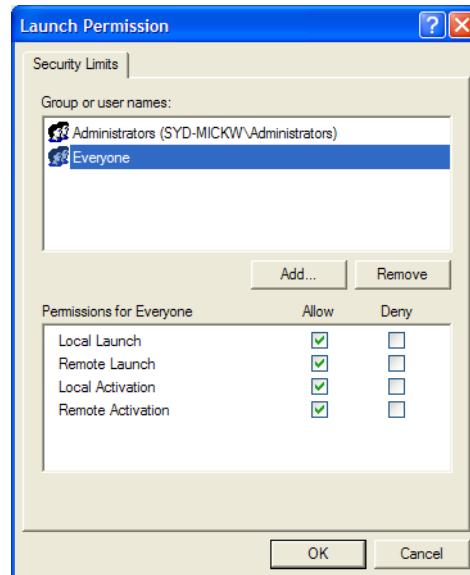
- 13 Click **Edit Limits** in the **Access Permissions** section. In the **Group or user names** list, select **ANONYMOUS LOGON** and then select the **Remote Access Allow** check box, as shown here.



This setting is needed for `OPCEnum.exe` to function, and for some OPC Servers and clients that set their DCOM authentication level to None in order to allow anonymous connections. If you don't use `OPCEnum.exe` you might not need to enable remote access for anonymous users.

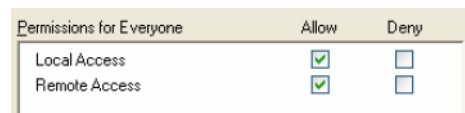
- 14 Click **Edit Limits** in the **Launch and Activation Permissions** section. In the **Group or user names** list, select **Everyone** and then select all check boxes in the **Allow** column, as shown here.



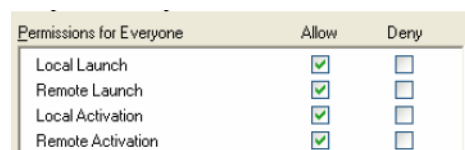


Now you're ready to edit the default permissions for **Access** and **Launch**. For each user (or group) that participates in OPC communication (for example, "OPC Users"), make sure that both the **Local Allow** and **Remote Allow** check boxes are selected.

- 15 In the **Access Permissions** section, click **Edit Default** and configure the settings as shown here:



- 16 In the **Launch and Activation Permissions** section, click **Edit Default** and configure the settings as shown here:



You have now configured your OPC Client.

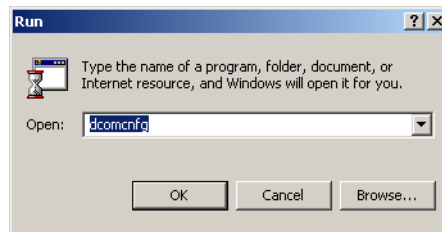
Some older OPC Clients cannot locate CitectSCADA OPC Server on a client machine. If this is the case, copy the `ctopc.reg` file located in the `bin` directory to the client machine. Double-click the file to add OPC Server-specific data to the registry.

**Note:** After configuring your OPC Client, you must restart this machine for your changes to take effect.

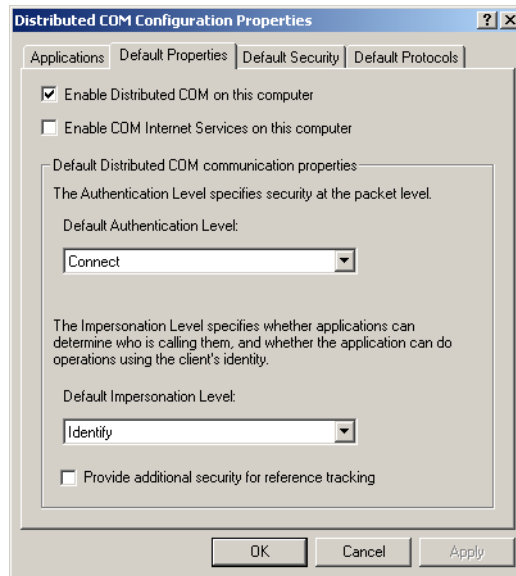
#### Windows 2000

This section describes how to configure the OPC Client on Windows 2000.

- 1 Choose **Start | Run**. In the Run dialog, type **dcomcnfg** and click **OK**.

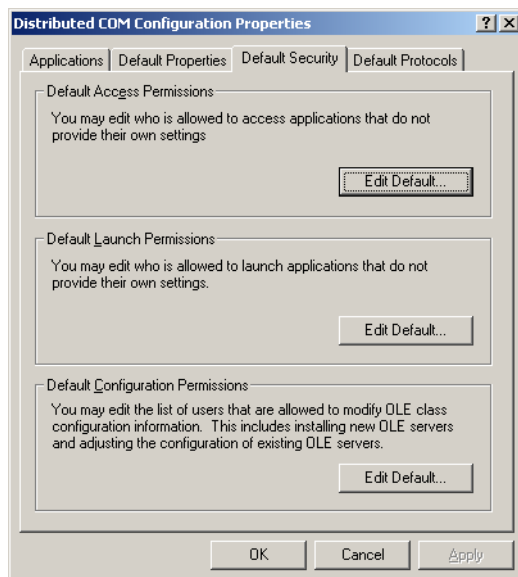


- 2 In the tree pane, click **Component Services** under the **Console Root**.
- 3 In the tree pane, click **Computers** under **Component Services**.
- 4 Right-click **My Computer** in the pane on the right and choose **Properties** from the context menu.
- 5 Select the **Default Properties** tab and make sure the settings appear as below. Click **OK** to save your settings.

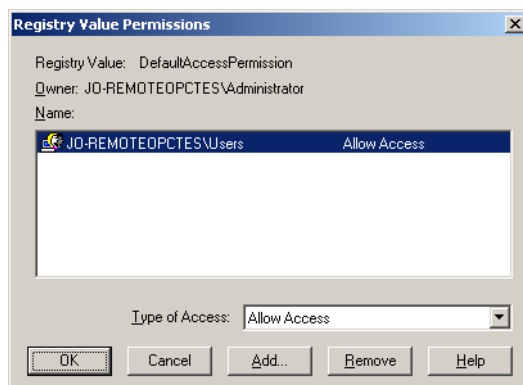


You must edit the **Access Permissions** and **Launch and Activation Permissions** settings.

- 6 Select the **Default Security** tab, and then click **Edit Default** in the **Default Access Permissions** section.



- 7 In the **Registry Value Permissions** dialog, select the user you want to edit permissions for and then choose **Allow Access** from the **Type of Access** menu, as shown here.

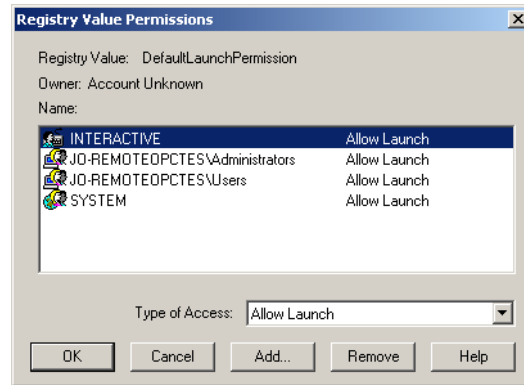


This setting is needed for `OPCEnum.exe` to function, and for some OPC Servers and clients that set their DCOM authentication level to None in order to allow anonymous connections. If you don't use `OPCEnum.exe` you might not need to enable remote access for anonymous users.

Click **OK** to save your changes.

Now you must edit the default launch permissions.

- 8 Click **Edit Default** in the **Default Launch Permissions** section. Make sure the settings are as shown here.



Click **OK** to save your changes.

You have now configured your OPC Client.

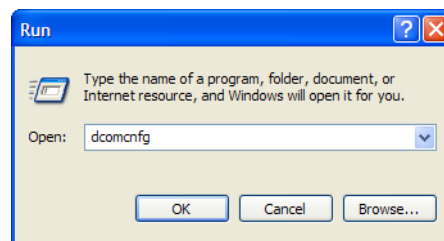
Some older OPC Clients cannot locate CitectSCADA OPC Server on a client machine. If this is the case, copy the `ctopc.reg` file located in the `bin` directory to the client machine. Double-click the file to add OPC Server-specific data to the registry.

**Note:** After configuring your OPC Client, you must restart this machine for your changes to take effect.

#### Windows Server 2003

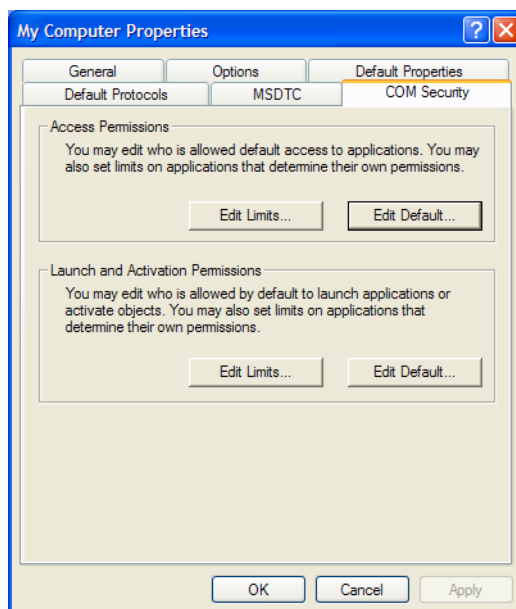
This section describes how to configure the OPC Client on Windows Server 2003.

- 1 Choose **Start | Run**. In the Run dialog, type **DCOMCnfg** and click **OK**.



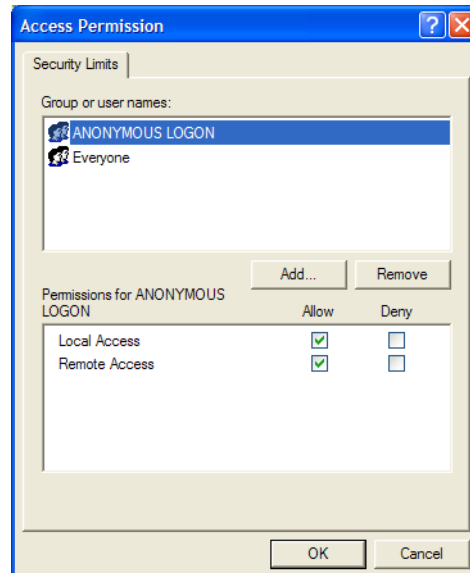
- 2 Click **Component Services** under the Console Root to expand it.
- 3 Click **Computers** under Component Services.
- 4 Right-click **My Computer** in the pane on the right and choose **Properties** from the menu.

- 5 Select the **COM Security** tab.



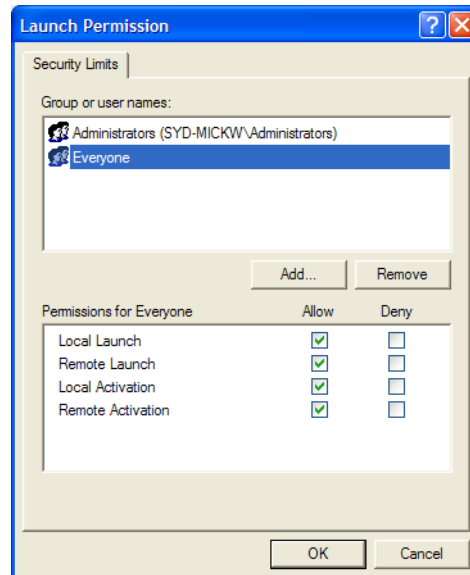
You must edit the **Access Permissions** and **Launch and Activation Permissions** settings.

- 6 Click **Edit Limits** in the **Access Permissions** section. In the **Group or user names** list, select **ANONYMOUS LOGON** and then select the **Remote Access Allow** check box, as shown here.



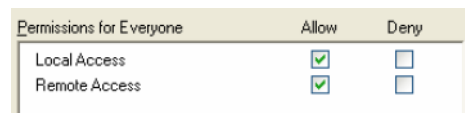
This setting is needed for `OPCEnum.exe` to function, and for some OPC Servers and clients that set their DCOM authentication level to None in order to allow anonymous connections. If you don't use `OPCEnum.exe` you might not need to enable remote access for anonymous users.

- 7 Click **Edit Limits** in the **Launch and Activation Permissions** section. In the **Group or user names** list, select **Everyone** and then select all check boxes in the **Allow** column, as shown here.

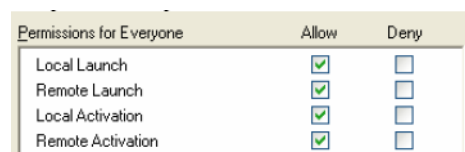


Now you're ready to edit the default permissions for **Access** and **Launch**. For each user (or group) that participates in OPC communication (for example, "OPC Users"), make sure that both the **Local Allow** and **Remote Allow** check boxes are selected.

- 8 In the **Access Permissions** section, click **Edit Default** and configure the settings as shown here:



- 9 In the **Launch and Activation Permissions** section, click **Edit Default** and configure the settings as shown here:



You have now configured your OPC Client.

Some older OPC Clients cannot locate CitectSCADA OPC Server on a client machine. If this is the case, copy the `ctopc.reg` file located in the `bin` directory to the client machine. Double-click the file to add OPC Server-specific data to the registry.

**Note:** After configuring your OPC Client, you must restart this machine for your changes to take effect.

## Create a data group

Before you can access data from an OPC Server, you need to define the data group to which it belongs. A data group is a way to organize data retrieved from the OPC Server into logical selections of related data.

**Note:** OPC Clients usually allow properties to be set on a per group basis, such as the rate at which the data is collected. You should therefore consider arranging and naming your groups according to these requirements.

As this procedure is OPC Client-specific, you should refer to the documentation accompanying your OPC Client.

## Add data items to the group

Within each group you can assign one or more data items. Each variable tag defined in CitectSCADA runtime is available for selection as a data item.

As this procedure is OPC Client-specific, you should refer to the documentation accompanying your OPC Client.

## Troubleshooting

If you experience connectivity issues with your OPC Server, OPC Client, or both, refer to the CitectSCADA Knowledge Base.



# Chapter 35: Communicating with I/O Devices

---

CitectSCADA can communicate with any control or monitoring I/O device that has a communication port or data highway, including PLCs (programmable logic controllers), loop controllers, bar code readers, scientific analyzers, remote terminal units (RTUs), and distributed control systems (DCS).

I/O devices can be easily classified into two distinct categories for their communication connection method with CitectSCADA: local or remote.

- **Local:** I/O devices are directly connected to a CitectSCADA [I/O server](#).
- **Remote:** I/O devices are connected to CitectSCADA via an intermediate communications means (radio link, modem and phone line, and so on).

Both of these types can be configured to be permanent, periodic, or on request.

## Communication types

CitectSCADA supports four types of I/O device communication:

- [serial communication](#)
- [PLC interface board](#)
- [data acquisition board](#)
- [dynamic data exchange \(DDE\) Server](#)

Whether the I/O device is local or remote, communicating with I/O devices is usually via a simple serial connection using the RS-232, RS-422, or RS-485 standard.

With CitectSCADA there are many I/O device communications options, through the CitectSCADA computer's COM port, through a high-speed serial board, or through a communications board supplied by the I/O device manufacturer. Whichever option you choose, use the Express I/O Device Setup wizard.

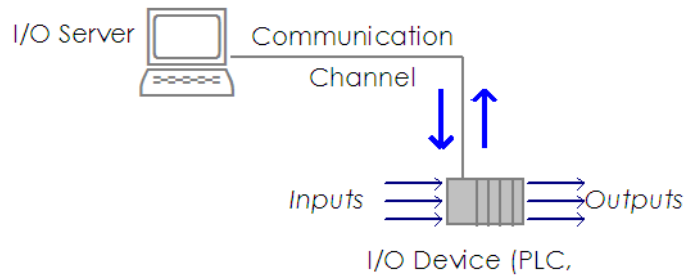
**Note:** Most I/O device communications standards are serial-based. For clarity however, only simple serial communications are considered here. More complex serial communications, such as [ethernet](#), are detailed where appropriate.

See Also [How CitectSCADA Communicates with I/O Devices](#)

## How CitectSCADA Communicates with I/O Devices

CitectSCADA communicates directly with the I/O device(s) in your plant or factory. This system has three major components:

- CitectSCADA computer (I/O server)
- Communications channel
- I/O device



To enable CitectSCADA to communicate with an I/O device, you need a device driver. This is the interface between CitectSCADA and the I/O device which implements the communication [protocol\(s\)](#) of the I/O device.

CitectSCADA device drivers allow a single driver to communicate using several hardware boards, each with several hardware ports, with each port communicating with several I/O devices.

Inputs to the I/O device provide information about your plant, such as the location of a product, speed of a machine, status of a drive, or temperature of an oven. Outputs from the I/O device usually perform the tasks required to operate your plant, such as starting electric motors or varying their speed, or switching valves and indication lamps. In some I/O devices (such as PLCs), a program stored in the I/O device controls the outputs. The logic (control strategy) of this stored program and the status of the inputs determine the value of each output.

The value of each input and output is stored in a separate memory register in the I/O device. Each memory register is referenced by its address.

Most I/O devices provide a communication port or data highway for communicating with other devices or computers. By using this communication pathway, CitectSCADA can read and write to the memory registers in the I/O device.

By reading and writing to memory registers in all your I/O devices, CitectSCADA collects data from your plant or factory for monitoring and analysis, and provides high-level (supervisory) control of your equipment and processes.

You do not usually need to read (or write) to all registers in the I/O device: CitectSCADA lets you specify which inputs and outputs you want to monitor or control. After defining these register addresses, you can use them for system control, operator displays, trend analysis, data logging and alarm indication.

**Note:** I/O devices such as programmable logic controllers (PLCs) usually have an internal program that controls the low-level processes within your plant. A PLC program continually scans the input registers of the PLC, and sets the output registers to values determined by the PLC program logic. While CitectSCADA can replace any PLC program, this is not recommended. PLCs are designed for high-speed response (typically 1 to 100ms) and replacing this functionality with CitectSCADA could negatively affect your control system's performance. Only use CitectSCADA to complement your PLC program (that is, for high level control and system monitoring).

See Also [How CitectSCADA reads from the I/O device](#)

## How CitectSCADA reads from the I/O device

The computer directly connected to the I/O device is the I/O server for that I/O device. The I/O server keeps up-to-date information on all its I/O devices by regularly retrieving data from each I/O device and storing it in a [cache \(I/O device data cache\)](#). Then, whenever a CitectSCADA client ([display client](#), trends server, [reports server](#), and so on) requires data from an I/O device, the I/O server uses the information stored in the cache to provide the requested data.

**Note:** CitectSCADA uses the computer directly connected to the I/O device as the CitectSCADA I/O server for that I/O device. This server might have one or more I/O devices connected to it. For any CitectSCADA client (display client, trends server, reports server, and so on) to request data (status) from any I/O device, the client requests the data from the I/O server connected to the I/O device, rather than directly from the I/O device. The server retrieves the data from the I/O device and stores it.

Before starting a task, CitectSCADA reads all the required data from the I/O device (that is, any data needed to complete a required Cicode task or process). For example, when you schedule a report, CitectSCADA reads all I/O device data that the report might need, and any data that a Cicode function (called by the report) might need before the first line of the report starts running.

This requirement might have side effects you should allow for. Firstly, as CitectSCADA must read the I/O device when you schedule a report, for example, there will be a short delay before the Cicode executes, until the read data (associated with it) is read from the I/O device. For example, if you have a keyboard command as follows:

Key Seq:	ENTER
Command:	Prompt("Value of PLC_Var" + PLC_VAR: ####);
Privilege:	
Comment:	Display value of PLC_VAR from the PLC

CitectSCADA first asks the I/O server to read the value of PLC\_VAR from the PLC. It will then perform another task while waiting for the PLC to send the data back. When the data is returned, it will execute the Cicode. Because of this lag, you might have time to initiate another task before this one completes. This

effect could cause problems if the data was to be displayed at, say, AN 50, and while waiting for the data you changed pages. Then the value of PLC\_VAR would be displayed on the new page. This is normally only a problem with complex Cicode operations. If there is no data to read from the I/O device, the Cicode or process executes immediately.

If CitectSCADA cannot read the data from the I/O device (for example, the I/O device is offline or the data is bad), CitectSCADA still starts the report, and generates a hardware error. For example, #COM will be displayed if the variable is read from a text object returning its numeric value, but you should test for errors if it will impact your Cicode. If you call the function ErrCom(), it will tell you if all the I/O device variables associated with your report or Cicode are OK. So you could do the following in a report:

```
{CICODE}
IF ErrCom() <> 0 Then
  PrintLn("This Report contains bad data");
END
{END}
```

Sometimes CitectSCADA will not read the I/O device before starting a piece of Cicode. This includes callback events, Alarm ON/OFF action and any Cicode task created with TaskNew() without using mode 4. Under these conditions, CitectSCADA wants to start the Cicode immediately, and as it is a critical operation, CitectSCADA will not read the I/O device first. Instead, it will retrieve the data from the local copy of the variable(s). Be aware that this data might be stale. If you require I/O device data under these conditions, create a new task using TaskNew(mode 4).

See Also [How CitectSCADA writes to the I/O device](#)

## How CitectSCADA writes to the I/O device

CitectSCADA performs writes to the I/O device asynchronously (that is, when you write to the I/O device, the write takes time to get to the I/O device, during which CitectSCADA continue to perform other operations). If the other Cicode assumes that the write has completed immediately, you might encounter some side effects.

If you have the following Cicode:

```
PLC_VAR = 1234;
Prompt("Variable is " + PLC_VAR : #####);
```

the first line is a write to the PLC.

When CitectSCADA executes the first line, it generates a request to the I/O server to write the value 1234 into the PLC variable PLC\_VAR. CitectSCADA then executes the next line of Cicode before the PLC write is completed.

CitectSCADA does this so that the Cicode is not stopped while waiting for a slow I/O device. As the write to the PLC has not completed, you might think that the next line of Cicode will display the last value of PLC\_VAR and not the value

1234. The Cicode display the correct value (1234) because whenever CitectSCADA writes to the PLC, it first updates its local copy of the variable: any following Cicode will get the correct value.

Sometimes this solution will not work as CitectSCADA might keep multiple copies of an I/O device variable, and only updates the one associated with the current Cicode. The other variables will contain the old value of the I/O device variable until they are refreshed (with a read from the I/O device). There is a separate data area for each display page, Cicode file, alarms, trends and reports. If you write to an I/O device variable from a page keyboard command, the copy of the I/O device variable associated with that page will be updated; however, the copy associated with other pages and all the Cicode functions is not updated until the next read (as determined by the [Page]ScanTime parameter). If you call a Cicode function that assumes the write has completed, it will get the last value.

The workaround is to write to the I/O device variable in the Cicode function. All Cicode functions share the same I/O device variables, so the writes will operate as expected.

```
FUNCTION
TestFunc(INT nValue)
PLC_VAR = nValue;
END
```

Another side effect of this operation is that you might think that CitectSCADA has successfully written to the I/O device, when in fact it might later fail because of a hardware fault or configuration error, (that is, write to an input register). Under some conditions, you might want to check that the write has completed. When a write fails to the I/O device, a hardware error is generated, but the associated Cicode cannot be notified or use the `IsError()` function as that Cicode has executed past the I/O device write. You might verify a critical write by calling the function `TagRead()` after the write and then verify the value of the variable. `TagRead()` forces Cicode to re-read the I/O device variable so that you can check the new value. `TagRead()` is a blocking function. It blocks the calling Cicode task until the operation is complete.

```
PLC_VAR = 1234;
sTestStr = TagRead("PLC_VAR");
IF sTestStr <> 1234 THEN
Prompt("Write failed");
END
```

Here the data will be read from the physical PLC, not from the I/O server cache, as the I/O server will invalidate any cached data associated with a PLC write. This will allow you to test for a completed write. Note that other Cicode tasks running at the same time will not be waiting on the `TagRead`, so they might see the old or new value, depending on if they are using the same copy of the PLC

variable. You can also stop CitectSCADA from writing to the local copy of the variable by using the function `CodeSetMode(0, 0)`.

## Performance Considerations

Many factors that are outside of your control influence the performance of control and monitoring systems. The computer, the I/O device(s), and the communication pathway between them are obvious factors. The faster they can transfer data, the faster your system operates. (CitectSCADA always maintains a [data transfer](#) rate as fast as the I/O device hardware can support.) The data transfer rate is hardware dependent: once you have installed the hardware, it is beyond your control.

However, there is one area where you can directly affect the performance of your runtime system: the arrangement of data registers in the I/O device(s).

See Also [Caching data](#)  
[Grouping registers](#)  
[Remapping variables in an I/O device](#)

## Caching data

On large networked systems with many display clients, you can improve communications turn-around time by using memory caching.

When caching is enabled, all data that is read from an I/O device is stored temporarily in the memory of the I/O server. If another request is made (from the same or another display client) for the same data within the cache time, the CitectSCADA I/O server returns the value in its memory, rather than rereading the I/O device. Data caching results in faster overall response when the same data is required by many display clients.

A cache time of 300 milliseconds is recommended. Avoid using long cache times (in excess of 1000 milliseconds), because the data can become “stale.”

**Note:** Do not use data caching for memory or a [disk I/O device](#).

### How data caching works

Data caching prevents unnecessary rereads of I/O device data within a short period of time. Unnecessary reads can be generated when more than one client requests the I/O server to read data from a PLC or similar I/O device within a short (typically 300ms) period of time.

Normally, upon request from a client, an I/O server reads status data from an I/O device, and passes it back to the requesting client.

If the server receives subsequent requests from other clients before the original data is returned to the first client, it optimizes the read by automatically sending the original data back to all requesting clients. (Page General Blocked Reads shows this count).

If a client requests the same data immediately after the server returned the data to a client, the server rereads the device unnecessarily.

Setting the data cache time to 300ms (or similar) prevents identical repetitive reads within that cached time frame. If further clients request the identical data from the same server up to 300ms after the server has sent that data to an earlier client, the cached data on the server is sent immediately in response to the subsequent requests.

**Note:** Multiple clients don't have to be separate CitectSCADAs on a network.

They may be the alarms and trend clients in the same computer, so this optimization will affect even a single node system.

CitectSCADA also uses read-ahead caching. When the data in the cache is getting old (close to the cache time), the I/O server will re-request it from the I/O device. This optimizes read speed for data that is about to be re-used (frequent). To give higher priority to other read requests, the I/O server requests this data only if the communication channel to the I/O device is idle.

#### Keeping a permanent record of the data

To keep a permanent copy of cached device data, you can save the I/O server's cache to disk. For every `[IOServer]SavePeriod`, the data is saved to persistence caches, one for each cached device.

Saving the data to disk ensures that you can shut down and restart the I/O server without having to contact each I/O device again to get its current values. Instead, you can read the values from the device's [persistence cache](#).

**Note:** When read-through caching is enabled for a remote or scheduled I/O device, the persistence cache for that device is saved to disk when the active I/O server disconnects from the device. This occurs regardless of the value set in `[IOServer]SavePeriod`. You can enable read-through caching by setting the parameter `[Dial]ReadThroughCache`.

See Also

[Grouping registers](#)

## Grouping registers

When you configure a CitectSCADA system, you must define each variable (register address) that CitectSCADA will read when your system is running. When your runtime system is operating, CitectSCADA calculates the most efficient method of reading registers. CitectSCADA optimizes communication based on the type of I/O device and the register addresses.

When CitectSCADA requests data from an I/O device, the value of the register is not returned immediately; an overhead is incurred. This overhead (associated with protocol headers, checksum, device latency, and so on) depends on the brand of I/O device, and is usually several times greater than the time required to read a single register. It is therefore inefficient to read registers individually, and CitectSCADA usually reads a contiguous block of registers. Because the

overhead is only incurred once (when the initial request is made), the overhead is shared across all registers in the block, increasing the overall efficiency of the [data transfer](#).

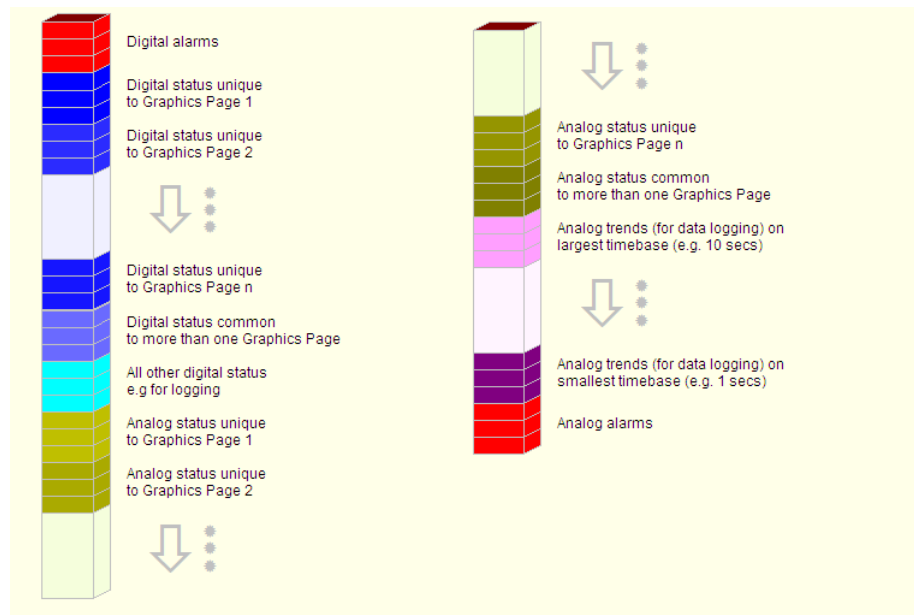
However, reading a block of registers where only a small percentage of the block is actually used is also inefficient. If the registers that your CitectSCADA system will read are scattered throughout the memory of your I/O device, excessive communication will be required. CitectSCADA must either read many contiguous blocks (and discard the unused registers), or read registers individually, degrading system performance. You can avoid this by grouping the registers that CitectSCADA will read.

CitectSCADA continually reads all registers associated with alarms. (If an alarm condition occurs, CitectSCADA can display the alarm immediately.) You should therefore group all registers that indicate alarm conditions.

Registers associated with status displays (objects, trends, and so on) are only read as they are required (that is, when the associated [graphics page](#) is displayed) and are best grouped according to the pages on which they are displayed.

Registers used for data logging are read at a frequency that you define. They are best grouped according to the frequency at which they are read.

The following diagram shows an ideal register grouping for a CitectSCADA system:





While memory constraints and the existing PLC program might impose limitations, grouping registers into discrete blocks, even if they are not consecutive blocks, will improve system performance.

When designing your system, allow several spare registers at the end of each block for future enhancements.

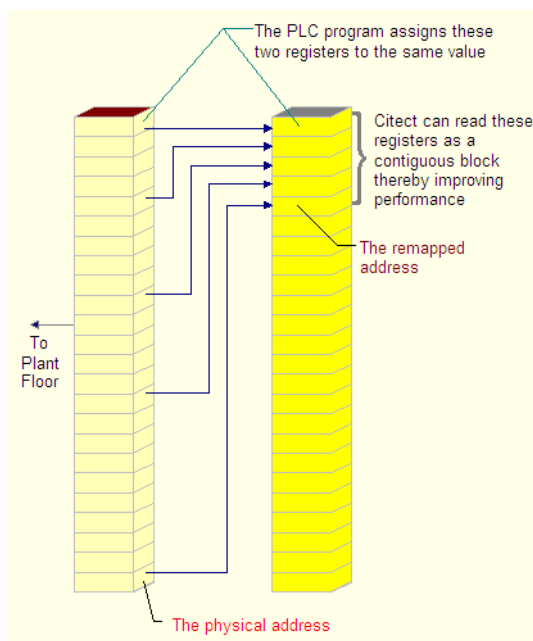
See Also [Remapping variables in an I/O device](#)

## Remapping variables in an I/O device

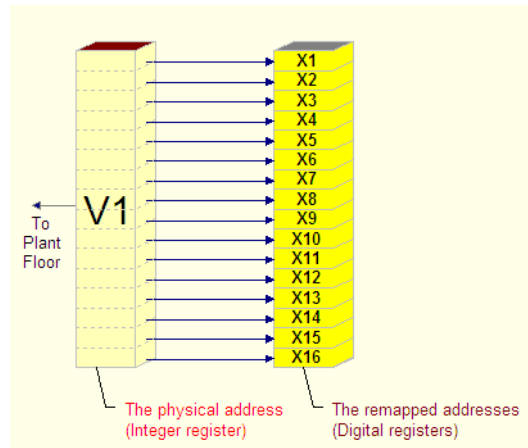
Some PLCs allow you to remap (or copy) an I/O device variable to another register address. CitectSCADA allows you to remap to:

- Group registers more efficiently to increase performance.
- Allow CitectSCADA to interpret a variable type (for example, an analog variable) as a different variable type (for example, a digital variable). For example, you can create additional digital addresses if an I/O device has run out of digital addresses.

To remap a variable in your PLC, you must design (or modify) the logic in the PLC to associate both addresses. CitectSCADA can then read or write the variable to and from the remapped address instead of the physical address.



You can also reassign one type of variable (for example, an integer) to another type of variable (for example, a digital variable).

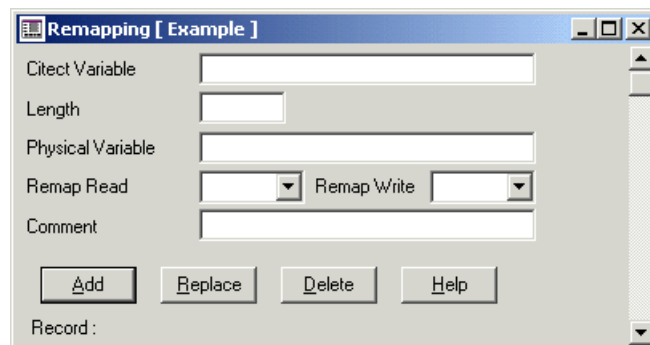


To remap in CitectSCADA, first create the variables in your project as you would normally. Then you can set up the remapping, specifying that any variable with an address in the desired range will be remapped. The I/O server will redirect the addresses at runtime as per the remapping instruction.

**Note:** Not all PLCs and/or CitectSCADA drivers support remapping. CitectSCADA does not recommend using remapping unless necessary. Contact Citect Support if you need to evaluate whether the PLC and/or CitectSCADA driver support remapping.

#### To remap a variable in CitectSCADA:

- 1 Open Project Editor.
- 2 Choose **Communication | Remapping**. The Remapping dialog box appears.
- 3 Complete the dialog box, and then click **Add**.



See Also [Remapping properties](#)  
[Remapping example](#)

## Remapping properties

You use the Remapping dialog box to remap your variables.

Remapping has the following properties:

### CitectSCADA variable

The first remapped (CitectSCADA) variable defined in the variable tags database (using the Tags dialog box); for example: Motor\_1\_Run.

Alternatively, use the direct <Unit Name>|<Address>| format (using values specific to your I/O device); for example: IODev|X1|.

**Note:** The address entered here is remapped. At runtime the I/O server will read/write data through the physical address instead.

### Length

The number of remapped variables. CitectSCADA reads enough physical variables to remap this number of CitectSCADA variables.

The length must be less than the [maximum request length](#) of the protocol. The protocol overview displays the maximum request length of the protocol.

### Physical Variable

The first physical variable in the PLC, for example: ReMapIntV7.

This variable does not need to be defined in the variable tags database. You can use the <Unit Name>|<Address>| format (using values specific to your I/O device). For example, IODev|V7|.

### Remap Read

Determines whether to perform the remapping for reads:

- Read normal (not remapped) variables. The actual address of the CitectSCADA variables will be read directly from the I/O device, instead of through the Physical Variable. (Use this mode if your I/O device does not support remap reads.)
- Read remapped variables (through the physical variable).

### Remap Write

Determines whether to perform the remapping for writes:

- Write to normal (not remapped) variables. The actual address of the CitectSCADA variables will be written directly in the I/O device, instead of through the physical variable. (You must use this mode if your I/O device does not support remap writes.)
- Write to remapped variables (through the physical variable).

**Note:** To determine if the device supports remapped reads or writes, see the I/O device data type Help.

See Also [Remapping example](#)

Remapping example

Using the CCM protocol with a GE 9030 PLC, the following remapping could be used to optimize communication when reading some digitals. This example is based on the assumption that the PLC is set up correctly for remapping.

Variable tags database

The digitals are defined as follows.

Variable Tag Name	Motor_1_Running_Feedback
Data Type	Digital
I/O Device Name	Area1
Address	M1
Variable Tag Name	Motor_1_Fault
Data Type	Digital
I/O Device Name	Area1
Address	M3
Variable Tag Name	Motor_4_ Running_Feedback
Data Type	Digital
I/O Device Name	Area1
Address	M4
.	.
.	.
.	.
Variable Tag Name	Motor_4_ Running_Feedback
Data Type	Digital
I/O Device Name	Area1
Address	M16

Notice that the address M2 is not used. This will not cause any problems.

Remapping database

The remapping is defined as follows.

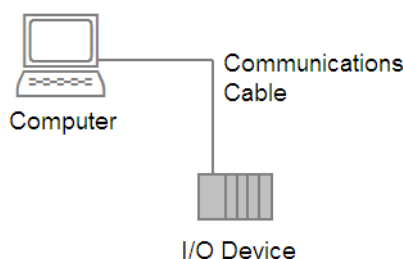
CitectSCADA Variable	Area1 M1
Length	16
Physical Variable	Area1 R10
Remap Read	True
Remap Write	False

The physical variable is an integer data type; it does not need to be defined in the variable tags database (but it can be).

## Serial Communications

### Using a COM port

The simplest CitectSCADA systems use a single computer connected to the I/O device(s). You can connect an I/O device directly to a [communications port](#) with a standard RS-232 communications cable.



#### How to set up CitectSCADA to use your computer's COM port:

- 1 Make sure that the **Boards** configuration has **COMx** as the **Type**, and the **Address** set to **0**. The **I/O Port**, **Interrupt** and **Special Opt** can all be left blank.
- 2 Enter the **Port Number** in the **Ports** configuration. The COM port number will usually be either **1** or **2**, and is set in the Ports section of the Control Panel. Use the **Special Opt** field to modify the behavior of the COMx driver.

**Note:** You only need to define the COMx board once. You can then add several ports that use the same CitectSCADA board. For example, a COM port and two serial boards would be defined as one COMx board in CitectSCADA, with multiple ports.

See Also [COMX driver special options reference](#)  
[TCP/IP driver special options reference](#)  
[Using a Serial Board](#)

### COMX driver special options reference

Special Options (in the Ports form) are space separated and start with the dash character (-) immediately followed by the option characters. Only a small percentage of users will need to use the following options:

- **-cATS0=1**: Send the string 'ATS0=1<CR>' on startup to allow the modem to detect the [baud rate](#) the port is running at. Abandon transmit if DCD is low. Wait for incoming call to raise DCD.
- **-d**: Data will be transmitted only when DSR is high.
- **-di**: Received data is ignored when DSR is low.
- **-dMS**: When transmitting a message the driver will wait up to 2000 ms for DSR to go high, then wait another MS milliseconds before transmitting.

- **-e:** Provides access to the output signal lines. The format is "**~EIAWXYZ**" where **WXYZ** represents one of the following options:

S	Simulate XOFF received
S	Simulate XON received
RT	Set RTS high
RT	Set RTS low
D	Set DTR high
D	Set DTR low
B	Set the device break line
B	Clear the device break line

**Example:** "**~EIADTR1**" sets DTR high

- **-h:** Data will be transmitted only when CTS is high.
- **-hMS:** When transmitting a message the driver will wait up to 2000 ms for CTS to go high, then wait another MS milliseconds before transmitting.
- **-i:** The string sent whenever the port is initialized. The tilde (~) and '\M' characters represent special instructions:
  - ~: Delay for 500 milliseconds
  - \M: Send carriage return

**Examples:**

**~Fred:** Wait 500 milliseconds and then send 'Fred'

**Fred\MMary:** Send 'Fred', a carriage return, and then 'Mary'

**Note:** This option is not available for dialable devices (i.e. when the port number is -1).

- **-nt:** With some serial interfaces, line faults can cause the COMx read thread to shutdown. If this happens, the driver does not recover after the fault. However, with the -nt (no terminate) option set, the thread is not shutdown, allowing the system to recover when the fault is rectified.
- **-nts:** If errors occur when the COMx driver is starting up, it will not terminate, but will continue attempts to open the COMx port.
- **-r:** Driver will raise DTR only when transmitting.
- **-ri:** DTR is raised when there is enough room in the input buffer to receive incoming characters and drop DTR when there is not enough room in the input buffer.
- **-rPRE,POST:** When transmitting a message the driver will raise DTR for PRE milliseconds, transmit message, wait for POST milliseconds then drop DTR.

## TCP/IP driver special options reference

- **-sc:** Activates software flow control using XON and XOFF
  - **XonLim:** A number in bytes. This represents the level reached in the input buffer before the XON character is sent (30 bytes).
  - **XoffLim:** The maximum number of bytes accepted in the input buffer before the XOFF character is sent. This is calculated by subtracting (in bytes) 100 from the size of the input buffer.
- **-t:** Driver will raise RTS only when transmitting.
- **-ti:** RTS is raised when there is enough room in the input buffer to receive incoming characters and drop RTS when there is not enough room in the input buffer.
- **-to:** RTS is raised only when there are characters to transmit.
- **-tPRE,POST:** When transmitting a message the driver will raise RTS for PRE milliseconds, transmit message, wait for POST milliseconds then drop RTS.

Special Options (in the Ports form) are space separated and start with the dash character (-) immediately followed by the option characters. Use the following special options for TCP/IP:

- **-A:** Allows the TCPIP driver to be used from Cicode.
- **-Ia.b.c.d:** defines remote [IP address](#) to connect to.
- **-FC:** Allows for a fake connection. This creates a pretend connection (no actual IP connection is made). Its intended use is when a driver wants to support a dummy connection but not talk to a device, or have a virtual unit. A virtual unit would allow access to driver addresses which do not need to talk to a device.
- **-K:** sets socket SO\_KEEPAIVE flag.
- **-LIa.b.c.d:** defines local IP address.
- **-LPn:** defines local PORT.
- **-Ma.b.c.d:** defines multicast IP address.
- **-Pn:** defines remote PORT to connect to.
- **-RC:** Activates the reconnection retries on reception of FD\_CLOSE event. FD\_CLOSE is only received if the Keepalive option is activated. -RC can resolve issues where the TCP/IP driver is notified of the connection close. For a more comprehensive handling, the -k option is needed. Retries can also be activated with the following high-level driver call:

```
COMSetParam( (SHORT)ChannelNumber, (UCHAR*)( "DO_RECONNECT" ),
NULL );
```

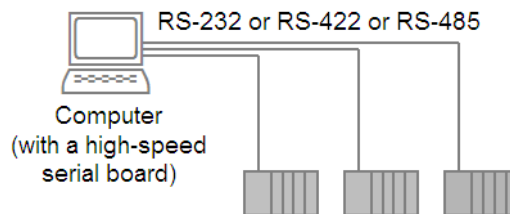
- **-U:** sets this port for UDP (datagram) operation

where:

- **a.b.c.d**: standard IP address in [dot notation](#) using decimal numbers 0-255. (Do not use a leading 0 when adding an IP address).
- **n**: decimal value for the port ID for the required service.
- **-T**: sets this port for TCP (stream) operation.

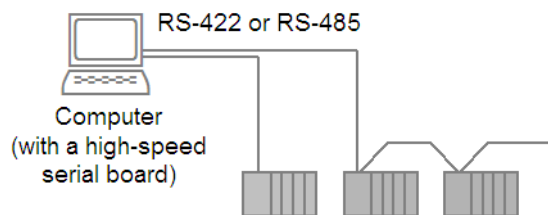
## Using a Serial Board

The [communications port](#) on the computer is not designed for high-speed communications and reduces system performance. Instead, install a high-speed serial board (such as a [Digiboard](#)). High-speed serial boards have several ports (usually 4, 8, or 16) to let you connect several I/O devices to your CitectSCADA system.



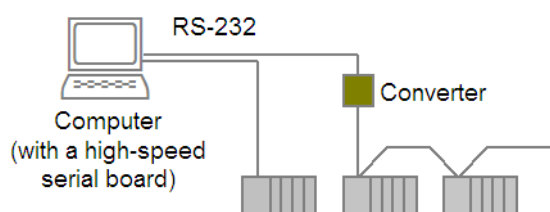
You can use identical I/O devices or I/O devices supplied by different manufacturers; CitectSCADA supports all popular I/O devices. You can connect any number of I/O devices; the only limitation is the size of your computer. High-speed serial boards are available for RS-232, RS-422, or RS-485 communication.

If you have several I/O devices from the same manufacturer and these I/O devices support multi-drop communication, you can connect them to an RS-422 or RS-485 high-speed serial board installed in your computer. (The RS-232 standard does not support multi-drop communication.)





Not all high-speed serial boards support RS-422. You can use an RS-232/RS-422 or RS-232/RS-485 converter to achieve the same arrangement.



**Note:** Using a converter can introduce handshaking/timing problems.

See Also [Serial board setup](#)  
[Serial Port Loop-Back Test](#)

## Serial board setup

To set up CitectSCADA to use a serial board:

- 1 Install the board in your computer and set it up under Windows as per the accompanying instructions. Use the latest driver from the board manufacturer.
- 2 Make sure that the boards configuration has COMx as the Type, and the Address set to 0. The **I/O Port**, **Interrupt**, and **Special Opt** can be left blank.
- 3 Enter the **Port Number** in the Ports configuration. The COM port number is usually greater than 2 and set in the **Ports** section of the Control Panel. Use the **Special Options** field to modify the behavior of the COMx driver.

### Notes

- If using your computer's COM port, you don't need to install additional software.
- You only need to define the COMx board once. You can then add several ports that use the same CitectSCADA board. For example, a COM port and two serial boards would be defined as one COMx board in CitectSCADA, with multiple ports.

See Also [Using Digiboards with Windows](#)  
[Using Proprietary Boards](#)

## Using Digiboards with Windows

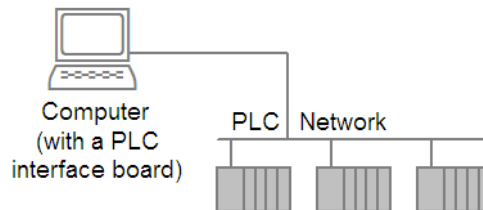
CitectSCADA Version 5 uses the standard Digiboard drivers supplied by Digiboard with Windows. CitectSCADA no longer supplies the drivers for these boards.

The COM/Xi and MC/Xi are not supported for WIN95 or WINNT, but the PC/Xe and PC/Xi are.

Please note that you can use other third-party serial board; you are not limited to using Digiboard serial boards.

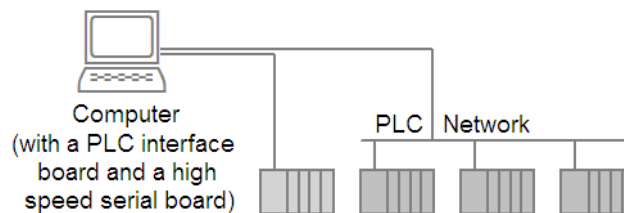
## Using Proprietary Boards

With some brands of PLCs you can install a proprietary interface board in your computer. This [PLC interface board](#) is supplied by the PLC manufacturer; you can connect it to a single PLC or a PLC network.



**Note:** With some PLCs, a high-speed serial board provides better performance than a PLC interface board when the system is connected to more than one PLC.

You can mix both PLC interface boards and high-speed serial boards in a single computer. You can, for example, connect a PLC network to a PLC interface board, and individual I/O devices to a high-speed serial board.



There are many possible hardware arrangements for a CitectSCADA application. CitectSCADA is a flexible system and imposes few restraints on the type (or manufacturer) of I/O devices that you can use, or on the way you connect them to the computer.

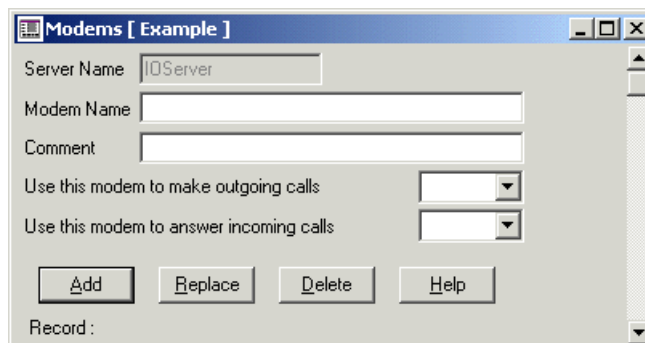
See Also [Proprietary board setup](#)

### Proprietary board setup

To set up CitectSCADA to use a **proprietary board**:

If you are using a proprietary board (that is, supplied by the PLC manufacturer):

- 1 Install the board in your computer and set it up under Windows as per the accompanying instructions. Use the latest driver from the manufacturer.
- 2 If possible, run diagnostics on the board before configuring CitectSCADA to check that the board works correctly.
- 3 Check that the **I/O Port** and **Interrupt** settings are correct.
- 4 Configure the **Boards** and **Ports** as instructed by the PLC-specific help.



## Serial Port Loop-Back Test

You can use the serial port loop-back test to test your serial hardware configuration. This test can be used with any COM port, whether it is local, or on a multi-port serial board (such as a [Digiboard](#)). The test can be performed internally or externally with loop-back cable attached.

### Test setup

- 1 No other protocols should be configured. Temporarily delete other boards and units while performing this test.
- 2 Configure a unit for each port to be tested. The I/O devices form should look as follows:
  - **Name:** <unique name for the I/O device>
  - **Number:** <unique network number for the I/O device>
  - **Address:** NA
  - **Protocol:** LOOPBACK
  - **Port Name:** <the "Port Name" in the Ports form>
- 3 The following Citect.ini options are supported:
  - [LOOPBACK]

LoopBack = Set this to 1 if internal loop-back is to be performed (make sure this is deleted after running the test). When set to 0, a loop-back connector which ties pins 2 and 3 together is required at each port.

**Note:** The COMx driver does not support internal loop-back. The external loop-back is the default mode.

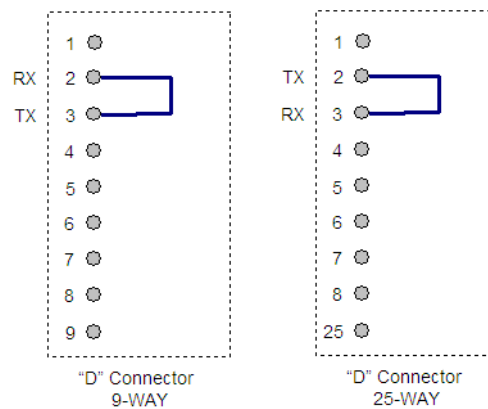
**Size** = Sets the maximum frame size. The length of each frame transmitted is random between 1 to 'Size'-1. The default size is 512.

- 4 Start up CitectSCADA. Each port will transmit a frame of random length. This process is repeated when the entire frame is received.
- 5 Open the kernel, type "page driver" and press **Enter**. Type **V** to set the display mode to 'verbose'. The following statistics appear:
  - Number of characters transmitted.
  - Number of frames transmitted.
  - Number of characters received.
  - Elapsed time in milliseconds.
  - Characters received per second.
  - Start time in milliseconds.
  - Total number of errors.
  - Error code of last error.

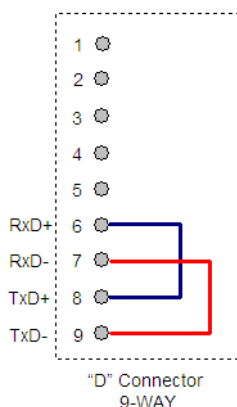
**Note:** The total number of errors should be 0. If the error is not zero, your serial hardware is faulty.

#### Serial port loop-back cable

The diagram below shows the loop-back connections to use with RS-232.



The diagram below shows the loop-back connections to use with RS-422 or RS-485.



## Setting Up Communications

To set up your I/O device communication:

- 1 Open the Project Editor.
- 2 Choose **Communication** | **Express Wizard** or open Citect Explorer.
- 3 Double-click the **Express I/O Device Setup** icon in the Communications folder of the current project.
- 4 Complete the Wizard.

**Note:** Each I/O device/protocol combination requires a unique setup for the boards, ports, and I/O devices forms. See the specific Help for each device.

See Also [Manually Configuring Communications](#)

## Manually Configuring Communications

Usually the Express Communications Wizard is sufficient to set up your communications. However, if you need to manually configure communications, do the following:

- 1 Define an I/O server in the [I/O Server Properties](#) form. This defines the name of the CitectSCADA server that the I/O device will communicate with.
- 2 Complete the [Boards Properties](#). This defines which board (on your CitectSCADA computer) to use to communicate (mother board, network card, serial board or a PLC communication card). Diable remote I/O devices must use a COMX board.

- 3 Complete the [Ports Properties](#). Often boards have multiple communication ports and you must specify the port to use. Some equipment can have several logical (virtual) ports assigned to the one physical port. If using modems, you must specify a unique port name for each and -1 for the port number. You must also specify the communication parameters and any special behavior of that port.
- 4 Complete the [I/O Devices Properties](#). This defines the I/O device that CitectSCADA is talking to, by specifying the address. The protocol is also defined at this level.
- 5 Run the Computer Setup Wizard to complete configuration. This allows you to define your CitectSCADA computer as the I/O server defined above. This is usually done after you compile the project.
- 6 If using diallable remote I/O devices, you must complete the Modems dialog box. This defines how CitectSCADA uses a modem to communicate with remote I/O devices.

**Note:** If there is no data to read or write, CitectSCADA will not communicate with an I/O device regardless of whether it is defined or not. You must create a variable tag and use it somewhere in CitectSCADA before CitectSCADA will do a read request. For example, use an integer variable to display a number on a page.

## I/O Server Properties

To define an I/O server, you must specify a name on the I/O Server form. This name will be used to reference the I/O server and should be logical. For example, for a single I/O server, you might use the name IOserver. When specifying the server name, you cannot use localized characters.

If you are using multiple I/O servers for redundancy (or to split the communications), you must add a database record for each. Select a unique name for each I/O server, for example IOserver1 and IOserver2.

**Note:** You must add the record to the project database (use the Add Button at the bottom of the form) or replace the record (use the Replace Button at the bottom of the form) if you have changed the record.

### Adding an I/O server

To define a computer as an I/O server, you must run the Computer Setup Wizard on that computer. The wizard allows you to choose from a selection of all of the I/O servers you have configured in your project.

## Boards Properties

The properties of a board depend on the type of board installed in the I/O server computer.

Boards have the following properties:

### Board Name

A name for the board. For example Server1\_Board1.

If you have more than one board in your I/O server computer, the name of each board must be unique. If you have multiple I/O servers, the board name need only be unique within each server. For example Server1\_Board2

### **Board Type**

The type of board.

If you are using a serial board or your computer's COM port, you should enter COMx.

### **Address**

The starting address of the Board. For example 0xCC00.

You must specify the address to match the switch settings on the board when it was installed in your computer. If you are using a serial board or your computer's COM port, you should enter 0 as the address.

**Note:** If more than one board is installed in the same computer, use a different memory address for each board.

### **I/O Port**

The I/O port address of the Board.

You must specify the address to match the switch settings on the board when it was installed in your computer.

**Note:** If you are using your computer's COM port you should not enter the port address here. You should specify the port number in the Ports form.

### **Interrupt**

The [interrupt](#) number used by the Board. This is not required if using your computer's COM port.

### **Special Opt**

Any special options supported by the board. Please check the Hardware Arrangements Help Topic for your specific I/O device to see if specific options are required.

### **Comment**

Any useful comment.

## **Ports Properties**

The properties of a port depend on the type of board installed in the I/O server, and on the I/O device connected to the port. Ports have the following properties:

### **Port Name**

A name for the port connected to your I/O device(s). Each port must have a unique name (i.e. you cannot assign the same Port Name to two ports in your system). You can use any name (up to 16 characters), for example: Board1\_Port1

If you have more than one board in your computer, you can use the port name to identify the board, for example: Board2\_Port1

**Port Number**

The port number that the I/O device is connected to. Do not assign the same Port Number to two ports on a board, unless connecting to a diallable remote I/O device via a modem (see the NOTE below). Ports on different boards can be assigned the same number.

If you are using your computer's COM port you should enter the port number here - the port number is defined in the Ports section of the Windows Control Panel.

**Note:** If you are connecting to a diallable remote I/O device (via a modem), you must define a unique port on the I/O server for each diallable remote I/O device, and the port number must be -1 for each.

**Board Name**

The name you used for the board. This is required to link the port to the board. For example Server1\_Board1

**Baud Rate**

The baud rate of the communication channel (between the CitectSCADA I/O server and the I/O device).

**Note:** The I/O device hardware and serial board may support other baud rates. If you do choose an alternative baud rate, ensure that both the I/O device and serial board support the new baud rate.

**Data Bits**

The number of data bits used in data transmission. You must set your I/O device to the same value.

**Stop Bits**

The number of stop bits used to signify the completion of the communication. You must set your I/O device to the same value.

**Parity**

The data [parity](#) used in data transmission.

**Special Opt**

Any special options supported by the port. Please check the Hardware Setup Help Topic for your specific I/O device to see if specific options are required.



See Also [COMX driver special options reference](#)

#### Comment

Any useful comment.

## I/O Devices Properties

Configuring an I/O device involves specifying its properties using the Project Editor. The properties depend on both the protocol and I/O device.

#### To configure an I/O device:

- 1 In the Project Editor, select **Communication | I/O Devices**. The I/O Devices dialog displays.
- 2 In the **Name** field, type in a name for your I/O device (PLC). The name must be unique in the CitectSCADA system, unless the I/O device is defined in other I/O servers (to provide redundancy). If redundancy is used, the I/O device must then have the same I/O device number and address for each I/O server. You should use different I/O device names for your primary and standby I/O devices, otherwise I/O device Cicode functions cannot differentiate between them.
- 3 In the **Number** field, enter a unique number for the I/O device (0-16383). The number must be unique in the CitectSCADA system, unless the I/O device is defined in other I/O servers (to provide redundancy). If redundancy is used, the I/O device must then have the same I/O device number and address for each I/O server. You may use the same device name, but if you want to use I/O device Cicode functions, it is easier to have different I/O device names.
- 4 In the **Address** field, enter the address of the I/O device. What you enter in this field is determined by the type of I/O device (and protocol) used, as each has a different addressing strategy.
- 5 In the **Protocol** field, select the protocol you are using to communicate to the I/O device. Many I/O devices support multiple protocols, dependent on the communication method chosen.
- 6 In the **Port Name** field, specify the port on the board to which the I/O device is connected. This is required to link the I/O device to the port. For example Board1\_Port1.

**Note:** There is a limit of 255 COMx ports on a server. To avoid this limitation restricting your number of remote I/O devices, you can connect multiple remote I/O devices to the same port as long as communication details (Telephone number, Baud rate, Data bits, Stop bits, and Parity) are identical.

- 7 In the **Memory** field, specify whether the I/O device runs in Memory mode. The default value is FALSE. If you select TRUE, the I/O device will be created in memory and its values stored in memory at runtime. This may be useful if you are testing your system, before connecting physical I/O devices, as memory mode stops CitectSCADA from communicating with physical I/O devices. See [Using Memory Mode](#).

- 8 In the **Comment** field, type in any useful comment. This field is optional and is not used at runtime.
- 9 Click Add to add a new record, or Replace to replace an existing one.

**Note:** The following fields are implemented with extended forms (press F2).

### **Linked**

Determines whether or not the I/O device is linked to an external data source. If you link to an external data source, CitectSCADA is updated with any changes made to the external data source when a refresh is performed.

If you cut an existing link, you can choose to make a local copy of all the tags in the database or you can delete them from CitectSCADA's variable tags database altogether.

If an I/O device is linked to an external data source the Database Type, External Database, Connection String, Tag Prefix and Live Update fields will be greyed out.

### **Link External Tag Database**

The path and filename of the external data source for the I/O device.

Alternatively, you can enter the IP address/directory, computer name, or URL of a data server, etc. (e.g. "C:\Work.CSV" or "127.0.0.1", "139.2.4.41\HMI\_SCADA" or "http://www.abicom.com.au/main/scada" or "\coms\data\scada").

Click **Browse** to select a path and filename.

### **Connection String**

Enter a connection string to provide connection details for the data source. This is similar to an ODBC connection string. For example:

```
UserID = XXX; Password = YYY
```

or

```
ServerNode=111.2.3.44; Branch=XXX
```

Not all data sources require a connection string.

### **Link Database Type**

The format of the data referenced by the external data source.

### **Tag Prefix**

The prefix that will be inserted in front of the names of all linked tags in CitectSCADA's variable tags database (for this I/O device only). To change the prefix, you should delete it first, perform a manual refresh, then add the new prefix.

### **Automatic Refresh of Tags**

Determines whether the linked tags in CitectSCADA's variable tags database will be updated when the external data source is changed (i.e. you manually change a field, etc.). This refresh will occur the first time you link to the data source, and then whenever you attempt to read the changed variables (e.g. you compile your project, display the variable using the Variable Tags form, modify or paste the tag, etc.).

Without an automatic refresh, you will need to perform a manual refresh to update the linked tags in CitectSCADA.

### Live Update

**Note:** This field is only available if you have installed one of the CitectSCADA FastLinux products. See <http://www.citect.com> for details on FastLinux.

Controls whether or not the linked tags in CitectSCADA and an external tag database will be synchronized if either database is changed. To enable live linking, choose Yes from the Live Update menu.

When Live Update is enabled and the CitectSCADA variable tag database is accessed (for example, during project compilation or when a dropdown list is populated), CitectSCADA queries the external tag database to determine if it has been modified. If so, CitectSCADA merges the changes into the local variable tag database. Conversely, any changes made to the local tag variable database will be incorporated seamlessly into the external tag database.

### Startup mode

The type of I/O device redundancy.

Primary	Enable immediate use of this communications channel. This is the default mode if no mode is specified.
Standby	This channel will remain unused until activated by the failure of the I/O device configured with the primary channel.
StandbyWrite	This channel will remain unused until activated by the failure of the I/O device configured with the primary channel. All write requests sent to the primary channel are also sent to this channel. DO NOT use this mode for scheduled I/O devices.

**Note:** To use Standby or StandbyWrite modes, you must also configure a Primary I/O device with the same I/O device name, number and address.

### Priority

Sets the order standby devices are promoted in if a primary I/O device fails during runtime.

If there is more than one standby I/O device configured for a cluster, you can use this field to give precedence to a particular standby device. If this field is left blank, priority will be automatically allocated to the device during project compilation, based on the priority settings of other standby devices and/or the order the devices were configured in.

See [I/O Device promotion](#).

### Log Write

Enables/disables the logging of writes to the I/O device. When enabled, all writes are logged in the CitectSCADA SYSLOG.DAT file (in the Windows directory). See [Tag Functions](#) for information about TagWriteEventQue and logging tag data.

**Note:** Logging all writes to an I/O device may slow communications as the CitectSCADA system will be writing large amounts of data to disk. However, logging of writes is useful when debugging a system.

### Log Read

Enables/disables the logging of reads from the I/O device. When enabled, all reads are logged in the CitectSCADA SYSLOG.DAT file (in the Windows directory).

**Note:** Logging all reads to an I/O device may slow communications as the CitectSCADA system will be writing large amounts of data to disk. However, logging of reads is useful when debugging a system.

### Cache

Enables/disables data caching. When enabled, a cache of the I/O device's memory is retained at the I/O server, thus improving communications turn-around time.

**Note:** Data caching should be enabled for scheduled I/O devices, but disabled for memory or disk I/O devices.

### Cache Time

The cache time specified in milliseconds. When caching is enabled, all data that is read from a I/O device is stored temporarily in the memory of the I/O server. If another request is made (from the same or another Display Client) for the same data within the cache time, the CitectSCADA I/O server returns the value in its memory - rather than read the I/O device a second time. Data caching results in faster overall response when the same data is required by many Display Clients.

A cache time of 300 milliseconds is recommended.

The Cache Time for a scheduled I/O device is automatically calculated. You can view a scheduled I/O device's cache time using the Kernel Page Unit command.

### Scheduled

Determines whether the I/O device is configured for scheduled communications. This is normally set using the Express Communications Wizard.

NOTE: If you do not specify a schedule for a diallable remote I/O device, the connection will be established at startup and will remain connected until shut-down.

See Also [Scheduled Communications](#)

### **Time**

The I/O server will attempt to communicate with the I/O device at this time, and then at intervals as defined below. This time is merely a marker for CitectSCADA. If you run up your project after this time, the I/O server will NOT wait until the next day to begin communicating. It will operate as if your project had been running since before the start time.

### **Period**

The time between successive communication attempts.

Examples (all based on a Synchronize at time of 10:00:00):

- 1 If you enter 12:00:00 in the Repeat every field, and start your project at 9am, the I/O server will communicate with the I/O device at 10am, then once every 12 hours after that, i.e. 10pm, then again at 10am of the following day, etc.
- 2 If you enter 12:00:00, and start your project at 4pm, the I/O server will communicate with the I/O server at 10pm, then again at 10am of the following day, and so on. The I/O server will assume that communications were established at 10am, so it continue as if they had been - communicating once every 12 hours after 10am.
- 3 If you enter 3 days, and start your project at 9am on a Wednesday, the I/O server will communicate with the I/O device at 10am, then once every 3 days after that, i.e. 10am on the following Saturday, then at 10am on the following Tuesday, etc.
- 4 If you enter the 6th of December in the Repeat every, and start your project during November, the I/O server will communicate with the I/O device at 10am on December 6, then again on December 6 of the following year, etc.

Select On Startup for a permanent connection. To disconnect a permanent connection, you must call the IODeviceControl() function with type 8.

### **Connect Action (254 Chars.)**

Cicode to be executed once communication with the I/O device has been established (and before any read or write requests are processed).

### **Disconnect Action (254 Chars.)**

Cicode to be executed once communication with the I/O device has been terminated (and after all read and write requests are processed).

**Phone Number (64 Chars.)**

Windows 2000 Only

The telephone number that needs to be dialled to initiate contact with the I/O device. (i.e. for diallable remote I/O devices)

**Caller ID (32 Chars.)**

Windows 2000 Only

A unique identifier which identifies a remote I/O device when it dials back to the I/O server. The caller ID can be any combination of alpha-numeric characters and/or the character '\_' (underscore).

This ID will only be used if the I/O device initiates the call to the I/O server. If the modem initiates the call, you must set the caller ID on the modem.

**Note:** If you are multi-dropping off a single modem, you should use your I/O devices to issue the caller ID, not the modem. The problem with using the modem to issue the ID is that it will send the same ID no matter which I/O device the call is relevant to. This makes it hard for you to identify the I/O device that triggered the call.

By using the I/O device to issue the ID, the I/O server will receive a unique caller ID for each I/O device. However, not all I/O devices are capable of issuing caller IDs. If you are multi-dropping, you should use I/O devices that can issue caller IDs.

**Background Poll**

Specifies that all the tags for a particular device are continuously polled at a minimum background poll rate. The options are True or False. Set this field to True if you are operating on a slow network, with a slow device, or where tags may normally only be polled infrequently.

**Rate**

When Background Poll is set to True, specifies the minimum rate by which the device should be polled. You may select any predefined value from the drop-down list, or enter your own in the format of HH:MM:SS. If you do not enter a value, the default value of 30 seconds will be used.

See Also

[Communicating with Diallable Remote I/O Devices](#)

**Using Memory Mode**

When configuring an I/O device, you have the option to set memory mode. This means that the I/O device will be created in memory and its values stored in memory at runtime.

Refer to [I/O Devices Properties](#) for more information on how to configure an I/O device.

Devices using memory mode are not connected to any hardware and write their values to a cache. The I/O device values can be read by many processes. The difference between a local variable and a device in Memory Mode is that an I/O device in Memory Mode will reside in the I/O Server's memory and will observe all standard networking and redundancy rules of a standard I/O device.

Memory mode is useful when you are configuring a system for the first time, as you can design and test your system before connecting a physical I/O device.

As with local variables, values in an I/O device using only memory mode are not retained when you shut down. For more information on local variables, refer to [Configuring Local Variables](#).

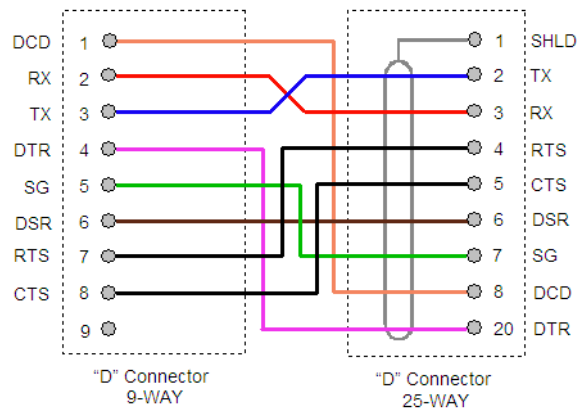
See Also [I/O Devices Properties](#)

## Wiring Cables

If using proprietary hardware (that is, a communication card installed in your CitectSCADA computer), refer to the accompanying documentation. The hardware manufacturer will provide the necessary wiring information and often the communication cable.

### Using a 9-to-25-pin converter

In serial protocols help for an RS-232 wiring diagram, usually only the pin arrangement for a [DB-25](#) (25-pin 'D' type) connector is given. To use your Com port or a 9-pin serial card, a 9 to 25 diagram is usually supplied also. The diagram is as follows:



**Note:** The 9- and 25-pin connections above are considered standard. The 9-pin arrangement is common to most computer com ports.

The converter is *straight through*, meaning pins that are labelled the same are connected together (that is, TX goes to TX).

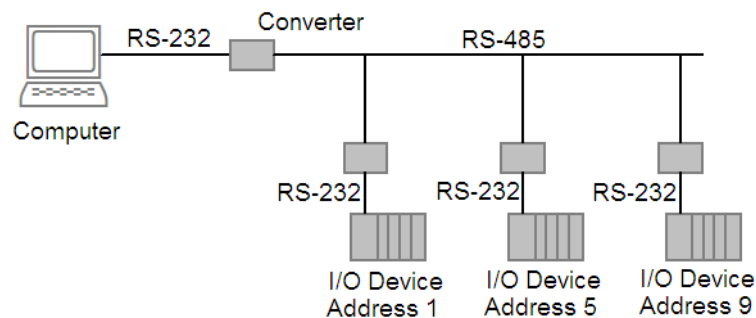
The diagram shows how to wire a serial cable to a 9-pin port. You do not need to create a 25-pin connection. Consider the following scenario:

- You want to connect to your computer's com port (ideal for a small system) using RS-232.
- The wiring diagram given in the help shows only a 25-pin connection.
- Create the cable as normal, but instead of using the 25-pin connector use a 9-pin. Instead of joining the wires to the pins on the 25-pin diagram, use the above diagram to see which pins to use on the 9 pin. e.g. instead of using pin 2 on the 25 pin (TX) use pin 3 on the 9 pin (TX).

### Using an RS232/485 converter

Using an RS232/485 converter is common, offering a cheaper alternative for using RS-485 without having an RS-422/485 serial board. RS-485 has significant advantages over RS-232, such as longer distances, faster transmission, noise immunity. Even more significant is that RS-232 does not support multi-drop.

The following diagram shows how to use converters to allow a configuration that would not be achievable with RS-232.



This arrangement is only available if the protocol supports multiple I/O devices. RS-422 can be used also if supported by the protocol. Obviously the maximum [data transfer](#) rate is less than that of a high-speed serial board.

**Note:** Wiring details vary from manufacturer to manufacturer. Generally the devices are defined as DCE and should be wired as such.

### DTE and DCE

Some people are often confused by DTE and DCE. They are defined as follows:

- DTE or data terminal equipment. This represents the equipment that ultimately acts as a data source or data sink (i.e. to further process the data). Computers and PLCs are usually regarded as DTE.
- DCE or data communications equipment. This represents a device that transmits data between a DTE and a physical [communications link](#). Usually responsible for establishing and maintaining a data transmission connection. Normally DCE refers to a modem.



Often you'll use DCE equipment in your control communications. These devices should be simple to wire as the only difference DCE and DTE is the use of the TX (transmit) and RX (receive) pins. Except where stated otherwise, all wiring diagrams in CitectSCADA help are for DTE. To use the same cable for DCE simply reverse the TX and RX connections, or follow the following rule of thumb:

- DTE to DCE: Join DCE-RX to DTE-RX and DCE-TX to DTE-TX
- DTE to DTE: Join DCE-RX to DTE-TX and DCE-TX to DCE-RX

## Common Serial Communication Standards

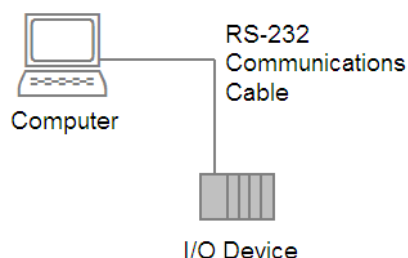
This section describes the following communication standards:

- [RS-232C \(or EIA-232C or RS-232\)](#)
- [RS-422 \(or EIA-422\)](#)
- [RS-485 \(or EIA-485\)](#)

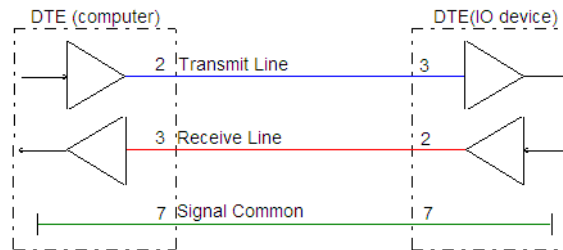
### RS-232C (or EIA-232C or RS-232)

RS-232C is the most common serial data communication interface standard. This standard defines the electrical and mechanical details of the interface but does not define a protocol. The standard covers the electrical signal characteristics, the mechanical interface characteristics (pin out etc.) and functional description of control signals etc.

- Point-to-point communication. Between only 2 devices.



- Communication is [full duplex](#). A single wire for each direction and a ground wire. This means that generally only 3 wires need to be connected for most applications. The diagram below shows the 'standard' pins for a [DB-25](#) connector.



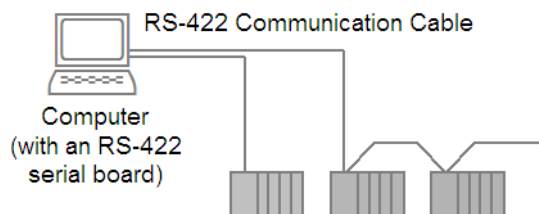
- Less than 75m maximum length at 19.2K maximum Baud rate. Up to 900 meters can be achieved at 900 Baud.

See Also [RS-422 \(or EIA-422\)](#)  
[RS-485 \(or EIA-485\)](#)

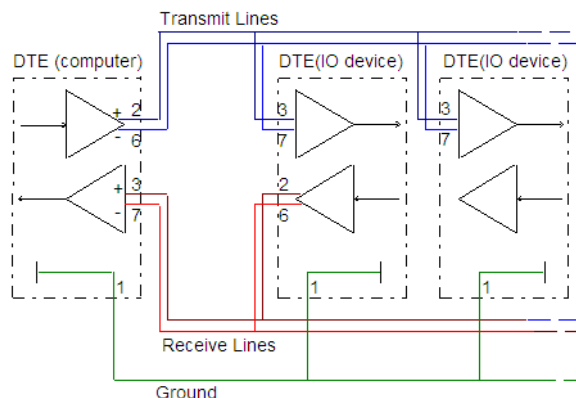
## RS-422 (or EIA-422)

RS-422 is recommended as it has significant benefits over RS-232C. This standard covers the electrical signal characteristics and functional description of control signals only. It does not define the protocol, but the protocol used should support multiple unit addressing to fully use this standard.

- Uses differential signals (difference between to line voltages) which provide greater noise immunity.
- Limited multi-drop communication. This means that there may be multiple receivers (but only 1 transmitter) on each line.



- Communication between two devices is full-duplex. Two wires are used for each direction and also one ground wire. This means that generally only 5 wires need to be connected for most applications. The diagram below shows a commonly used pin arrangement for a [DB-9](#) Connector.



Only one transmitter is allowed per line though there may be multiple receivers. This means only two devices may have full-duplex ([half duplex](#) for 2 wire) while the other devices have only simplex communication, as shown above.

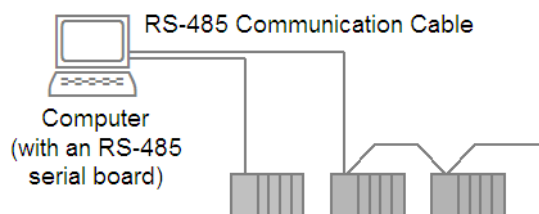
- Distances up to 1200m and transfer rates up to 10Mbps are achievable.
- The protocol used with this standard must take care of who (i.e. which device) is allowed to transmit at any one time. This allows each device to act as a transmitter when requested.

See Also [RS-232C \(or EIA-232C or RS-232\)](#)  
[RS-485 \(or EIA-485\)](#)

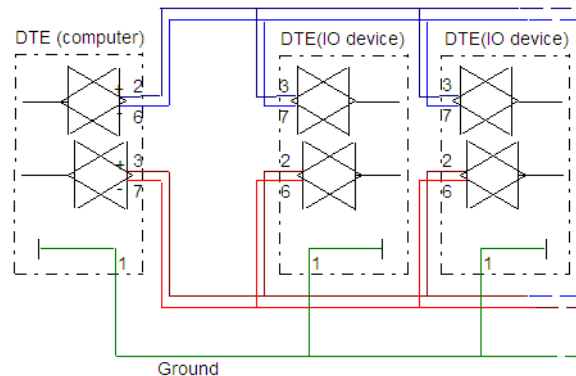
## RS-485 (or EIA-485)

RS-485 is an improved version of RS-422. This standard covers the electrical signal characteristics and functional description of control signals only. It does not define the protocol, but the protocol used should support multiple unit addressing and bus contention to fully use this standard. The major advantage is that all devices can transmit and receive on the same line.

- Electrically similar to 422. Logic levels, transfer rates and maximum distance are almost identical.
- RS-485 supports multiple transmitters and receivers on each line. This improves on the RS-422.



- Communication may be either half-duplex or full-duplex. Two wires are used for each direction and also one ground wire. This means that only three wires need to be connected for most half-duplex applications. Five wires are needed for most full-duplex applications. The diagram below shows a commonly used pin arrangement for a [DB-9](#) Connector.



**Note:** RS-485 supports both full and half-duplex. The above diagram shows a full-duplex arrangement. Unlike RS-422 each I/O device is able to transmit and receive on each line. If the arrangement were half-duplex, only one pair of transmission lines would be needed (rather than two pairs shown above).

The protocol used with this standard must take care of who (that is, which device) is allowed to transmit at any one time. This allows each device to act as a transmitter when requested.

See Also [RS-232C \(or EIA-232C or RS-232\)](#)  
[RS-422 \(or EIA-422\)](#)

## Using a Disk I/O Device

In addition to connecting to actual I/O devices, CitectSCADA supports the configuration of disk I/O devices, which exist only within your computer.

A disk I/O device provides permanent storage. The value of each variable in the disk I/O device is stored on your computer's hard disk. Using a disk I/O device is the same as configuring an I/O device in memory mode but with persistent data. See [Using Memory Mode](#) for more information.

Once configured, disk I/O devices appear exactly as any other I/O device in your CitectSCADA system, but are not connected to any field equipment. Disk I/O devices can contain any type of variable supported by CitectSCADA, and you can configure them to emulate any I/O device that CitectSCADA supports. You can also specify a generic protocol for a disk I/O device.

A disk I/O device is useful when the status of your plant needs to be restored after shutdown or system failure. You can configure your CitectSCADA system to continually update a disk I/O device with critical variables that define the status of your plant. When you restart your system after a shutdown or system failure, CitectSCADA can restore this status immediately.

You can also use disk I/O devices for storing predefined data that must be recalled immediately when a process is required (for example, in a simple recipe system).

**Note:** If you create a RAM disk in the computer for the disk I/O device, you do not need to create or copy the disk file to the RAM disk. CitectSCADA automatically creates a disk file on startup.

See Also [Disk I/O device setup](#)  
[Redundant Disk I/O Devices](#)  
[Using Memory Mode](#)

Disk I/O device setup

To set up communications with a device, follow the basic steps given in the I/O device setup procedure.

Sometimes you might need to edit the communications forms directly. They require the following specific information.

- You do not need to complete a Boards dialog box.
- You do not need to complete a Ports dialog box.
- Complete the I/O Devices dialog box as follows.

Perform the setup by entering the following information:

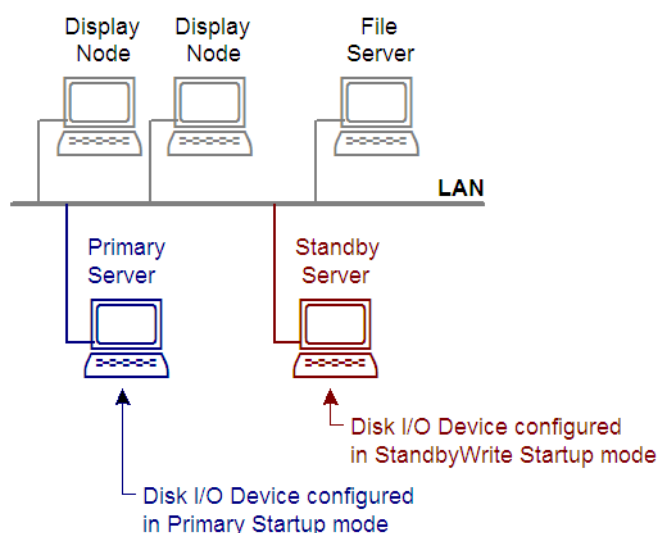
Text box	Required information
I/O Device name	A name for your Disk I/O device, for example: DISK_PLC. Each I/O device must have a unique name in the CitectSCADA system.
I/O device number	A unique number for the disk I/O device (1-4095). Each I/O device must have a unique number in the CitectSCADA system.

Text box	Required information
I/O device address	<p>The path and filename of the disk file, for example: [RUN]:DSKDRV.DSK</p> <p>If you are using redundant disk I/O devices, specify the path to both the primary file and the Standby file in the configuration of both disk I/O devices. For example, if this is the primary disk I/O device, enter: <code>Primary_File, Standby_File</code> If this is the Standby Disk I/O device enter: <code>Standby_File, Primary File</code></p> <p><b>Primary_File</b> is the name (and path) of the primary disk I/O device file. You may use path substitution in this part of the field. <b>Standby_File</b> is the name (and path) of the Standby Disk I/O device file. You may use path substitution in this part of the field.</p> <p>Note the following: If the specified disk I/O device file is not found, CitectSCADA creates a new (empty) file with the specified filename. To use redundant disk I/O devices, you must use a network that supports peer-to-peer communication. Disk files must have write permission (on both primary and standby servers). The frequency with which CitectSCADA writes data to the disk I/O device(s) is determined by the <code>[DiskDrv]UpDateTime</code> parameter.</p>
I/O device protocol	<p>To use the CitectSCADA generic protocol, enter: <b>GENERIC</b></p> <p>- or -</p> <p>To select a specific protocol supported by CitectSCADA, see the I/O devices online Help.</p>
I/O device port name	You must use: <b>DISKDRV</b>
I/O device comment	Any useful comment.
I/O device startup mode	<p>If not using redundant disk I/O devices, leave this property blank. If you are using redundant disk I/O devices, use either: <b>Primary</b>: Enable immediate use of this disk I/O device <b>StandbyWrite</b>: This disk I/O device will remain unused until activated by the failure of the computer with the primary disk I/O device configured. All write requests sent to the primary disk I/O device are also sent to this disk I/O device. To use StandbyWrite mode, you must also configure an I/O disk device in the primary server. Both I/O devices must have the same I/O device name and number.</p>
I/O Device Log Write, Log Read, Cache and Cache Time	Leave these properties blank.

See Also [Redundant Disk I/O Devices](#)

## Redundant Disk I/O Devices

If using a network, you can configure a redundant disk I/O device to eliminate data loss (in the event of server failure). This diagram illustrates the use of redundant disk I/O devices:



When the system is operating, CitectSCADA reads and writes runtime data to the disk I/O device configured in the primary server. CitectSCADA also writes runtime data to the disk I/O device configured in the standby server. (CitectSCADA maintains both disk I/O devices identically.)

If the primary server fails, the Disk I/O device in the standby server is activated without interrupting the system. When the primary server becomes active, CitectSCADA automatically returns control to the primary server, and copies the Disk I/O device from the standby server to the primary server. The Disk I/O device in the standby server reverts to its standby role.

### To define a redundant disk I/O device:

- 1 Configure a new disk I/O device
- 2 Select **StandbyWrite** for the Startup Mode.

For redundant disk I/O devices, you must use Microsoft Networking (or another peer-to-peer network), and the hard disk of the standby server (the directory where the disk I/O device file is stored) must be shared. Use Windows Explorer to set the directory to shareable.

See Also

[I/O Devices Properties](#)  
[Performance Considerations](#)

## Citect Driver Accreditation

Each driver-I/O device combination supported by CitectSCADA is subject to the CitectSCADA Driver Quality and Accreditation System (CiTDriversQA96), which promotes high reliability and efficient performance.

Because drivers can be written by any third-party developer, not all CitectSCADA drivers undergo the same quality control procedures. To enable users to distinguish between drivers of different standards, the following categories are used:

- [Accredited - Level 1](#)
- [Accredited - Level 2](#), or
- Functionally Stable.

**Note:** The I/O device Help indicates the category in which the driver-I/O device belongs.

## Advanced Driver Information

This section provides advanced information about drivers.

See Also

[Variable \(Digital\) Limitations](#)  
[Validating distributed project data for tag-based drivers](#)  
[Generic driver errors](#)  
[Standard driver errors](#)

### Variable (Digital) Limitations

Devices often have memory areas that are of a designated data type, like Byte, Integer, or Word. Some protocols do not support the reading and writing of data in these memory areas using a different data type. This situation is most common in the case of reading and writing of individual bits within the data types like Bytes, Integers, and Words.

In this case, reading individual bits within these larger data types is done by reading the designated data type and getting the CitectSCADA driver to break it down into individual bits. Writing to bits within the larger data types is more complicated, as writing to one bit within the larger data type will at the same time overwrite the other bits within that same data type. To overcome this limitation, a 'read-modify-write' scenario can be used to write to a bit within the larger data type. Using this approach, the CitectSCADA driver will read the larger data type, modify the appropriate bit within the larger data type, and then write the larger data type back to the device.

This 'read-modify-write' method has a serious operational concern: if the device modifies the larger data type after the CitectSCADA driver has read it, but before CitectSCADA has written the new value, any changes made by the device are overwritten. This issue could be serious in a control system, and it is



recommended that the device and CitectSCADA be configured so that only one of these systems writes to the data types of this kind.

Consider the following example:

- 1 The initial state of a PLC register is 0x02h.
- 2 The CitectSCADA driver reads the value of this register (effectively making a copy) in preparation for a change to bit 3.
- 3 However, before the CitectSCADA driver writes its change back to the PLC, the PLC code changes the value of bits 0 and 4 of this register to 0x13h.
- 4 The CitectSCADA driver then changes bit 3 of its copy of the register to 0x0Ah. When it writes to the PLC, it overwrites the PLC's copy of the whole register (not just the changed bit). Because the PLC code modified bits 0 and 4 in the interval between CitectSCADA's read and write, these changes are overwritten.

## Validating distributed project data for tag-based drivers

CitectSCADA uses numeric index values to uniquely identify all the variable tags in a project. They are used as a reference point when requesting data from the I/O server for a tag-based driver.

These index values are automatically generated when a project is compiled, which means they need to be carefully monitored when running a project across a number of client and server machines. Circumstances may arise where a distributed project has index values that represent different tag addresses on different computers. For this reason, CitectSCADA has a number of automatic checks in place that validate a project's tag index values and flag any discrepancies.

An initial security check takes place on client machines at a unit level, allowing a tag index mismatch to be isolated to a particular client before any requests are sent to the I/O server. This confirms that the unit address, the unit type, the raw data type and the tag address match for all index values across the client and server machines. Any problems are flagged by a hardware alarm on the client machine.

Each page is also checked to confirm that it was compiled against the current version of the variables database. There is also a check performed whenever a tag-based driver loads the variable database to ensure it matches the current tag addresses. The parameter TagAddressNoCase allows you to adjust the case-sensitivity of these checks.

In addition, CitectSCADA will also check if a project is currently running on the local machine when a compile is attempted, as this is one of the circumstances that may lead to mismatched index values.

If the project uses a tag-based driver and is currently in runtime, CitectSCADA will stop the compiler and generate an error in the error database noting that

Citect32.exe was still running. The .ini parameter [General]CitectRunningCheck allows you to override this feature, however it is recommended that you leave it enabled to ensure your tag index values remain valid.

## Generic driver errors

The following errors are generic to all CitectSCADA drivers. A driver error must be mapped to a generic error before CitectSCADA can interpret it.

Error	Description
GENERIC_ADDRESS_RANGE_ERROR (0x0001   SEVERITY_ERROR)	A request was made to a device address that does not exist. For example, an attempt was made to read register number 4000 when there are only 200 registers in the device.
GENERIC_CMD_CANCELED (0x0002   SEVERITY_ERROR)	The server cancelled the command while the driver was processing it. This can happen if the driver takes too long to process the command. Check the timeout and retries for the driver.
GENERIC_INVALID_DATA_TYPE (0x0003   SEVERITY_ERROR)	A request was made specifying a data type not supported by the protocol. This error should not occur during normal operation.
GENERIC_INVALID_DATA_FORMAT (0x0004   SEVERITY_ERROR)	A request contains invalid data; for example, writing to a floating-point address with an invalid floating-point number. Check the CitectSCADA database.
GENERIC_INVALID_COMMAND (0x0005   SEVERITY_ERROR)	The server sent a command to the driver that it did not recognize. This error should not occur during normal operation.
GENERIC_INVALID_RESPONSE (0x0006   SEVERITY_ERROR)	A problem with the communication channel is causing errors in the transmitted data.
GENERIC_UNIT_TIMEOUT (0x0007   SEVERITY_ERROR)	A device is not responding to read or write requests. The driver sent a command to the device and the device did not respond within the timeout period.
GENERIC_GENERAL_ERROR (0x0008   SEVERITY_ERROR)	Unmapped driver specific errors are normally reported as a general error. Refer to the protocol-specific errors listed with the protocol you are using.
GENERIC_WRITE_PROTECT (0x0009   SEVERITY_ERROR)	A write operation was attempted to a location that is protected against unauthorized modification. Change the access rights to this location to permit a write operation.
GENERIC_HARDWARE_ERROR (0x000A   SEVERITY_UNRECOVERABLE)	A problem exists with either the communication channel, server or device hardware. Examine all hardware components. The server's operation may no longer be reliable.
GENERIC_UNIT_WARNING (0x000B   SEVERITY_WARNING)	The communication link between the server and the device is functioning correctly; however, the device has some warning condition active, for example, the device is in program mode.
GENERIC_UNIT_OFFLINE (0x000C   SEVERITY_SEVERE)	The device is in off-line mode, preventing any external communication. This error will cause any stand-by units to become active. CitectSCADA will attempt to re-initialize the unit.

Error	Description
GENERIC_SOFTWARE_ERROR (0x000D   SEVERITY_SEVERE)	An internal software error has occurred in the driver. This error should not occur during normal operation.
GENERIC_ACCESS_VOILATION (0x000E   SEVERITY_ERROR)	An attempt has been made by an unauthorized user to access information. Check the user's access rights.
GENERIC_NO_MEMORY (0x000F   SEVERITY_UNRECOVERABLE)	The server or driver has run out of memory and cannot continue execution. Minimize buffer and queue allocation or expand the server computer's memory (physical or virtual memory).
GENERIC_NO_BUFFERS (0x0010   SEVERITY_ERROR)	There are no communication buffers left to allocate. The performance of the server may be reduced; however, it can still continue to run. Increase the number of communication buffers.
GENERIC_LOW_BUFFERS (0x0011   SEVERITY_WARNING)	This error may occur in periods of high transient loading with no ill effects. If this error occurs frequently, increase the number of communication buffers.
GENERIC_TOO_MANY_COMMANDS (0x0012   SEVERITY_WARNING)	Too many commands have been sent to the driver.
GENERIC_DRIVER_TIMEOUT (0x0013   SEVERITY_ERROR)	The server is not receiving any response from the driver. This error should not occur during normal operation.
GENERIC_NO_MORE_CHANNELS (0x0014   SEVERITY_SEVERE)	Each driver can only support a fixed number of communication channels. You have exceeded the limit. The command or data request has not been completed.
GENERIC_CHANNEL_OFFLINE (0x0015   SEVERITY_SEVERE)	A communication channel is currently off-line, disabling communication. The server cannot initialize the communication channel or the channel went off-line while running. All devices (units) connected using this channel will be considered to be off-line so this will cause any stand-by devices to become active. CitectSCADA will attempt to re-initialize the channel.
GENERIC_BAD_CHANNEL (0x0016   SEVERITY_SEVERE)	The server has attempted to communicate using a channel that is not open.
GENERIC_CHANNEL_NOT_INIT (0x0017   SEVERITY_SEVERE)	The server is attempting to communicate with a channel that has not been initialized. This error should not occur during normal operation. The command or data request has not been completed. If the problem persists, contact Citect Support.
GENERIC_TOO_MANY_UNITS (0x0018   SEVERITY_SEVERE)	A channel has too many devices attached to it. This error should not occur during normal operation.
GENERIC_INVALID_DATA (0x0019   SEVERITY_ERROR)	The data requested is not in a valid format or expected type.
GENERIC_CANNOT_CANCEL (0x001A   SEVERITY_WARNING)	The server tried to cancel a command, but the driver could not find the command. This error should not occur during normal operation.
GENERIC_STANDBY_ACTIVE (0x001B   SEVERITY_WARNING)	Communication has been switched from the primary to the stand-by unit(s). The server returns this message when a hot changeover has occurred.

Error	Description
GENERIC_MSG_OVERRUN (0x001C   SEVERITY_ERROR)	A response was longer than the response buffer. If this error occurs on serial communication drivers, garbled characters may be received. Check the communication link and the baud rate of the driver.
GENERIC_BAD_PARAMETER (0x001D   SEVERITY_ERROR)	There is a configuration error, e.g. invalid special options have been set.
GENERIC_STANDBY_ERROR (0x001E   SEVERITY_WARNING)	There is an error in a stand-by unit.
GENERIC_NO_RESPONSE (0x001F   SEVERITY_ERROR)	There is no response from the communications server.
GENERIC_UNIT_REMOTE (0x0020   SEVERITY_ERROR)	Cannot talk with remote unit (for example dial-up I/O devices). Only used for scheduled I/O devices.
GENERIC_GENERAL_WARNING (0x0024   SEVERITY_WARNING)	The driver is performing the action requested, but needs to notify of a potential problem. For example, some drivers may use this to warn of stale data.

## Standard driver errors

The following errors are low-level errors which are generic to all CitectSCADA drivers. These errors are all mapped to Generic errors so that CitectSCADA can recognize them. Most drivers also have a set of driver specific errors in addition to these errors.

Error	Description
0 (0x00000000) NO_ERROR	No error condition exists.
1 (0x00000001) DRIVER_CHAR_OVERRUN	Transmitted characters could not be received fast enough. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
2 (0x00000002) DRIVER_CHAR_PARITY	Parity error in received characters. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
3 (0x00000003) DRIVER_CHAR_BREAK	A break was detected in the receive line. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
4 (0x00000004) DRIVER_CHAR_FRAMING	Framing error. Check the baud rate. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
5 (0x00000005) DRIVER_MSG_OVERRUN	The message received from the device was too long. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
6 (0x00000006) DRIVER_BAD_CRC	Checksum in received message does not match the calculated value. Error mapped to Generic Error GENERIC_INVALID_RESPONSE.
7 (0x00000007) DRIVER_NO_STX	Start of text character not present. Error is mapped to Generic Error GENERIC_INVALID_RESPONSE.
8 (0x00000008) DRIVER_NO_ETX	End of text character not present. Error is mapped to Generic Error GENERIC_INVALID_RESPONSE.
9 (0x00000009) DRIVER_NOT_INIT	Driver has not been initialized. This error is mapped to Generic Error GENERIC_UNIT_OFFLINE.

Error	Description
10 (0x0000000A) DRIVER_BAD_TRANSMIT	Cannot transmit message. This error is mapped to Generic Error GENERIC_UNIT_OFFLINE.
11 (0x0000000B) DRIVER_CANNOT_RESET	Cannot reset serial driver. This error is mapped to Generic Error GENERIC_CHANNEL_OFFLINE.
12 (0x0000000C) DRIVER_BAD_LENGTH	Response length is incorrect. This error is mapped to Generic Error GENERIC_GENERAL_ERROR.
13 (0x0000000D) DRIVER_MSG_UNDERRUN	Message length too short. This error is mapped to Generic Error GENERIC_INVALID_RESPONSE.
15 (0x0000000F) DRIVER_INVALID_COMMAND	The command from the server is invalid. This error is mapped to Generic Error GENERIC_INVALID_COMMAND.
16 (0x00000010) DRIVER_NO_TIMER	Cannot allocate timer resource for the driver. This error is mapped to Generic Error GENERIC_HARDWARE_ERROR.
17 (0x00000011) DRIVER_NO_MORE_CHANNELS	Too many channels specified for device. This error is mapped to Generic Error GENERIC_NO_MORE_CHANNELS.
18 (0x00000012) DRIVER_BAD_CHANNEL	The channel number from the server is not opened. This error is mapped to Generic Error GENERIC_BAD_CHANNEL.
19 (0x00000013) DRIVER_CANNOT_CANCEL	Command cannot be cancelled. This error is mapped to Generic Error GENERIC_CANNOT_CANCEL.
20 (0x00000014) DRIVER_CHANNEL_OFFLINE	The channel is not online. This error is mapped to Generic Error GENERIC_CHANNEL_OFFLINE.
21 (0x00000015) DRIVER_TIMEOUT	No response have been received within the user configure time. This error is mapped to Generic Error GENERIC_UNIT_TIMEOUT.
22 (0x00000016) DRIVER_BAD_UNIT	The unit number from the server is not active or is out of range. This error is mapped to Generic Error GENERIC_UNIT_OFFLINE.
23 (0x00000017) DRIVER_UNIT_OFFLINE	The unit is not online. This error is mapped to Generic Error GENERIC_UNIT_OFFLINE.
24 (0x00000018) DRIVER_BAD_DATA_TYPE	The data type from the server is unknown to the driver. This error is mapped to Generic Error GENERIC_INVALID_DATA_TYPE.
25 (0x00000019) DRIVER_BAD_UNIT_TYPE	The unit type from the server is unknown to the driver. This error is mapped to Generic Error GENERIC_INVALID_DATA_TYPE.
26 (0x0000001A) DRIVER_TOO_MANY_UNITS	Too many units specified for channel. This error is mapped to Generic Error GENERIC_TOO_MANY_UNITS.
27 (0x0000001B) DRIVER_TOO_MANY_COMMANDS	Too many commands have been issued to the driver. This error code can also occur if you are running a restricted version of a driver (i.e. one that will run for a limited time) for all issued reads and writes. This error is mapped to Generic Error GENERIC_TOO_MANY_COMMANDS.
29 (0x0000001D) DRIVER_CMD_CANCELED	Command is cancelled. This error is mapped to Generic Error GENERIC_COMMAND_CANCELLED.
30 (0x0000001E) DRIVER_ADDRESS_RANGE_ERROR	The address/length is out of range. This error is mapped to Generic Error GENERIC_ADDRESS_RANGE_ERROR.
31 (0x0000001F) DRIVER_DATA_LENGTH_ERROR	The data length from the server is wrong. This error is mapped to Generic Error GENERIC_INVALID_RESPONSE.

Error	Description
32 (0x00000020) DRIVER_BAD_DATA	Cannot read the data from the device. This error is mapped to Generic Error GENERIC_INVALID_DATA.
33 (0x00000021) DRIVER_DEVICE_NOT_EXIST	Device specified does not exists. This error is mapped to Generic Error GENERIC_HARDWARE_ERROR.
34 (0x00000022) DRIVER_DEVICE_NO_INTERRUPT	Device specified does not support interrupt. This error is mapped to Generic Error GENERIC_HARDWARE_ERROR.
35 (0x00000023) DRIVER_BAD_SPECIAL	Invalid special options in port database. This error is mapped to Generic Error GENERIC_BAD_PARAMETER.
36 (0x00000024) DRIVER_CANNOT_WRITE	Cannot write to variable. This error is mapped to Generic Error GENERIC_GENERAL_ERROR.
37 (0x00000025) DRIVER_NO_MEMORY	The driver has run out of memory and cannot continue execution. Minimize buffer and queue allocation or expand the computer's memory (physical or virtual memory). This error is mapped to Generic Error GENERIC_NO_MEMORY.

## Scheduled Communications

CitectSCADA allows you to schedule communications with your I/O devices (regardless of the type of connection: modem, radio link, etc.). For example, if you have multiple I/O devices on a single network or line, you can schedule reads so that critical I/O devices are read more often than non-critical I/O devices. Alternatively, a water supplier with radio link connections to dam level monitors might schedule hourly level reads from CitectSCADA in order to conserve band-width.

See Also [Specifying a schedule](#)  
[Writing to the scheduled I/O device](#)  
[Reading from the scheduled I/O device](#)

### Specifying a schedule

To configure scheduled communications with an I/O device, you must flag it as a "Scheduled" device. This is done using the **Express Communications Wizard**. If your I/O device is not capable of scheduled communications, the scheduling options will not be presented by the wizard.

**Note:** Scheduled communications will not work for drivers that are designed to handle Report-By-Exception protocols. For communicating with your I/O device outside of schedule use the [IODeviceControl](#) function.

**To schedule communications with an I/O device:**

- 1 Select the Project Editor (or press the Project Editor icon).
- 2 Choose **Communication | Express Wizard** or open Citect Explorer.
- 3 Double-click the **Express I/O Device Setup** icon in the Communications folder of the current project.

- 4 Follow the instructions to work through the screens, selecting the relevant I/O device, and so on. When the scheduling screen displays, check the **Connect I/O Device to PSTN** box, even if your I/O device is not connected via a modem (but leave the Phone number to dial and Caller ID fields blank).
- 5 Fill out the schedule fields to achieve the desired schedule. For example (all based on a **Synchronize at** time of 10:00:00).

If you enter 12:00:00 in the **Repeat every** field, and start your project at 9am, the I/O server will communicate with the I/O device at 10am, then once every 12 hours after that, i.e. 10pm, then again at 10am of the following day, etc.

- If you enter 12:00:00, and start your project at 4pm, the I/O server will communicate with the I/O server at 10pm, then again at 10am of the following day, etc. i.e. it will assume that communications were established at 10am, so it continues as if they had been, communicating once every 12 hours after 10am.
- If you enter 3 days, and start your project at 9am on a Wednesday, the I/O server will communicate with the I/O device at 10am, then once every 3 days after that, i.e. 10am on the following Saturday, then at 10am on the following Tuesday, etc.
- If you enter the 6th of December in the **Repeat every**, and start your project during November, the I/O server will communicate with the I/O device at 10am on December 6, then again on December 6 of the following year, and so on.

- 6 Select **On Startup** for a permanent connection. To disconnect a permanent connection, you must call the `IODeviceControl()` function with type 8.
- 7 If your I/O device is not connected via a modem, you must go to the Ports form (select Ports from the Communication menu) and change the Port number to the actual number of the COM port.

See Also [Writing to the scheduled I/O device](#)

## Writing to the scheduled I/O device

Whenever an I/O device is actively communicating (as per its schedule), you can write to it directly. However, if you try to write to it when it is not communicating, your write request will be queued until it is. For example, you might decide to schedule one write per hour. If someone at a Display Client changes a tag's value during that hour, that change will not be written to the I/O device until the hour has expired.

As write requests are not written to the I/O device until it is communicating, you should ensure that all pending writes have been written before shutting down.

**Note:** Don't control hardware using a scheduled I/O device, as the exact state of the hardware may not be known. Although you can read the state from the cache, it may have changed since the cache was created.

See Also [Reading from the scheduled I/O device](#)

## Reading from the scheduled I/O device

When the I/O server initiates communication with the I/O device, it immediately writes any queued write requests, then reads all the I/O device's tags. These values are then stored in a cache so that you can still access them between communications.

**Note:** Because the I/O server reads all tags on initiation of communication, the point count is increased, even though some of the tags read may not actually be used.

### Keeping data up-to-date during prolonged connections

Normally, communication is terminated as soon as all read and write requests are complete. Sometimes, however, you may want to prolong the communication (for example by calling the `IODeviceControl()` function with Type 7).

In this situation (if Read-Through Caching is disabled), when client computers request device data from the I/O server, it retrieves the data from its cache, not from the I/O device. This occurs even though the I/O server maintains a connection to the device.

To retrieve fresh data from the I/O device, you can force a periodic read using `Cicode`, or change the cache timeout by setting the `IODeviceControl()` function to Type 11.

For example:

```
INT hTask;
// Initiate communications and read tags.
// Sleep time will depend on how fast your
// modems connect.
FUNCTION
DialDevice(STRING sDevice)
INT bConnected = 0;
INT nRetry = 5;
    hTask = TaskHnd("");
    IIODeviceControl(sDevice, 7, 0);
    Sleep(20);
    WHILE bConnected <> 1 AND nRetry > 0 DO
        bConnected = IIODeviceInfo(sDevice, 18);
        nRetry = nRetry - 1;
        Sleep(10);
    END

    IF bConnected = 1 THEN
```



```

        WHILE TRUE DO
            Sleep(2);
            IODeviceControl(sDevice, 16, 0);
        END
    END
END
// Kill the read task and terminate the connection.
FUNCTION
HangupDevice(STRING sDevice)
    TaskKill(hTask);
    IODeviceControl(sDevice, 8, 0);
END

```

You can also force the I/O server to read data directly from an I/O device by enabling Read-Through Caching. With `[Dial]ReadThroughCacheset`, while the I/O server is connected to a device it will supply data to requesting clients directly from the device. The cache is not updated during this time, but is refreshed with the most recent device data just before the server disconnects.

**Note:** If using modems, you might need to adjust or deactivate the inactivity timer in your modems to stop them from disconnecting whilst no data is being read. The inactivity timer is controlled by the S30 register, if your modem doesn't support this register, please consult your modem's manual.

#### Avoiding unnecessary multiple reads of I/O device data

To avoid unnecessary reads of an I/O device, you can use data caching to temporarily store data read from the device in the memory of the I/O server. This means that if the I/O server receives more than one request for device data in a short time period, instead of contacting the I/O device a second time and reading identical data, it can retrieve the data from the cache.

## Communicating with Diallable Remote I/O Devices

A diallable remote I/O device is one which is connected to CitectSCADA through a PSTN (Public Switched Telephone Network), and is accessible to CitectSCADA only through pre-selected and pre-configured modems.

Once connected, CitectSCADA can write to, and read from, diallable remote I/O devices just as it does with any other I/O device: local or remote.

Communications can be:

- **On request:** initiated by CitectSCADA using `IODeviceControl()` or by the remote I/O device (for instance, to report an alarm condition).
- **Periodic:** for instance, to transfer the logged events for a period.
- **Permanent:** for instance, to monitor and control the water level at a remote dam.

The only limiting factor would be an inability to connect a modem to an I/O device due to incompatible communications capabilities. Many I/O devices have fixed settings and can only communicate at a pre-set rate determined by the manufacturer. If a modem cannot match these settings, communication cannot be established.

To make communications setup easier, you can connect diallable remote I/O devices with identical communications to the same modem and port. Where I/O devices are connected to the same modem, CitectSCADA can communicate with each I/O device one after the other using the same phone connection, rather than hanging up and re-dialling. This reduces the number of necessary telephone calls and increases the speed and efficiency of communications.

You must have at least one modem at the I/O server end, and at least one at the I/O device end.

See Also [Modems at the I/O server](#)  
[Modems at the I/O device](#)  
[I/O device constraints for multi-dropping](#)  
[Configuring multidrop remote I/O devices](#)  
[I/O server redundancy for diallable remote I/O devices](#)  
[Trouble-shooting diallable remote I/O device communications](#)

## Modems at the I/O server

To decide how many modems to use at the I/O server end, decide what function each modem will perform. A single modem can do any one of the following functions:

Dial-out	Makes calls to remote I/O devices in response to a CitectSCADA request; for example, scheduled, event-based, operator request, and so on. Also returns calls from remote I/O devices.
Dial-in	Only receives calls from remote I/O devices, identifies the caller, then hangs up immediately so it can receive other calls. CitectSCADA then returns the call using a dial-back or dial-out modem.
Dial-back	Only returns calls from remote I/O devices.
Dial-in and dial-out	Receives calls from remote I/O devices and makes scheduled calls to remote I/O devices.
Dial-in and dial-back	Receives and returns calls from remote I/O devices.

Your modem setup depends on your system requirements. When making your decision, you should consider the following guidelines:

- If you need to communicate with multiple remote I/O devices at once, you will need a separate modem for each I/O device. Otherwise you'll have to wait as I/O devices are contacted one after the other, and incoming calls may be held up.
- If you have scheduled requests to I/O devices and you also need to urgently return calls from remote I/O devices that dial in, you will need at least one

modem for each of these functions. For example, if you have a large number of remote I/O devices and you require fast responses by CitectSCADA, you should provide an additional modem at the I/O server. This would reduce the chances of it being engaged when an I/O device dials in (say, with an urgent alarm).

- In a big system with many remote I/O devices or a system where calls from remote I/O devices can be critical, it's a good idea to dedicate at least one modem to Dial-Back. This will give you quick responses to Dial-In calls (from remote I/O devices). It also means your dial-out schedules won't be disrupted (if you use the same modem for returning calls and scheduled calls, the scheduled calls are forced to wait until the dial-back call is complete).

See Also [Modems at the I/O device](#)  
[Example configurations for modems at the I/O server](#)

## Modems at the I/O device

You can have multiple I/O devices connected to a single modem if they have the same communication requirements (phone number, [baud rate](#), data bits, stop bits, and parity). A separate port and modem must be used for each remote I/O device with different communication requirements.

When deciding how many modems to use, you should consider the following:

- Once an I/O device has been contacted, the connection can be retained for other I/O devices in the group. If the I/O server needs to request data from another I/O device with the same communication details, it will wait until the current request has been completed, then it will use the same connection to make the second request.
- You can configure your modem to initiate telephone calls (call CitectSCADA), and/or receive telephone calls. This is done independently of CitectSCADA.

**Note:** Wherever your modem is, you must ensure that its **Data bits**, **Parity**, **Stop bits**, and **Serial Port Speed** settings are compatible with the remote I/O device as defined by the device's manufacturer.

See Also [Modems at the I/O server](#)

## Example configurations for modems at the I/O server

The examples below demonstrate how to set up the modems at your I/O server to accommodate different combinations of I/O devices.

### Example 1

All your remote I/O devices have the same communication requirements (data bits, stop bits, parity, and baud rate) - 19200 8 E 1.

You don't expect any critical calls from your I/O devices, or you only have a few remote I/O devices. This means you can use a single modem at the I/O server

end. This modem would be set up to answer and return incoming calls and make scheduled and other CitectSCADA initiated calls.

To configure your modem, define it in Windows. Assuming that the logical modem is called 'Standard Modem', configure it as follows:

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM1	Standard Modem	19200	8	E	1

You would then configure it in CitectSCADA as a [dial-out modem](#) and [dial-in modem](#):

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem	TRUE	TRUE	FALSE

### Example 2

In this example, your I/O devices use a total of two different communication specifications - 9600 7 O 1 and 19200 8 E 1.

You don't expect important calls from I/O devices or you have only a few I/O devices. This means you can get by with a single modem at the I/O server end. This modem has to receive and return calls from all I/O devices as well as initiate calls (dial out) to all I/O devices.

To configure your modem, you must first define it in Windows (through the Windows Control Panel). Remember, you're not just defining the physical modem here. You have to define a separate Windows (virtual) modem for each communication specification.

So far, this gives you two virtual modems - one for 9600 7 O 1 and one for 19200 8 E 1. However, Windows won't let you define both of these modems as dial-in. It only lets you define one dial-in modem per port. If you choose the first, it won't be able to receive calls with the second, and vice versa.

This means you have to set up a separate virtual modem that can answer calls no matter which communication specification is used. This modem would be set with a generic communication specification of 9600 8 N 1.

So in Windows, you'll end up with three logical modems (two for Dial-Out and one for Dial-In). Assuming that the logical modems are called 'Standard Modem' to 'Standard Modem #3', you would configure them as follows:

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM1	Standard Modem	9600	7	O	1
COM1	Standard Modem #2	19200	8	E	1
COM1	Standard Modem #3	9600	8	N	1

You would then configure the modems in CitectSCADA as follows.

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem	TRUE	FALSE	FALSE
Standard Modem #2	TRUE	FALSE	FALSE
Standard Modem #3	FALSE	TRUE	FALSE

### Example 3

In this example, there are five different communications frameworks - 9600 7 O 1, 19200 8 E 1, 4800 8 N 1, 9600 8 N 1, and 19200 8 N 1.

If you expect important calls from I/O devices or you have many I/O devices, you would set up three modems at the I/O server end:

- One on COM3 dedicated to receiving calls from **9600 7 O 1** I/O devices.
- One on COM2 for dialling out to **4800 8 N 1**, **9600 8 N 1**, and **19200 8 N 1** I/O devices.
- One on COM1 for dialling out to **9600 7 O 1** and **19200 8 E 1** I/O devices.

The two dial-out modems would return calls as well as initiate calls in response to scheduled requests, and so on.

To configure your modems, you must first define them in Windows (through the Windows Control Panel). Remember, you're not just defining the physical modem here. You have to define a separate Windows (virtual) modem for each communication framework.

Assuming that the logical modems are called 'Standard Modem' to 'Standard Modem #6', you would configure them as follows:

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM1	Standard Modem	9600	7	O	1
COM1	Standard Modem #2	19200	8	E	1
COM2	Standard Modem #3	4800	8	N	1
COM2	Standard Modem #4	9600	8	N	1
COM2	Standard Modem #5	19200	8	N	1
COM3	Standard Modem #6	9600	7	O	1

You would then configure the modems in CitectSCADA as follows:

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem	TRUE	FALSE	FALSE

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem #2	TRUE	FALSE	FALSE
Standard Modem #3	TRUE	FALSE	FALSE
Standard Modem #4	TRUE	FALSE	FALSE
Standard Modem #5	TRUE	FALSE	FALSE
Standard Modem #6	FALSE	TRUE	FALSE

#### Example 4

In this example, your I/O devices use three different communication frameworks: 9600 7 O 1, 19200 8 E 1, and 9600 8 N 1. However, in this example, you are expecting critical calls from I/O devices, so you need a modem dedicated to returning calls.

Here you must configure your modems like this:

- One modem on COM1 to dial all remote I/O devices (for scheduled calls, and so on).
- One modem on COM2 to receive calls from remote I/O devices.
- One dedicated modem on COM3 to return these calls.

To configure your modems, first define them in Windows (through the Windows Control Panel). Remember, you're not just defining the physical modem here: you must define a separate Windows (virtual) modem for each communication framework. This means you have to configure:

- Three logical modems on the port to which the physical dial-out modem is attached.
- One logical modem on the port to which the physical [dial-in modem](#) is attached.
- Three logical modems on the port to which the physical dial-back modem is attached.

Assuming that the required total of seven logical modems are called 'Standard Modem' through to 'Standard Modem #7', configure these modems as follows:

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM1	Standard Modem	9600	7	O	1
COM1	Standard Modem #2	19200	8	E	1
COM1	Standard Modem #3	9600	8	N	1
COM2	Standard Modem #4	9600	8	N	1

Port	Modem Name	Max. Speed	Data Bits	Parity	Stop Bits
COM3	Standard Modem #5	9600	7	O	1
COM3	Standard Modem #6	19200	8	E	1
COM3	Standard Modem #7	9600	8	N	1

You would then configure the modems in CitectSCADA as follows:

Modem Name	Dial-out	Dial-in	Dial-back
Standard Modem	TRUE	FALSE	FALSE
Standard Modem #2	TRUE	FALSE	FALSE
Standard Modem #3	TRUE	FALSE	FALSE
Standard Modem #4	FALSE	TRUE	FALSE
Standard Modem #5	FALSE	FALSE	TRUE
Standard Modem #6	FALSE	FALSE	TRUE
Standard Modem #7	FALSE	FALSE	TRUE

## I/O device constraints for multi-dropping

If you are multi-dropping off a single modem, you should use your I/O devices to issue the caller ID, not the modem. The problem with using the modem to issue the ID is that it will send the same ID no matter which I/O device the call is relevant to. This makes it difficult to identify the I/O device that triggered the call.

By using the I/O device to issue the ID, the I/O server will receive a unique caller ID for each I/O device. However, not all I/O devices are capable of issuing caller IDs. If multi-dropping, use I/O devices that can issue caller IDs.

### To configure diallable remote I/O devices for communication with CitectSCADA:

- 1 Run the Express Communications Wizard.
- 2 Complete the wizard, selecting the relevant I/O server, then the I/O device, creating each new instance when required.
- 3 On the Scheduling page of the wizard, select the **Connect I/O Device to PSTN** check box.
- 4 Select an appropriate schedule for CitectSCADA to communicate with the remote I/O device. (For a permanent connection - whenever CitectSCADA is running - select **On Startup**.) For example (all based on a **Synchronize at** time of 10:00:00):
  - If you enter **12:00:00** in the **Repeat every** field, and start your project at 9am, the I/O server will communicate with the I/O device at 10am, then

once every 12 hours after that; that is, 10pm, then again at 10am of the following day, and so on.

- If you enter 12:00:00 and start your project at 4pm, the I/O server will communicate with the I/O server at 10pm, then again at 10am of the following day, and so on. CitectSCADA will assume that communications were established at 10am, so will continue as if they had been, communicating once every 12 hours after 10am.
  - If you enter 3 days and start your project at 9am on a Wednesday, the I/O server will communicate with the I/O device at 10am, then once every 3 days after that; that is, 10am on the following Saturday, then at 10am on the following Tuesday, and so on.
  - If you enter the 6<sup>th</sup> of December in the **Repeat every** field and start your project during November, the I/O server will communicate with the I/O device at 10am on December 6, then again on December 6 of the following year, and so on.
- 5 Select **On Startup** for a permanent connection. To disconnect a permanent connection, call the `IODeviceControl()` function with type 8.
  - 6 Type in the phone number required for CitectSCADA to dial the remote modem attached to the remote I/O device. Include any pre-dial numbers necessary to obtain connection to an outside PSTN line (dial tone) before dialling (for example, 0 (zero)) - if appropriate.
  - 7 On the next wizard page, if the device is configured to dial-in to CitectSCADA, create a unique identifying caller name for the remote I/O device so that it can be identified by CitectSCADA.
  - 8 Follow the instructions on the next page of the wizard and click **Finish**.

See Also [Configuring multidrop remote I/O devices](#)

## Configuring multidrop remote I/O devices

Multidropping remote I/O devices from the same remote modem enables CitectSCADA to communicate with each I/O device one after the other, using the same phone connection, rather than hanging up and re-dialling.

Although you can configure multidrop remote I/O devices using the Express Communications Wizard, we recommend that you do it manually. The wizard would create a new port for each I/O device. This would mean you couldn't have any more than 255 I/O devices.

- 1 Run the Express Communications Wizard.
- 2 Configure every other I/O device manually.
- 3 Open the Citect Project Editor.
- 4 Select **Communications** | **I/O Server** and scroll to the I/O server that will be communicating with the I/O device.



- 5 Select **Communications** | **I/O Devices**. Complete the dialog box.
- 6 To increase the efficiency and capacity of your system you can allocate the same port name to all I/O devices with the same communication settings.

**Note:** If you are multi-dropping and you want to be able to dial in to the I/O server, you should use your I/O devices to issue the caller ID, not the modem. The problem with using the modem to issue the ID is that it will send the same ID no matter which I/O device the call is relevant to. This makes it difficult to identify the I/O device that triggered the call.

By using the I/O device to issue the ID, the I/O server will receive a unique caller ID for each I/O device. However, not all I/O devices are capable of issuing caller IDs. If multi-dropping, use I/O devices that can issue caller IDs.

#### To set up a modem connected to your diallable remote I/O devices:

You can connect multiple I/O devices to the same modem. This means CitectSCADA can communicate with these I/O devices one after the other using the same phone connection, rather than hanging up and re-dialling. This will reduce the number of necessary telephone calls and increase the speed and efficiency of communications.

- 1 Connect the modem to a PC with a telephony program installed (e.g. HyperTerminal or PhoneDialler). This is where you will configure the modem to answer calls from CitectSCADA and/or initiate calls.
- 2 If the modem is required to make calls to CitectSCADA, configure it to initiate the phone call to a pre-determined CitectSCADA I/O server Dial-In type modem (following manufacturer instructions).
- 3 Depending on your hardware either the modem or an intelligent PLC can be responsible for initiating calls to CitectSCADA and identifying the caller. Whichever is responsible must have a caller ID set. The caller ID can be any combination of alpha-numeric characters and/or the character '\_' (underscore).

Some modems have dip-switch settings, and some have initiation strings which can include auto-diallable numbers that are stored within the modem's non-volatile memory. Consult the manual provided with the modem for exact details.

You can use either the Express Communications Wizard or the I/O devices form to set the caller ID for an I/O device.

If multi-dropping off a single modem, use your I/O devices to issue the caller ID, not the modem. The problem with using the modem to issue the ID is that it will send the same ID no matter which I/O device the call is relevant to, making it difficult to identify which I/O device triggered the call.

By using the I/O device to issue the ID, the I/O server will receive a unique caller ID for each I/O device. However, not all I/O devices are capable of

issuing caller IDs. If you are multi-dropping, you should use I/O devices that can issue caller IDs.

- 4 Set the modem's **Data bits**, **Parity**, **Stop bits**, and **Serial-Rate** to match manufacturer specifications for communication with the I/O devices.

Some modems do not allow you to manipulate their communications settings via methods such as extended AT commands or dip switches. If this is the case, the only way of setting the required values is to communicate with the modem *using* the values (for example, via Hyperterminal). Once this is done, the modem remembers the last values used to communicate with its serial port.

- 5 Connect the modem to the I/O devices.

To configure a modem at the I/O server you must set it up in Windows and then set it up in CitectSCADA.

If all of your I/O devices are the same, you only have to do this once for each modem. However, if your I/O devices talk using different communication specifications (data bits, parity, stop bits, and serial-rate), your modem has to be able to talk using each of these details as well. To set this up, you have to create a modem in Windows and CitectSCADA for each specification. (See [Example configurations for modems at the I/O server](#))

#### To set up a modem in Windows

- 1 Each modem connected to a CitectSCADA I/O server PC must FIRST be configured within Windows using **Start | Settings | Control Panel | Phone and Modem Options**.
- 2 Select the **Modems** tab, and click **Add** to launch the Install New Modem wizard.
- 3 Check the box labelled **Don't detect my modem; I will select it from a list**, then click **Next**.
- 4 Select **Standard Modem Types** in the list of manufacturers.  
**Note:** Do not select a brand name modem from the Manufacturers list, even if the name of the modem you're installing is included in the list. Do not click **Have Disk**.
- 5 Select the **Standard xxxx bps modem** rate from the list of models to exactly match the bit per second rate of the I/O device that is going to be communicating via this modem. Check the device to determine the device communication rate. If you are still unsure, select the 9600bps model. This can be changed later if required.
- 6 Do not click **Have Disk**. Click **Next**.
- 7 Select the **COMx Port** that the modem is connected to. Click **Next**.

- 8 Click **Finish**. Windows displays the modem in the list of modems on the Modems Properties form.
- 9 Note that no option was given for the selection and setting of the Data bits, Parity, Stop bits information. The Modems wizard automatically defaults to 8-none-1 for all Standard Modem types. To change these settings to match the Data bits, Parity, Stop bits requirements of the remote I/O device, select a modem in the list, then click the **Properties** button.
- 10 Click the **Advanced** tab and click **Change Default Preferences**.
- 11 Click the **Advanced** tab at the next dialog to gain access to the Data bits, Parity, Stop bits settings for the modem.
- 12 Change the Data bits, Parity, and Stop bits settings using the drop-down options, so that they exactly match those being used by the remote I/O device and its remote modem. Don't change any advanced settings. (The default is Hardware flow control.)
- 13 Click **OK**. If a modem of the same rate is installed to the same port as an existing modem, Windows asks for confirmation that you want to install the same thing more than once. Click **Yes** to install a duplicate copy of the modem.
- 14 Preconfigure the modem(s) to be used at the remote diallable I/O device(s). These will be used to test modem configuration settings in the next step.
- 15 With CitectSCADA not running, confirm that the local and remote modems will properly communicate with each other by using a terminal communications program such HyperTerminal or PhoneDialer (both supplied with Windows).

Once the modem(s) are set up and tested with proven communications in Windows, they can then be set up in CitectSCADA.

#### To set up a modem in CitectSCADA:

Before completing this procedure, make sure you have set up your modem in Windows (as described above).

- 1 Open the **Citect Project Editor**.
- 2 Select **Communications** | **I/O Server** and scroll to the I/O server the modem is attached to.
- 3 Select **Communications** | **Modems**.
- 4 Complete the dialog box.

CitectSCADA allows you to set up a maximum of 256 modems on the I/O server for communication with remote diallable I/O devices. Before CitectSCADA can successfully establish communication, any targeted remote I/O devices must also be properly configured within CitectSCADA on the I/O server.

See Also [Modems Properties](#)

## Modems Properties

Using this form, you can configure the modem at your I/O server to make and receive calls from remote diallable I/O devices.

Modems have the following properties:

### **Server Name (16 Chars.)**

The name of the I/O server to which the modem is attached.

### **Modem Name (64 Chars.)**

The name of the modem you are configuring (as it appears in the Windows Control Panel | Phone and Modem Options).

### **Comment (48 Chars.)**

Any useful comment.

### **Use this modem to make outgoing calls**

Determines whether this modem will be used to initiate calls from the I/O server to a diallable remote I/O device. (Dial-Out)

This may include calls that are scheduled, event driven, or in response to I/O devices that dial in.

### **Use this modem to answer incoming calls**

Determines whether this modem will be used to receive calls from a diallable remote I/O device. (Dial-In)

**Note:** The following fields are implemented with extended forms (press F2).

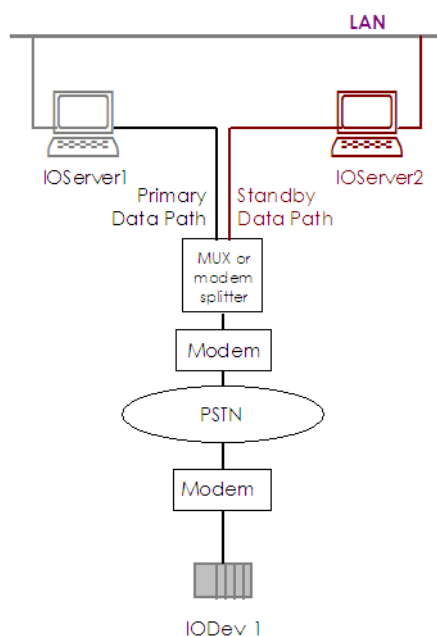
### **Use this modem to call back I/O devices**

Determines whether this modem will be used to initiate calls from the I/O server to a diallable remote I/O device in response to a call received from the I/O device. (Dial-Back)

## I/O server redundancy for diallable remote I/O devices

You can change the number of rings the I/O server will wait before answering the call (using the **[Dial]RingCount** parameter). If you are using redundant I/O servers, the primary I/O server will be called by default. However, by setting **[Dial]RingCount** to a different value on each of the I/O servers, you can make the standby I/O server answer the call if the primary does not.

Consider the following setup:



If you set the ring count to 3 on IOServer1 and 4 on IOServer2, IODev\_1 will attempt to call IOServer1. If IOServer1 has not answered the call after 3 rings, IOServer2 will answer it.

See Also [Trouble-shooting diallable remote I/O device communications](#)

### Trouble-shooting diallable remote I/O device communications

The problems most often encountered when using a diallable remote I/O device communications involve speed, parity, and control signals from the connected equipment. If changing the speed and parity does not solve the problem, the modem's answering codes or command echoing might be the source of the difficulty.

Following is a list of settings that might be helpful in resolving problems. (Since not all modems support the same in commands in the same way, this is only a guide. Consult the modem manual for exact details.)

#### On the modem at the PC end

```
ATV1 //Enables long-form (verbose) result codes
ATQ0 //Result codes are sent on the RS-232 connection
ATE0 //Commands sent from the computer are not echoed back to the RS-232 connection
AT&C1 //DCD will follow carrier on the line
AT&K0 //Handshaking OFF
ATW0 //Upon connection, only DTE speed is reported
AT%C0 //Compression OFF
AT&D0 //DTR always on
```

To ensure a call from a remote I/O device does not get through while CitectSCADA is shut down (and losing the data being forwarded), set the following parameter to zero:

```
ATS0 = 0 // Auto answer OFF
```

Note, however, this will also impact applications that might use the modem other than CitectSCADA, as the modem cannot answer a call while CitectSCADA is not driving its functionality.

#### **On the modem at the I/O device end**

```
ATV0 //Enables short-form result codes
ATQ1 //No result codes are sent on the RS-232 connection
ATE0 //Commands that are sent from the computer are not echoed
back to the RS-232 connection
AT&C1 //DCD will follow carrier on the line
AT&K0 //Handshaking OFF
ATW0 //Upon connection, only DTE speed is reported
AT%C0 //Compression OFF
AT&D0 //DTR always on
ATS0 //Set to greater than 0 (sets the number of rings required before the modem answers
an incoming call).
```

#### **Alternative (backward compatibility) method of permanent connection**

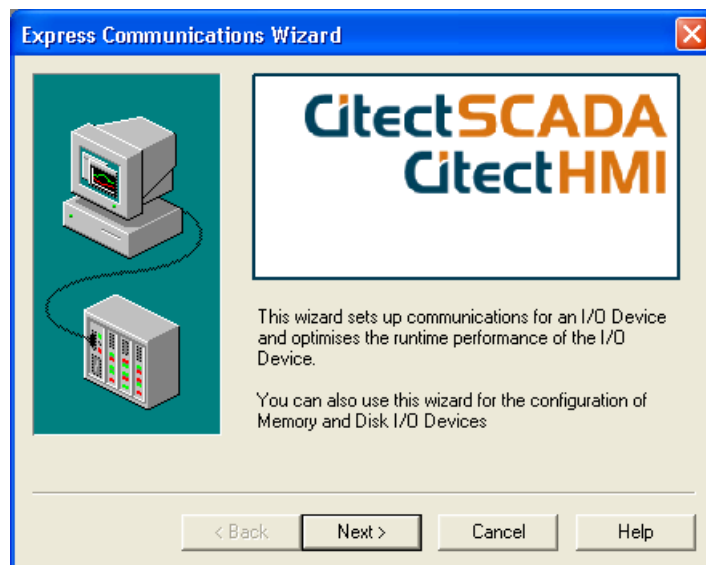
If you are setting up your modem to dial a diallable remote I/O device, you should follow the procedures described in [Communicating with Diallable Remote I/O Devices](#). This method is available for backward compatibility.

# Chapter 36: Using the Communications Express Wizard

---

You use the Express Communications Wizard to set up communications with your I/O devices. Based on your selections, the Express Communications Wizard provides default values and a setup tailored to your I/O device communications requirements.

To access the Communications Express Wizard, from the Citect Project Editor choose **Communication | Express Wizard**.



See Also [Express Communications Wizard - introduction](#)

## Express Communications Wizard - introduction

You will be asked to select an [I/O server](#), choose a name, and indicate the type of I/O device (External, Memory, Disk). From the list of available manufacturers you choose the manufacturer, model and communications method for the I/O device. If you are connecting external devices or using a proprietary board in your computer you may be requested to nominate addresses and [communications port](#).

After completing your setup, the Summary Page summarizes the configuration of your I/O device and/or internal boards. Click **Finish** to save the listed configuration, or click **Back** to change a previous selection.

See Also [Express Communications Wizard - Server selection](#)

## Express Communications Wizard - Server selection

Select an existing I/O servers as defined in the current Project, or create a new I/O server.

When you create a new I/O server, CitectSCADA automatically suggests the name **IOServer1**. You can enter a different name if you want. The name you specify must be 16 characters or less and use alphanumeric characters (A-Z, a-z, 0-9). You can also use the underscore character ( \_ ).

**Note:** If you add a new I/O server, run the Computer Setup Wizard on the appropriate computer before you attempt to run the project.

See Also [Express Communications Wizard - Device selection](#)

## Express Communications Wizard - Device selection

Enter the name of the new I/O device that you want to create, or accept the name **IODev** which CitectSCADA automatically suggests.

[Express Communications Wizard - I/O device type](#)

## Express Communications Wizard - I/O device type

Select the type of I/O device. You may choose an **External I/O Device** or a **Disk I/O Device**.

See Also [Express Communications Wizard - I/O device communications selection](#)

## Express Communications Wizard - I/O device communications selection

From the list of available manufacturers, select the manufacturer, model, and communications method specific to the I/O device.

If a memory or disk I/O device has been selected, the CitectSCADA Generic Protocol is included at the top of the tree.

See Also [Express Communications Wizard - TCP/IP address](#)



## Express Communications Wizard - TCP/IP address

Enter the [IP address](#) for the I/O device, in standard Internet dot format (for example, 192.9.2.60). This address is set on (or specified by) your I/O device.

The Port number and Protocol (TCP or UDP) fields have been set to the default values for the I/O device. You should change these fields only if necessary.

For details about addressing your specific I/O device, click **Driver Address Help** to browse driver-specific information for your I/O device.

See Also [Express Communications Wizard - I/O device address](#)

## Express Communications Wizard - I/O device address

Enter the address for the I/O device. What you enter in this field is determined by the type of I/O device (and protocol) used, as each has a different addressing strategy.

For details about addressing your specific I/O device, click **Driver Address Help** to browse driver-specific information for your I/O device.

See Also [Express Communications Wizard - I/O device connection schedule](#)

## Express Communications Wizard - I/O device connection schedule

This form allows you to define the details of the communications schedule for your I/O device and indicate that your I/O device is remote by checking the PSTN box.

### Connect I/O Device to PSTN

Check this box to indicate that the I/O device is a diallable I/O device (connected to a PSTN - Public Switched Telephone Network).

Even if your I/O device is not connected via a modem, you must still check this box to schedule communications (but leave the Phone number to dial and Caller ID fields blank). Once you have completed your I/O device setup using this Wizard, you must go to the Ports form and change the Port number to the actual number of the COM port.

You can choose to define the communication period in terms of **months**, **weeks**, **days**, or **hours**, **minutes**, and **seconds**. Alternatively, you can choose to communicate only at **startup** (permanent connection). Click a radio button to make your selection, then enter the start time and period as described below.

### Synchronize at

The I/O server will attempt to communicate with the I/O device at this time, and then at intervals as defined below. This time is merely a marker for CitectSCADA. If you run up your project after this time, the I/O server won't wait until the next day to begin communicating. It will operate as if your project had been running since before the start time.

### Repeat every

The time between successive communication attempts.

Examples (all based on a **Synchronize at** time of 10:00:00):

- If you enter 12:00:00 in the **Repeat every** field, and start your project at 9am, the I/O server will communicate with the I/O device at 10am, then once every 12 hours after that, i.e. 10pm, then again at 10am of the following day, etc.
- If you enter 12:00:00, and start your project at 4pm, the I/O server will communicate with the I/O server at 10pm, then again at 10am of the following day, etc. i.e. it will assume that communications were established at 10am, so it continue as if they had been - communicating once every 12 hours after 10am.
- If you enter 3 days, and start your project at 9am on a Wednesday, the I/O server will communicate with the I/O device at 10am, then once every 3 days after that, i.e. 10am on the following Saturday, then at 10am on the following Tuesday, etc.
- If you enter the 6<sup>th</sup> of December in the **Repeat every**, and start your project during November, the I/O server will communicate with the I/O device at 10am on December 6, then again on December 6 of the following year, etc.
- Select **On Startup** for a permanent connection. To disconnect a permanent connection, you must call the `IODeviceControl()` function with type 8.

### Phone number to dial

The telephone number that needs to be dialled to initiate contact with the I/O device.

**Note:** These values can also be set using the I/O Devices form in the Project Editor.

See Also

[Caller ID and commands](#)

[Express Communications Wizard - Link to external database](#)

## Caller ID and commands

### Caller ID

A unique identifier which identifies a remote I/O device when it dials back to the I/O server. The caller ID can be any combination of alpha-numeric characters and/or the character '\_' (underscore).

This ID will only be used if the I/O device initiates the call to the I/O server. If the modem initiates the call, you must set the caller ID on the modem.

**Note:** If you are multi-dropping off a single modem, use your I/O devices to issue the caller ID, not the modem. The problem with using the modem to issue the ID is that it will send the same ID no matter which I/O device the call is relevant to, which makes it hard to identify the I/O device that triggered the call.

By using the I/O device to issue the ID, the I/O server will receive a unique caller ID for each I/O device. However, not all I/O devices are capable of issuing caller IDs. If you are multi-dropping, you should use I/O devices that can issue caller IDs.

#### [Event Commands] On connect

Cicode to be executed once the modem is connected and the I/O device has come online (i.e. before any read or write requests are processed).

#### [Event Commands] On disconnect

Cicode to be executed before the connection to the I/O device is terminated (and after all read and write requests are processed).

**Note:** These values can also be set using the I/O Devices form in the Project Editor.

See Also [Express Communications Wizard - Link to external database](#)

## Express Communications Wizard - Link to external database

This screen allows you to link to an external data source.

### Link I/O Device to an external tag database

Determines whether or not you want to link the I/O device to an external data source. If you link to an external data source, CitectSCADA is updated with any changes made to the external data source when a refresh is performed.

If you cut an existing link, you can choose to make a local copy of all the tags in the data source or you can delete them from CitectSCADA's variable tags data source altogether.

### Database type

The format of the data referenced by the external data source.

**Note:** If you select the **Mitsubishi Fastlinx** database option, then select the browse button in the following **External database** field, a modal dialog will display the tree-view listing of all Mitsubishi Fastlinx Servers found on the network connected to the computer.

Selecting any database type other than Mitsubishi Fastlinx displays a standard Windows Open File dialog if the **Browse** button is used.

#### External tag database

The path and filename of the external data source for the I/O device. Alternatively, you can enter the IP address/directory, computer name, or URL of a data server, etc. (e.g. "Work.CSV" or "127.0.0.1", "139.2.4.41\HMI\_SCADA" or "http://www.abicom.com.au/main/scada" or "\\coms\data\scada").

#### Connection string

Enter a connection string to provide connection details for the data source. This is similar to an ODBC connection string. For example:

```
UserID = XXX; Password = YYY
```

or

```
ServerNode=111.2.3.44; Branch=XXX
```

Not all data sources require a connection string.

**Note:** If the **Mitsubishi Fastlinx** database option is selected, the correct connection string can be generated by selecting the browse button and filling in the form.

#### Add prefix to externally linked tags

Check this box if you want to insert a prefix in front of the names of all linked tags in your Variable.DBF.

#### Tag prefix

The prefix that will be inserted in front of the names of linked tags in your Variable.DBF (for this I/O device only). To change the prefix, you should delete it first, perform a manual refresh, then add the new prefix.

#### Automatic refresh of tags

Determines whether the linked tags in CitectSCADA's variable tags database will be updated when the external data source is changed (i.e. you manually change a field, etc.). This refresh will occur the first time you link to the data source, and then whenever you attempt to read the changed variables (e.g. you compile your project, display the variable using the Variable Tags form, or paste the tag, etc.).

Without an automatic refresh, you will need to perform a manual refresh to update the linked tags in CitectSCADA.

## Live Update

**Note:** This field is only available if you have installed one of the CitectSCADA FastLinx products. See <http://www.citect.com> for more information on FastLinx.

Controls whether or not the linked tags in CitectSCADA and an external tag database will be synchronized if either database is changed. To enable live linking, choose **Yes** from the **Live Update** menu, and ensure that the **Automatic refresh** check box is not selected. (Live Update and Automatic Refresh are mutually exclusive.)

When Live Update is enabled and the CitectSCADA variable tag database is accessed (for example, during project compilation or when a dropdown list is populated), CitectSCADA queries the external tag database to determine if it has been modified. If so, CitectSCADA merges the changes into the local variable tag database. Conversely, any changes made to the local tag variable database will be incorporated seamlessly into the external tag database.

See Also [Express Communications Wizard - Serial device](#)

## Express Communications Wizard - Serial device

Since your protocol is based on serial communications you must select which port on your computer will be used for the I/O device.

The serial ports listed have been detected from your operating system registry. If you have correctly installed a proprietary serial board (for example a [Digiboard](#)) and the associated driver, the available port will be listed.

See Also [Express Communications Wizard - Summary](#)

## Express Communications Wizard - Summary

Summarizes the I/O device setup using the information you provided. The summary shows the CitectSCADA communications setup and the recommended configuration of your I/O device and/or internal boards.



# Chapter 37: Building Your CitectSCADA Project

---

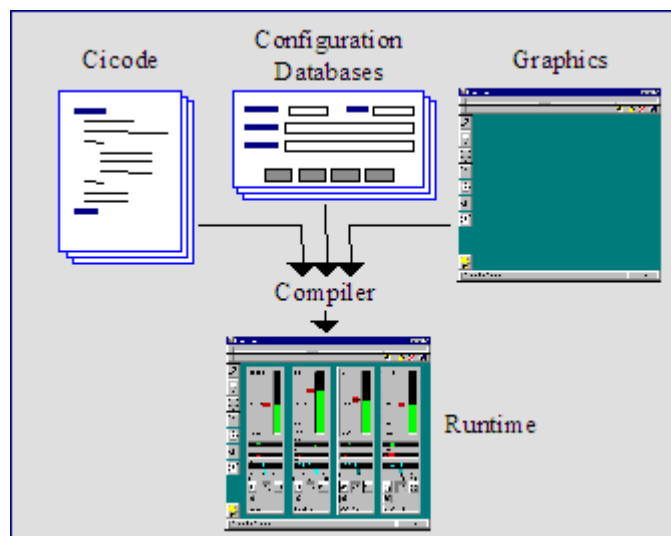
This section describes how to build your CitectSCADA project.

See Also

[Compiling the Project](#)  
[Running the System](#)  
[Running Your System Over the Internet](#)  
[Using an Alternative INI File](#)  
[Debugging the Runtime System](#)  
[Debugging I/O Devices and Protocols](#)  
[Restarting the System Online](#)  
[CitectSCADA Software Protection](#)  
[Using the CitectSCADA Kernel](#)  
[Gathering Runtime Information](#)

## Compiling the Project

The CitectSCADA compiler compiles (or builds) the elements of your project into a runtime system.



Compilation checks the project for errors and optimizes your system for fast and efficient operation. The time required to compile a project depends on its size

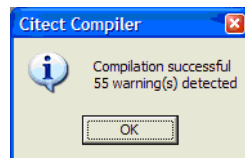
and on the speed of your computer. Typically, compiling only takes several minutes.

When the CitectSCADA compiler runs, it normally opens all files in exclusive mode. In this mode only CitectSCADA has access to the files (while the compiler is running). This improves the performance of the compiler, but can cause a problem if two people try to compile different projects at the same time, as both compilations must open the Include project. The [General] ShareFiles parameter tells the compiler to open all files in shared mode. This option allows shared network users to run the compiler at the same time, but it can increase the time required for the compilation.

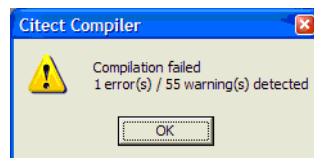
**To compile a project:**

- 1 Select the Project Editor.
- 2 Click **Compile**, or choose **File | Compile**.

A successful compile results in the following message being displayed:



An unsuccessful compile results in the following message being displayed:



Note that the Citect Compiler result dialogs report the number of errors and warnings separately to avoid confusion. A CitectSCADA project can compile successfully even though the compile may generate warnings; however, compile errors prevent a project from compiling successfully.

If there are compile errors, you must first fix the errors, and then recompile. CitectSCADA automatically compiles the project (if uncompiled) when you try to run it.

See Also [Incremental compilation](#)  
[Debugging the compilation](#)  
[Compile Error Properties](#)  
[Compile Error Messages](#)  
[Setting the Project Editor options](#)



## Incremental compilation

You can compile the project incrementally. With incremental compilation, CitectSCADA only compiles the database records that were added (or changed) since the last compilation. The remainder of the project is not re-compiled.

**Note:** Some database records are dependent on other database records. If you change a dependent record, CitectSCADA compiles the entire database.

Before you run a system on a live plant, you should perform a complete compilation (switch off Incremental Compile). When you restore a project from floppy disk, you must perform a complete compilation the first time (switch off Incremental Compile).

**To switch to Incremental Compile:**

- 1 Select the Project Editor.
- 2 Choose **Tools | Options**.
- 3 Select the **Incremental Compile** check box, and then click **OK**.

See Also [Debugging the compilation](#)

## Debugging the compilation

If the compiler detects any errors during compilation, the errors are written to an error file. The compiler will notify you of any errors as it compiles, and you can opt to cancel the compilation at any stage. If there are multiple or severe errors, the compiler might automatically cancel. Once the compiler is finished, you can locate each compile error and display information on it.

The compiler does not verify the operation of your project. Just because your project compiles does not mean it will work correctly at runtime. For example, the compiler checks that the tags you use are defined correctly, and that your Cicode has acceptable syntax. But, it does not check your tags for incorrect scaling, or that your Cicode has no potential divide by zero errors.

**Note:** Do not attempt to run your system until you have resolved all (if any) of the compile errors.

**To view compilation errors:**

- Select the Project Editor and choose **File | Compile Errors**.

**To get more information on an error:**

- 1 Click **Help** at the bottom of the Compile Errors dialog box.
- 2 Read the Help topic associated with the error.

**To locate the error (in the project):**

- Click **Go To** at the bottom of the Compile Errors dialog box.

See Also [Compile Error Properties](#)

## Compilation options

The compiler has a number of options available to help simplify the process of resolving compilation issues. Typically, these options modify the output of the compile error log, making it more practical to assess specific problems.

The options listed here can be adjusted via the [Project Editor Options dialog](#), which is accessible from the Project Editor's **Tools** menu.

### ■ Log "tag not defined" warnings during compile

If you select this option, the compiler will generate a 'tag not defined' warning in the error log for any tags detected that are not defined in the variable database.

As CitectSCADA V7.0 now allows you to include undefined tags on your graphic pages, this warning may be redundant and impractical. By unchecking this option, the warnings are still included in the displayed warning count, but they are not added to the error log.

### ■ Log deprecated warnings during compile

If you select this option, the compiler will generate a warning to identify any deprecated elements it detects in a project, i.e. any functions, parameters, or Kernel commands that are no longer supported.

By unchecking this option, the warnings are still included in the displayed warning count, but they are not added to the error log.

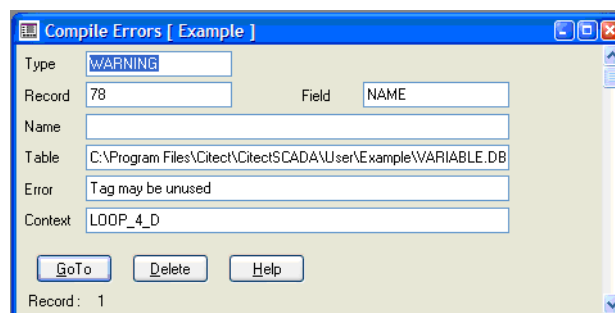
### ■ Warn about unused tags during full compile

This option enables the generation of warning entries for unused tags that are not used directly in a CitectSCADA project. The warning entries are included in the compile errors form when a full compile is run. By default this option is not selected.

See Also [Setting the Project Editor options](#)

## Compile Error Properties

You might encounter errors when you compile your CitectSCADA project. Compile errors are displayed on the Compile Errors dialog.



Compiler errors have the following properties:

**Type**

The type of error. Three types of errors can occur during compilation. These are:

Type	Meaning
ERROR	The compilation process continues, however the project will not compile successfully until you have corrected the error.
FATAL	The severity of this error is such that it halts the compilation process. The project cannot be compiled until you correct the error.
WARNING	The error was not serious enough to stop the project being compiled successfully, however you should investigate and correct the error.

**Record**

The number of the database record where the error has occurred.

**Name**

The name of the [graphics page](#), library, or report format file where the error has occurred.

**Field**

The database field where the error has occurred.

**Table**

The database table where the error has occurred.

**Error**

A brief description of the error.

**Context**

The location in the database field, report format file, or Cicode library where the error has occurred. The context of the error is enclosed in braces { . . . }.

See Also [Compile Error Messages](#)

## Compile Error Messages

You might see the error messages described below during project compilation.

Error	Description
Address on bad boundary	When reading a long or real from the memory of an I/O device, all addresses must be on odd or even boundaries. Addresses cannot be mixed. You can disable checking with the [General]CheckAddressBoundary parameter.
Analog address not supported	An INT or other analog tag is specified where a DIGITAL tag is expected. Check that the tag name is correct, or that a DIGITAL data type is specified for the tag.
Array size exceeded	A tag is indexed but that tag is not declared as an array, or no index has been specified when a tag is declared as an array, or the wrong number of dimensions are specified for an array, or more than four dimensions are specified for an array.

Error	Description
Bad analog format	The format is incorrectly specified for an analog variable. Check the Format field in the Variable Tags form.
Bad factor specification	A Cicode expression that contains an invalid expression has been used. Check the syntax of the expression.
Bad floating point value	A floating-point number cannot be found where one is expected, or the floating-point value is out of range.
Bad I/O device variable	The variable tag address is not a valid format for the I/O device protocol you are using; check the address format. (See the <b>Data Types</b> topic in the Help for each supported I/O device for a list of appropriate address formats for that device).
Bad integer value	An integer value cannot be found where one is expected, or the integer value is out of range.
Bad point limit	The incorrect point limit is specified in the <code>citect.ini</code> file. The point limit must correspond to your CitectSCADA license.
Bad raw data type	An invalid raw data type or a mismatch of data types is specified, for example, an attempt was made to convert an integer into a string.
Bad string conversion parameter	An invalid format parameter is specified in a string conversion. Check the format specification of the variable in the Variable Tags form.
Cannot compile all functions	Error(s) were detected while compiling the function library.
Cannot open file	The file cannot be opened. The file does not exist, or it has become corrupt, or your system is out of file handles.
Cannot read from file	The file cannot be read. The end of file was found, or it has become corrupt.
Cannot return value from void function	A RETURN statement cannot be used in a function that does not return a value. Remove the RETURN statement or declare a return data type for the function.
Cannot use an array inside function	You cannot declare an array within a function. Arrays can only be declared as library variables, i.e. at the beginning of the library file.
Cannot use RETURN outside of functions	A RETURN statement can only be used within a function.
Cannot write to file	The file cannot accept a write operation. The file has become corrupt or the disk is full.
Cicode data limit reached	An array in a Cicode module cannot exceed 60 KB. Reduce the size of the array.
Close bracket expected	The Cicode statement has a different number of open and close brackets. Another close bracket ')' or ']' is expected in the statement.
Close comment delimiter expected	A comment opened with /* must be closed with the */ delimiter. Add the */ delimiter or use a single line comment that starts with an exclamation mark (!) and has no end delimiter.
Close quotation mark expected	The Cicode statement has a different number of open and close quotation marks. Another close quotation mark (") is expected in the statement.
Database is empty	The database does not contain any records.
Database not found	The main project database cannot be found.
Database table full	The database is full. If the error persists, contact Citect Support.

Error	Description
Disk full	The disk is full. Remove unwanted files from the disk, or replace the existing disk with a larger disk.
DO expected	A DO statement must be used in a WHILE statement.
END expected	An END statement must be used at the end of a conditional statement or function definition.
Error reading file	An include file specified in a database field cannot be found or cannot be opened. Check that the file name is correct, and that the file has been specified correctly, i.e. <@FILENAME>.
Expression too big	An expression is too large for the compiler. Reduce the length of the expression by splitting the expression into two or more smaller expressions.
File already opened in SINGLE mode	The file has been opened by another user. Set the [General] ShareFiles parameter to 1 in the <a href="#">citect.ini file</a> to open files in shared mode.
File does not exist	The file cannot be found. Check that the file name is correct.
File is read only	An attempt was made to write to a read-only file. Check that the file name is correct or change the attributes of the file.
File locked	A file is in use by another network user.
File not indexed	The database must be indexed, but the index file associated with the database cannot be found. Pack the database.
File size error	A Cicode functions file, or Report format file, or an include file is too big. The maximum file size is 1 MB.
FUNCTION expected	A function must be declared with the FUNCTION keyword.
GLOBAL function is not allowed, use PUBLIC	You have declared a Global function within your Cicode. Global is not a valid function type. Instead, the type Public must be used.
Group not found	A group name was expected. Check that the group name is correct, or that the group has been specified correctly in the Groups form.
Include project not found	An included project (specified in the Included Projects database) does not exist. Check the name of the included project.
Incompatible types	There is a mismatch of data types in a statement. For example, a string is specified where a number is expected. This error can be generated when a variable tag is placed on a graphics page and the name of the tag is identical to the name of a Cicode function in the project or an included project.
Incorrect number of arguments for function	Too many or too few arguments have been passed to a Cicode function.
Index key has changed	The database has a corrupted index. Pack the database.
Invalid BOOLEAN value	A non-integer value was found where a TRUE or FALSE value is expected. For example, the controlling expression in an IF, WHILE, or FOR statement must be an integer.
Invalid font name	The font does not exist in the project. Check that the font name is correct, or specify the font with the Fonts form.
Invalid group definition	A group does not exist in the project. Check that the group name is correct, or specify the group with the Groups form.

Error	Description
Invalid time format	<p>The time is incorrectly specified in the Time, Period or Sample Period field of a Reports, Events, Trend Tags, SPC Trend Tags, or Devices form.</p> <p>Time formats must be in the format HH:MM:SS and must be in the range of 0:00:00 - 23:59:59. Only the hour is required, e.g. a value 16 means 16:00 (4:00 PM). Note that 24:00:00 is accepted for historical purposes, and maps directly to 0:00:00.</p> <p>Period formats must be either a valid date or a time in the format HH:MM:SS with the minutes and seconds in the range of 0 - 59. Only the seconds are required, e.g. a value of 22 means 22 seconds.</p> <p>Sample Period formats must either be a milliseconds value (e.g. 0.200 for 200 milliseconds) or a time in the format HH:MM:SS with the minutes and seconds in the range of 0 - 59. Only the seconds are required, e.g. a value of 22 means 22 seconds.</p>
Label argument error	The syntax of the argument is incorrect, or the incorrect number of arguments has been specified, or the number of characters in an argument is incorrect.
Label is defined twice	Label names must be unique. Check the Labels form for duplicated names.
Label too big	The label is too big. The size of a label must not exceed 8 KB.
Maximum report size exceeded	The report file size must be less than 63K. Reduce the size of the report or configure two reports.
MODULE function is not allowed, use PRIVATE	You have declared a Module function within your Cicode. Module is not a valid function type. Instead, the type Private must be used.
Must return value from function	If a Cicode function is declared to return a value, it must have a RETURN statement.
No I/O Devices defined	No I/O devices are defined in the project.
Not database format	The database has become corrupt or the file format is unknown. Pack the database.
Open bracket expected	<p>You must use parentheses () in Cicode functions, even when they have no parameters, e.g. MyFunction().</p> <p>This error can be generated when a variable tag is placed on a graphics page and the name of the tag is identical to the name of a Cicode function in the project or an included project.</p>
Operand expected	A Cicode operator must be followed by an operand.

Error	Description
Out of file handles	<p>CitectSCADA uses a file handle to open each file. When you try to open too many files or databases simultaneously, CitectSCADA can need more file handles than are available.</p> <p>You are most likely to run out of file handles if you have many included projects. When CitectSCADA compiles your project, it will open several files in each include project at the same time, so each extra project you include will increase the usage of file handles. If you get this error message when you have added another include project, you have run out of file handles. To verify that this is the problem, remove one of the included projects to see if CitectSCADA can then compile your project.</p> <p>With Windows running on a network, the setup of the number of file handles is located in various places. To increase the number of file handles in DOS, the setup is in the CONFIG.SYS file. If you are using Novell Netware you must also increase the file handles in the NET.CFG or SHELL.CFG file. You must also increase the number used by CitectSCADA with the [CtEdit]DbFiles parameter. Adjust the following settings the associated files:</p> <p>CONFIG.SYS FILES=120 NET.CFG or SHELL.CFG file handles=120</p>
Out of memory	CitectSCADA has run out of memory. Increase the amount of memory in the computer or use smaller databases.
Page Name cannot start with underscore	A Page Name must start with an alphanumeric character (A - Z, a - z, or 0 - 9).
Point limit reached	<p>The maximum number of points that can be referenced has been reached. The maximum <a href="#">point limit</a> is determined by your CitectSCADA license. Contact Citect Support.</p>
PRIVATE variable is not allowed, use MODULE	You have declared a Private variable within your Cicode. Private is not a valid variable type. Instead, the type Module must be used.
Protocol expected	The protocol field in the I/O devices database is blank. You must select a protocol for the I/O device.
PUBLIC variable is not allowed, use GLOBAL	You have declared a Public variable within your Cicode. Public is not a valid variable type. Instead, the type Global must be used.
Reached the end of table	The end of the database has been reached or the database has become corrupt. Pack the database. If the error persists, contact Citect Support.
Read remap is not supported for this variable	A mapped variable cannot be written when Remap Write is disabled, and cannot be read when Remap Read is disabled. Check the Remapping form.
Semicolon expected	All Cicode statements must be separated with semi-colons (;).
Software error	An internal CitectSCADA software error has been detected. Contact Citect Support.
Specification file invalid	A system file has become corrupt, or been deleted. Re-install CitectSCADA on your system. If the error persists, contact Citect Support.
Statement expected	CitectSCADA is expecting a statement. Check the Cicode for syntax errors.
String expected	Only strings can be used in database fields.

Error	Description
String too big	The string size has been exceeded. The maximum size of the string must not exceed 255 characters.
Super Genie must be on a page	Super Genie syntax (?) can only be used on pages. You cannot use a Super Genie in a report or Cicode function library. Use the TagRead() and TagWrite() functions instead.
Symbol search failed	A database record does not exist. Check that the <a href="#">record name</a> is correct.
Syntax error	A malformed Cicode expression has been specified. Check the structure of the expression.
Tag already defined	Tag names must be unique. Check the Variable Tags form for duplicated names.
Tag expected	A tag name was not found where one was expected, or an <a href="#">expression</a> has been passed to a function that expects a tag. Check that the tag name is correct, or specify the tag with the Variables Tag form.
Tag not found	The tag does not exist. Check that the tag name is correct, or specify the tag (with the Variables Tag form). If the tag does exist in the variables database, the index to the database might be incorrect. This can occur if you have edited the variables database using Excel or some other database editor. To re-index the database choose <b>File   Pack</b> (in the Project Editor).
THEN expected	A THEN statement must be used in an IF statement.
Too many arguments	Too many arguments are specified in a Cicode function. The maximum number of arguments allowed is 32.
Too many Cicode functions	More than 4500 user functions have been defined. To increase the number of functions allowed (up to 10000), use the CtEdit parameter MaxCicodeFunctions. This error is often due to Cicode functions being defined in a number of included projects. Extending this parameter might affect system performance. It should only be set when advised by Citect Customer Service.
Too many fields in database	Too many fields have been specified in the database. This error should only occur if the <code>citect.frm</code> file has been changed or become corrupt. Contact Citect Support.
Too many files open	The maximum number of .DBF files that can be open simultaneously has been exceeded. Increase the limit by changing the [CtEdit] DbFiles parameter.
Too many Include projects	More than 240 Include projects have been defined.
Too many records in database	Too many records have been specified in the database. This error should only occur if the <code>citect.frm</code> file has been changed or become corrupt. Contact Citect Support.
Trailing characters in Cicode	There are extra trailing characters in a Cicode statement, following the semi-colon.
Trailing characters in Name	The database record name contains invalid characters. Remove any invalid characters from the record name.
Unexpected beginning of file	The beginning of the database has been reached or the database has become corrupt. Pack the database. If the error persists, contact Citect Support.



Error	Description
Unexpected end of file	The end of the database has been reached or the database has become corrupt. Pack the database. If the error persists, contact Citect Support.
Unknown bin error	An output file could not be opened during compilation. Pack the database. If the error persists, contact Citect Support.
Unknown DBA error	An internal CitectSCADA software error was detected. Contact Citect Support.
Unknown field	A field is being referenced that does not exist. The database has been modified or has become corrupt. Pack the databases. If the error persists, contact Citect Support.
Unknown file	The include file cannot be found. Check the name of the include file, or that the included file is in the correct directory.
Unknown I/O device	The I/O device (unit) does not exist in the project. Check that the I/O device name is correct.
Unknown protocol	The protocol does not exist.

## Running the System

After compiling your project you can start your runtime system. Run the Computer Setup Wizard before running your system.

**Note:** Remember, the CitectSCADA software is protected against piracy. If you try to run your CitectSCADA without a protection key, CitectSCADA displays an error message and you will have to run in Demo Mode.

See Also [Startup and runtime configuration](#)  
[Running servers independently](#)  
[Running Your System Over the Internet](#)

### Startup and runtime configuration

You can specify a Cicode function to execute automatically when CitectSCADA starts up. This Cicode gets executed as soon as the Cicode system comes online. You should use the Computer Setup Wizard to specify the name of the startup function.

You can also run a report on startup. CitectSCADA searches for a default report called "Startup" when it starts up. If you have configured a report called "Startup", it is run automatically. You can change the name of the startup report (or disable it altogether) by using the Computer Setup Wizard.

You can customize many elements of CitectSCADA's runtime and startup behavior. The Computer Setup Wizard is usually all that is required, but you can also use Parameters for more control.

**To start your runtime system:**

- Click **Run**, or choose **File | Run**.

## Running servers independently

**To compile and run CitectSCADA online:**

- Click **Run**, or choose **File | Run**.

**Note:** CitectSCADA automatically compiles the project (if uncompiled) when you try to run it.

You can run individual processes in a multi-process system, allowing a particular server or client to be launched independently. This can be useful for hardware testing and system analysis.

The Runtime Manager can be launched during the configuration of a project via the **Tools** menu in **Citect Explorer**, **Graphics Builder** and **Project Editor**. Launching the Runtime Manager this way presents it in an idle state, allowing you to start a particular process by itself.

Once a process is started (by right-clicking on it and selecting **Start**), the Runtime Manager's monitoring pane displays the startup log for the selected processes, and launches Runtime.

**Note:** Only one instance of Runtime Manager can run on a machine at any time.

For more information on using Runtime Manager, click on its **Help** button.

See Also [Server Redirection](#)

## Running Your System Over the Internet

If you have a computer with Internet access, you can use it to run your project over the Internet from a remote location. Your computer would then be called an Internet Display Client. This is basically a runtime-only version of CitectSCADA; you can run your project from that computer, just as you would from any normal Display Client. However, an Internet Display Client cannot be a server, and it cannot be used to make configuration changes - you can only run your project.

**Note:** CitectSCADA also allows you to run your projects in a standard Web browser across a network of LAN-connected computers. See the [CitectSCADA Web Client](#).

See Also [CitectSCADA Internet Display Client](#)  
[CitectSCADA Internet server](#)  
[Startup and runtime configuration](#)  
[Server - client file updates](#)  
[Citect Internet Client setup properties](#)

## CitectSCADA Internet Display Client

The Internet Display Client is a runtime-only version of CitectSCADA. An Internet Display Client cannot be a server, and it cannot be used to make configuration changes - you can only run your project.

A computer can have a normal CitectSCADA installation as well as the Internet display client. Before you can run a project over the Internet, you must switch the **[Internet]Client** parameter on.

You can also run multiple instances of the Internet display client at the same time. This allows you to work with more than one project, in runtime-only mode, on the remote computer.

#### Notes

- If you are using a firewall, you must ensure that ports 2073 to 2079 are unblocked so that the Internet display client and the Internet server can communicate.
- If an ActiveX object has an associated data source, you need to ensure the data source can be located by the computer hosting the Internet Display Client. See the topic [Managing associated data sources](#).
- If you want to use the Process Analyst via the IDC, you must copy the Process Analyst view (.pav) files to the IDC in order to do so.
- To synchronize time on the IDC with the Time Server you must manually copy the [Time] section of citect.ini file from the server (after being configured by the Computer Setup Wizard) to the citect.ini file in the IDC bin directory. The IDC will not inherit this configuration from the server and has a default installation configuration.

See Also [CitectSCADA Internet server](#)

### CitectSCADA Internet server

Any [I/O server](#) can be a CitectSCADA Internet Server: you just need to use the Computer Setup Wizard. (A special protection key is required for the Internet Server. Please contact Citect Support for protection key details.)

If you are using a firewall, you must ensure that ports 2073 to 2079 are unblocked so that the Internet Display Client and the Internet Server can communicate.

See Also [Startup and runtime configuration](#)

### Startup and runtime configuration

You can customize many elements of CitectSCADA's runtime and startup behavior via the Computer Setup Wizard.

You can also configure an IDC installation to automatically connect to a CitectSCADA Internet Server at startup. To do this, you need to adjust the parameters [Internet]IPAddress, [Internet>Password and [Internet>ShowSetupDlg within the [citect.ini file](#) of the IDC computer.

The citect.ini file is stored in the bin directory of the Internet Display Client installation.

See Also [Server - client file updates](#)

## Server - client file updates

When you log on to the CitectSCADA Internet Server, all the files needed to run the project are downloaded to your computer. Because these files must be up to date, CitectSCADA periodically compares the files on the Internet Server with the downloaded files on the Internet Display Client. (This period is defined using the **[Internet]UpdateTime** parameter.) If a file has been changed since the last update, it is copied to the Internet display client.

### To set up your CitectSCADA Internet Server:

- 1 Run the Computer Setup Wizard and select **Custom Setup**.
- 2 Select **Server and Display Client** from **Network Computer** section.
- 3 Once you reach the Internet Server screen, select **Internet Server**.
- 4 Enter the TCP/IP address of your Internet server computer (e.g. 10.5.6.7 or *plant.yourdomain.com*). This information is downloaded and stored in the Internet Display Client's [citect.ini file](#) when a connection is made.
- 5 To determine the TCP/IP address of the Internet server computer:
  - For Windows NT4 or 2000, go to the Command Prompt, type **IPCONFIG**, and press **Enter**.
  - For Windows 95, select **Start | Run**, type **WINIPCFG**, and press **Enter**.
- 6 After completing the Computer Setup Wizard, define the passwords required by users of your Internet Display Clients using the `[Internet]Manager` and/or `[Internet]Display` parameters.
- 7 If the Runtime project on the Internet Server has links to any included projects, the Internet Display Client can only access the included project files if they are stored on the same directory level as the Runtime project. For example, if the current project is located at:
 

```
C:\Citect\User\<current project>
```

 then any included projects must be located on the same level:
 

```
C:\Citect\User\<included project>
```
- 8 Any files you want to make accessible to an anonymous FTP user should be placed in the Internet Server's `\Internet` directory, located at `C:\Citect\User\Internet` by default. This is where CitectSCADA stores the `idc.exe` file to allow remote installation of the Internet Display Client.

The `Internet` directory on the Internet Server is only accessible to anonymous FTP users if it shares the same directory level as the current Runtime project. For example, if you use CitectSCADA's default settings, the current project folder will be located at:

```
C:\Citect\User\<current project>
```

and the `Internet` directory will be located on the same level at:

```
C:\Citect\User\Internet
```

---

If the runtime project on the Internet server is stored elsewhere, an appropriately located Internet directory must be created.

**To install the Internet Display Client:**

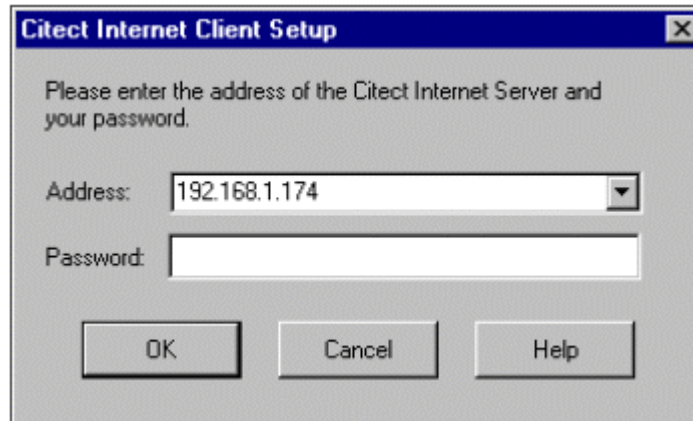
- 1 On the remote computer, run up your Internet browser.
- 2 Type in the FTP address of your CitectSCADA Internet Server (for example, **ftp://sanctus.citect.com.au/idc.exe**).
- 3 Save the file (**IDC.exe**) to a temporary folder.
- 4 Go to this temporary folder and double-click **IDC.exe**.
- 5 Follow the prompts to complete the installation.

**To run your project over the Internet:**

- 1 Ensure you have installed the Internet Display Client and set up your Internet server.
- 2 At the Internet Display Client, double-click the **Citect Runtime** icon in the CitectSCADA IDC program group.
- 3 In the **Citect Internet Client Setup** dialog, type the TCP/IP address of your CitectSCADA Internet Server (e.g. 10.5.6.7 or *plant.yourdomain.com*), then type your password.
- 4 Click **OK**. All the relevant data will be downloaded to your computer and your project will run (if the CitectSCADA Internet Server is running).

**To connect to a different CitectSCADA Internet Server once your project is running on the Internet Display Client:**

- 1 Click **Client Setup** from the runtime **Control** menu.
- 2 In the **Citect Internet Client Setup** dialog box, type the TCP/IP address of your CitectSCADA Internet Server (e.g. 10.5.6.7 or *plant.yourdomain.com*), then type your password.
- 3 Click **OK**. All the relevant data is downloaded to your computer and your project will run.



See Also [Citect Internet Client setup properties](#)

## Citect Internet Client setup properties

You use the Citect Internet Client Setup dialog box to configure your internet client.

Complete the following properties to set up your Internet display client:

### Address

Enter the TCP/IP address of the CitectSCADA Internet Server (e.g. 10.5.6.7 or plant.yourdomain.com). The address of the last Internet Server used will be automatically entered here. The addresses of all Internet Servers previously used are retained in the menu (with the most recently used at the top).

### Password

Enter the password supplied to you by the CitectSCADA Internet Server administrator. Your password will be encrypted before it is sent across the Internet.

If you enter an incorrect password, the connection attempt will fail.

## Using an Alternative INI File

When CitectSCADA starts up, it reads default values from `citect.ini`. (By default, CitectSCADA looks for the ini file in the `\CitectSCADA 7\bin` directory. If it can't find it there, it searches the windows directory.) If you are running multiple projects, you can specify an alternative ini file for each project. The new ini file name must be passed to the CitectSCADA system applications; i.e., to `CtExplor.exe` and `Citect32.exe` as a command line argument.

### To specify an alternative INI file for CitectSCADA :

- 1 Click the icon that you use to start Citect Explorer or Runtime.

- 2 Right-click to display the shortcut menu and select **Properties**.
- 3 Click the **Shortcut** tab.
- 4 Add the name of the INI file to the command line property of the appropriate icon using the **/i** option. For example, to start CitectSCADA using the initialization file `my.ini`, enter the following line:

```
c:\CitectSCADA\CitectSCADA 7\bin\ citect.exe /
ic:\citect\user\myproj\MY.INI
```

Note that the Computer Setup Wizard uses the same `.ini` as specified for Citect Explorer. You can specify different `.ini` files for both the Runtime and Explorer programs. However, if you initiate Runtime from the Explorer, it uses the `ini` that is specified for the Explorer.

## Debugging the Runtime System

This section describes how to solve commonly encountered problems with your runtime system.

See Also [Hardware alarms](#)  
[SysLog.DAT file](#)  
[Debugging I/O Devices and Protocols](#)

### Hardware alarms

When a system error occurs (that is, a malfunction in CitectSCADA operation) CitectSCADA generates a hardware alarm. Hardware alarms are usually displayed on a dedicated Hardware Alarm page, available as a standard template.

The hardware alarm page indicates what is happening in your CitectSCADA system. If a communication fault occurs, if Cicode can't execute, if a graphics page is not updating correctly, or if a server fails, this page shows you. Hardware alarms consist of a unique description and error code.

The hardware alarms do not have detailed information, but serve to point you in the right direction. For example, if you have a Conflicting Animation alarm, CitectSCADA will not tell you the cause. You must observe which page causes the hardware alarm, and locate the animations yourself.

**Note:** Your system should have no recurring hardware alarms.

There are two hardware alarm fields that are not always shown on the hardware alarms pages. `ERRPAGE` will display the name of the page that was displayed when the error occurred. This is useful for finding animation faults. `ERRDESC` provides information that is specific to the type of the alarm. For example, if the alarm is an I/O device error, `ERRDESC` shows the name of the device.

See Also [SysLog.DAT file](#)

## SysLog.DAT file

The `syslog.dat` is a file maintained by CitectSCADA, that contains a useful log of CitectSCADA System information. The variety of information that can be logged to the `syslog.dat` is extensive; from low level driver traffic and Kernel messages, to user defined messages.

The **Log Read** and **Log Write** fields in the I/O Devices Properties dialog box control whether logs are made for each I/O device.

**Note:** CitectSCADA locks `syslog.dat` while running. However, you can still view it by using the SysLog command in the Kernel.

This file is restricted in size (to 300k by default). When it reaches the size limit, CitectSCADA renames it `syslog.bak`, and starts a new `syslog.dat`. You can make this restriction larger or smaller by using the `[Debug]SysLogSize` parameter. For example, the following lines in the `citect.ini` will set the `syslog.dat` size to 1000k:

```
[DEBUG]
SysLogSize=1000
```

See Also [Debugging the Runtime System](#)

## Server Redirection

### Using address forwarding

CitectSCADA allows you to temporarily transfer the communications of a server to another computer to assist with hardware maintenance and system analysis.

By including an address forwarding section within a `Citect.ini` file, you can override a server's project-configured address, redirecting network traffic to a different address and port.

For example, if you would like to test a new server before adding it permanently to a project configuration, you could make the required address forwarding adjustments within the `citect.ini` file of a single display client to direct it to the new hardware. The server could be tested within the context of a system without having to recompile the project.

You need to ensure that you make the required adjustments within the `citect.ini` file of all computers that are likely to be impacted by address forwarding.

For example, if you have a computer configured to run two I/O servers, and you would like to temporarily redirect one of them, then both server machines and all connecting clients must have the required address forwarding adjustments made to their `citect.ini` files, so that the servers are aware of each other.

Address forwarding is implemented using the following syntax within a `citect.ini` file:

```
[AddressForwarding]
<ClusterName>.<ServerName>=<ipaddress>:<port>
```



You can also redirect the Peer Port connection for an I/O Server using the following syntax:

```
[AddressForwarding]
<ClusterName>.<ServerName>_PeerPort=<ipaddress>:<port>
```

For Alarm Servers that have alarm properties enabled, you can redirect the alarm properties connector using the following:

```
[AddressForwarding]
<ClusterName>.<ServerName>_AlarmProps=<ipaddress>:<port>
```

Address forwarding is only interpreted and used during startup of Citect Runtime. It is recommended (but not required) that the Computer Setup Wizard is run before running up a project to confirm your changes. This is useful, especially if a server requires communication to a redirected server acting as a time server.

**Note:**

- Address forwarding should only be used as a temporary engineering solution and should not be used as the primary mechanism for specifying server/client communications. It allows side stepping of configuration checks the compiler enforces.
- You should carefully track any adjustments you make, as they must be manually corrected once redirection is no longer required.
- The CitectSCADAWeb Client uses address forwarding to manage communication across corporate firewalls. Any manual adjustments may also impact the ability to connect a Web Client to a deployment on a Web Server.

## Debugging I/O Devices and Protocols

Before commissioning any system, you must test all communications between CitectSCADA and the I/O devices. Many people leave this last, only to uncover communication problems that they cannot resolve.

The following information is intended to encourage you to test your communications thoroughly before it becomes a time critical element in the job. It will also help you to debug communications and protocol problems yourself.

See Also [Creating a communications test project](#)  
[Debugging a COMx driver](#)  
[Debugging a TCP/IP driver](#)  
[Debugging a protocol driver using serial communications](#)  
[Debugging proprietary board drivers](#)  
[Contacting Citect support](#)

## Creating a communications test project

The first step in any testing of Communications is to make sure that CitectSCADA can bring your I/O devices online. To do this create a test project. The Test Project needs to be as simple as possible.

For example, if your system will eventually be communicating through a COM port, a KTX card, an SA85 card, and via TCP/IP do not try and make all this work on your first try. Make the Test Project with 1 element at a time. First add and test the COM port. When that works, make a new test project for the KTX card, then make a new test project for the SA85 card, and finally one for the TCP/IP. Once you are sure that each individual element works properly, start to add them together.

When creating a test project:

- 1 Use the Communications Express Wizard to set up your communications.
- 2 In the Project Editor go through the forms under the Communications menu.
- 3 Check that you have one I/O server defined.
- 4 Check that you have one Board defined. Verify that the information for the Board is correct.
- 5 Check that you have one Port defined. Verify that the information for the Port is correct.
- 6 Check that you have one I/O device defined.

**Note:** Do not add any variable tags or create any graphics pages. Do not add anything else.

While checking that you have only the absolute minimum information in the project to enable communications, make sure that there are no duplicated records. The most reliable way to do this is to open a form and then check the Record Number shown on the bottom left of the form. Make sure it is on Record 1, and then click the button in the scroll bar on the right hand side of the form and drag it down. When you get to the bottom of the scroll bar, let the button go. If it pops back up to the top of the form and the record numbers stays at 1 then you have only 1 record.

It is necessary to drag the scroll bar as all the forms are indexed on the I/O server name. Quite often there will be 'orphaned' records from a previous I/O server name still in the database files. If you find any extra records, delete them and then [pack](#) the project. The majority of all communications problems come from having duplicated or orphaned records in the communications database.

After setting up your communications, perform a communications test.

### Before running your test project

Before running your test project, do the following:

- 1 Make sure the Kernel is enabled on CitectSCADA startup so that you can follow the startup procedures step by step. To do this, edit your `citect.ini` file to contain the following:

```
[DEBUG]
Kernel=1
```

This will show the CitectSCADA Kernel on startup of CitectSCADA.

- 2 Run the Computer Setup Wizard. Ensure the computer is defined as a stand-alone CitectSCADA system, configured to run your test project.

### Running your test project

Start the project running. When the kernel appears, double-click the title bar in the Main window, then double-click the title bar of the kernel. Doing this in order is important, as it will maximize the Main window, then Maximize the whole kernel. If you don't do this then you will not see what is happening as the information in the Kernel cannot be scrolled, and once it is off the screen it is gone. This might require a bit of practice as the startup procedure might only take 1-2 seconds or less on a fast machine.

Once you have the project running (and assuming that everything worked) the last line in the Main window in the Kernel should tell you that your I/O devices are online. Do not confuse this with the message telling you that your Port channels are online. CitectSCADA should first report that it can communicate through the port you have setup, then report that it can communicate with the I/O device.

### When your test project does not communicate

Check everything and run it again. Depending on the type of board driver you are using there are different methods for debugging them. However, all of them are basically the same. Add one item at a time and test.

See Also [Debugging a COMx driver](#)

## Debugging a COMx driver

A COMx driver is the board driver used for most serial communications.

Version 2.01 of the COMx driver lets you dump debug information. Three files are produced for each com port: a write file, a read file and status file. The debug files are configured by settings in the `citect.ini` and are written to the default OS path. The following `citect.ini` entries are used:

```
[COMx]
WritePortName=X1,X2...
WriteDebugLevel=Y
WriteFileSize=Z
ReadPortName=X1,X2...
ReadDebugLevel=Y
```

```

ReadFileSize=Z
StatusPortName=X1,X2...
StatusDebugLevel=Y
StatusFileSize=Z

```

where:

- Xn = A port name as it appears in the Citect|Communications | Ports form. For example, PORT\_1.
- Y = 1 to enable debugging, 0 to disable debugging.
- Z = The file size in Kb (default is 1000 Kb).

### Example

```

[comx]
WritePortName=PORT_1,PORT_2
WriteDebugLevel=1
WriteFileSize=2000
ReadPortName=PORT_1
ReadDebugLevel=1
ReadFileSize=1000
StatusPortName=PORT_2
StatusDebugLevel=1
StatusFileSize=10

```

The above example would:

- Log to "write" files up to 2000Kb of the data sent by the CitectSCADA driver to both PORT\_1 and PORT\_2;
- Log to a "read" file 1000Kb of the data received by the CitectSCADA driver from PORT\_1; and
- Log up to 10Kb of status data from PORT\_2 to a "status" file.

This would also result in the following text files being created in the OS Path (generally WINDOWS or WINNT):

- WPORT\_1.dat
- WPORT\_2.dat
- RPORT\_1.dat
- SPORT\_2.dat

In general, the format for the file names is "R", "W" or "S" followed by the Port name requested, followed by ".dat". The file names represent the corresponding "Read", "Write" and "Status" files.

### File formats

The **Write** file will adopt the following format:

```

LINE 1      WRITE Debug file Started PORTNAME Debug Level 1 DOW MONTH DOM
              HH:MM:SS.msec
LINE 2      HH:MM:SS.msec

```

LINE 3                    Out W In X nBytes Y Status Z  
 LINES 4...                AA BB CC ..

where:

W & X =                    Values of buffer pointers  
 Y =                        Number of bytes written  
 Z =                        Return status of the WriteFile  
 AA BB CC =                Values in hex of each byte written

### Example:

```
WRITE Debug file Started PORT1_BOARD1 Mon Dec 15 16:07:07.998
16:07:09.810
Out 0 In 8 nBytes 8 iStatus 997
0e 02 00 00 00 10 00 00
16:07:10.802
Out 8 In 16 nBytes 8 iStatus 997
0e 02 00 00 00 10 00 00
.
.
```

The **Read** file will adopt the following format:

LINE 1                    READ Debug file Started PORTNAME Debug Level 1 DOW MONTH DOM  
                           HH:MM:SS.msec  
 LINE 2                    HH:MM:SS.msec  
 LINE 3                    Out W InRx X nBytes Y Status Z  
 LINES 4...                AA BB CC ..

where:

W & X =                    Values of buffer pointers  
 Y =                        Number of bytes read  
 Z =                        Number of characters remaining in the buffer  
 AA BB CC =                Values in hex of each byte written

### Example

```
READ Debug file Started PORT1_BOARD1 Mon Dec 15 16:07:07.998
16:07:09.830
Out 0 In 0 nBytes 8 iStatus 0
0e 02 00 00 00 10 00 00
16:07:10.822
Out 0 In 8 nBytes 8 iStatus 0
0e 02 00 00 00 10 00 00
```

The **Status** file will adopt the following format:

LINE 1                    STATUS Debug file Started PORTNAME Debug Level 1 DOW MONTH DOM  
                           HH:MM:SS.msec  
 LINE 2                    HH:MM:SS.msec

LINE 3                      modemStatus X

### Example

```
STATUS Debug file Started PORT_1 Debug Level 1 Wed Nov 05
15:28:55.310
15:29:55.950
modemStatus 34
.
.
```

More parameters might be added later, so check the COMx information and the Citect Knowledge Base regularly.

See Also [Debugging a TCP/IP driver](#)

## Debugging a TCP/IP driver

A TCP/IP driver is the low-level driver that is used for any TCP/IP communications. It might be over [ethernet](#), Token Ring or Arcnet. This driver communicates between CitectSCADA and Winsock, so it really uses all the normal networking functionality of Windows. The parameters used for TCP/IP are:

[TCPIP]

- Log=1: Dumps all traffic between the CitectSCADA TCPIP.DLL and Winsock to a text file called TCPIP.DAT in your Windows directory. Note that this file does not have a size limit and could potentially use all available disk space.
- Debug=1: Opens a window at runtime and shows communication between CitectSCADA and Winsock. It displays exactly the same data that is sent to the log file, but allows you to see it as it is happening. The number of line displayed is limited, though.

### Debugging steps

Use the following steps for debugging:

- 1 Check if you can PING your target I/O device. If you can't, CitectSCADA cannot talk to it. To do this open up a Command Window and type **PING aaa.bbb.ccc.ddd** where a.b.c.d is the IP address of the I/O device you are trying to connect to. If PING does not work, you need to go back to your Windows Networking and fix that.
- 2 Keep using your Simple As Possible Project (SAPP).
- 3 Delete any old tcpip.dat files.
- 4 Set the Debug=1 and Log=1 parameters in your citect.ini file.
- 5 Start the project. From the information in the maximized Main window of the Kernel and the TCP/IP Debug window, you can see if CitectSCADA is sending requests to your I/O device to initialize communications with it.

If there are no requests being sent, there is a configuration problem with your software, and you should check that there were no errors on startup of CitectSCADA. If there were errors on startup, look them up in the Help. Also check that your computer is an I/O server (and that it matches the one in your project). To do this run the Computer Setup Wizard, and configure the computer for a standalone configuration.

If there are requests, you can see all communications between CitectSCADA and Winsock in a separate window, which displays the requests made by CitectSCADA to connect to the I/O device and the corresponding response from the I/O device. Any errors have a Winsock Error code that you can look up in the Microsoft Knowledge Base. If you see a **Connection OK** message, CitectSCADA should be able to come online.

See Also [Debugging a protocol driver using serial communications](#)

## Debugging a protocol driver using serial communications

To debug a protocol driver that uses serial communications, do the following:

- 1 Keep using your Simple As Possible Project (SAPP).
- 2 Set the `DebugStr=*` all for your protocol.
- 3 Backup and delete `syslog.dat` and `syslog.bak`. This ensures you start with a fresh log file.
- 4 Start the project. From the information you can see in the maximized Main window of the kernel you should be able to see if CitectSCADA is sending requests to your I/O device to initialize communications with it.

If there are no requests being sent then there is a configuration problem with your software, and you should check that there were no errors on Startup of CitectSCADA. If there were errors on startup look them up in the Online Help. Also check that your computer is an I/O server (and that it matches the one in your project). To do this run the *Computer Setup Wizard*, and configure the computer for a stand-alone configuration.

If there are requests being sent but no reply, then CitectSCADA is trying to communicate. When CitectSCADA is sending requests but getting no reply, these are the most common causes:

- **The request CitectSCADA is sending is not getting to the I/O device -** Check the Address field in the I/O Devices form, and make sure it is correct. If the I/O device is one that needs a unique identifier (such as a node address), or you need some type of routing path, then make sure it is correct.

Check that you have the same parameters in the Ports form that the I/O device is using. If you have 8 data bits and the I/O device uses 7 data bits, communications will not work.

Check that your cable is OK. The easiest way to do this is to create a new project and use the Loopback protocol. You can use this to verify the **Tx** and

**Rx** lines' integrity by placing a jumper on these lines. Initially test this with a jumper between pins 2 and 3 on your PC. Then plug in your cable and test again with the jumper between the **Tx** and **Rx** lines. Keep moving the jumper until it is at the end of your communications bus. You can find more information on using the Loopback protocol in the Citect Knowledge Base.

Even if the Loopback protocol shows no errors, your cable might still be faulty. CitectSCADA usually places a far higher constant load on serial communications than programming software does, this usually means that CitectSCADA will require much more stringent handshaking than the programming software. So it is possible that the cable you use to program your I/O device works fine for programming, but not for CitectSCADA. Check the Wiring Diagram for your Protocol in the help.

Another major cause of cabling problems is 9 pin to 25 pin converters. Many of these converters are made specifically for serial mice. These typically only use the **Tx**, **Rx** and **Ground** signals. If you use one of these converters they do not support any handshaking at all and will most likely not work for your Protocol.

If all of the above checks OK, use the parameters for COMx (as mentioned above) to create log files. Examine these log files and make sure that what CitectSCADA thinks it is sending is actually what it is sending. The log files produced by using these parameters get their information from a lower level than CitectSCADA and show you exactly what is going through the COMx driver.

- **The Response from the I/O device is not getting to CitectSCADA** - This is unlikely and usually caused by a cabling problem. Check your cabling as above. Also, check that you are specifying everything you need within CitectSCADA. Many protocols require CitectSCADA to send a unique identifier in its request packet. If this identifier is incorrect then the response can never get back to CitectSCADA.
- **The I/O device does not understand the Request** - All CitectSCADA protocols can check if an I/O device is running. Typically the protocol attempts to read data from the I/O device, usually a status register or other register that should be there. However, many pseudo-standard protocols, such as Modbus, do not conform to the exact specification for that protocol. Many protocols supplied with CitectSCADA have some extra parameters to allow you to choose the specific initialisation request from CitectSCADA. These can be found in either the Online Help or the CitectSCADA Knowledge Base. Check with the manufacturer of your I/O device to make sure it will respond to the request that CitectSCADA sends. If you are unsure of the request that CitectSCADA is sending for its initialisation, use DebugStr=\* to get the actual variable address that CitectSCADA is asking for in its initialisation.



Check the protocol you are using. CitectSCADA may have many different protocols for communicating to an I/O device. PLCs such as the AB PLC5 can use different serial protocols, depending on the method you are trying to use. Make sure you are using the correct one. If you are unsure, try the other possible protocols.

- **The I/O device is not functioning properly** - There is usually some sort of software from the I/O device manufacturer that can be used to diagnose any problems with the I/O device.

See Also [Debugging proprietary board drivers](#)

## Debugging proprietary board drivers

Proprietary board drivers (such as the Allen Bradley KTX card, Modicon SA85 card, Siemens TIWAY card, and so on) have their own low-level drivers. Each driver has debugging parameters to make it easier to debug problems. Check the CitectSCADA Knowledge Base and Help for parameters. The CitectSCADA Knowledge Base has articles describing how to debug these board drivers.

The debugging process is exactly the same as with a serial connection:

- 1 Keep using your Simple As Possible Project (SAPP).
- 2 Set any debugging parameters for the protocol and board drivers.
- 3 Start CitectSCADA with clean log files.
- 4 Find any errors and then look them up in the manufacturer's documentation.

## Contacting Citect support

If you can't debug your problems, contact Citect support and provide the following information:

- Blank Citect Solution Request (CSR), which you can obtain through Citect Support.
- `syslog.dat`, `syslog.bak`, `tcPIP.dat`, or COMx log files.
- A copy of the SAPP, and the `citect.ini` file.

See [Getting Technical Support](#).

## Restarting the System Online

With the online restart facility, you can change your project configuration and examine the results in the CitectSCADA runtime system without having to shut down CitectSCADA: you can update your system while CitectSCADA is running.

**Note:** If you've configured CitectSCADA to use multiprocessor support, it is strongly recommended that you do not use the restart facility; use the Runtime Manager instead.

The time taken for the system changeover depends on the size of the project and the extent of the changes to the project:

- If you only change graphics pages, CitectSCADA does a *partial restart* (changing only the pages in the runtime system). Changeover is instantaneous.
- If you change any databases (for example, add a new alarm tag, trend tag, or Cicode function), CitectSCADA does a *full restart* to run the updated project.

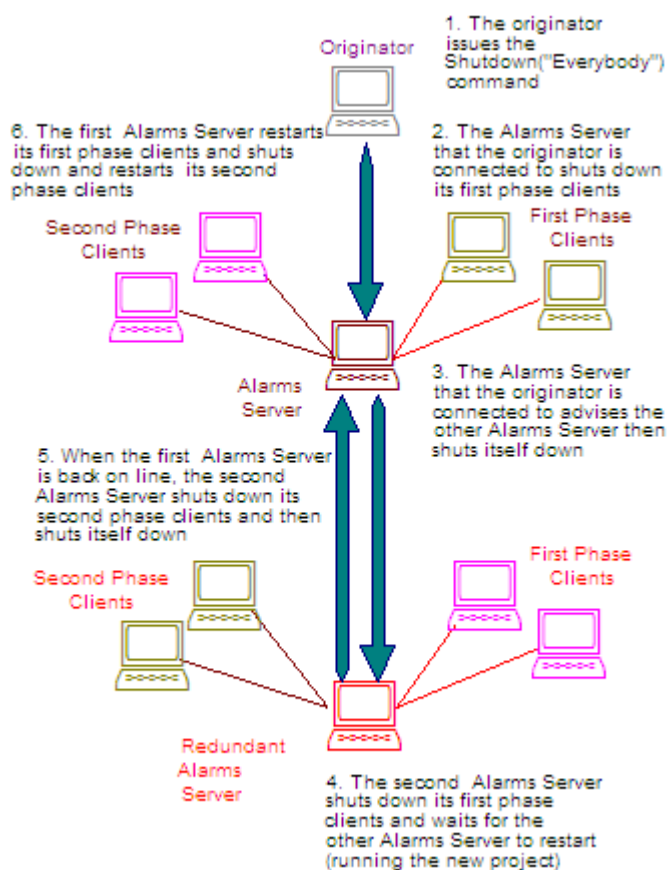
**Note:** Do not use this feature if you plan on making significant changes to a project.

See Also [Restarting a networked system online](#)

## Restarting a networked system online

If you are using CitectSCADA on a network, you can use a structured restart procedure that ensures control of the plant is maintained and no data is lost during changeover. You can use any CitectSCADA computer on the network to initiate the online restart.

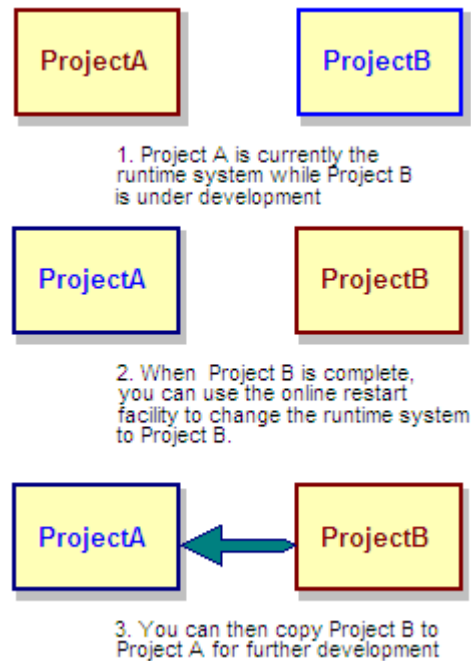
CitectSCADA automatically manages the online restart in the following sequence:



**Note:** If you've configured CitectSCADA to use multiprocessor support, it is strongly recommended that you do not use the restart facility; use the Runtime Manager instead.

### Using multiple projects

The most effective method of using the online restart facility is to use two projects. The first project becomes the current runtime system while the second project is in the development stage. You can manage both projects as follows:



**Note:** If you've configured CitectSCADA to use multiprocessor support, it is strongly recommended that you do not use the restart facility; use the Runtime Manager instead.

### Initiating the online restart

To initiate the online restart, the originator (any CitectSCADA computer on the network) issues a shutdown command with the Shutdown function, for example:

```
Shutdown("Everybody", "MyProject", 2);
```

Where possible, balance all Display Clients across both phases of the shutdown. The [Shutdown]Phase parameter defines the phase to which each CitectSCADA computer responds.

You can exclude selected computers (for example I/O servers) from the online restart procedure with the [Shutdown]NetworkIgnore parameter.

For security, you can prevent selected computers from initiating the online restart procedure with the [Shutdown]NetworkStart parameter.

**Note:** If you've configured CitectSCADA to use multiprocessor support, it is strongly recommended that you do not use the restart facility; use the Runtime Manager instead.

#### Using a Callback function

You can use a [callback function](#) (with the `OnEvent` function) to perform housekeeping tasks before the system shuts down. You would normally call `OnEvent()` in the main startup function (defined with the `[Code]Startup` parameter). Each time a `Shutdown()` call is made, the callback function is run.

```
/* A user shutdown procedure. */
INT
FUNCTION
MyStartupFunction()
. . .
. . .

OnEvent(25, MyShutdown);

. . .
. . .
END

INT
FUNCTION
MyShutdown()
STRING sPath;

// Perform housekeeping tasks
. . .
. . .
. . .

sPath = ProjectCurrentGet();
If sPath = "ProjectA" Then
    ProjectSet("ProjectB");
Else
    ProjectSet("ProjectA");
END

Shutdown("Everybody", sPath, 2);
END
```

## CitectSCADA Software Protection

CitectSCADA uses a hardware key to safeguard against license infringement. The hardware key is a physical key that plugs into either the parallel port or USB

port of your computer. The hardware key contains details of your user license, such as type and I/O [point limit](#).

See Also [CiUSAFE dialog properties](#)  
[Demo mode](#)

## CiUSAFE dialog properties

The CiUSAFE dialog box has the following properties:

### Serial Number

The serial number of the computer's hardware key. It will only appear if the key was delivered after September 11 2000, or has been updated since this time. If this is not the case, you can read the number from the label on the hardware key. You need to enter the serial number at the Citect web site to update the key.

### KeyID

Each time you launch CiUSAFE, a Key ID will display in the **KEYID** field. You might need to provide the Key ID plus the serial number when updating the hardware key. This depends on the status of the key in the CitectSCADA license database, and you are prompted if the Key ID is required. Click **Save KeyID** to save the Key ID and serial number to a text file, which you can refer to when visiting the Citect web site.

### Authorization Code

To update the hardware key, enter the 106-character authorization code. You are asked for this code once you have entered the Key ID and serial number, and your license and Customer Service agreement have been verified. Click **Update** to update your hardware key.

### Return Code

The Return Code indicates the result of the key update:

0	The key was updated successfully.
1,3	Either the KeyID or the Authorisation code you entered is invalid.
2	Either the KeyID or the Authorisation code you entered has been corrupted.
4,16	Either the KeyID or the Authorisation code you entered is invalid.
9	No hardware key could be found.

To close the program, click **Exit**.

## Demo mode

You can run CitectSCADA without the hardware key in demonstration (Demo) mode. Demonstration mode lets you use all CitectSCADA features normally, but with restricted runtime and I/O.

**Note:** If you configure CitectSCADA to run as multiple processes on one CPU or multiple CPUs, you cannot use CitectSCADA in demo mode. If you run CitectSCADA as one process, you can use demo mode as with previous versions of CitectSCADA.

The following demonstration modes are available:

- 15 minutes with a maximum of 50,000 real I/O.
- 10 hours with no static points and a maximum of 1 dynamic real I/O. This is useful for demonstrations using memory and disk I/O. CitectSCADA starts in this mode if no static points are configured.
- If you want to demonstrate DDE, CTAPI, or ODBC writes to CitectSCADA in this mode, you can only write 1 point. To write to more than 1 point, you must force CitectSCADA to start in 15 minute-50,000 I/O demo mode by creating at least one static I/O point.

For this to work, you must configure a real variable tag, with an accompanying PLC or I/O device. The tag must be used by a page or in Cicode. If you do not have a real I/O device connected, CitectSCADA gives a hardware error, which you can disable using the `IODeviceControl` function.

- 8 hours with a maximum of 42,000 real I/O. This is only available through special CitectSCADA Integration Partners (CIP) keys.

## Using the CitectSCADA Kernel

You use the CitectSCADA kernel to perform low-level diagnostic and debugging operations, and for runtime analysis of your CitectSCADA system. Use it to display all low-level data structures, runtime databases, statistics, debug traces, network traffic, I/O device traffic and so on. You can also call in-built Cicode function or user-written Cicode functions.

### Notes:

- You should be experienced with CitectSCADA and Cicode before attempting to use the Kernel as these facilities are powerful, and if used incorrectly, can corrupt your system.
- Only use the Kernel for diagnostics and debugging purposes, not for normal CitectSCADA operation.

You should restrict access to the Kernel: anyone using the Kernel has total control of CitectSCADA (and subsequently your plant and equipment).

### Access to Cicode and Cache commands

In order to prevent unauthorized use of Cicode and Cache commands in the Kernel, only Kernel users have access to these commands. The Kernel user must be defined in the User database (with the username 'kernel' and a non-blank password) for the CitectSCADA project in which they want to access the commands. The Kernel user does not need to have any areas or privileges defined.

During runtime if the Kernel user attempts to access Cicode or Cache Kernel commands, the Kernel will ask for the Kernel user password. If the Kernel user is not defined in the User database for that project (or if the user provides the incorrect password), access to the Cicode (or Cache) commands is denied.

See Also [Displaying the kernel window](#)  
[Inside the kernel](#)  
[Using Kernel Commands](#)  
[Kernel commands](#)

## Displaying the kernel window

You can display the kernel window in several ways. CitectSCADA can open the window automatically at startup, provide a command option on the Control menu, or you can define a runtime command to display the Kernel window when required.

- [Displaying the kernel from the control menu](#)
- [Displaying the kernel at startup](#)
- [Defining a runtime command](#)
- [Closing the kernel window](#)

### Displaying the kernel from the control menu

To add a **Kernel** option to the Control menu of the runtime system, use the Computer Setup Wizard. Run the wizard, select **Custom** mode, and select the **Kernel on menu** option on the Security Setup - Control Menu page.

You can then display the Kernel window by selecting the Kernel option from the Control Menu (top-left corner) at runtime. If you do not have a title bar displayed, you can access the Control Menu by pressing ALT-SPACE (make sure the **Alt-Space enabled** option is selected on the Security Setup - Keyboard page).

**Note:** Clear these options (the default) after commissioning to prevent accidental or unauthorized use of the Kernel.

### Displaying the kernel at startup

To display the Kernel window automatically when CitectSCADA starts up, set the [Debug]Kernel parameter to 1. The Kernel window is opened at startup and closed at shutdown. The display is off (0) by default.

**Note:** Reset this parameter after commissioning to prevent accidental or unauthorized use of the Kernel.

### Defining a runtime command

To display the Kernel window, define a runtime command that calls the `DspKernel()` function, passing 1 in the **iMode** argument:

```
Command                                DspKernel(1);
```



Comment	Displays (opens) the Kernel window
---------	------------------------------------

To close the Kernel window, call the `DspKernel()` function again, passing **0** in the **iMode** argument:

Command	<code>DspKernel(0);</code>
Comment	Closes the Kernel window

**Note:** You should put the highest privilege level on the `DspKernel` command to prevent your operators from opening the Kernel window.

#### Closing the kernel window

You can close the Kernel window by choosing **Close** from the Control menu of the main Kernel window.

See Also [Inside the kernel](#)

## Inside the kernel

When displayed, the CitectSCADA Kernel consists of an application (client) window called Main and one or more child windows. At startup, the CitectSCADA Kernel window displays information about your CitectSCADA startup processes. The Kernel window also displays CitectSCADA runtime system messages, providing a continuous operational history of your CitectSCADA system.

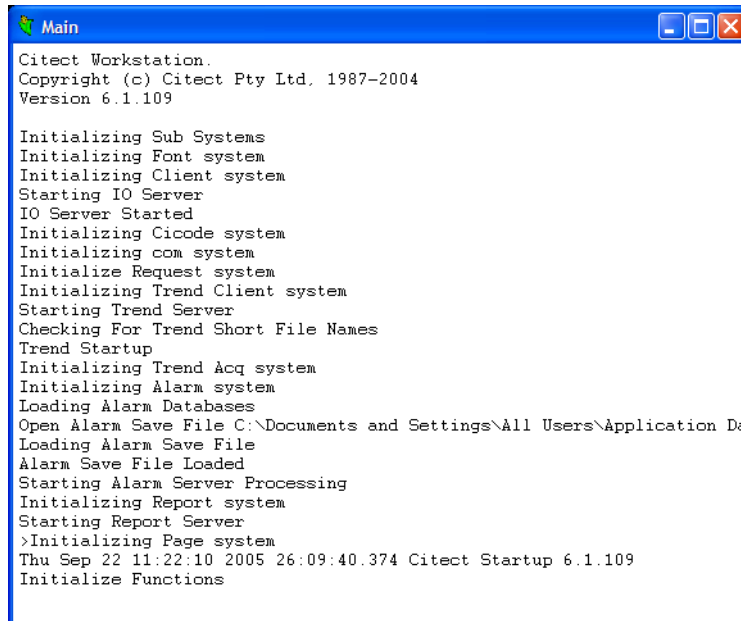
The Kernel window contains a command line interface (similar to the Command prompt) where you can type in Kernel commands to perform a Kernel operation or to display other child windows.

**Note:** Access to the Cache and Cicode Kernel commands is password-protected for security. For details, see [Access to Cicode and Cache commands](#).

By default, CitectSCADA runs in one instance on a single processor. With this configuration, a single Kernel window reports on all CitectSCADA processes.

If CitectSCADA runtime is configured to run in multiple instances, each instance has a separate Kernel window that displays instance-specific system messages; for example, trend-related system messages appear in the Trend window in the Kernel. You can switch between Kernel windows by using the Kernel menu on the Kernel window menu bar.

A typical startup Kernel window for CitectSCADA using a default configuration looks like this:



```

Main
Citect Workstation.
Copyright (c) Citect Pty Ltd, 1987-2004
Version 6.1.109

Initializing Sub Systems
Initializing Font system
Initializing Client system
Starting IO Server
IO Server Started
Initializing Cicode system
Initializing com system
Initialize Request system
Initializing Trend Client system
Starting Trend Server
Checking For Trend Short File Names
Trend Startup
Initializing Trend Acq system
Initializing Alarm system
Loading Alarm Databases
Open Alarm Save File C:\Documents and Settings\All Users\Application Da
Loading Alarm Save File
Alarm Save File Loaded
Starting Alarm Server Processing
Initializing Report system
Starting Report Server
>Initializing Page system
Thu Sep 22 11:22:10 2005 26:09:40.374 Citect Startup 6.1.109
Initialize Functions
  
```

For a default configuration, the Kernel window displays the following messages:

- **Initializing Sub Systems** - The primary parts of CitectSCADA are getting started.
- **Initializing Font System** - Creating all fonts that have been defined within CitectSCADA. These are fonts used for displaying items such as alarms, and pre-V5.0 dynamic text.
- **Initializing Client System.**
- **Starting IO Server** - Only visible if the CitectSCADA computer is an I/O server. If the computer is an I/O server and this message does not display, the computer is improperly set up: run the Computer Setup Wizard to check your configuration. **IO Server Started** - The server has started and is functioning correctly. It is unlikely that this will fail.
  - **Initializing I/O Server** - Starting to check what is required for the I/O server to work, and initializing any cards that are required.
  - On a Client, these messages will be replaced with **Calling '<I/O Server>' Connected.**
- **Initializing Cicode System** - All Cicode has been loaded into memory, and is prepared to run.

- **Initializing com System** - Making sure that all ports and hardware is responding and functioning correctly.
- **Initializing Request System** - The system that handles requests from the Client part of CitectSCADA to the Server parts of CitectSCADA.
- **Initializing Trend Client System** - The Trend Client is slightly different than the normal client, so it needs separate initialization.
- **Starting Trend Server** - You will only see these messages if the CitectSCADA computer is a trends server. If the computer is a trends server, and these messages do not display, most likely the computer is improperly set up. You should run The Computer Setup Wizard to check your configuration.
  - **Trend Startup** - CitectSCADA is checking for all the trend files, and making new ones if they can't be found.
  - **Initializing Trend Acq System** - Every trend you define has its own sample rate. Here CitectSCADA is setting up the system so it can poll the data at the correct rate for each trend pen.

On a Client, these messages will be replaced with **Calling '<Trend Server>' Connected.**

- **Initializing Alarm System.**
- **Loading Alarm Databases** - You will only see these messages if the CitectSCADA computer is an alarms server. This is loading all alarm data into memory. If the computer is an alarms server, and these messages do not display, most likely the computer is improperly set up. You should run The Computer Setup Wizard to check your configuration.
- **Open Alarm Save File**  
**Loading Alarm Save File**  
**Alarm Save File Loaded** - CitectSCADA gets the alarm save file (that was created in the specified directory), and examines it in order to see the status of existing alarms. If you have a redundant alarms server, then this alarms server will interrogate the other instead of using the alarm save file - since the information on the other server will be newer than any file.  
**Starting Alarm Processing** - The server is now processing (and serving) alarm data.  
 On a Client, these messages will be replaced with **Calling '<Alarm Server>' Connected.**
- **Initializing Report System.**
- **Starting Report Server** - You will only see this message if the CitectSCADA computer is a reports server. If the computer is a reports server, and this message does not display, most likely the computer is improperly set up. You should run The Computer Setup Wizard to check your configuration.

On a Client, this message will be replaced with **Calling '<Report Server>' Connected.**

- **Initializing Page System** - CitectSCADA will now display the Startup Page. At this time CitectSCADA will cover up the Kernel if it is displayed.
- **Initializing Functions** - Executing any Cicode functions that have been defined as running at start up.

The next line of information is the start up time and CitectSCADA version number.

- **Channel PORT# is Online**  
**Channel PORT# is Online**  
**Channel PORT# is Online** - You will only see these messages if the CitectSCADA computer is an I/O server. These are messages telling you that any ports you have defined in the I/O server have come online. If you get a messages saying that the port is not online, or could not be opened, check the configuration of your project. *PORT#* is the Port Name specified in the Ports form.
- **Unit 'UNIT#' Port PORT# is Online**  
**Unit 'UNIT#' Port PORT# is Online**  
**Unit 'UNIT#' Port PORT# is Online** - Only visible if the CitectSCADA computer is an I/O server. This indicates that the I/O device with the Unit Number of *UNIT#* (as defined in the I/O Devices form), is connected to port *PORT#*.
- **Communication System Online** - CitectSCADA has completed startup operations and is now fully operational (running).

#### What to look for

All systems in CitectSCADA should start smoothly. When commissioning a system, you should check the Kernel. If any element repeatedly fails at startup, your CitectSCADA system is not working correctly and you should investigate.

Common problems that can cause startup errors are:

- Incorrect computer setup (usually solved by the Computer Setup Wizard).
- Networking faults or bad hardware.
- Communication faults (usually just a configuration issue).

Use the Main window to check that all your I/O devices come online correctly when starting. First the ports must initialize, then the I/O device itself will come online. If there is a problem, CitectSCADA displays a message such as "PLC not responding", "I/O Device Offline" or similar.

Some I/O devices might take two attempts to come online. If so, CitectSCADA waits (usually 30 seconds) and tries again. If the I/O device does not come online after the second attempt, check your configuration (at both ends) and cabling.

**Note:** The Kernel continues to report changes in the status of I/O devices to the Main window. This information might also be reported as alarms to the Hardware Alarms page.

See Also [Using Kernel Commands](#)

## Using Kernel Commands

Commands are issued at a command line interface (similar to the DOS prompt), usually from the Main Kernel window. Some commands display their results in the Main Kernel window; others open a child window for information display (or for further commands). You can open a maximum of five windows at once.

You can use several keyboard keys to scan and reuse commands from the command history, to speed up the issuing of Kernel commands. (The command history is a list of commands that you have previously issued). These keyboard keys are listed below:

Key	Description
Up arrow	Scans backward through the command history. (Commands are displayed in the command line.)
Down arrow	Scans forward through the command history. (Commands are displayed in the command line.)
F3	Puts the last command you issued in the command line.
Left arrow	Moves the cursor back one character at a time (in the command line).
Right arrow	Moves the cursor forward one character at a time (in the command line).
Delete	Deletes the character to the right of the cursor (in the command line).
Backspace	Deletes the character to the left of the cursor (in the command line).
Insert	Switches from over-strike mode to insert character mode (in the command line).

**Note:** When typing a command in the Main window, if a message appears in the middle of your command, you can still execute the command normally. Use the Shell command to open a new command window.

See Also [Kernel commands](#)

## Kernel commands

The table below describes the kernel commands.

Command	Description
<a href="#">Cache</a>	Changes the cache timeout for each I/O device.
<a href="#">Cicode</a>	Opens a child window that you can use to call Cicode functions.
<a href="#">Cls</a>	Clears all text from the Main or Cicode windows.
<a href="#">Debug</a>	Enables the debugging of raw <a href="#">data transferred</a> between CitectSCADA and a driver.
<a href="#">DriverTrace</a>	Lists the driver control blocks (DCBs) on the I/O Server awaiting delivery to a driver.
<a href="#">Exit</a>	Closes a Cicode or Shell window.

Command	Description
<a href="#">Help</a>	Displays a list of some of the commands available in the Kernel.
<a href="#">INI</a>	Displays the local <code>citect.ini</code> file
Kernel Alarm	Deprecated in v7.0
Kernel Trend	Deprecated in v7.0
Kernel Report	Deprecated in v7.0
Kernel IO Server	Deprecated in v7.0
Kernel Client	Deprecated in v7.0
<a href="#">Log</a>	Enables or disables the logging of I/O device reads and writes.
NetBIOS	Obsolete in v7.0.
<a href="#">Page General</a>	Displays general statistics information.
<a href="#">Page Driver</a>	Displays information about each driver in the CitectSCADA system.
<a href="#">Page Memory</a>	Displays the memory debug heap.
Page Netstat	Obsolete in v7.0.
<a href="#">Page Table</a>	Displays information about CitectSCADA's internal data structures.
<a href="#">Page RDB</a>	Displays information about CitectSCADA's Runtime Databases.
<a href="#">Page Unit</a>	Displays information about each I/O device in the CitectSCADA system.
<a href="#">Pause</a>	Pause debug output.
Probe	Deprecated in v7.0
<a href="#">Shell</a>	Opens a new command (shell) window.
<a href="#">Stats</a>	Resets all system statistics.
<a href="#">SysLog</a>	Displays the <code>syslog.dat</code> file.

### Cache

Changes the cache timeout for each I/O Device with which Citect is communicating.

**Syntax** **Cache** *<I/O Device name>* *<Timeout>*

Where:

*<I/O Device name>* . Any valid I/O Device defined in the project (using the I/O Devices form), or \* for all I/O Devices.

*<Timeout>* . . . . . Timeout in milliseconds, or 0 (zero) to disable timeout.

This command allows you to tune the cache timeout while the I/O Server is communicating with the I/O Devices. If you set the timeout to 0 (zero), the cache is disabled. If you specify a cache timeout for an I/O Device that has the cache disabled, the cache is enabled.

Any changes made to an I/O Device only apply while the I/O Server is running. If you restart the I/O Server, the cache timeout reverts to the value configured in the project. Once you have determined the optimum cache timeout, you should make the change permanent by setting the value in the I/O Devices form (for the particular I/O Device).

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

#### Cicode

Opens a child window that you can use to call Cicode functions, on either a local or remote computer. Any in-built or user-written function can be called from this window.

**Syntax** **Cicode** [*<Name>*]

Where:

*<Name>* . . . . . Optionally the name of a Citect server (e.g. Alarm, Report, Trend) or a client computer name.

If you enter the Cicode command with no Name argument, a local Cicode window is created. All Cicode commands are executed on the local computer.

If you enter a server or computer name as the Name argument, you can create a Cicode window to the remote server or computer. All Cicode commands entered in a remote Cicode window are executed on the remote computer. For example, to create a Cicode window where all commands execute on the Alarms Server, use:

```
Cicode Alarm
```

If you issue the command from a server, you can create a window to the "MyComputer" computer:

```
Cicode MyComputer
```

If the remote computer can be found, its name is displayed in the title of the Cicode window, otherwise a local window is created.

**Note:** You can only specify a computer name if you are issuing the command on a server. This function only supports Client to Server or Server to Client connection.

Each Cicode command is executed with its own Cicode task, so you can start tasks that take a long time to complete. The Cicode prompt returns immediately after the Cicode task has started and the task continues to run in the background. If the function is completed immediately, the return result of the function is displayed. If the function continues to run, the result is not displayed and cannot be returned - the message "Task still running" and the task handle is returned instead.

**Note:** Remember that there is no Privilege check on any command issued from this window, so you have full access to the system.

The Cicode prompt 0:> shows the current window number with which any object is associated. To change the current window, use the [WinGoto\(\)](#) function (or any other Cicode function that affects the current window).

The Cicode window does not recognise any variable names, so when you call a Cicode function, you can only pass constants (e.g. numbers or strings). When you call a function that expects a string, you should pass a string constant, e.g. `Prompt("Hello from the Kernel")`. If the string is only a single word, you do not have to use delimiters, e.g. `Prompt(Hello)`. The Cicode window tries to convert whatever you enter (as arguments) into the correct data type. If it cannot convert the arguments, it passes either 0 (zero) or an empty string to the function.

**Note:** Some Cicode functions are implemented as label macros by the compiler. These macros allow backward compatibility when the number of arguments to a function has been changed. Because the Cicode window does not expand macros, you cannot call these functions directly. You must use the macro expansion. If the function you are trying to use cannot be found, try again by adding an underscore (\_) to the front of the function name, e.g. `_DevClose(1)`. You can also shut down the Citect system from this window by using the `Shutdown()` function.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

#### Cl

Clears all text from the Main or Cicode windows, and moves the cursor to the top left-hand corner.

**Syntax** **Cl**

Use this command to clear the current window when it has become cluttered (from displaying debug data or too many commands).

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

#### Debug

Enables the debugging of all raw data that is transferred between Citect and a selected driver. All protocol traffic is displayed in the Kernel window and logged to the SysLog.DAT file.

**Syntax** **Debug** <Port> <Mode> [<Display>]

Where:

<Port> ..... A port name configured in the project (using the Ports form)

<Mode> ..... The mode:

ALL - Trace all low level communication traffic to the Kernel window.



READ - Trace the low level communication traffic of all read commands to the Kernel window.

WRITE - Trace the low level communication traffic of all write commands to the Kernel window.

ERROR - Trace any low level communication traffic that contains a protocol error. This mode only generates traces when an error occurs, so you can leave this option on for long periods of time (to find difficult problems).

TO - Display the high-level requests going to the protocol driver.

FROM - Display the high-level request coming from the protocol driver.

OFF - Stop the debug trace of any type of command.

<Display> . . . . . Optionally the display scenario:

MONO - Send the debug to a monochrome monitor (if one is attached) as well as the Kernel.

MONOONLY - Send the debug to the monochrome monitor only.

To place a port in debug mode, enter DEBUG followed by the port name and the mode you require. If you do not know the name of the port, enter DEBUG (without any arguments), and Citect displays a list of all available ports.

Only the I/O server communicates with the I/O Devices, so this command is generally used only on the I/O Server.

When you enable a debug trace mode, Citect displays all protocol traffic in the Kernel window and logs it to the SysLog.DAT file. This tends to reduce Citect's performance (as there may be a lot of data), and therefore you should not enable debug trace on an I/O Server that is critical to your current operation. Use this command only during commissioning or on a non-vital section. Excessive use of this command may cause the I/O Device to go off line.

When the debug trace is sent to a Monochrome monitor, it is displayed directly from the interrupt routine of the driver and therefore at a much faster rate. This trace (if MONOONLY is used) causes less CPU overload of Citect while the trace is active, and provides instantaneous output. This method is useful if you are developing your own driver and your driver crashes before the debug trace is displayed in the Kernel.

You can use the [Shell](#) command to create an extra command window while the trace is active. This allows you to enter more commands (in the new window).

You can use the [Pause](#) command to stop the debug output (to view the data).

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

**DriverTrace**  
Lists the commands and information issued to a driver by CitectSCADA. It can be useful in debugging a driver, for example, it can help determine why a driver is unable to fully initialize.

**Note:** Running syslog traces can draw heavily on a CPU usage. You should monitor the impact on CPU performance when implementing a large number of traces.

**Syntax**     **DriverTrace**  
[OFF|CMDS|VER[BOSE]|PORT=<name>|MASK=<xxxxxxxx>|DUMP]

Where:

*blank* ..... Returns the current DriverTrace settings. This can be useful to determine the current mask or port settings before implementing a new trace. Just key DriverTrace into the Kernel and hit the return key to view the current settings.

*OFF* ..... Turns the DriverTrace off.

*CMDS* ..... Turns the DriverTrace on, but does not display the contents of the data buffer.

**Note:** Specific driver commands must be enabled with the Kernel command DriverTrace[Mask], or the DriverTraceMask INI parameter.

*VER[BOSE]* ..... Turns the DriverTrace on, and displays the contents of the data buffer.

**Note:** Specific driver commands must be enabled with the Kernel command DriverTrace[Mask], or the DriverTraceMask INI parameter.

*PORT* ..... allows traces to be limited to a particular driver port, by defining a port name. The default setting, PORT=\*, will trace all ports.

*MASK* ..... a 4-byte hexadecimal number that represents a bit mask used to either include or exclude driver commands from the DriverTrace. The driver commands and their values are as follows:

Command	Bit Position
CTDRV_INIT	00000001
CTDRV_OPEN	00000002

Command	Bit Position
CTDRV_INIT_CHANNEL	00000004
CTDRV_INIT_UNIT	00000008
CTDRV_READ	00000010
CTDRV_WRITE	00000020
CTDRV_CONVERT	00000040
CTDRV_CANCEL	00000080
CTDRV_CPU	00000100
CTDRV_DATABASE	00000200
CTDRV_STOP_UNIT	00000400
CTDRV_STOP_CHANNEL	00000800
CTDRV_CLOSE	00001000
CTDRV_FORMAT	00002000
CTDRV_STATS	00004000
CTDRV_DEBUG	00008000
CTDRV_INFO	00010000
CTDRV_STATUS_UNIT	00020000
CTDRV_INIT_CARD	00040000
CTDRV_UPDATE_INFO	00080000
CTDRV_UI_READ	00100000
CTDRV_UI_WRITE	00200000
CTDRV_EXIT	00400000
CTDRV_LINE_STATUS	00800000
CTDRV_SEND_BREAK	01000000
CTDRV_REINIT_CHANNEL	02000000
CTDRV_SET_PARAM	04000000
CTDRV_GET_PARAM	08000000
CTDRV_OPEN_PORT	10000000
CTDRV_CLOSE_PORT	20000000
CTDRV_STATUS_DISCONNECT	40000000

For example, the value you would use to include only the CTDRV\_OPEN, CTDRV\_INIT\_UNIT and CTDRV\_READ commands would be: 0000001A.

Most users will want to exclude the CPU function call, as this happens often. Do this by setting a mask of 7ffffeff.

The default <mask> is 7FFFFFFF.

*DUMP* ..... Lists the driver control blocks (DCBs) on the I/O Server awaiting delivery to a driver and actioning from the driver.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

**Exit**

Closes the Cicode or Shell windows.

**Syntax**   **Exit**

**Note:** You cannot use this command to close the Main window. See [Closing the kernel window](#).

To close all other windows, select Close from the window's control-menu box, or press Esc, or double click the control-menu box.

See Also   [Kernel commands](#)  
[Displaying the kernel window](#)

**Help**

Displays a list of some of the commands that are available in the Kernel.

**Syntax**   **Help**

See Also   [Kernel commands](#)  
[Displaying the kernel window](#)

**INI**

Displays local citect.ini file. (If you are using a Citect Internet Display Client to run a project over the internet, the .ini file in the bin directory will display). You can use the Page Up, Page Down, and arrow keys to move through the file, but you cannot edit or save it.

**Syntax**   **INI**

**Note:** This window is the same that is used to display the syslog.dat file. You can display either the citect.ini or syslog.dat in this window, but you cannot display both at the same time.

See Also   [Kernel commands](#)  
[Displaying the kernel window](#)

**Log**

Enables or disables the logging of I/O DEVICE reads and writes to the syslog.dat file.

**Syntax**   **Log** *<I/O Device>* *<Mode>*

Where:

*<I/O Device>* . . . . . The name of the I/O Device to log

*<Mode>* . . . . . Either: READ, WRITE, or OFF

See Also   [Kernel commands](#)  
[Displaying the kernel window](#)

## Page General

Displays general statistics information on the overall performance of Citect.

### Syntax **Page General**

#### **General Statistics**

Server Name	The server name of this computer or, if it is a client, the name of the primary server this client talks to.
Node Name	The computer name of this computer. You can set this through the Computer Setup Wizard. This is only used if you have Citect in a networking configuration.
Time	The (current) time of day and the date.
CPU Index	An indication of the performance of the Computer. This number provides a rough indication of the performance of the computer. On a Compaq 486/25M it will be 25.
Running	The total time Citect has been running
Memory Total	The total amount of free memory (including virtual memory).
Memory Physical	The total amount of free physical memory. The physical RAM that is free on the computer (not including virtual memory).
Memory Resources	The % of free Windows system resources.
Scheduler Cycles	<p>The number of times that the Citect scheduler is looking for a task to execute. A good indication of how fast the computer can respond to an event. This number depends on the speed of the computer and the CPU resources that Citect is using.</p> <p>This value is completely different depending on whether you are running 32-bit or 16-bit Citect. This is because Citect can use the 32-bit operating system to perform process scheduling, an operation that Citect must perform itself in the 16-bit environment.</p> <p>16-bit: The busier Citect is, the lower the number. Typical values of 10,000 to 50,000; a highly loaded system may drop to below 1000.</p> <p>32-bit: The busier Citect is, the higher the number. Typical values of 100 to 1000; a highly loaded system may rise above 5000.</p>
CPU Usage	<p>The percentage of the total available computer processing power that is being used on this computer. Under Windows 95, this figure may be unreliable, as some 16-bit applications will have no effect on this value! As the percentage increases to a high level, the performance of Citect may level off or become sluggish. If the CPU usage is consistently very high (greater than 70%), there could be a problem with your system or you may be overloading the CPU. For best performance, your system should run between 0% and 40%. However, under NT it can be perfectly OK to run at 100% CPU utilisation, since Citect can only access the CPU when the operating system lets it.</p>
Tasks Per Sec	The number of tasks per second that the Citect scheduler is executing, to show how busy Citect is. This number should be between 30 and 200 tasks per second. If the computer is an I/O Server, the task number is higher (because each protocol uses many tasks).

---

CPU Usage	The percentage of the total available computer processing power that is being used on this computer. Under Windows 95, this figure may be unreliable, as some 16-bit applications will have no effect on this value! As the percentage increases to a high level, the performance of Citect may level off or become sluggish. If the CPU usage is consistently very high (greater than 70%), there could be a problem with your system or you may be overloading the CPU. For best performance, your system should run between 0% and 40%. However, under NT it can be perfectly OK to run at 100% CPU utilisation, since Citect can only access the CPU when the operating system lets it.
Tasks Per Sec	The number of tasks per second that the Citect scheduler is executing, to show how busy Citect is. This number should be between 30 and 200 tasks per second. If the computer is an I/O Server, the task number is higher (because each protocol uses many tasks).

### **I/O Server Statistics**

The following statistics are only incremented if the computer is configured as an I/O Server:

Write Requests	The first number is the total number of write requests sent to the I/O Server from all client Citect computers, including the local Citect client. The second number is the write requests per second (that the I/O Server is performing). The number of requests depends on the rate at which clients are sending write requests to the I/O Server (typically 0 to 20). If the number of requests always exceeds 10, then this may be causing performance problems (usually caused by Cicode performing too many writes to the I/O Devices).
Physical Writes	The first number is the total number of physical writes made to the I/O Devices. Because the I/O Server can optimise the number of write requests made to the I/O Devices, this number is usually smaller than the number of write requests. The second number is the number of physical writes per second (that the I/O Server is performing). The number of writes depends on the rate at which clients are sending write requests to the I/O Server (typically 0 to 20). If the number of requests always exceeds 10, then this may be causing performance problems (usually caused by Cicode performing too many writes to the I/O Devices).
Blocked Writes	The total number of times a request was made for the same I/O Device address while the I/O Server was already writing that address. The server blocks the two requests together as an optimisation.
Register Reads	The total number of register points that the I/O Server has read from all I/O Devices.
Register Reads per Sec	The number of register reads per second that the I/O Server is processing. This provides a general indication of performance. It is dependent on the protocol.
Cache Reads	The number of read requests serviced from the read cache.
Cache Reads(%)	The percentage of reads serviced from the cache. If the cache read % is large, (e.g. greater than 40%), the I/O Device cache timeout could be set too high. Try reducing the I/O Device cache time to bring the % cache below 40%. The % of read cache depends on the configuration of your Citect system and the number of clients connected. If you have many clients looking at the same I/O Device data, the cache % may be very high, however this does not cause a problem. For example, if you have 10 Citect clients viewing the same page, a cache read of 90% would be acceptable.
Cache Flush	The number of times a cache buffer was flushed because a write request was directed to the same location. When you write to the I/O Device and that address is in the read cache, Citect flushes the data from the cache. The next time the data is read, it is reloaded from the I/O Device to reflect the new value.
Cache RD Ahead	The number of read-ahead updates made to the cache buffer. When read ahead caching is enabled, the I/O Server will try to read any I/O Device data which is coming close to cache 'timeout' time. The I/O Server will only read this data if the communication channel to the I/O device is idle - to give higher priority to other read requests.

Cache Buffers	The number of active cache buffers allocated. Each block of data that is read from the I/O Device requires a cache buffer when stored in the cache. The number of cache buffers active at once depends on the dynamic operation of the Citect computers and their project configurations (typically 0 to 100).
Cache Short	The number of times the cache needed to allocate additional buffers but no buffers were available. When this happens the server cannot cache the data. This does not generate errors but it does lower performance. If this field increments too quickly, increase the available memory.
Response Times	<p>The time taken by the I/O Server to process read and write requests (i.e. the time from when a request arrives at the server to when it is sent back to the client). This time depends on the physical response time of the I/O Device and how long the request had to wait in a queue for other requests to be completed. This time increases as the server's loading increases, and is the best indication of the total system response.</p> <p>The average, minimum, and maximum times are displayed. Typical values depend on the I/O Device protocol, however Citect should have an average response of 500 to 2000 ms. Slower protocols or a heavily loaded system may make the response time higher, but you should be able to get response times of less than two seconds even for very large systems.</p>
Points	The maximum number of I/O points that can exist in your Citect system. The combined Static and Dynamic count cannot exceed this number. The point limit is part of the Citect software protection, and is programmed into your hardware key.
Max Full	The maximum number of fully functional Citect computers (Full Licenses) that can exist in your Citect system. This number is part of the Citect software protection, and is programmed into your hardware key.
Current Full	The current number of fully functional Citect computers (Full Licenses) that are running in your Citect system.
Peak Full	The peak number of fully functional Citect computers (Full Licenses) that your Citect system has experienced since it was re-started.
Static	The number of static I/O points.
Max Mngr	The maximum number of Manager Clients that can exist in your Citect system. This number is part of the Citect software protection, and is programmed into your hardware key.
Current Mngr	The current number of Manager Clients that are running in your Citect system.
Peak Mngr	The peak number of Manager Clients that your Citect system has experienced since it was re-started.
Dynamic	The number of dynamic I/O points.
Max Dsp	The maximum number of Display-only Clients that can exist in your Citect system. This number is part of the Citect software protection, and is programmed into your hardware key.
Current Dsp	The current number of Display-only Clients that are running in your Citect system.



Peak Dsp                      The peak number of Display-only Clients that your Citect system has experienced since it was re-started.

See Also    [Kernel commands](#)  
[Displaying the kernel window](#)

#### Page Driver

Displays information about each driver in the Citect system. This window is only displayed if the Citect computer is configured as an I/O Server with physical I/O Devices attached.

#### Syntax    **Page Driver**

Use Page Up and Page Down keys to scan the driver list.

#### Driver Statistics

Port Name	The name of the physical port the driver is using for communication.
Protocol	The name of the protocol.
Title	The protocol title and version string. All protocol drivers for Citect Versions 3.xx, 4.xx, and 5.xx, are Version 2.0 type drivers.
Read Requests	The first number is the total number of read requests sent to the driver from all client Citect computers, including the local Citect client. The second number is the read requests per second (that the I/O Server is performing).
Physical Reads	The first number is the total number of physical reads made to the I/O Devices. Because the I/O Server can optimise the number of read requests made to the I/O Devices, this number is usually smaller than the number of read requests. The second number is the number of physical reads per second (that the I/O Server is performing).
Blocked Reads	The total number of times a request was made for the same I/O Device address while the driver was already reading or about to read that address. The driver blocks the two requests together as an optimisation.
Digital Reads	The total number of digital points that the I/O Server has read from all I/O Devices.
Digital Reads per Sec	The number of digital reads per second that the I/O Server is processing. This provides a general indication of performance. It is dependent on the protocol.
Write Requests	The first number is the total number of write requests sent to the driver from all client Citect computers, including the local Citect client. The second number is the write requests per second (that the I/O Server is performing).

Physical Writes	<p>The first number is the total number of physical writes made to the I/O Devices. Because the I/O Server can optimise the number of write requests made to the I/O Devices, this number is usually smaller than the number of write requests.</p> <p>The second number is the number of physical writes per second (that the I/O Server is performing).</p>
Blocked Writes	The total number of times a request was made for the same I/O Device address while the driver was already writing that address. The driver blocks the two requests together as an optimisation.
Register Reads	The total number of register points that the I/O Server has read from all I/O Devices.
Register Reads per Sec	The number of register reads per second that the I/O Server is processing. This provides a general indication of performance. It is dependent on the protocol.
Cache Reads	The number of read requests serviced from the read cache.
Cache Reads(%)	<p>The percentage of reads serviced from the cache. If the cache read % is large (e.g. greater than 40%), the I/O Device cache timeout could be set too high. Try reducing the I/O Device cache time to bring the % cache below 40%.</p> <p>The % of read cache depends on the configuration of your Citect system and the number of clients connected. If you have many clients looking at the same I/O Device data, the cache % may be very high, however this does not cause a problem. For example, if you have 10 Citect clients viewing the same page, a cache read of 90% would be acceptable.</p>
Error Count	The total number of errors encountered by the driver.
Short buffers	The number of times the server needed a buffer for the driver but none were available. This can reduce communication performance and result in loss of requests. Increase available memory if this field increments too quickly (ie more than 10 per minute).
Driver Errors	The number of low level driver errors encountered before retries were performed. This field may continue to increment even if no errors are reported because the driver retries and it may complete the command on the second attempt. If this field increments quickly (i.e. 10 or more per minute) without other errors being reported, you may have a low level error that affects performance.
Out of Buffers	<p>The number of times the driver requires a buffer but none were available. The driver must discard the data. If this field increments too quickly, increase the number of pending commands the driver can process. This field only increments with client drivers. I/O Device drivers do not require pending buffers.</p>
Time Outs	The number of timeouts encountered by the driver during operations. This field may continue to increment with no visual errors as the driver retries the operation. If this field increments excessively, there may be a communication problem.
Retrys	The number of retries executed by the driver. If this field is incrementing, there may be a communication problem.

Maximum Pending Commands	<p>This is the number of requests that are kept in a buffer waiting to be serviced by the protocol. There is a pre defined default value for this for every protocol. If a protocol is capable of handling multiple requests or commands at a time then this number may be high. If the protocol is capable of only handling 1 command at a time then this will probably be 1 or 2.</p>
Blocking Size (Bytes)	<p>This is the range that the I/O Driver will use when blocking read or write requests together at runtime. The default value for this is typically set after quite a lot of experimentation. The value is calculated as the optimum range of data that the I/O Device can respond to, to get the fastest response times from your I/O Device.</p> <p>For example, if the Blocking Size is 100 and you have a graphics page that has Address1 and Address 99 on it, Citect will read both of these addresses in one request. If you have Address 1 and Address 101 on the page Citect will issue 2 separate read requests. The block size may not be the maximum packet size for the protocol, since a particular type of I/O Device may respond faster to smaller requests of data than larger requests. There is a drawback to this method; many I/O Devices must have their Memory tables created by the user. If the user does not define all memory addresses in a range, then Citect may try and read a block of memory from the I/O Device that does not exist (giving a hardware alarm). This is because Citect will ask for the whole range of addresses between the starting address and the ending address. So, if in our previous example the I/O Device did not have address 76, it would report back to Citect that it could not read address 76. The I/O Driver does not know that it doesn't need this address and will retry the command, and in some cases will eventually put the I/O Device offline.</p> <p>Make sure that you always have defined the memory addresses that Citect will need to read.</p>
Timeout Period (ms)	<p>This is the period of time that the I/O Driver will wait before re-requesting data, if no answer comes from the I/O Device.</p>
Maximum Retrys	<p>The number of times the I/O Driver will attempt to get data from the I/O Device.</p> <p>Combining this with Timeout gives you the total period before Citect will put an I/O Device offline.</p> <p>If the Timeout=2000 and Retry=2 then Citect will wait 2 seconds for a response, then retry, wait 2 seconds, retry, wait 2 seconds, Offline. Total time between losing communications and deciding it is offline is now 6 seconds. You can modify these parameters, but if you set them too low you will generate unneeded retries and possibly get I/O Device Offline messages.</p>
Poll Time (ms)	<p>This is the time in milliseconds that Citect will check the port for data or write data to the port. If this is 0 then the protocol is operating in Interrupt mode.</p>

Transmit Delay (ms)	This is the time that Citect will hold a packet of data between receiving a response from the last request and sending the new request. This is usually 0, however some protocols can become saturated and start to misbehave. In these cases the default value has been calculated while the protocol was being tested, and modifying this value to something smaller will cause problems. However, making it bigger will only have a very slight impact on the overall response times in your system, but may make the communications more stable.
Watchtime (seconds)	This is the period of time that Citect will wait after deciding an I/O device is offline before trying to re-establish communications. This is typically 30 seconds. It can be made smaller but must never be made smaller than the period that Timeout and Retry will be - otherwise you will never be able to re-establish communications with an I/O Device.
Response Times	The time taken by the driver to process read and write requests (i.e. the time taken to process a single read or write operation to the I/O Device). This time depends only on the physical response time of the I/O Device, because no queue waiting time is included. This field reflects any tuning of the communication channel (e.g. increasing the baud rate should reduce the response time). The average, minimum and maximum times are displayed.
Channel Usage	This field displays the percentage of the total capacity of the hardware channel that is currently being used. The I/O server always tries to keep the utilisation as high as possible, however if the client Citect computers are requesting data slower than the channel can supply, the total will be below 100%. It is possible for the channel usage to rise above 100% as some I/O Device drivers can process more than one command at the same time (having two or more commands using the channel at the same time).
Bytes Per Second	The number of bytes transferred (each second) by the driver. This number provides a simple performance indication that is useful when tuning the driver.
Special Variables	By enabling verbose mode (press V) or by pressing the down arrow, twenty special variables are displayed. The meaning of these variables is driver-specific.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

#### Page Memory

Displays memory debugging information. This window is designed for use by Citect specialists, and requires a high degree of expertise to use.

**Syntax** **Page Memory**

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

### Page Table

Contains information about Citect's runtime data structures. This area is very extensive and is initially a bit difficult to navigate. Currently there are about 50 tables of information - most of which are of little use to you.

#### Syntax **Page Table** [<Name>]

Where:

<Name> . . . . . Optionally the name of the table to display.

Use Page Up and Page Down keys to navigate through the table list. Use Up Arrow and Down Arrow keys to scroll the table data.

The MASTER\_TABLE (Table 1) lists all of the tables. The following tables are particularly useful:

#### **Page Table Stats**

This very useful table contains the cycle and execution times of every task that is running in Citect. The Execution time is the time taken for the entire task to run. The Cycle time is the time between when a task starts and when it starts again. The CPU is the percentage of total available CPU that the task is using (fast tasks often have 00 CPU).

The Citect 0 entry is the display task (graphics page updates) That is the total time taken for the Client to request data from the I/O Server, the I/O Server to get the data and sent it back to the Client, and the Client to update the display.

**Note:** Citect 0 corresponds to the display task for the main window. Citect 1 for the first child window, Citect 2 for the third, and so on.

The CodeX entries correspond to Cicode tasks, where X is the handle of the task. You can find out which task corresponds to which handle by viewing Cicode table.

**Note:** There will be a Trend Acq entry for every different trend sampling period you have defined in your project.

#### **Page Table Cicode**

This table contains the a list of all Cicode tasks currently running. It contains the task name, handle and running state, as well as some statistics. CPU\_Time is the total time that the task has run for - it is incremented each time the task runs. The CPU is the percentage of total available CPU that the task is using (fast tasks often have 00 CPU).

#### **Page Table Tran**

Obsolete in v7.0.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

Page RDB  
Displays information about each of Citect's runtime databases (RDBs).

Syntax **Page RDB**

Use the Page Up and Page Down keys to move between the tables.

The runtime databases contain your compiled project configuration information. There are two types of RDB; resident and non-resident.

Resident databases are loaded at Citect startup and remain in memory. Examples of resident databases are Alarms, Trends, Reports, and Functions. The names of resident databases always start with the underscore character (\_).

Non-resident databases are those that are associated with pages. These databases are loaded (and unloaded) as required - when the page is displayed.

Each database is divided into 10 (or so) tables; <name>, TEXT, REQUEST, PIECE, CODE, SCALE, RUN, WRITE, FUNC, SYMB, and <name>.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

Page Unit  
Displays information about each I/O Device in the Citect system. This information is displayed if the Citect computer is configured as an I/O Server or simply as a client. If the computer is a client, then all the I/O Devices for all the I/O Servers are displayed. If the computer is an I/O Server, then only the I/O Devices for that I/O Server are displayed. You can display I/O Devices from other I/O Servers by using the Verbose mode (press V to enable Verbose mode). If the computer is a client, then not all of the I/O Device Information is updated (because only the I/O Server has this information).

Syntax **Page Unit**

If the Citect computer is a client, the status and error codes are only local to the computer and do not reflect the true status of the I/O Device on the I/O Server. Note that all configured I/O Devices are displayed in this window, not just the I/O Devices for the particular I/O Server. (Any remote I/O Devices do not reflect the true status of the I/O Device).

Use the Page Up and Page Down keys to scan the I/O Device list.

**I/O Device Information**

I/O Device	The name of the I/O Device defined in the project (with the I/O Devices form).
IO Server	The name of the I/O Server that is servicing this I/O Device.
Comment	A description of the I/O Device defined in the project (with the I/O Devices form).

I/O Device No	The I/O Device number defined in the project (with the I/O Devices form).
PLC Number	The physical I/O Device address defined in the project (with the I/O Devices form).
Port Name	The communication port to which the I/O Device is connected.
Protocol	The protocol used for communication with the I/O Device.
Server Status and Client Status	<p>The status of the I/O Device. The Server Status is only valid if the computer is an I/O Server and it is servicing this I/O Device. The Client Status field is valid for Clients only, and indicates the status of the I/O Device that is attached to the I/O Server. The I/O Device status can be one of the following:</p> <p>RUNNING - Indicates that the communication link with the I/O Device is good.</p> <p>STANDBY - Indicates that the communication link with the I/O Device is good, but communication with that I/O Device is currently being performed by another port. This port is in standby mode.</p> <p>STARTING - Indicates that the server is currently establishing a communication link (with the I/O Device).</p> <p>STOPPING - Indicates that the server is currently relinquishing control of the communication link (with the I/O Device).</p> <p>OFFLINE - Indicates that the server cannot establish a communication channel with the I/O Device. If a standby port or server is available, Citect tries to communicate to the I/O Device using that port.</p> <p>REMOTE - Indicates that the status of the I/O Device is OK, but it is not currently connected.</p>
Primary	Indicates if the I/O Device is in primary mode; Yes = Primary, No = Standby. If the I/O Device is in primary mode, the server starts a communication channel with the I/O Device as soon as the server is activated. If an I/O Device is in standby mode, the I/O Device remains inactive when the server starts (until a primary I/O Device fails).
Client Using	The name of the I/O Server that this client is using. This allows you to identify the primary and standby I/O Servers.
Generic Error	The last generic error code returned by the driver. Because all protocol drivers have their own special errors, they cannot be recognised by the I/O Server. The drivers convert their special errors into generic errors that can be identified by the server.
Error Handle	The error handle that is assigned by the I/O Server to each error. This handle is not used by Citect (at this time).
Driver Error	The driver-specific error code. Each driver has its own special error codes. Refer to the driver specific errors (for the particular protocol) for an explanation of each of the error codes.
Error Message	The error message associated with the generic error code.
Error Count	The total number of errors from the I/O Device.
Restarts	The number of times the server has tried to establish a connection with the I/O Device. This number is normally 1, because the server establishes a connection at startup. If this field displays a number greater than 1, there is a problem with the communication channel.

Server Status and Client Status	<p>The status of the I/O Device. The Server Status is only valid if the computer is an I/O Server and it is servicing this I/O Device. The Client Status field is valid for Clients only, and indicates the status of the I/O Device that is attached to the I/O Server. The I/O Device status can be one of the following:</p> <p>RUNNING - Indicates that the communication link with the I/O Device is good.</p> <p>STANDBY - Indicates that the communication link with the I/O Device is good, but communication with that I/O Device is currently being performed by another port. This port is in standby mode.</p> <p>STARTING - Indicates that the server is currently establishing a communication link (with the I/O Device).</p> <p>STOPPING - Indicates that the server is currently relinquishing control of the communication link (with the I/O Device).</p> <p>OFFLINE - Indicates that the server cannot establish a communication channel with the I/O Device. If a standby port or server is available, Citect tries to communicate to the I/O Device using that port.</p> <p>REMOTE - Indicates that the status of the I/O Device is OK, but it is not currently connected.</p>
Primary	<p>Indicates if the I/O Device is in primary mode; Yes = Primary, No = Standby. If the I/O Device is in primary mode, the server starts a communication channel with the I/O Device as soon as the server is activated. If an I/O Device is in standby mode, the I/O Device remains inactive when the server starts (until a primary I/O Device fails).</p>
Client Using	<p>The name of the I/O Server that this client is using. This allows you to identify the primary and standby I/O Servers.</p>
Generic Error	<p>The last generic error code returned by the driver. Because all protocol drivers have their own special errors, they cannot be recognised by the I/O Server. The drivers convert their special errors into generic errors that can be identified by the server.</p>
Error Handle	<p>The error handle that is assigned by the I/O Server to each error. This handle is not used by Citect (at this time).</p>
Driver Error	<p>The driver-specific error code. Each driver has its own special error codes. Refer to the driver specific errors (for the particular protocol) for an explanation of each of the error codes.</p>
Error Message	<p>The error message associated with the generic error code.</p>
Error Count	<p>The total number of errors from the I/O Device.</p>
Restarts	<p>The number of times the server has tried to establish a connection with the I/O Device. This number is normally 1, because the server establishes a connection at startup. If this field displays a number greater than 1, there is a problem with the communication channel.</p>



Response Times	The time taken by the driver to process read and write requests (i.e. the time taken to process a single read or write operation to the I/O Device). This time depends only on the physical response time of the I/O Device, because no queue waiting time is included. This field reflects any tuning of the communication channel (e.g. doubling the baud rate should half the response time). The average, minimum, and maximum times are displayed. <b>Note:</b> One I/O Device with a slow response can slow down your entire system. For example, if you have an I/O Device with a response of 2000ms, any pages in your system that use data from that device, will have a minimum update time of 2000ms.
Cached	This field indicates if the I/O Device data is cached.
Cache Timeout	If the I/O Device is cached, this field displays the cache timeout value. Data is held in the cache for this timeout period before being discarded and re-read from the I/O Device. Only read data is cached.
Blocking Constant	The current blocking constant value for this I/O Device, as specified in the protocol.
Dial-up Connection	The status and history of the dial-up connection. SUCCESS - The number of successful dial-up attempts. FAIL - The number of failed dial-up attempts. TOTAL - The total number of dial-up attempts. NEXT - The time of the next scheduled dial-up attempt.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

Pause  
Pauses debug output in the Kernel window.

**Syntax** **Pause**

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

Shell  
Opens a new command (shell) window.

**Syntax** **Shell**

You can use shell windows in a similar manner to a Main window. Shell windows are useful for displaying debugging information, or entering commands when the Main window is displaying debug trace data. You can close the shell windows by selecting Close from the window's system icon or with the [Exit](#) command.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

### Stats

Resets all system statistics (used in the page general, page drivers, page table (stats), and page I/O Device windows) to 0 (zero).

#### Syntax **Stats**

This command allows you to reset the statistics after Citect has been running for a long time, and therefore provides an indication of the statistics now (instead of an average over the total time that Citect has been running).

**Note:** Some I/O Server statistics are automatically reset every few minutes.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

### SysLog

Displays local SysLog.DAT file. The SysLog.DAT is read from disk before being displayed, but is not updated once it is displayed. You must close and re-open the window to force it to update. You can use the Page Up, Page Down, and arrow keys to move through the file, but you cannot edit or save it.

#### Syntax **SysLog [Delete]**

Use the optional Delete keyword to clear (purge) the contents of the SysLog.DAT file.

**Note:** This window is the same that is used to display the Citect.INI file. You can display either the Citect.INI or Syslog.DAT in this window, but you cannot display both at the same time.

See Also [Kernel commands](#)  
[Displaying the kernel window](#)

## Gathering Runtime Information

The CitectSCADA Kernel can display information about your CitectSCADA runtime system. The following areas are useful places to gather information:

- **General** - Statistics and information on the overall performance of CitectSCADA. For example, this page shows memory usage, summaries of protocol and I/O device statistics, as well as CPU usage. Access this by using the Page General command.
- **Table** - Contains information about CitectSCADA runtime data structures. This area is extensive and is initially difficult to navigate. However, Page Table **Stats** is insightful. Access this by using the Page Table command.
- **Driver** - Specific statistics and information about the individual protocols running on the I/O server. Each individual port has its own page of information. Access this by using the Page Driver command.

- **Unit** - Similar to the driver information, this shows specific statistics and information about each I/O device. Access this by using the Page Unit command.

See Also [System tuning](#)

## System tuning

CitectSCADA is designed for optimal performance, so it not necessary for most users to tune their system. However, special circumstances might require that you adjust your system for optimal performance. The Kernel allows you to locate areas that need tuning, and the tuning itself is usually done through parameters. For example, you can improve performance of the Display Client by using the [Page]ScanTime and [Alarm]ScanTime parameters.

### Cache tuning

The cache should be tuned large enough so that unnecessary reads are not generated, and small enough that old data is not returned while keeping the communication channel busy. If the cache is too large, the communication channel might become idle for a while and so waste its bandwidth. Also if the cache is too large, a CitectSCADA client might start to short cycle on reads request, which will generate unnecessary network or internal traffic load.

Read short cycling occurs when a client requests data from the I/O server, and the data is returned from the cache, so it is returned quickly. The client will process the data (display it on screen) then ask for the same data again. If the I/O server again returns the same data from the cache, the client will process the same data again which is redundant and a waste of CPU and the network (to transmit the request and response). When short cycling starts to occur, the CPU and network loading will rise while the PLC communication traffic will start to fall.

To tune the cache you must balance the cache time between unnecessary reads and short cycling. The method described below assumes you know how to use the CitectSCADA debugging Kernel.

- 1 Turn off all unit caching, use the CACHE command in the Kernel so you don't have to re-compile your project.
- 2 Run one CitectSCADA client only on the network, use the Client in the I/O server for the test.
- 3 Display a typical page to generate normal PLC loading for your system.
- 4 In the Kernel use the STATS command to reset all the CitectSCADA statistics.
- 5 In the Kernel display the page 'PAGE TABLE STATS'. This page shows the cycle and execution time of various CitectSCADA tasks, some of which consume PLC data. The tasks called 'Citect *n*' where *n* is a number are the tasks which get data from the PLC and display on screen. Look at the Avg Cycle time, this is the third column from the left. Assume that the Avg cycle

time is 1200 ms. This will mean that the current page is gathering all PLC data and displaying its data on the screen in 1200ms.

- 6 The cache time should always be below this average cycle time to prevent short cycling. On average it should be less than half this time, ie 600 ms.
- 7 Set the cache time to half the cycle time (600ms). You might not see any improvement in performance with a single client, as caching will only improve performance with multi clients. You might see improvements if you are also running trends, alarms or reports which are requesting the same data.
- 8 Add another CitectSCADA client that is displaying the same data. Reset the STATS and check the Average cycle time. Each new client should not increase the cycle time, it should drop slightly. Also look at PAGE GENERAL, to see that each new client should service its reads from the cache; i.e., the % cache reads increases.
- 9 If the average cycle time drops to less than half the original time then short cycling is occurring and you should decrease the cache time until this stops. Tuning the cache is a trial and error process - as you change it, the read cycle time will also change. The cache time will also depend on what the current PLC traffic is. The current traffic is dynamic as CitectSCADA will only read what is required depending on the current page, trend, alarm and reports running. You should always be on the safe side and set the cache a bit lower to stop short cycling under lower loading conditions.

# Chapter 38: CitectSCADA Runtime Manager

---

The CitectSCADA Runtime Manager is used to start up, stop, monitor, and shutdown CitectSCADA processes. This chapter describes how to use CitectSCADA Runtime Manager and covers:

- [Launching Runtime Manager](#)
- [CitectSCADA Runtime Manager Interface](#)
- [Hiding CitectSCADA Runtime Manager](#)
- [Shutting Down CitectSCADA Runtime Manager](#)
- [Monitoring Runtime Processes](#)
- [Starting a Process](#)
- [Stopping a Process](#)
- [Restarting a Process](#)
- [Cancelling a Process](#)

## Launching Runtime Manager

When CitectSCADA is launched, CitectSCADA Runtime Manager starts automatically and the CitectSCADA Runtime Manager interface appears. It then reviews the settings in the `citect.ini` file and the project and starts the required processes of CitectSCADA on the specified CPUs. The monitoring pane displays the verbose startup message for each process as they start (for more information see [Monitoring Startup](#)).

You can also start the Runtime Manager during the configuration of a project via the Tools menu in Citect Explorer, Graphics Builder and Project Editor. This will launch the Runtime Manager in an idle state, allowing you to start processes independently in a multi-process system. This can be useful for system testing and analysis, as it allows you to start a particular server or client without having to perform a full compile of a project.

**Note:** Only one instance of CitectSCADA Runtime Manager can run on a machine at any time.

When all the CitectSCADA processes have started, the CitectSCADA Runtime Manager window minimizes to an icon in the Notification area (bottom-right corner of the screen). The CitectSCADA Runtime Manager continues to run for the duration of the CitectSCADA application. While CitectSCADA Runtime Manager is running, it is available as a task bar icon in the Notification area.

Clicking the icon restores the CitectSCADA Runtime Manager window to the foreground and allows for the continued monitoring of the CitectSCADA processes.

#### Existing CitectSCADA processes

If CitectSCADA Runtime Manager finds any CitectSCADA processes already running, it checks to ensure the existing processes match the configuration specified in the CitectSCADA project and the `citect.ini` file.

If the existing runtime processes are configured as per the required configuration, CitectSCADA Runtime Manager:

- Accepts these processes and displays them in the monitoring pane.
- Displays any other processes configured in the `citect.ini` file in the monitoring pane but does **not** start these processes; these processes must be started manually. See [Starting a Process](#).

If the existing processes are **not** configured as per the required configuration, CitectSCADA Runtime Manager ignores the configuration and the Monitoring Pane displays the CitectSCADA Runtime currently running. To launch the configuration within the `citect.ini` file, you must shut down all existing processes and relaunch CitectSCADA.

## CitectSCADA Runtime Manager Interface

The CitectSCADA Runtime Manager interface is described below.

Each individual process on the workstation is represented by a line item in the monitoring pane. Every line item (process) has several columns showing the following information:

Column	Description
CPU	The CPU on which the process is running.
ProcessID	The Windows ProcessID.
Process	Lists each CitectSCADA component running in the process. Where CitectSCADA is running in single process mode and all components are sharing a process, they appear comma-separated on a single line in this column.
Type	The Type of process (Alarm, Report, Trend, IO Server or Client). Where CitectSCADA is running in single process mode and all components are sharing a process, the type is known as 'MIXED'.
Status	The status of each process. See <a href="#">Monitoring Runtime Processes</a> for a list of each status reported by CitectSCADA Runtime Manager.
Message	Displays message information relevant to the status of each process, such as the date and time of project compilation, as well as project name. For example, during startup of each process the verbose startup message appears in this column.

The CitectSCADA Runtime Manager interface also includes the following buttons:

- **Start All** - starts all the CitectSCADA processes for this workstation. (For details see [Restarting a Process](#).)  
Once a CitectSCADA process is running, this button becomes **Shutdown All**, which shuts down CitectSCADA Runtime Manager. (For details see [Shutting Down CitectSCADA Runtime Manager](#)).
- **Hide** - hides CitectSCADA Runtime Manager. (For details see [Hiding CitectSCADA Runtime Manager](#)).
- **Help** - opens the CitectSCADA Runtime Manager online help.

## Hiding CitectSCADA Runtime Manager

The **Hide** button minimizes CitectSCADA Runtime Manager to an icon in the Notification area. CitectSCADA Runtime Manager does not stop running, the interface is just hidden until required by the user.

Clicking the task bar icon in the Notification area restores the CitectSCADA Runtime Manager to the foreground to allow for the monitoring of the CitectSCADA processes.

## Shutting Down CitectSCADA Runtime Manager

You cannot shut down CitectSCADA Runtime Manager until all CitectSCADA processes have been shut down.

### To shut down the CitectSCADA Runtime Manager

- 1 Start the shutdown of the CitectSCADA processes:
  - Click **Close** on CitectSCADA display component, or
  - Click **Shutdown** on the CitectSCADA Runtime Manager interface.

**Note:** For security reasons the **Shutdown** button in the CitectSCADA Runtime Manager can be disabled. If the `citect.ini` parameter [\[Debug\]Shutdown](#) is set to 0, the button is disabled in CitectSCADA Runtime Manager. This parameter can be set in either the Computer Setup Wizard or Computer Setup Editor.

In both cases the CitectSCADA Runtime Manager stops all CitectSCADA processes that are currently running, not responding, or not starting up. The monitoring pane comes to the foreground and displays the verbose shutdown message for each process.

- 2 If the `citect.ini` parameter [\[RuntimeManager\]ExitAfterShutdown](#) is set to 1 (its default value), CitectSCADA Runtime Manager automatically shuts down when all CitectSCADA processes are exited.
- 3 If the `citect.ini` parameter [\[RuntimeManager\]ExitAfterShutdown](#) is set to 0, CitectSCADA Runtime Manager remains operational, but the **Close** button in the top right of the Window appears. To shut down CitectSCADA Runtime Manager, click **Close**.

**Note:** When the Log Off | Reboot | PowerOff options are executed from Cicode during the shutdown of the CitectSCADA display component, the CitectSCADA Runtime Manager is maximized and displays the Verbose/GAS shutdown messages for each instance. Then, when complete, it automatically shuts down the CitectSCADA Runtime Manager.

## Monitoring Startup

During the startup of each CitectSCADA process, the verbose startup message for each process appears in the Message column of the monitoring pane.

**Note:** If networking is disabled, you can only start up CitectSCADA as a single process.



## Monitoring Runtime Processes

CitectSCADA Runtime Manager monitors the status of each CitectSCADA process running on the workstation. The CitectSCADA Runtime Manager reports the status of each process in the **Status** column of the monitoring pane.

The CitectSCADA Runtime Manager displays the following states.

Status	Definition
Starting	The CitectSCADA process being monitored is currently being started up.
Stopped	The CitectSCADA process being monitored has: <ul style="list-style-type: none"> <li>- Been stopped through shutdown.</li> <li>- Crashed.</li> <li>- Disappeared due to a Task Manager kill.</li> </ul>
Running	The CitectSCADA process being monitored is currently running.
Stopping	The CitectSCADA process being monitored is currently being shut down.

When the status of a process changes to **stopped** abnormally, a notification balloon appears from the task bar icon to inform you a process has stopped.

When the status of a process changes to **not responding**, a notification balloon appears from the task bar icon to inform you a process is not responding.

While a process is running the Message column indicates the project being run and the date and time that project was compiled. A rollover is also available on each process which indicated how long the process has been running for.

## Starting a Process

If a CitectSCADA process is stopped or not responding, use the CitectSCADA Runtime Manager to start that process.

**To start a process which is stopped or is not responding:**

- 1 Right-click the process to start and choose **Start** from the shortcut menu.  
The CitectSCADA Runtime Manager restarts the CitectSCADA process. The verbose startup message appears for that process until the startup completes, and the status for the process is set to running.  
**Note:** A process that has stopped abnormally is restarted automatically.
- 2 Click **Hide** to hide the CitectSCADA Runtime Manager.

## Stopping a Process

If a CitectSCADA process is running, use the CitectSCADA Runtime Manager to stop the process.

### To stop a process which is running:

- 1 Right-click the process to stop and choose **Stop** from the shortcut menu.  
The CitectSCADA Runtime Manager stops the CitectSCADA process. The status for the process is set to Stopped.
- 2 Click **Hide** to hide the CitectSCADA Runtime Manager.

## Restarting a Process

If a CitectSCADA process is running, right-click the process and choose **Restart** from the shortcut menu to restart the process.

## Starting all Processes

If all CitectSCADA processes being monitored are in a stopped state, click **Start All** to start all processes.

## Cancelling a Process

If a CitectSCADA process is starting, right-click the process and choose **Cancel** from the shortcut menu to cancel the process.

By default, a **Cancel** button appears on the startup message box for CitectSCADA Runtime Manager and for each CitectSCADA process. If the `citect.ini` parameter **[Page]StartUpCancel** is set to 0, the button is unavailable in the startup message box. This parameter can be set in either the Computer Setup Wizard or Computer Setup Editor.

## Viewing the Kernel

If a CitectSCADA process is running, the kernel window can be used to diagnose the runtime information for that process.

### To view the kernel for a process

- 1 In CitectSCADA Runtime Manager, right click on the process you wish to view the kernel window for.
- 2 Select **Kernel** from the context menu which appears.
- 3 The Kernel Window for the selected process will appear.

## Troubleshooting

### Incorrect configuration

If CitectSCADA Runtime Manager encounters an invalid configuration within the `citect.ini` file during startup, a warning message is displayed, an error is logged and the system does not start up.

---

## Incorrect License

If CitectSCADA Runtime Manager encounters a licensing problem, an error is logged and the system does not start up.



# Using the Web Client

This section contains information on the Web Client and describes the following:

[CitectSCADA Web Client](#)



# Chapter 39: CitectSCADA Web Client

---

## Introduction

The CitectSCADA Web Client allows you to view a live CitectSCADA project within a Web browser. It provides easy access to CitectSCADA Runtime for LAN-connected users requiring read/write access to current production information.

For example, a senior manager could monitor a facility and access current production information from any computer on the LAN without the need for extensive downloads or software installation.

See Also [System architecture](#)

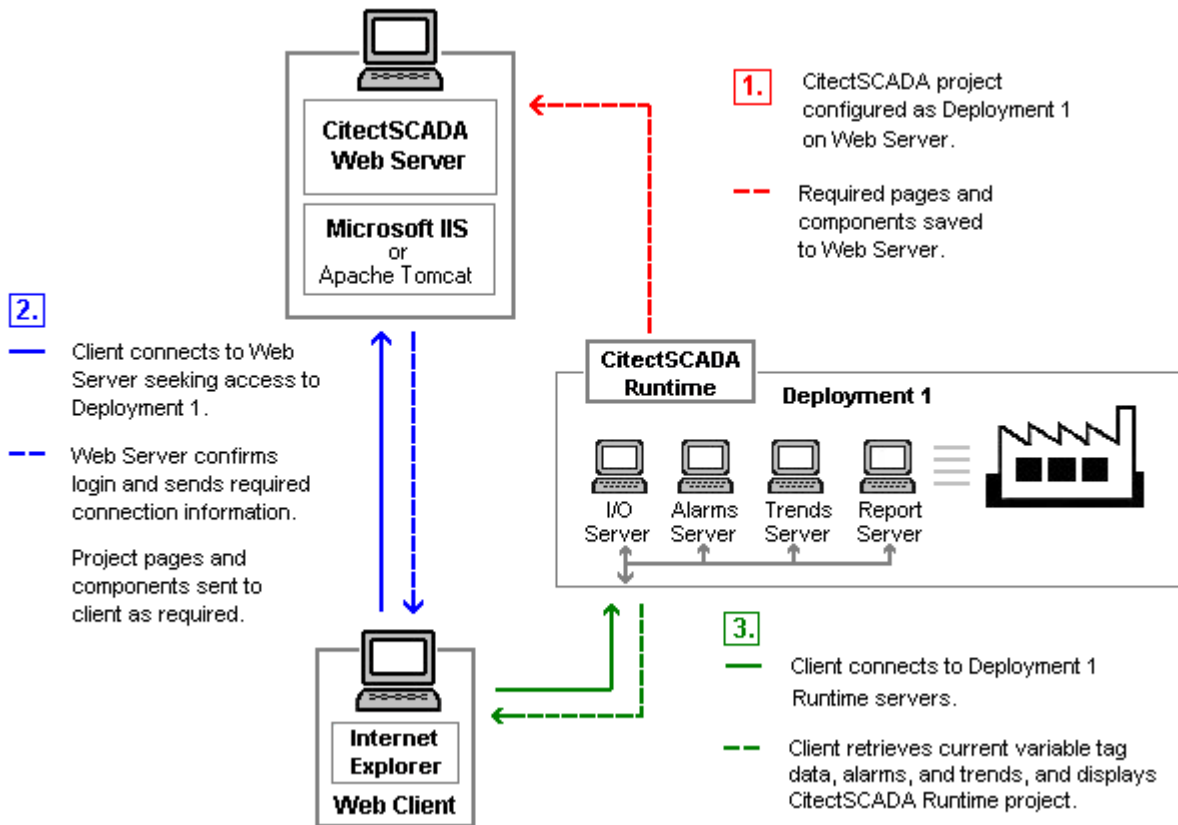
## System architecture

To display a live CitectSCADA project in an Internet browser, you must combine the content of the project pages and the current data these pages present using standard, Web-based communication protocols. To understand the communication architecture for the CitectSCADA Web Client, it's easiest to consider the role each of the following components play in achieving this outcome:

- **CitectSCADA Web Server** - Performs the server-side functionality of the system. As well as facilitating communication, it directs a client to the graphical and functional content of a CitectSCADA project and the location of the runtime servers. This information is stored on the Web Server when a CitectSCADA project is configured as a "deployment". A CitectSCADA Web Server can contain multiple deployments.
- **CitectSCADA Runtime Servers (including the I/O Server, Alarms Server, Trends Server and Report Server)** - Monitor the physical production facility and contain the live variable tag data, alarms and trends that the Web Client will display.
- **Web Client** - provides the platform to merge a deployed project's pages and content with the raw data drawn from the runtime servers. Again, standard Web technologies are required, so the client uses Microsoft Internet Explorer.

The following diagram shows how these components interact.

CitectSCADA Web Client communications architecture.



Once the Web Client has connected to the Runtime servers, steps 2 and 3 become an ongoing process, with the required content being called upon as the user navigates the project pages.

The citect.ini file settings used by a Web Client are taken from the citect.ini file on the Web Server at the time of connection.

This diagram has the system components set up on different computers purely for the sake of explaining the communications model. In reality, the flexibility of the architecture allows these components to be distributed in any required arrangement; they can even share a common location.



## Getting Started

The CitectSCADA Web Client Help is designed to guide you through the steps required to successfully set up a Web Client system.

For detailed information on installing and configuring the web server, refer to the The CitectSCADA Installation and Configuration Guide which is available in a PDF format on the Installation CD, or in the Documentation folder of your CitectSCADA installation.

To ensure a successful installation, you should first familiarize yourself with the [System architecture](#), and then work your way through the following steps, as they will logically guide you through the correct set up procedure.

**Note:** In order to implement a web client solution you must first install Microsoft .NET Framework 2.0 on the web client machine.

### 1 Decide which Web Server platform to use

Both Microsoft IIS and Apache Tomcat are supported as a platform for the CitectSCADA Web Server. Deciding which to use is an important first step towards setting up your system.

- [Choosing which Web Server platform to use - IIS or Tomcat?](#)

### 2 Preparing a CitectSCADA project for deployment

This section explains the adjustments that need to be made to a CitectSCADA project prior to deployment on the Web Server.

- [Preparing a Project for Deployment](#)

### 3 Configuring a deployment

This section describes how to deploy a project on the Web Server, by identifying its source location and associated servers.

- [Configuring a deployment](#)

### 4 Multi-language support

If you are using IIS as your Web Server platform, there are several language options you can implement on the Web Server interface.

- [Implementing Multiple Language Support](#)

If you have followed the procedures outlined above and experience problems, use the [Frequently Asked Questions](#) section to help resolve any issues you might be having.



---

## Choosing which Web Server platform to use - IIS or Tomcat?

Two technologies can be used as the platform for the CitectSCADA Web Server:

- Microsoft's **Internet Information Server** (IIS)
- The Apache Jakarta Project's **Tomcat Web Server**

You should decide which platform you're going to use before setting up your Web Server, as the setup is significantly different for the two.

You should base your decision on whether or not you prefer an open source solution (that is, Apache Tomcat), or the level of familiarity you have with either technology.

However, the following considerations may also determine your choice.

- CitectSCADA Web Server on Tomcat does not support the names of deployment, CitectSCADA project or files (for example, graphics pages) containing non ASCII characters.

**Note:** For instructions on how to install and configure a web server on either platform, refer to the CitectSCADA **Installation and Configuration Guide**.



## Preparing a Project for Deployment

Before deploying a project on a Web Server, you will need to make adjustments in the CitectSCADA configuration environment to ensure it is ready for Web-based delivery. You need to consider the following issues:

- [Functionality limitations of the Web Client platform](#)
- [Preparing a project's user files for delivery](#)
- [Running the Web Deployment Preparation tool](#)

**Note:** To use the Web Client, your CitectSCADA system must be configured to communicate using TCP/IP. This means the network addresses in your project must be defined using standard IP addressing.

### Functionality limitations of the Web Client platform

Due to the architecture required to support Web-based execution of CitectSCADA projects, the Web Client cannot offer the full functionality of a standard CitectSCADA system.

You should consider the following list of unsupported features and Cicode functions to assess if this will be detrimental to the performance of your project. Some adjustments might be required.

#### Feature limitations

The following features are not supported:

- Cicode Debugger
- Remote shutdown
- Fuzzy Logic
- Kernel windows
- Keyboard shortcuts that clash with Internet Explorer's keyboard shortcuts
- Web Client is unable to act as a CitectSCADA Server
- Pages based on the default Menu Page template will only show buttons for pages previously visited
- The **Page Select** button on the default Normal template only lists pages previously visited
- The CSV\_Include project's Update Page List menu item will not work

**Note:** If your project is based on the CSV\_Include template, you must create a customized menu to access pages from the menu bar.

Cicode Function Limitations

Several Cicode functions are unavailable with the Web Client, or limited in their capabilities:

Cluster Functions	Cluster functionality not supported
DebugBreak	Cicode debugger not supported
DelayShutdown	Programmatic Shutdown not supported
FTP Functions	All FTP functions are not supported
Fuzzy Logic Functions	Removed from control due to size
KerCmd	Kernel windows not supported
ProjectRestartGet	Programmatic Shutdown not supported
ProjectRestartSet	"
ProjectSet	"
Shutdown	"
ShutdownForm	"
SwitchConfig	Configuration environment not available
TraceMsg	Kernel windows not supported
UserCreate	Changes to user profiles must be made on the machine where the project is compiled and auto-deployed.
UserCreateForm	"
UserDelete	"
UserEditForm	"
UserPassword	"
UserPasswordForm	"
GetWinTitle	Windows other than the main window only
WinFree	"
WinMode	"
WinMove	"
WinPos	"
WinSize	"
WinTitle	"
WndShow	"
WndViewer	Invokes multimedia applications Feature not supported

See Also [Preparing a project's user files for delivery](#)

Preparing a project's user files for delivery

If the content of your CitectSCADA project incorporates user-created files, such as DBF files, HTML files, or CSV files, you must manually place these into a special zip file called **Misc.zip** for delivery to the Web Server. Similarly, if a project contains ActiveX objects, these also must be included in a zip file called **ActiveX.zip**.

#### To prepare any user-created files for deployment:

- 1 Identify all the user-created files that are associated with the project you want to deploy.  
These files could include CSV or DBF files associated with tables presented on project pages, or HTML content.
- 2 Use a compression tool to zip these files up into a single file called **Misc.zip**.
- 3 Place **Misc.zip** in the main folder for the project. For example, in the case of the Example project, this would be:

C:\Documents and Settings\All Users\Application  
Data\Citect\CitectSCADA\User\Example

**Note:** If your project has included projects that use ActiveX objects, ensure these are also zipped up in an **Activex.zip** file in the included project's directory.

The files are now ready for deployment on the Web Server.

#### To prepare any included ActiveX objects for deployment:

- 1 Identify all the ActiveX objects associated with the project you want to deploy.
- 2 Use a compression tool to zip these files up into a single file called **ActiveX.zip**.
- 3 Place **ActiveX.zip** in the main folder for the project. For example, in the case of the CSV\_Include project, this would be:

C:\Documents and Settings\All Users\Application  
Data\Citect\CitectSCADA\User\CSV\_Include

**Note:** If an ActiveX object has an associated data source, ensure the data source can be located by the computer hosting the Web Client. See the topic [Managing associated data sources](#) under the section on ActiveX objects in the CitectSCADA User Guide Help.

See Also [Running the Web Deployment Preparation tool](#)

### Running the Web Deployment Preparation tool

The final step in preparing a project for deployment involves running it through the Web Deployment Preparation tool. This takes a freshly compiled project and creates the required files and directories for Web-based delivery.

#### To run a project through the Web Deployment Preparation tool:

- 1 Ensure the project you want to deploy has all its associated user files and ActiveX objects zipped up for delivery (see [Preparing a project's user files for delivery](#)).

- 2 Locate the project you want to deploy in Citect Explorer and do a fresh compile.
- 3 Go to the Citect Explorer **Tools** menu and select **Web Deployment Preparation** (or click the following icon on the Explorer toolbar):



- 4 A progress indicator appears. The size of the project significantly affects how long this process takes; a large project with many files can take over ten minutes to process, depending on your hardware. (You can abort the deployment preparation if you want.)
- 5 When complete, a dialog appears stating the preparation was successful. Click **OK**.

The project is now ready for deployment on the Web Server. If you change a project, you must do a fresh compile and run the Web Deployment Preparation tool again.

**Hint:** You can run the Web Deployment Preparation tool automatically when you compile a project. To do this, go to the Citect Project Editor **Tools** menu and select **Options**. Select the **Prepare for Web Deployment** option and click **OK**. Note that this increases the time taken for each compile, particularly for large projects.



## Configuring a deployment

A deployment represents the implementation of a CitectSCADA project on the Web Server. It incorporates the files and components required to display a project, and keeps a record of the location of the servers where CitectSCADA Runtime data is generated.

The deployments configured on a Web Server are listed on the Web Client home page, which is the page that appears when you initially log in. The configuration details for a deployment can be displayed by clicking the small plus (+) icon to the left of the deployment name.

The type of action you can implement for a deployment depends on the permissions granted by your log in. For example, if you log in as a Manager Client, you can only view a deployment. If you are an administrator, you can edit deployments and create new ones.

The following list describes the functionality associated with each of the icons presented on the home page.:



**Add New Deployment** - takes you to the Deployment Configuration page where you can create a new deployment (Administrator Clients only).



**Help** - displays this help page on how to configure and use the Web Client.



**Edit Deployment** - takes you to the Deployment Configuration page and allows you to edit the selected deployment (Administrator Clients only).



**Delete Deployment** - Deletes the selected deployment (Administrator Clients only).



**Start Display Client** - Displays the selected deployment with Display Client permissions (Display Client and Administrator Client only)



**Start Manager Client** - Displays the selected deployment with Manager Client permissions

Additionally, the **System Messages** panel provides notification of events that impact the current status of the Web Server.

**Note:** The citect.ini file settings used by a Web Client are taken from the citect.ini file on the Web Server at the time of connection. This includes which clusters a Web Client has access to. To switch clusters in a project viewed using a Web Client, use the functions [ClusterActivate](#) and [ClusterDeactivate](#).

See Also

[Preparing a Project for Deployment](#)  
[Creating a new deployment](#)  
[Deploying a project from within CitectSCADA](#)

## Creating a new deployment

[Displaying a deployment](#)  
[Editing an existing deployment](#)  
[Updating a deployment to reflect project changes](#)  
[Deleting a deployment](#)

To configure a deployment of a CitectSCADA project on a Web Server, you must log in to the Web Client with Administrator permissions. This will provide you with access to the full functionality of the home page.

### To add a new deployment

- 1 Click the **Add New Deployment** icon.



This will display the Deployment Configuration page.

- 2 Type a name in the **Deployment** text box, and include a **Description** if required. A deployment name cannot contain any of the following characters: \ \* ? | . , / " ' : ; < > # &

**Note:** If you've upgraded your version of the Web Client, you can still view your legacy deployments that you created using version 6.1 or earlier of the Web Client. For details, see [Web Client Upgrade Issues](#).

- 3 Identify the source of the CitectSCADA project's content in the **Project Path** field.

If the project is located locally on the Web Server, you can use a normal path address. The path must point directly to the project within the CitectSCADA User directory. For example, the location of the MyExample project would be:

```
C:\Documents and Settings\All Users\Application
Data\Citect\CitectSCADA\User\MyExample
```

**Note:** If you are remotely administering the Web Server and use a local path address, make sure the path represents the location of the project on the Web Server computer, not the computer you are currently using.

If the project is not located on the Web Server, you need to use a UNC address that identifies the host network computer and the directory it can be found in. For example,

```
\\ComputerName\<path to application data>\User\MyExample
```

**Note:** You must share the directory a project resides in to allow the Web Server access to it. Ideally, you should create a share from the directory (called WebShare, for example) and then use the following project path:

```
\\ComputerName\WebShare
```

Remember that if you are trying to access the project directory from a remote computer, a “local” administrator log in will not provide you with appropriate access on a different computer. You should use a network user profile that will be recognized by other computers on the same domain.

- 4 Determine if any of the I/O, alarm, report or trend servers associated with the project are protected by a firewall. If they are, you need to confirm with the firewall administrator if the CitectSCADA ports have been opened to allow direct access, or if the firewall is using address forwarding.

If forwarding is being used, you will need to identify each server by typing the name in the **Server** field, using the following format:

```
<ClusterName>.<ServerName>
```

If you have alarm properties enabled on an alarm server, you will need to configure an alarm properties connector as a separate server to let the Web Server know which port it is running on. Type the alarm server name in the **Server** field, using the following format:

```
<ClusterName>.<AlarmServerName>_AlarmProps
```

For example:

```
ClusterOne.AlarmServerOne_AlarmProps
```

Type in the **Address** and **Port** for each server, as supplied by the firewall administrator.

**Note:** The Web Client will automatically add any servers that are redirected in this way to the [AddressForwarding] section of the local citect.ini file. See [Using address forwarding](#) for more information.

You can add additional servers to the list by selecting the **Add New Server** icon.

- 5 Use the **Client Control** text box to specify the use of a particular version of the Web Client component when the deployment is displayed.

The menu lists all the different versions of the Web Client control currently installed on the Web Server. Typically, you should choose the version of the control that matches the version of CitectSCADA your project was compiled on.

- 6 Click **Apply Changes**.



This is important, as you’ll lose your changes if you jump straight back to the home page.

All the project files are retrieved from the path indicated, and copied to the Web Server ready for access by the Web Clients.

Once complete, information about the size of the project appears in the File Paths banner above the Project Path field. The number to the left indicates how many files are included in the project; the number to the right indicates the total size of the project.

The deployment is saved. When you return to the Web Client home page, by clicking the home icon, your new deployment is listed.

See Also [Deploying a project from within CitectSCADA](#)  
[Displaying a deployment](#)

## Deploying a project from within CitectSCADA

The Web Client architecture lets you deploy a project from within the CitectSCADA configuration environment, avoiding the need to use the Web Client interface to setup a system.

This process requires you to adjust two parameters in the Citect.ini file:

- [WebServer]WebClientCab
- [WebServer]DeployRoot

These parameters identify the client component used with the project and the location of the deployment root directory. When the project is compiled and prepared for deployment, it is placed directly on the Web Server.

### Notes

- If you've upgraded your Web Client to version 7, you can still view your legacy deployments. For details, see [Web Client Upgrade Issues](#).
- When implementing this option, pay attention to your citect.ini file configuration, as any errors with these parameters are difficult to diagnose. To avoid input errors, use the Web Deployment Tool on the Citect Explorer tool bar with the Web Server's Web Deployment GUI.
- If the project name contains non-English characters, deploying from within CitectSCADA might fail. Under these circumstances, use the Web Server interface to create the deployment.

### To deploy a project from within CitectSCADA:

- 1 Confirm that your CitectSCADA system is configured to use TCP/IP. If you run the Computer Setup Wizard, the **Networking** page will identify which communications protocol is being used.
- 2 Adjust the [WebServer]DeployRoot parameter within the citect.ini file. This parameter represents the directory where the deployment will be located on the WebServer.

If you have set up an IIS-based Web Server, the default location will be the Deploy directory within the installed directories. For example:

```
[webserver]
DeployRoot="c:\ProgramFiles\Citect\CitectSCADA
7\WebServer\deploy"
```

If you are using an Apache Tomcat Web Server, this will be:

```
[webserver]
DeployRoot="C:\Program Files\Apache Software Foundation\Tomcat
5.5\webapps\CitectSCADA\deploy"
```

**Note:** When setting the [WebServer]DeployRoot ini parameter, the path must contain “deploy” as the last subfolder name, otherwise the deployment will fail. Use a mapped drive instead of a UNC address if deploying to a network destination from a Windows 2000 system. Do not map a drive directly to the deployment location, as the path must finish with a “deploy” subfolder.

- 3 Adjust the [WebServer]WebClientCab parameter within the citect.ini file. This parameter represents the directory path and client component to use when a deployment is run, in relation to the installed Client directory. For example:

```
[webserver]
WebClientCab=700/CitectSCADAWebClient_7_0_176.cab
```

Note the use of a forward slash in the defined path.

- 4 Compile your project and then prepare it for deployment. Go to the Citect Explorer **Tools** menu and select **Web Deployment Preparation** or select the following icon on the Explorer toolbar.



Your project should now appear as a deployment within the Web Client home page next time you log in.

**Note:** You can run the Web Deployment Preparation process automatically when you compile a project. To do this, go to the Citect Project Editor **Tools** menu and choose **Options**. Select the **Prepare for Web Deployment** option and click **OK**. Be aware, however, that this might increase the time required for a project to compile.

- 5 Determine if any of the I/O, alarm, report or trend servers associated with the project are protected by a firewall. If they are, you need to confirm with the firewall administrator if the CitectSCADA ports have been opened to allow direct access, or if the firewall is using port forwarding.

If port forwarding is being used, you will need to log in to the Web Client as an Administrator, select the project, and then the **Edit Deployment** button:



This will take you to the deployment configuration page.

- 6 Identify each server that port forwarding is being used for by typing the name in the **Server** field, using the following format:

`<ClusterName>.<ServerName>`

If you have alarm properties enabled on an alarm server, you will need to configure an alarm properties connector as a separate server to let the Web Server know which port it is running on. Type the alarm server name in the **Server** field, using the following format:

`<ClusterName>.<AlarmServerName>_AlarmProps`

For example:

`ClusterOne.AlarmServerOne_AlarmProps`

- 7 Type in the **Address** and **Port** for each server, as supplied by the firewall administrator.

**Note:** The Web Client will automatically add any servers that are redirected in this way to the [AddressForwarding] section of the local citect.ini file. See [Using address forwarding](#) for more information.

You can add additional servers to the list by selecting the **Add New Server** icon.

See Also [Displaying a deployment](#)

## Displaying a deployment

When you display a deployment, it downloads the required Web Client component file from the Web Server, enabling you to run the associated CitectSCADA project in your Web browser.

**Note:** The citect.ini file settings used by a Web Client are taken from the citect.ini file on the Web Server at the time of connection. This includes which clusters a Web Client has access to. To switch clusters in a project viewed using a Web Client, use the functions [ClusterActivate](#) and [ClusterDeactivate](#).

### To display a deployment:

- 1 Locate the deployment you want to display in the list of available deployments.
- 2 Click the relevant icon (**Start Display Client** or **Start Manager Client**) to display the deployment.



The display options available to you depend on your login permissions. If you select the Manager Client icon (the one with the gold lock), you can only read the current values for the CitectSCADA project.

Once the required project files and components have been downloaded, the CitectSCADA project appears. You can now navigate the project pages as required.

**Note:** An error message might appear if the current user on the client machine does not have Windows administrator rights when a new or updated component file (.cab file) is downloaded. Ensure the current Windows user has administrator rights if a new deployment is run or an updated .cab file needs to be downloaded.

See Also [Editing an existing deployment](#)

## Editing an existing deployment

If required, you can edit the settings for a deployment. For example, you can change the name of the deployment or specify a new address for a runtime server.

To edit a deployment's settings, you must be logged in as an Administrator Client.

### To edit an existing deployment

- 1 Select the deployment you want to edit in the list of available deployments.
- 2 Click the **Edit Deployment** icon.



This takes you to the Deployment Configuration page.

Change the fields as required. For field descriptions, see [Creating a new deployment](#).

**Note:** If you give a deployment a new **Name**, it is duplicated instead being updated and overwritten. This allows you to easily copy an existing deployment; however, you must delete the original deployment with the old name if it's no longer required.

- 3 Click **Apply Changes**. (This is important, as you'll lose your changes if you jump straight back to the home page.)



The Web Server retrieves a fresh set of pages and components for the CitectSCADA project, which will include any recent changes.

See Also [Updating a deployment to reflect project changes](#)

## Updating a deployment to reflect project changes

If you change a source CitectSCADA project, you must update its associated deployment to ensure the changes are reflected on the Web Server.

Updating a deployment ensures the latest project pages and components are retrieved by the Web Server and available for distribution. This is important as discrepancies might occur between the project pages and the data being pulled from the runtime servers if the content is not up to date.

### To update a deployment:

- 1 Ensure that the project you want to update has been compiled and processed within the CitectSCADA by the Web Deployment Preparation tool. See [Running the Web Deployment Preparation tool](#).
- 2 Select the deployment you want to update.
- 3 Click the **Edit Deployment** icon.



This takes you to the Deployment Configuration page.

- 4 Click **Apply Changes**.



The Web Server retrieves a fresh set of pages and components for the CitectSCADA project, which will include any recent changes.

See Also [Editing an existing deployment](#)  
[Deleting a deployment](#)

## Deleting a deployment

To delete a deployment from a Web Server, you must log in as an Administrator Client.

### To delete a deployment from the Web Server:

- 1 Select the deployment you want to delete from the list of available deployments.
- 2 Click **Delete Deployment**.



A dialog asks you to confirm that you want to delete the deployment. Click **OK**.

See Also [Configuring a deployment](#)



## Implementing Multiple Language Support

The Web Client deployment configuration interface can be displayed using languages other than English. The following languages are supported by default:

- French
- German
- Spanish
- Chinese
- Japanese
- Korean

You can also implement other languages by translating the resource message file that defines the text displayed. In the case of the languages listed above, this file has already been translated with a version for each language stored in the installed `locales` folder.

See Also

[How default languages are implemented](#)  
[Using a language different to the current system locale setting](#)  
[Implementing a non-default language](#)

### How default languages are implemented

When you connect a client computer to the Web Server, the script on the web page automatically detects the [language code](#) currently defined as the default for the browser. This code is drawn from the system locale setting defined in **Control Panel | Regional Options** on the client machine.

Once the browser's language code has been determined, the script attempts to match it with those available on the Web Server. If a match is made, the associated language is automatically used for the Web Client deployment configuration interface. If a match cannot be made, it defaults to English.

For example, if your Windows **Locale** setting is Chinese (PRC), the language code set for your browser would be "**zh-cn**". This is compared to the current list of language codes on the Web Server, which by default is the following:

Language	Windows Language Code
English	en
French	fr
German	de
Spanish	es
Simplified Chinese	zh
Japanese	ja
Korean	ko

Having failed to match “zh-cn”, the script tries to load Simplified Chinese language, “zh”, as a match. The interface will automatically display in Chinese.

See Also [Using a language different to the current system locale setting](#)

## Using a language different to the current system locale setting

You can display the content of the Web Client’s deployment configuration pages using a language that’s different to the current system locale setting for the computer. To do this, use a URL query string in the address field of your browser.

**To switch to a language other than the default:**

- 1 Decide which language you want to use and determine its associated language code. (See [How default languages are implemented](#) for a list of the codes for the default languages supported by the Web Server).

For example, if you want to use Chinese, the code required would be **zh**.

**Note:** If the language you want to use is not one of the supported languages, you must create and translate your own message file. See [Implementing a non-default language](#).

- 2 Use a URL query to indicate the language you want to use for the Web Client deployment pages. For example, if the address field on your browser currently reads:

```
http://localhost/CitectSCADA
```

add a “/?lang=” query to the end of the address. For example, Chinese would be:

```
http://localhost/CitectSCADA/?lang=zh
```

**Note:** If you use a code that represents a regional variation of one of the default languages and that specific code cannot be matched, the Web Server can only implement the available default version of the language. For example, using the language code for Chinese (PRC), “zh-cn”, results in the Simplified Chinese being used, “zh”.

Your Web browser now displays the Web Client’s deployment configuration pages using the appropriate language.

See Also [Implementing a non-default language](#)

## Implementing a non-default language

If you need to use a language on the Web Client’s deployment configuration interface other than one of the default languages supported by the Web server, you can implement your own translation of the messages file that defines the text that appears.

### To display a language other than those supported by default:

- 1 Using a text editor that supports the language you want to edit, open one of the existing message files located in the Web Server's `locales` directory; the default path is:

`C:\Program Files\Citect\CitectSCADA 7\Web Server\locales`

The file you open should include the language that will be easiest to translate. The language code at the start of each file name can be used to identify the language each file represents; for example, the English language file is called **enmsg.xml**.

- 2 Save the file back to the `locales` directory, using the appropriate language code in the name.

To name the file correctly, check the list of [Windows Language Codes](#) for the appropriate code. This will allow your translated resource file (XXmsg.xml) to be automatically loaded when the Web Client home page is launched, provided it matches the current system locale setting.

For example, to implement Hebrew on the Web Client's configuration pages, you would name your file **hemsg.xml**. To use the Taiwanese variation of Chinese, you would call the file **zh-twmsg.xml**.

- 3 Now change the file content. Firstly, set the correct encoding format.

The encoding format is defined in the top line of the file, which appears as follows:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
```

If the language uses English characters, the format you would use is ANSI, which is defined as "iso-8859-1" (see example above).

If the language uses non-English characters, you would use Unicode, which is defined as "UTF-8" (see example below).

```
<?xml version="1.0" encoding="UTF-8" ?>
```

- 4 Now translate the text that appears on the Web Client interface.

The content that needs to be translated is divided across two sections within the file: "labels" and "messages". The labels section includes the content used to describe and identify the elements of the interface; the messages section includes the notifications that appear in the system messages panel.

To translate these sections, alter the text between the enclosing XML tags. Do not alter the tags themselves.

**Note:** Make sure you maintain any "%" characters, as these are used to insert system information.

For example, the English file:

```

<!-- Labels -->
<span id="TITLE">CitectSCADA Web Client Deployment</span>
<span id="SYSMSG">System Messages</span>
<span id="DEP">Deployment</span>
<span id="DESC">Description</span>
<span id="ACTION">Action</span>
<!-- Messages -->
<sysmsg id="DELOK">% deleted.</sysmsg>
<sysmsg id="DELCAN">% will NOT be deleted.</sysmsg>
<sysmsg id="DEPNUL">You can't % an empty deployment.</sysmsg>
...

```

would appear as follows in Spanish:

```

<!-- Labels -->
<span id="TITLE">Despliegue del Cliente Web CitectSCADA</span>
<span id="SYSMSG">Mensajes del Sistema</span>
<span id="DEP">Despliegue</span>
<span id="DESC">Descripción</span>
<span id="ACTION">Acción</span>
<!-- Messages -->
<sysmsg id="DELOK">% eliminado.</sysmsg>
<sysmsg id="DELCAN">% NO será eliminado.</sysmsg>
<sysmsg id="DEPNUL">No puede % un despliegue vacío.</sysmsg>

```

Once you have translated the file and saved it with the appropriate name to the `locales` folder, your Web Server will be able to support the language.

**Note:** When you save your file, make sure the text editor you used saves the file in the appropriate format, i.e. ANSI or Unicode (UTF-8).

## Web Client Upgrade Issues

If you have upgraded your Web Client from an earlier version to version 7.0, read the following sections about installation and how to use the upgraded Web Client tool with existing deployments.

### Installation

If you intend to use the Web client on Windows 2000 or Windows 2003 Server, you must first install the latest Windows Installer module on your machine. This is available via the Windows Update feature in Windows 2000 or Windows 2003 Server.

Upgrading to the new version of the Web Client adds a new folder named `700` to the `client` folder of the WebServer folder.

If you're planning on installing Web Client version 7.0 but want to view legacy (that is, pre-version 7.0) deployments, before installing the new version, back up your old deployments to a safe location. Then, after installing the new version, copy your old deployment(s) back to the `deploy` folder (see above), and the legacy `.cab` file(s) to the corresponding folder in the `client` folder; this will make your old deployments available for use.

### Creating new deployments

When creating a new deployment, note that the `.cab` file you use (**Client Control**) for the deployment must correspond to the correct version of the project you want to access. For example, to create a deployment based on a version 6.0 CitectSCADA project, from the **Client Control** menu choose **600/filename.cab**, to create a deployment based on a version 7.0 CitectSCADA project, from the **Client Control** menu choose **700/filename.cab**.

### Deploying a project from within CitectSCADA

If you are deploying a project from within CitectSCADA, edit the `[webserver]` section in your `citect.ini` file to specify the correct `cab` file for the version of the Web Client you're using. For example, for a version 7.0 deployment, specify a `.cab` file located in the `700` folder; for a 6.0 version, specify the `600` folder.

## Frequently Asked Questions

This section answers frequently asked questions concerning the Web Client. One section is dedicated to issues pertaining to Windows 2003 Server, and the other to general issues:

- [Windows 2003 Server-related issues](#)
- [General issues](#)

#### Windows 2003 Server-related issues

This section describes issues relating to the Windows 2003 server product.

**Q. My Web Client Deployment Page displays incorrectly on Windows 2003 Server: Show Server Details is missing and the icons for Start Display Client, Delete Deployment, and Edit Deployment are also missing. How do I fix this?**

A: There are two problems that could be occurring here:

- When IIS 6.0 is installed, it defaults to a secure "locked" mode, meaning it can serve up only static content. ASP, ASP.NET, and FrontPage Server Extensions are all disabled and must be explicitly and separately enabled. The CitectSCADA Web Server needs ASP enabled on the IIS.

#### To enable ASP for IIS6 on Windows 2003 Server:

- 1 Choose **Start | Control Panel**, then double-click **Add or Remove Programs**.
  - 2 In the Add or Remove Programs dialog box, click **Add/Remove Windows Components**.
  - 3 In the Windows Components Wizard dialog box, select **Application Server**, and then click **Details**.
  - 4 In the Application Server dialog box, select **Internet Information Services (IIS)** and click **Details**.
  - 5 In the Internet Information Services (IIS) dialog box, select **World Wide Web Service** and click **Details**.
  - 6 In the World Wide Web Service dialog box, make sure **Active Server Page** is selected.
- On Windows 2003 Server, the default setting is to have all the web locations except localhost as an untrusted site. Consequently you must modify your browser's security settings.

#### To update your Trusted Sites settings for Windows 2003 Server:

- 1 Choose **Tools | Internet Options**.
- 2 Click the **Security** tab and then **Trusted Sites | Sites**.
- 3 In **Add this Web site to the zone** field, add the web server's IP address as follows:

`http://<ip address>`

**Note:** If you have other problems on your Web Client pages, you should verify your security settings even if you are not running Windows 2003 Server.

**Q: When I try to start the Web Client, I get the error message "Starting Citect Web Client Failed: Can not initialize citect system", and then the Web Client fails. How do I correct this?**

A: First check that you haven't accidentally deleted the #DisplayClient folder from the installed Web Server directory, as this will cause this error. By default, this directory is located at:

```
C:\Program Files\Citect\CitectSCADA\WebServer\
deploy\#displayclient
```

If this is not the case, this issue is due to a MIME configuration problem: the initialization files are not being recognized in Windows 2003 as registered file extensions. To correct this, add the correct MIME type extension by doing the following:

- 1 Run the IIS manager and go to **Web Sites | Default Web Site | Citect | deploy**.
- 2 Choose **Properties** from the folder's right-click menu.
- 3 Go to **HTTP Headers | MIME types**.
- 4 In the Extension field, enter ".\*".
- 5 In the MIME type field, enter "all".
- 6 Restart your Web server and client.

#### General issues

This section describes general issues relating to the Web Client product.

**Q. When I try to run a deployment in Internet Explorer, I get the following error: "Problems with this page might prevent it from being displayed properly...". What is the problem?**

A. The cause of this problem stems from downloading the client component (the .cab file) associated with the deployment. If the current user on the client computer does not have Windows' local administrator rights when the download takes place, this error message appears.

The solution is to ensure that the person who runs a deployment for the first time is a Windows local administrator on the client machine. Once the components have been downloaded, the problem will not reoccur and any user can access the deployment; unless the .cab file is updated and a newer version must be downloaded.

**Q. I deployed a project from within CitectSCADA using the appropriate citect.ini [WebServer] parameters, but the project does not appear in the list of deployments on the Web Server. A dialog informed me that the deployment was successful. What has happened?**

A. This problem can occur if you make an error with the syntax for the [WebServer]DeployRoot parameter. If, for example, you use a curly bracket instead of a square bracket, (for example, "[WebServer}DeployRoot"), the compiler cannot read the parameter and deployment files are sent to the CitectSCADA project directory instead:

C:\Documents and Settings\All Users\Application Data\Citect\CitectSCADA\User\<Project Name>

The deployment is flagged as successful, but it cannot be located by the Web Server.

You should check the location above for evidence of this problem, as a subfolder called "Web Deploy" will have appeared. If this is the case, you should review the syntax used in your `citect.ini` file.

**Q. I deleted a user from the list of users configured for access to the Web Server, but they can still log in. How do I deny them access?**

A. Sometimes a user can connect to the Web Server even after their user account has been deleted. This is due to the operating system failing to acknowledge, for a period of around half an hour, that a user has been deleted.

The solution is to deny full access to the user before deleting them. That way, they cannot gain access. See the topic "Deleting a user account" in the Web Client section of the CitectSCADA Installation and Configuration Guide.

**Q. When I try to run the Web Client component for the first time, I get a "System Settings Change" message instructing me to restart my computer. What should I do?**

A. This is a known problem affecting computers that contain old versions of some system files required by the Web Client Control. If these files are used by another application during installation, this System Settings Change message appears. Click OK to restart your machine to allow the newer versions of the required files to be installed during system reboot. The problem will disappear.

**Q. One of the ActiveX Object's included in my project cannot locate its associated data source. Where is it?**

A. If an ActiveX object has an associated data source, you need to ensure the data source can be located by the computer hosting the Web Client. See the topic [Managing associated data sources](#) for details.

**Q. Why does a pop-up saying "Client control (CitectSCADAWebClient\_7\_0\_xxx.cab) is not in the option list!" when I try to edit my deployment from the Web Client Deployment Configuration Page?**

There are two possible reasons for this dialog:

- 1 You might have set the [Web Server] parameter WebClientControl incorrectly. The Web page might not recognize the name or version of the .cab file. You should also check that you've used a forward slash in front of the cab file name, as this is required for the Web Server to locate the correct file.
- 2 A user might have deleted the required .cab file from C:\ProgramFiles\Citect\CitectSCADA7.0\WebServer\client\700 (or the specified location). Therefore, the Web page cannot find it.



**Q. The Process Analyst interface normally displays in a foreign language as I translated the language resource DLL, but it displays in English on the Web Client platform. How do I correct this?**

A. A Process Analyst control running inside a CitectSCADA Web Client supports runtime language switching, but you must configure which languages the Web Client will download to the client machine.

**To configure the languages to download:**

- 1 Create a zip file in the `c:\Program Files\Citect\CitectSCADA 7\Web-Server\client\700` folder called **bin.zip**.
- 2 Add to the zip file all the language resource DLL files that you want the client to download and use. (You can find these files in your `\Program Files\Common Files\Citect` folder.)

**Note:** The bin.zip file and its contents are not version-checked. This means you must manually remove the bin.zip from the Web Client machines if your server contains a more recent bin.zip file. To do this:

- 1 Find the installation directory of the Analyst.dll file on your Web Client machines and look for a file called **bin.zip** in this directory.
- 2 Delete this file.
- 3 Reconnect to the Web server to download the latest bin.zip file.

**Q. I have keyboard shortcuts configured in my CitectSCADA project, but they do not work properly when the project is deployed in the Web Client. What's wrong?**

A. Keyboard shortcuts configured for Internet Explorer (IE) take precedence over keyboard shortcuts configured within your CitectSCADA projects. For example, the Example project has F11 assigned to call up Help on a selected animation point on a graphics page. If the project is run as a Web Client deployment, F11 will toggle the view to full screen, as is the case normally with IE.

This is a limitation of using Internet Explorer to host CitectSCADA projects. The easiest solution is to return to the CitectSCADA configuration environment and assign your shortcuts so that no clashes occur. See the Internet Explorer Help for details of preconfigured keyboard shortcuts.

**Q. I can't print from the Web Client. Why not?**

A. You *can* print from the Web Client, but not by using your browser's **File | Print** command. Instead, in your CitectSCADA project, create a Print control that uses the Cicode `WinPrint()` function to print the page you want.

**Q. The new page that I added to my CitectSCADA project does not appear in the Page Select list or the default menu page in the Web Client. How can I correct this?**

A: If the page you added to your CitectSCADA project does not appear in Web Client, you can manually type in the page name in the Page Select list to view this new page. In this version of the Web Client, the new page is not added to the default menu page.

**Q. How does the Web Client deal with ActiveX controls (for example, CiMeterX.ocx) and user files (Recipes.dbf, for example) that are required by a user project?**

A. If your user project requires files such as these, you need to create special zip files to contain them. Create an `ActiveX.zip` file to contain the ActiveX files required by your project, and a `Misc.zip` file to contain other files that your project needs; for example, `recipes.dbf`, `Chinese.dbf`, `Japan.dbf`, and so on. Add these files under the main project path (for example, `C:\Documents and Settings\All Users\Application Data\Citect\CitectSCADA\User\Example`).

**Note:** You can have subfolders within the zip files, but your project must be configured to use the same relative path structure.

During compilation, any zip files that contain supporting files required by a CitectSCADA project are copied to the Webdeploy subfolder. During startup, the Web Client will check the timestamp of any zip files to determine if the zip files have been updated; if the files have been updated, the zip files will be downloaded.

**Q. My project was created using CitectFacilities and incorporates the Citect TimeScheduler. The TimeScheduler is not working when I run the project on the Web Client. What's wrong?**

A. If you want to run the Time Scheduler on a CitectSCADA Web Client, you must ensure that the user profile you log in with has appropriate network access to the configuration tool, and the location of the configuration files. The user must be able to execute the configuration tool, and write to the configuration files.

**Q. The Web Client Deployment Page displays incorrectly on Windows 2000 Advanced Server. 'Show Server Details' is missing, and the icons for Start Display Client, Delete Deployment and Edit Deployment are also missing. What is wrong?**

A. This appears to be caused by Windows Automatic Update installing several components at the same time after a fresh install of the operating system. Even though Internet Explorer might have been upgraded to the latest version (for example, 6.0.2800.1106) it might still behave as a version 5 browser; for example, it offers limited support for "iframes". If you call up **About Internet Explorer** from the Help menu, and a Version 5-style dialog appears with a version 6 release number, then your computer is affected in this way.

A complete uninstall/reinstall of Internet Explorer will correct the problem.

# Using the CSV\_Include Project

This section contains information on the CSV\_Include project and describes the following:

- [Introducing CSV\\_Include](#)
- [Using Pages and Templates](#)
- [Creating a New Project](#)
- [CSV\\_Include Reference](#)



# Chapter 40: Introducing CSV\_Include

---

The CSV\_Include Project is a preconfigured project that is installed with CitectSCADA Version 5.5 or later. The CSV\_Include project includes a set of templates and pages styled for the Windows XP environment and will help you reduce the time required to configure a new project.

When a new CitectSCADA project is created, the CSV\_Include project is automatically incorporated as an included project. This means all the project's templates and associated content are available for implementation when creating your graphics pages in Graphics Builder.

As well as including a standard graphics page template for creating plant mimics, the project also includes predefined trend and alarm display pages, an administration tools page, a file page for displaying text and Rich Text Format files, and a selection of popup windows. All have identical navigation and alarm menus for consistent “look and feel” across an entire project. The project even supports multimonitor display, allowing you to simultaneously display several graphics pages across several computer screens.

**Note:** Do not modify the CSV\_Include project for use as a runtime project. It will not compile successfully, and should be set aside for use as a template for new projects. CitectSCADA upgrades install a new version of the CSV\_Include project, so you will lose changes you make to the project when this happens.

## Where to Find Information

Use the following links for details about the preconfigured content in the CSV\_Include Project:

- Predefined pages and templates. See [Using Pages and Templates](#).
- Common toolbars. See [Common Toolbars](#).

It also describes the processes used to create a project based on CSV\_Include:

- Creating a new project. See [Creating a New Project](#).
- Creating pages. See [Creating Pages](#).
- Creating custom menus. See [Creating Custom Menus](#).
- Creating an alarm group. See [Creating an Alarms Group](#).
- Creating a trend group. [Creating a Trends Group](#).

Reference material is also included, describing the citect.ini parameters and Cicode functions available to customize a project.



# Chapter 41: Using Pages and Templates

---

When you create a new project based on CSV\_Include, the following templates and pages are available for you to use as you create your graphics pages. This section describes the pages and their buttons, menus, and tools.

The CSV\_Include project has the following templates:

Template Name	Description
Normal	Standard graphics page
Alarm	Active alarm page
Disabled	Disabled alarm page
Summary	Summary alarm page
Hardware	Hardware alarm page
Trend	Trend page (8 pens)
Double Trend	Split-page trend page (16 pens)
File	Displays a file
Admin Tools	Engineering tools page
Poptrend	Popup trend
Instanttrend	Instant trend
Popup_Small	Small popup window
Popup_Mid	Medium popup window
Popup_Large	Large popup window
Popup_Xlarge	Extra large popup window

The project also includes the following preconfigured pages:

Page Name	Description
CSV_AdminTools	Engineering tools page
CSV_Alarm	Active alarm page
CSV_AlarmDisabled	Disabled alarm page
CSV_AlarmSummary	Summary alarm page
CSV_AlarmHardware	Hardware alarm page
CSV_Trend	Trend page (8 pens)
CSV_TrendDouble	Split-page trend page (16 pens)
CSV_File	File page

See Also [Creating a New Project](#)  
[Creating Pages](#)

## Normal Page Template

A template designed for creating user-required content and plant mimics. This page contains the standard navigation and alarm toolbars featured on all CSV\_Include templates. If you are creating a project based on CSV\_Include, use this template for your standard graphics pages.

The Normal template creates a page in 1024 x 768 display format without a title bar, the default size for all pages in the CSV\_Include project.

See Also [Creating Pages](#)

## Alarm Page Templates

Templates are included for the following types of alarm displays:

- **Active Alarm Page (Alarm):** Used to create a page displaying all of the audible alarms that are unacknowledged or acknowledged and still in alarm state. The preconfigured page CSV\_Alarm is based on this template.
- **Hardware Alarm Page (Hardware):** Used to create a page displaying details of any system errors that are unacknowledged or acknowledged and still in alarm state, for example, if a communication fault occurs, if Cicode can't execute, if a graphics page is not updating correctly, or if a server fails, this page will alert you to the problem. The preconfigured page CSV\_AlarmHardware is based on this template.
- **Disabled Alarm Page (Disabled):** Used to create a page displaying all of the alarms that are presently disabled in the system. The preconfigured page CSV\_AlarmDisabled is based on this template. The alarms will appear on this page when they have been disabled from the normal alarm system due to a maintenance shutdown, nuisance tripping etc.
- **Alarm Summary Page (Summary):** Used to create a page displaying a historical log of alarms that have occurred. The preconfigured page CSV\_AlarmSummary is based on this template. This page can be used for trouble shooting purposes.

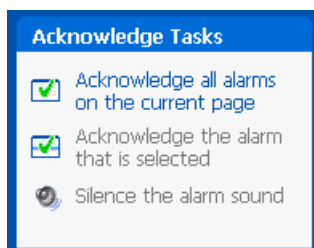
### Common functionality

The alarm page templates all share panels to the left of the alarms list that support the following functionality:

#### Acknowledge Tasks

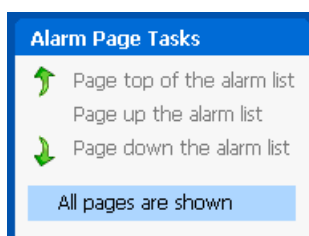
Offers the option to acknowledge all alarms on the current page, to acknowledge just the selected alarm, or to silence the audible alarm. It is featured on the Active Alarms page and the Hardware Alarms page.





### Alarm Page Tasks

Allows the user to navigate through the list of alarms a page at a time. The blue box indicates if more than one page is available to view.



### Alarm List Filter Tasks

Allows the user to filter the current list based on plant area or by alarm category. To apply a particular filter, you firstly must configure an alarm group.



## Trend Page Templates

The CSV\_Include project includes templates for the four following types of trend display:

- **Trend:** An eight-pen trend display. The preconfigured page CSV\_Trend is based on this template.
- **DoubleTrend:** A 2 x 8-pen trend display spread across a divided screen. The preconfigured page CSV\_TrendDouble is based on this template.
- **PopTrend:** A four-pen pop-up display that can be launched from other graphics pages.
- **InstantTrend:** A popup window for instant display of a variable tag.

**Note:** To implement the Instant Trend feature in one of your own projects, you must add the CSV\_InstantTrend project as an Included project. (See [Including a project in the current project.](#))

The Trend, Double Trend and PopTrend displays allow you to select and display trend tags from your CitectSCADA project on a color-coded linear chart. Being based on information coming from the CitectSCADA trend server, these displays are supported by stored historical data that can be recalled if required.

The Instant Trend display is unique as it allows the selection of up to four variable tags for display. This allows you to visually monitor a tag without having to set it up within your CitectSCADA project as a trend tag. You can even load tags directly into the display by hovering the mouse over a tag value on a graphics page and keying in a plus sign (see the parameter [TrendX]KeySeq).

There are limitations to this feature, however, as data is only available from the time the display is launched and is lost when the display is closed. The Instant Trend feature has a duration setting that limits the amount of time the display can remain open, you can adjust the default setting for this via the parameter [TrendX]Duration.

For details on implementing instant trending, see [Setting Up Instant Trending.](#)

## Common functionality

The trend display templates share common controls that support the following functionality:

- [Selected Trend Field](#)
- [Range/Scale markers](#)
- [Span markers](#)
- [Set span button](#)
- [Trend cursor](#)
- [History mode](#)
- [Zoom](#)
- [Autoscale](#)
- [Scale defaults](#)
- [Export to file](#)
- [Paste to clipboard](#)
- [Plot trend](#)
- [Trend group](#)

### Selected Trend Field

Displays information relating to the trend tag(s) currently being mapped in the trend chart, including a name, the current value and the scale range.

Trend	Current Value	Scale Range
➡ CITECT CPU USAGE	42.0 %	0.0 - 100.0%

The field is color coded to match the graphical display, hence the colorful nature of the listed fields.

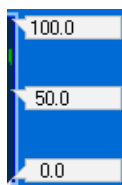
To load a trend into a Selected Trend field, right-click the field. A menu appears allowing you to select a trend or clear the field. Once you have loaded all your selected trends, you can make a particular pen the focus of the graph by clicking this field. An arrow to the left of the field indicates it is currently the focus trend.

Note that this field appears slightly different when using instant trending, as the graph will be displaying current data for a variable tag, not a trend tag. In this case, the selection field appears as below, with the tag name, current value, description and sample period displayed.

Variable Tag	Current Value	Description	Sample Period
➡ CPU_Usage	21 %	Percentage CPU usage	1

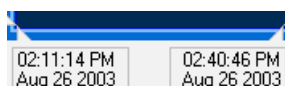
### Range/Scale markers

These markers appear along the vertical axis and show the scale of the display for the currently selected pen, highlighting the lowest to highest values for the scale range.



### Span markers

These markers appear along the horizontal axis, and show the span of the trend display in time. The left-hand marker shows the current start time and date, the right shows the current end time and date. Typically the right-hand marker shows the current time; however, this can be affected by zooming or displaying in Historical mode.



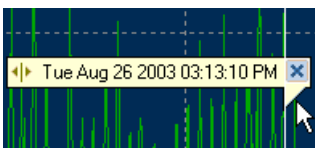
### Set span button

Displays an input dialog that sets the span of the current display. When you enter a unit of time, the display will cover the defined period ending with the current time.



### Trend cursor

Allows you to select a location on the graphical trend display and determine the time and date for that particular point. The arrows to the left end of the display allow you to move the cursor left or right. The Exit icon to the right closes the trend cursor.

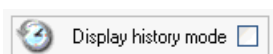


### History mode

Allows you to scroll forward or backward through a graph of a trend tag's history. To switch into History Mode, select the **Display History Mode** checkbox. The display changes to lower version of the control shown to the left.

The four buttons allow you to move backwards or forwards through the trend graph, with the two outer buttons shifting a page at a time, and the two inner buttons shifting half a page. The clock icon allows you to edit the end time for the historical display.

Deselecting the box returns to the normal trend view.



### Zoom

Zooms in on or out of the current display. Zoom in (+) increases the focus of the display and continues zooming in on subsequent clicks. Zoom out (-) returns to the default.



#### Autoscale

Autoscales the current view, which means the scale adjusts to the lowest and highest values reached.



#### Scale defaults

Returns the scale for the display to the default for the currently selected trend.



#### Export to file

Exports the data for the currently displayed trend to a file. A Save As dialog will appear, allowing you to save the data as a D Base III file (.DBF file), a comma separated value file (.CSV file) or a text file (.TXT).



#### Paste to clipboard

Sends the data for the currently displayed trend to the Windows clipboard.



#### Plot trend

Plots the trend to a printer. Use this button to print a trend graph instead of the Print Page button. A dialog allows you to configure the printer setup.



#### Trend group

Allows you to load a trend group. A dialog lists the currently configured groups and allows you to clear currently displayed variable tags. See [Creating a Trends Group](#).



## File Page Templates

Use this template to create a page that can display text (.txt) or rich format text (.rtf) files.

To understand how a file page works, you must distinguish between the file page and the file that is presented on the page; for the file page merely acts as a blank palette to a variety of files.

The file page is displayed whenever the function `CSV_Nav_File` is called. When executed, `CSV_Nav_File` determines the file that is to be displayed on the file page, its location, the title applied to the page, and whether or not the file is editable.

For example, you may configure a menu item that calls up the following:

```
?CSV_Nav_File(MyPageTitle,[Run]:\file.txt,2)
```

This would call up the file page, put the title “MyPageTitle” on the title bar, load the file called `File.txt` from the `Run` directory, and allow the file to be edited (with the last argument set to 2, the file can be saved).

The parameter `[Navigation]FilePage` determines the page that's used as the palette for this process. The preconfigured page `CSV_File` is the default. You can change the setting for `Navigation[FilePage]`, however, you must ensure that any page you specify for this parameter is based on the `CSV_File` template, otherwise `CSV_Nav_File` will not be able to execute properly.

## Admin Tools Page Template

This template is used to create a page that features network and local machine statistics, and allows the user to launch Citect configuration tools and Windows applications. The preconfigured page `CSV_AdminTools` is based on this template.

The panels to the left of this page launch the following configuration tools:

- **Windows applications:** Allows the user to launch the listed Windows Applications from within the runtime environment.
- **Citect Configuration:** Allows the user to launch the listed Citect configuration tools from within the runtime environment. This includes viewing and editing the `Citect.INI` file, running the Computer Setup Wizard, creating alarm groups, trend groups, and launching the Menu Configuration tool.
- **Citect Kernel:** Launches the various components of the Citect Kernel, allowing the system to be debugged from within the runtime environment.
- **System / Hardware:** I/O Device Stats launches a dialog displaying IO device statistics. The Next and Previous buttons allow you to step through the IO devices connected to the system, while the Search button allows you to view statistics for a particular device.

The Tag Debug tool allows you to browse a list of available tags and read the current value for a selected tag. Depending on your access privileges, you may also be able to write a new value to the tag.

The right-hand side of this page displays statistics about the CitectSCADA system:

- **System Information:** Provides details of the CPU usage, memory usage and available disk space for the current runtime machine.
- **Citect Information:** Provides information about the CitectSCADA system including version information and the number of trend, alarm and report clients currently connected
- **I/O Server:** Provides information about the status of the CitectSCADA IO Server your system is connected to. The name of the IO Server machine appears in the title bar of this panel. The information displayed includes:
  - Maximum - the overall maximum response time (ms)
  - **Average:** Overall average response time (ms).
  - **Minimum:** Overall minimum response time (ms).
  - **Read Request:** Total read requests per second.
  - **Read Physical:** Total physical read requests per second.
  - **Write Request:** Total write requests per second.
  - **Write Physical:** Total physical write requests per second.
  - **Reset button:** clears the current values and recalculates them.
- **Timekeeping:** The **Set Date** and **Set Time** fields function as buttons that call up an input dialog for entering a new date and time for the local computer. The **Set Master Time** field sends the local time to the CitectSCADA time server, which in turn sends the date and time to all CitectSCADA machines on the current network.

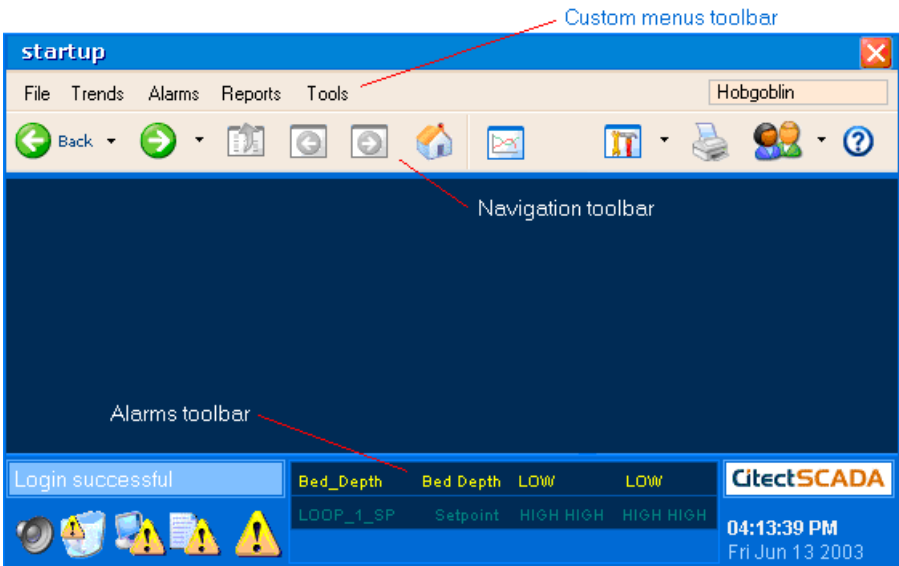
## Common Toolbars

Pages in the CSV\_Include project include common toolbars for easy navigation and feature access, as well as a consistent appearance.

The following three toolbars remain on screen during operation:

- **Navigation toolbar:** Provides navigation buttons and direct access to key pages such as the Trends page and Admin Tools page.
- **Alarm toolbar:** Provides access to Alarms pages and displays the last three active alarms.

- **Custom Menu toolbar:** Provides menus capable of navigating to a specific page or calling a Cicode function. The content of the menu is generated at runtime using a lookup table.












See Also [Navigation Toolbar](#)  
[Alarms Toolbar](#)  
[Creating Custom Menus](#)

### Navigation Toolbar

The Navigation toolbar includes buttons that allow the user to move between a project's pages. If the current user has insufficient privilege or there is no option configured for a particular button, it is unavailable to the user.

Icon	Name	Description
	Back	Goes back to the page displayed before the current page. The arrow to the right of the button allows you to select from a menu of recently visited pages. You can set the maximum number of pages included in this menu by adjusting the parameter [Navigation]LastPageStackSize.
	Forward	Goes forward to the page that was displayed prior to the back button being pressed. The arrow to the right of the button allows you to select from menu of pages recently navigated back from. You can set the maximum number of pages included in this menu by adjusting the parameter [Navigation]LastPageStackSize.

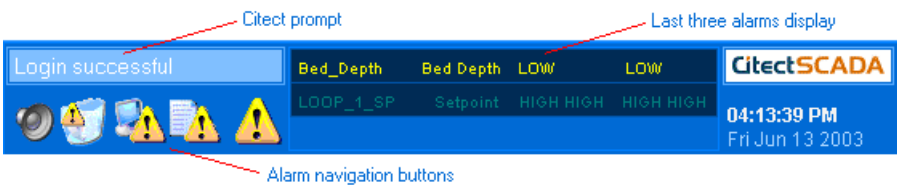


Icon	Name	Description
	Parent Page	Changes the display to the "parent page" of the current page. You can assign a parent page to a graphics page by setting an environment variable in Graphics Builder. To do this, open the page you want to assign a parent to. Go to the properties dialog for the page ( <b>File   Properties</b> ), and click the <b>Environment</b> tab. Add a new variable called "ParentPage" with a value of the page name of the parent page.
	Previous/Next	Moves backwards or forwards through pages in a browse sequence if a sequence is configured. To configure a browse sequence, go to the Page Properties dialog ( <b>File   Properties</b> ) for each page in the sequence and set the <b>Next</b> and <b>Previous</b> fields on the <b>General</b> tab accordingly.
	Home	Displays the "home" page. By default, this page is the startup page CSV_Start. To change the home page to a different page, use the [Navigation]HomePage parameter.
	Trends Page	Displays the "trend" page. By default, the preconfigured page CSV_Trend appears when this button is pressed. To display a different page, adjust the parameter [Navigation]TrendPage.
	Network Page	Displays a page called "Network" if one exists. By default, this page does not exist, which means the button will not appear. To use this button to call a function, or to launch a page with a name other than network, adjust the parameter [Navigation]NetworkPage.
	Tools	Displays the Admin Tools page, named CSV_AdminTools. You can adjust the parameter [Navigation]ToolsPage to call a function with this button or change the page displayed; however, this is not recommended. This button also features a menu to launch the Tag Debug tool and Instant Trend display. It also allows you to switch tool tips on and off.
	Print Page	Prints the current page. The parameter [Printer]Port needs to be configured correctly for printer selection.
	Login	Displays the standard CitectSCADA login prompt. The menu to the right offers extra options such as Logout, Change Password, Edit User (restricted by login) and Create User (restricted by login).
	Help	This button can be configured to display user assistance information for your project. By default, it will point to the topic <a href="#">Adding user assistance to a page</a> within the CitectSCADA






See Also [Alarms Toolbar](#)  
[Creating Custom Menus](#)

## Alarms Toolbar

The alarms toolbar provides access to current alarm information and navigation buttons for single-click access to alarm pages. It also includes standard CitectSCADA prompt and current date and time.



The alarms toolbar contains the following features:

Icon	Name	Description
n/a	Last Alarms	The center panel of the toolbar displays the last three alarms triggered within the system, providing the operator with an immediate visual cue to alarm occurrences. You can modify the format of this list and control which alarms appear. For example, you can set the list to only display alarms of a particular type, category or priority. For details, see the parameters <code>LastAlarmFmt</code> , <code>LastAlarmCategories</code> , <code>LastAlarmPriorities</code> , and <code>LastAlarmType</code> .
	Active Alarms	Changes the display to the active alarms page, <code>CSV_Alarm</code> . This button is animated and will blink when there is an unacknowledged alarm in the system. This action performed by this button is determined by the parameter <code>[Navigation]AlarmPage</code> .
	Alarms Summary	Changes the display to the alarm summary page, <code>CSV_AlarmSummary</code> , which provides a historical log of alarm occurrences. This action performed by this button is determined by the parameter <code>[Navigation]SummaryPage</code> .
	Hardware Alarms	Changes the display to the hardware alarms page, <code>CSV_AlarmHardware</code> . This is an animated button which will blink when an unacknowledged hardware alarm occurs. This action performed by this button is determined by the parameter <code>[Navigation]HardwarePage</code> .
	Disabled Alarms	Changes the display to the disabled alarms page, <code>CSV_AlarmDisabled</code> . This action performed by this button is determined by the parameter <code>[Navigation]DisabledPage</code> .
	Alarm Silence	Silences the audible alarm buzzer that sounds when an alarm occurs. See <a href="#">Implementing Audible Alarms</a> .

See Also [Creating Custom Menus](#)

# Chapter 42: Creating a New Project

Creating a project based on the CSV\_Include project is simple; by default, it is incorporated as an included project in all new projects. This means whenever you start a new project, the CSV\_Include pages and templates are ready to use as required.

To make it easier to configure a project, and to ensure that the CSV\_Include project works correctly, follow these steps:

- 1 Create a privileged user; see [Creating a Privileged User](#).
- 2 Run the Computer Setup Wizard; see [Running the Computer Setup Wizard](#).
- 3 Set up instant trending; see [Setting Up Instant Trending](#).
- 4 Set up multiple monitor display; see [Displaying a Project on Multiple Monitors](#).
- 5 Implement audible alarms; see [Implementing Audible Alarms](#).

Don't modify the CSV\_Include project for use as a runtime project; set it aside for use as a template for new projects.

If creating a project based on the CSV\_Include templates, don't include pages based on templates that use a different style, including the earlier Include project. Doing so might affect functionality.

See Also [Creating Pages](#)

## Creating a Privileged User

Some CSV\_Include project content is protected via a user login. Without a valid login, some project functionality will be disabled. For example, the Tools page is mostly inactive if you log in as a user with restricted privileges.

By default, the following elements within the CSV\_Include project are protected by global privileges.

Element	Global Privilege	Associated Function
Admin Tools page	8	[Privilege]EngTools
Editing users	8	[Privilege>EditUser
Project shutdown	0	[Privilege]Shutdown
Acknowledge alarms	1	[Privilege]AckAlarms
Disable alarms	8	[Privilege]DisableAlarms

When configuring a CSV\_Include project, make sure your users have appropriate access to the available functionality. Ensure that your users can acknowledge alarms if required, and that they have access to the full functionality of the Admin Tools page.

To adjust the global privileges for the elements previously listed for a more complex security architecture, adjust the [Privilege] parameters in the `citect.ini` file. Click the relevant link in the previous table above for details.

See Also [Running the Computer Setup Wizard](#)

## Running the Computer Setup Wizard

As with all CitectSCADA projects, you must complete the Computer Setup Wizard on any machine where a CSV\_Include project will be run. See [Running the Computer Setup Wizard](#) for more information.

How you set up a computer using the Wizard is mostly up to you; however, due to some required settings, you must run the Computer Setup Wizard as a custom setup. The Wizard pages you must pay attention to are:

### Events Setup page

The CSV\_Include project includes three preconfigured events.

- CSV\_TrendXClient: enables the Instant Trend feature.
- CSV\_TrendXServer: enables the Instant Trend feature.
- CSV\_AlarmClient: enables audible alarm siren.

Ensure that you select and enable the required events listed above. To support the instant trend feature, you must enable both the CSV\_TrendXClient and CSV\_TrendXServer events on the computer acting as your trend server. Only CSV\_TrendXClient needs to be enabled on your runtime machines. To support audible alarms, enable CSV\_AlarmClient.

You should select all of the events unless you need to disable the associated functionality.

### Security Setup - Control Menu page

Provides a Display Title Bar option. As the CSV\_Include pages feature an XP-styled title bar, deselecting this option allows your project to run successfully in full-screen mode (run CSV\_Include-based projects in full-screen mode).

See Also [Creating a Privileged User](#)  
[Setting Up Instant Trending](#)  
[Implementing Audible Alarms](#)

## Setting Up Instant Trending

Instant trending allows you to monitor a variable tag visually from your CitectSCADA project, without configuring the tag it as a trend tag. Without the

support of the CitectSCADA trend server to facilitate this functionality, instant trending requires you to perform the following setup:

- 1 **Include the CSV\_InstantTrend project.** To implement the Instant Trend feature in one of your own projects, you must add the CSV\_InstantTrend project as an Included project. (See [Including a project in the current project.](#))
- 2 **Enable the CSV\_TrendX events.** As described in [Running the Computer Setup Wizard](#), you must enable the events CSV\_TrendXClient and CSV\_TrendXServer for instant trending to work. Ensure that both events are enabled on the trend server computer, and that CSV\_TrendXClient is enabled on any runtime machines that will use instant trending.
- 3 **Call the Instant Trend display.** Ensure that any buttons or menu items you configure to launch the Instant Trend display call the function CSV\_TrendX\_Display(). You should call this function with no arguments set, as this will implement the default settings. This ensures the correct page is called with appropriate display settings.
- 4 **Set the default display duration.** The Instant Trend window only stays open for a set period of time. Use the [\[TrendX\]Duration](#) parameter to adjust the default period of time as appropriate.
- 5 **Configure the Tag Selection dialog.** You can decide whether or not to load a list of current tags into the Tag Selection dialog when using instant trending. Having no tags appear in the selection dialog might be useful if your project has many tags. See the parameter [\[TrendX\]TagListEnable](#).
- 6 **Check the key sequence used to load tags.** You can load tags directly into the Instant Trend display by putting the mouse cursor on a tag value on a graphics page and pressing the plus (+) key. However, make sure that this key sequence does not clash with other application functionality. To change the key sequence for this feature, see the parameter [\[TrendX\]KeySeq](#).

Instant trending does not support variable tags of the type LONG or REAL.

See Also [Trend Page Templates](#)

## Displaying a Project on Multiple Monitors

The CSV\_Include project can display several pages simultaneously across multiple computer screens. If your hardware can run multiple monitors off a single computer, the CSV\_Include project can display a different page on up to six screens at a time.

To successfully run your project across multiple monitors, adjust the following parameters:

- **[MultiMonitors]Monitors.** Adjust this parameter to indicate how many monitors your project will be running on.

- **[MultiMonitors]StartupPage<sub>n</sub>**. This parameter determines which page will appear on each monitor at startup. Note that you have to duplicate this parameter for each monitor. For example, `StartupPage1` determines the page that appears on the first monitor at startup, `StartupPage2` determines what appears on the second monitor, and so on.

Many types of hardware support multiple-monitor display. The CSV\_Include project has been tested on a PC with multiple outputs from a single video card, but not on other architectures. The ability to display project pages on multiple screens might be affected by hardware architectures that have not been tested.

## Implementing Audible Alarms

The CSV\_Include project offers support for audible alarms. You can configure a project so that a selected wav file is sounded whenever an alarm of a particular priority is triggered. You can even assign different sounds to different alarm priorities, allowing the urgency of an alarm to be determined from its sound.

**To implement audible alarms in your project:**

- 1 Ensure the alarms you want to trigger an audible alert are assigned to a category and given a particular priority.
- 2 Adjust the parameter `[Alarm]Soundn`. This parameter determines the wav file used when an alarm sounds, based on the priority of the alarm (*n*). For example, `[Alarm]Sound1` identifies the .wav file that will be sounded whenever a priority 1 alarm is triggered.
- 3 If required, adjust the parameter `[Alarm]SoundnInterval`. This parameter determines how long the selected wav file sounds, based on the priority of the alarm (*n*). For example, `[Alarm]Sound1Interval` sets the length of time a priority 1 alarm is sounded for.
- 4 Enable the CSV\_AlarmClient event on any machine you want an audible alert sounded from. You can enable this event from the Events Setup page of the Computer Setup Wizard. See [Running the Computer Setup Wizard](#).
- 5 Ensure your users have appropriate privileges associated with their login to acknowledge alarms, otherwise they will not be able to silence an alarm. See [Creating a Privileged User](#) and the parameter `[Privilege]AckAlarms`.

See Also [Alarm Page Templates](#)

## Creating Pages

When considering the pages required for your project, first determine if you can use any of the predefined pages in the CSV\_Include project:

Page name	Description
CSV_Trend	Default 8-pen Trend page, called from the Trend button on the Navigation toolbar.

Page name	Description
CSV_Alarm	Active Alarms page, called from the Alarms toolbar.
CSV_AlarmHardware	Hardware Alarms page, called from the Alarms toolbar.
CSV_AlarmDisabled	Disabled Alarms page, called from the Alarms toolbar.
CSV_AlarmSummary	Alarms Summary page, called from the Alarms toolbar.
CSV_AdminTools	Admin Tools page, called from the Tools button on the Navigation toolbar.

There are also several pages that are accommodated by the Navigation toolbar that you'll need to create if required. Decide if the following pages would be useful in your project and create them using the appropriate name:

Page name	Description
Home	Called from the Home button on the Navigation toolbar.
Network	Called from the Network button on the Navigation toolbar. Note that this button does not display if a page called "Network" does not exist.

The links between the pages listed and the buttons that call them are defined by the [Navigation] parameter settings within the `citect.ini` file. If you want a button to call a page with a different name, adjust these parameter settings. For details see [CSV\\_Include Parameters](#).

## Creating new pages

Normal and Popup templates are designed with minimal content to enable the presentation of customer-required information and plant mimics.

Pages are created based on these templates within Graphics Builder by choosing the required template from the Use Template dialog (**File | New Page**). You can access the CSV\_Include templates by selecting `csv_xp_01` from the **Style** field.

**Note:** If you try to launch a CSV template from within Citect Explorer, you must drill down to `[Project Name]/Graphics/Templates/csv_xp_01/XGA` to locate it.

The CSV\_Include project can add rollover buttons to pages without drawing and configuring multiple elements. You can add text to a page that has a button appear behind it whenever the cursor passes by adding the function `DspButtonRollOver()` to a button's visibility property. No arguments are required.

See Also [Adding user assistance to a page](#)

## Adding user assistance to a page

The CSV\_Include project includes a Help button on the Navigation toolbar. As you create your pages, you can configure this button to provide information to the user about the content of a page, and its supported functionality.

The information provided to your operators must be created as individual HTML pages and compiled into a HTML help file (.chm). The compiler required to create a .chm file is available for download from the Microsoft Developer Network (MSDN). Go to <http://msdn.microsoft.com> and search for "HTML Help

Workshop". An SDK is available that includes the required downloads and information about creating an HTML help project.

Once you have created a .CHM file, it is recommended you place it in the project directory.

#### To configure the help button for a project page

- 1 Select the project you've created in Project Explorer.
- 2 Go to Project Editor and select **Parameters** from the **System** menu. The Parameters dialog will appear.
- 3 In the **Section Name** field, type in "HelpButton".
- 4 In the **Name** field, identify the page you would like to add help to using the following format:

SetTopic:<PageName>

where <PageName> is the name of the page included in your CitectSCADA project. (Do not include a space after the colon.)

- 5 In the **Value** field, identify the associated HTML Help file and topic using the following format:

<HelpFileName> | <HTMLPageName>

where:

- <HelpFileName> is the name of the compiled HTML Help file you've created (e.g. MyHelpFile.chm)
- <HTMLPageName> is the name of the compiled HTML page you'd like displayed as a help topic (e.g. Help\_Topic.html).

- 6 If required, add a **Comment**.
- 7 Click **Add**.

The help button will now launch the identified HTML file with the selected topic showing.

#### To disable the help button for a project page

- 1 Select the project you've created in Project Explorer.
- 2 Go to Project Editor and select **Parameters** from the **System** menu. The Parameters dialog will appear.
- 3 In the **Section Name** field, type in "HelpButton".
- 4 In the **Name** field, identify the page you would like to disable the help button for using the following format:

SetTopic:<PageName>

where <PageName> is the name of the page included in your CitectSCADA project. (Do not include a space after the colon.)



5 In the **Value** field, type in DISABLED.

6 If required, add a **Comment**.

7 Click **Add**.

The help button on the identified page will now be unavailable during runtime.

**Note:** Make sure you do not duplicate this parameter for a single project page.

See Also [Using Pages and Templates](#)

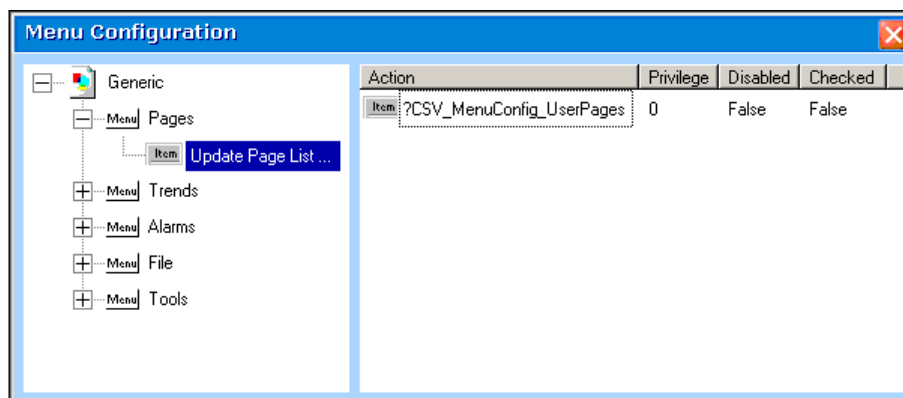
## Creating Custom Menus

The Custom Menu Toolbar, located directly beneath the page title bar, allows you to create drop-down menus capable of calling a Cicode function or navigating to a specific page (i.e. to a specific plant mimic).

You can disable the menu bar on all pages by adjusting the parameter [Page]MenuDisable. If you need to disable the menu on a particular page, you can apply this parameter as an environmental variable. To do this, open the required page in Graphics Builder, go to the Properties dialog (**File | Properties**), and insert the parameter on the Environment tab.

### Menu Configuration tool

The content of the menus can be configured via the Menu Configuration tool, which is launched from the Citect Configuration panel of the Admin Tools page.



The Menu Configuration tool has two panels. The panel on the left represents the menus configured for the current project in a directory structure. The panel on the right includes information about the item currently selected in the left panel. The picture above includes the default settings generated for any new projects based on the CSV\_Include templates.

The directory in the left panel is a graphical representation of a DBF lookup table that forms the basis of the menus displayed at runtime. The hierarchical structure is determined by the following fields within this table:

<b>Page</b>	The page field is defined as either "Generic" (as pictured above), or the name of a page within the project. Generic specifies that the menu is associated with all pages, a specific page name indicates the menus that will appear just on that particular page.
<b>Menuname</b>	Name(s) of the menus included on the specified page.
<b>Menuitem</b>	Item(s) that appear within each menu.
<b>Submenu</b>	Submenus that appear in a menu (optional). Adding a sub menu automatically removes the action defined for the menu item it is branched from, as the parent becomes a placeholder for the list of sub menus.

In the example, the Page is defined as "generic", indicating that the menus configured appear on all pages within the project.

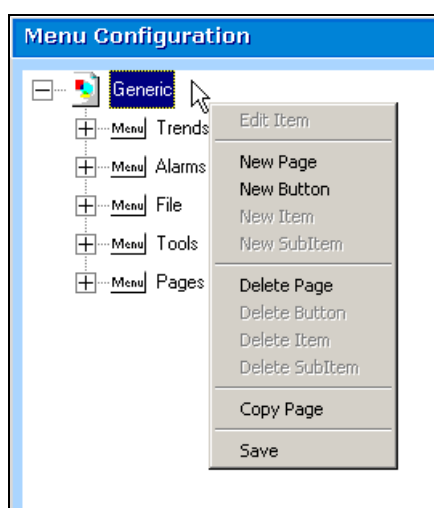
The Menu Names that appear are the five defaults: Pages, Trends, Alarms, File and Tools. The Pages menu is expanded, showing the Menu Item Update Page List. As Update Page List is the currently selected item, the fields associated with it are pictured in the right.

The fields you can expect to see associated with an item, as defined in the DBF table, are as follows:

<b>Action</b>	Either the name of the page to display or a Cicode function. If specifying a Cicode function, it must be prefixed by a question mark ("?"). To invoke a Cicode function that has arguments from the Menu System, use the following format: function name, space, comma-separated list of string arguments. For example, ?LoadPav dP45.pav.
<b>Privilege</b>	The login privilege required to trigger the associated action.
<b>Disabled</b>	Indicates if the item is currently disabled (embossed). For example, if the current login does not meet the required privilege set for the item (see above), this field will be set to True and the menu item will appear embossed to indicate its currently not active.
<b>Checked</b>	Indicates if the menu item is currently checked (with a tick). True indicates
<b>Btnwidth</b>	Specifies the width of the button (optional).

## Building custom menus

Creating a custom menu entails adding the required pages, menus, and items to the Configure Menus directory structure, and then assigning the required action and privileges to each item. This is all achieved via a right-click menu.



Right-click a branch in the directory to display the context menu, which shows the available actions for the current selection. In the example, clicking the Generic page allows you to add a new page, add a new button to the generic page, delete the page, copy it, or save the configuration.

The right-click menu includes the following actions:

<b>Edit Item</b>	Available when an Item is selected. This option displays the Edit Item dialog, which allows you to define the fields associated with the current Item. In particular, it allows you to specify the action associated with the item. (See Editing a menu item below)
<b>New Page</b>	Adds a new page to the menu configuration. Use this option to create new custom menus designed specifically for a particular page in your project. Note that the name you give the new page in the menu configuration must be identical to the name of the page in your project that you want the menus to be applied to.
<b>New Button</b>	Adds a new menu button to the current page.
<b>New Item</b>	Adds a new item to the currently selected menu.
<b>New Sub Item</b>	Adds a sub item to the currently selected item. A subitem overwrites the action configured for its parent item. The parent item becomes a label identifying the list of Sub Items assigned to it.
<b>Delete Page</b>	Deletes the currently selected Page from the menu configuration.
<b>Delete Button</b>	Deletes the currently selected Menu Button from the menu configuration.
<b>Delete Item</b>	Deletes the currently selected Item from the menu configuration.
<b>Delete Sub Item</b>	Deletes the currently selected Sub Item from the menu configuration.

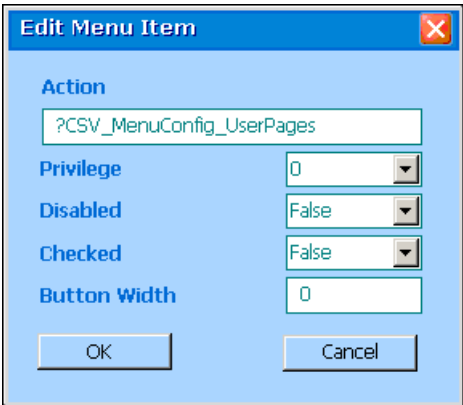
- Copy page

Copies the currently selected page and pastes it to the end of the menu configuration. This option is useful if you want a page to include the generic page menu configuration, but with additional menus that only appear when the particular page is displayed. To do this, copy the generic page, rename it to match the page you want to customize the menus for, then add the required menus and items.
- Save

Saves the current menu configuration.

Editing an item

You configure a menu item's functionality by using the Edit Item Menu dialog.



Use this dialog to identify the action an item will trigger, be it navigation to a specific page, or execution a function. To launch this dialog, right-click the item to configure and choose **Edit Item** from the menu.

You should fill out the fields as follows:

- Action

Indicates the name of the page to display, or a Cicode function. If specifying a Cicode function, it must be prefixed by a question mark ("?").
- Privilege

The login privilege required to trigger the action. Only users with appropriate access privileges will be able to use this item.
- Disabled

As this setting is determined automatically depending on the current user's access privileges, leave this set to **False** unless you want the item disabled by default.
- Checked

Leave this set to **False** unless you want the item checked by default.
- Button Width

Leave this set to zero (0) as this sets the menu button to the appropriate width. To set a specific width for the button, input the width in pixels.

See Also [Common Toolbars](#)

Creating an Alarms Group

The CSV\_Include project allows you to use "alarm groups" to display a specific set of tags defined by the alarm category and area settings configured within the runtime CitectSCADA project.

For example, you could create a group defined by all category one alarms. This group could then be used as a filter to create a list of all the category one alarms currently displayed on the Active Alarms page.

Alarm groups are configured by clicking the appropriate link on the Citect Configuration panel to the left of the Admin Tools page. The functionality can also be added to a custom menu for easier access.

**To configure an alarm group:**

- 1 Go to the Admin Tools page of a runtime project by clicking the **Tools** button.
- 2 Select **Configure an Alarm Group** from the Citect Configuration panel. The Configure Alarm Groups dialog appears.

- 3 In the **Alarm Group Description** box, type the name you want to use to identify the group.
- 4 In the **Categories** box, list the alarm categories configured in the CitectSCADA runtime project that you want to use to define the group.
- 5 In the **Area** box, type the areas defined in the CitectSCADA project you want to use to define the group.
- 6 Click **Add**.

The information to the right of the dialog indicates how many alarm groups are configured and where you are currently positioned in this list. You can scroll through the list of configured groups by using the up and down arrows.

To change a group, locate it in the list, change it as appropriate, then click **Replace**.

See Also [Creating a Trends Group](#)

## Creating a Trends Group

The CSV\_Include project allows you to use “trend groups” to display a specific set of trend tags. A trend group includes a set of up to eight variable tags that can be automatically loaded into a trend display without having to select each tag individually.

Trend groups are configured by clicking the appropriate link on the Citect Configuration panel to the left of the Admin Tools page. The functionality can also be added to a custom menu for easier access.

To configure a trend group:

- 1 Click **Tools**.
- 2 Select **Configure a Trend Group** from the Citect Configuration panel. The Configure Trend Groups dialog appears.

- 3 In the **Description** text box, type the name you want to use to identify the group.
- 4 In the **Trend Pen** text boxes, select the eight variable tags you want the group to trend. Click the button on the right of each field to view the available tags within the CitectSCADA runtime project.
- 5 In the **Area** text box, type the areas defined in the CitectSCADA project you want to use to define the group.
- 6 Click **Add**.

The information to the right of the dialog indicates how many trend groups are configured and where you are currently positioned in this list. You can scroll through the list of configured groups by using the up and down arrows.

To change a group, locate it in the list of groups or browse to it using the selection button to the right of the **Description** text box, make the required changes, then click **Replace**.

See Also [Creating an Alarms Group](#)

## Using Environment Variables

Whereas parameters apply specified rules to all template pages, environment variables apply rules to *specific* pages. In this way, you can use environment variables to specify functionality for specific pages.

For example, to display a specific page, you'd specify the page name as the environment variable. To call a specific Cicode function, you'd specify the function name (with a "?" prefix), space, then a list of comma separated arguments, like this: ?WinPrint LPT1,0,0,Trend.pal. (The "?" indicates that the variable is to be interpreted as a function call rather than a page to display.)

Let's look at a more detailed example: usually the standard **Print** button is used for printing a graphics page in WYSIWYG format. If you have a text/html file displayed on the page, or perhaps a trend, you can create your own print function (for example, PrintTrend) and specify it in the environment variable, as ?PrintTrend. To reduce the amount of ink used, specify the WinPrint function with a user-defined palette. In this case you would specify it as a parameter rather than as an environment variable so that it applies to all pages.

**Note:** With the standard CSV\_Include trend pages, you can print the screen using the standard **Print** button, or plot the trend using the **Trend Plot** button. You should give the user both alternatives as they both serve different purposes.





# Frequently Asked Questions

---

This section contains answers to some commonly asked questions about CitectSCADA functionality. The FAQs have been divided into several categories:

- [Pages](#)
- [Graphics](#)
- [Runtime](#)
- [Trends](#)
- [Controls](#)
- [Alarms](#)
- [Miscellaneous](#)

## Pages

### **Q: How do I open a page on startup?**

**A:** CitectSCADA searches for a page called "Startup" when it starts up. If CitectSCADA locates this page, it is opened automatically. You can change the name of the default startup page with the Computer Setup Wizard, run in Custom mode. See [General Options Setup](#) for details.

### **Q: How do I get the Page Next and Page Previous buttons to work on new graphics pages?**

**A:** Use the page Properties (File menu in the Graphics Builder) to define the next page and previous page.

### **Q: How do I display pages starting with '!'?**

**A:** Turn on the List system pages option, under **Options** in the **Tools** menu of the Graphics Builder.

### **Q: What does the '!' mean when it is the first character in a page name?**

**A:** The '!' means the page is a System page. Pages that begin with '!' will not appear on the default menu page or in the Page Select combo box.

### **Q: Is it possible to associate more than 8 variable tags to a Super Genie. I am using the `AssPopup ( )` function but it only allows me to use 8?**

**A:** You can use the [AssVarTags](#) function to associate up to 256 variable tags, or use arrays.

**Q: I have just created a Genie and want to add a privilege to it. In versions 3 and 4, I was able to hold down CTRL and double-click, but it doesn't seem to work in this version?**

**A:** This is an effect of the addition of version 5 property-based objects. To add a privilege field, you must add another field to the Genie form by adding %privilege% in the **Privilege** field of the Genie.

## Graphics

**Q: With the version 4 graphic objects, you could display the on/off state of strings in different colors. How can I do this with the new objects?**

**A:** You can create the same effect by using the **On/Off** type in the Fill tab of a text object. Just add the digital tag again and specify the colors you want.

**Q: Why are there dots displayed all over my bar graphs and symbols?**

**A:** The dots indicate a communication error for this object display. If the I/O device associated with the data is off-line, then it is displayed with dots over it.

## Runtime

**Q: How do I run a Cicode function on startup?**

**A:** Specify the Cicode function with the Computer Setup Wizard - run in Custom mode. Use the **Startup Functions Setup** page. See [Startup Functions Configuration](#) for details.

**Q: How do I run a report on startup?**

**A:** CitectSCADA searches for a report called "Startup" when it starts up. If CitectSCADA locates this report, it is run automatically. You can change the name of the default startup report with the Computer Setup Wizard - run in Custom mode. Use the Startup report field on the Reports Setup page. See [Reports Configuration](#) for details.

**Q: How do I disable operator reboot?**

**A:** You can disable the Ctrl+Alt+Del command by using a third-party utility. See the Citect [knowledge base](#) or contact Citect Support to obtain the latest recommended software.

**Q: How do I remove the Cancel button from the Startup Message Box?**

**A:** Use the Computer Setup Wizard - run in Custom mode. De-select the **Display Cancel** button on startup option on the Security Setup - Miscellaneous page. See [Miscellaneous Security Configuration](#) for details.

**Q: How do I remove the Shutdown command from the Control menu?**

**A:** Use the Computer Setup Wizard - run in Custom mode. De-select the Shutdown on menu option on the Security Setup - Control Menu page. See [Control Menu Security Configuration](#) for details.

**Q: How do I remove the Project Editor and Graphics Builder commands from the Control Menu?**

**A:** Use the Computer Setup Wizard - run in Custom mode. De-select the Project Editor/Graphics Builder on menu option on the Security Setup - Control Menu page. See [Control Menu Security Configuration](#) for details.

## Trends

**Q: How do I get trend data into a dBase database?**

**A:** Display the trend on screen and select the File Save/As tool. This tool displays a Save/As dialog box for you to enter the name of the file, and calls the [TrnExportDBF](#) function to save the data. (For more complex procedures, call this function directly.)

**Q: How do I get trend data into Excel?**

**A:** You can get data into Excel in three ways:

- Display the trend on the screen, select the Clipboard tool to copy the data, and use the Excel paste command to paste the data into Excel.
- Display the trend on the screen and select the File Save/As tool. Save the data in CSV format, and open the CSV file in Excel.
- Display the trend on the screen and select the File Save/As tool. Save the data in DBF format and open the DBF file in Excel.

(For other procedures, call the [TrnExportCSV](#) or [TrnExportDBF](#) function.)

**Q: How do I get trend data into MS Access?**

**A:** You can get data into Access in three ways:

- Display the trend on the screen, select the Clipboard tool to copy the data, and use the Access paste command to paste the data into Access.
- Display the trend on the screen and select the File Save/As tool. Save the data in CSV format, and open the CSV file in Access.
- Display the trend on the screen and select the File Save/As tool. Save the data in DBF format and open the DBF file in Access.

(For other procedures, call the [TrnExportCSV](#) or [TrnExportDBF](#) function.)

**Q: How do I update trend data?**

**A:** Use the [TrnSetTable](#) function to write data back to the trend system.

**Q: How do I archive trend data?**

**A:** Use the trend archive Cicode functions from the Examples database. Use the Find Function command to search for the `TrendArchive()` function.

**Q: How do I restore archived trend data to the system?**

**A:** Use the trend archive Cicode functions from the "Examples" database. Use the Find Function command to search for the `TrendArchive()` function.

**Q: How do I get trend data into a report?**

**A:** Use the [TrnGetTable](#) function.

**Q: I've modified my trend tags, but they're not available for display in the display client. Why not?**

**A:** If you modify trends and restart the trend server, the modified pens won't be available for selection in the display client until you restart the display client. Restarting the display client refreshes the list of pens from trends.dbf.

## Controls

**Q: How do I start a motor with the keyboard?**

**A:** Define a Page Keyboard command that sets the value of the digital variable to 1, for example:

- Key Sequence: ENTER or F5
- Command: `Conv_Motor = 1`

**Q: How do I start a motor with a button?**

**A:** Define a Button command that sets the value of the digital variable to 1 (e.g. `Conv_Motor = 1`).

**Q: How do I adjust a setpoint from a keyboard entry?**

**A:** Define a Page Keyboard command to set the setpoint to a new value, for example:

- Key Sequence: #### ENTER
- Command: `SP1 = Arg1`

**Q: How do I enter a command that is bigger than the width of the field?**

**A:** Use an Include file or write a Cicode function and call that function. For details, see [Using Include \(Text\) Files](#) and [Writing Functions](#) respectively.

## Alarms

**Q: How do I allow the operator to go to the alarm page from any page in the system using the keyboard?**

**A:** Define a Global keyboard command (for example, the F3 key) to display the page with the [PageAlarm](#) function, or use a page [template](#) that has an Alarm Page button. Global keyboard commands are defined in System Keyboard Commands.

**Q: How do I call a function when an alarm trips?**

**A:** Define all alarms as an alarm category, and call the function in the Alarm Action field.

**Q: How do I send alarms to a dBase file?**

**A:** Specify the dBase file in the Log Device field on the Alarm Category form.

**Q: How do I display alarms?**

**A:** Use the [PageAlarm](#) function to display the standard alarm page. Alternatively, for more control, draw your own alarm page and use the [AlarmDsp](#) function.

**Q: How do I display the alarm summary?**

**A:** Use the [PageSummary](#) function to display the standard alarm summary page. Alternatively, for more control, draw your own alarm summary page and use the [AlarmDsp](#) function.

**Q: How do I get alarms into a report?**

**A:** You can do either of the following:

- Read the alarm log (.DBF) files (logged for the alarm category).
- Use alarm functions such as [AlarmFirstCatRec](#). (You can only use these functions if the Alarms server and Reports server are on the same computer.)

**Q: How do I disable groups of alarms from the I/O device?**

**A:** Program the I/O device to set a bit when it wants to disable alarms. Use an event to monitor this bit (Trigger is Bit = 1), and call the [AlarmDisable](#) function as the Action (when the bit is set).

**Q: How do I display alarms?**

**A:** When you configure the project, create a default alarm page (with the Graphic Builder) based on the alarm template. Save the page with the name "Alarm". All alarms will automatically display on this page.

**Q: How do I display alarm summaries?**

**A:** When you configure the project, create a default alarm summary page (with the Graphic Builder) based on the alarm template. Save the page with the name "Summary". All alarms will automatically display on this page.

**Q: How do I acknowledge alarms?**

**A:** Display the standard alarm page and click the left mouse button over the alarm, or use the [AlarmAckRec](#) function.

**Q: How do I set up alarm redundancy?**

**A:** Use the Computer Setup Wizard (run in Custom mode) to set up a second Alarms Server. No project reconfiguration is required.

**Q: How do I attach comments to alarms at runtime?**

**A:** Define a Page Keyboard command that calls the [AlarmComment](#) function.

**Q: How do I sound a bell when alarm occurs?**

**A:** Define the alarm in an Alarm Category and call the [Beep](#) function in the Alarm Action field.

**Q: How do I display the last alarm on all pages?**

**A:** Define a continuous animation on each page (or template) - a standard feature of all standard alarm templates.

**Q: How do I advise operators that alarms are active?**

**A:** If you are using standard templates, the clock animates. Alternatively, for more control, call the [AlarmActive](#) function.

**Q: How do I change the analogue alarms limits at runtime?**

**A:** Define a Page Keyboard command that calls the [AlarmSetThreshold](#) function.

## Miscellaneous

**Q: Why is #COM displayed on my pages?**

**A:** The #COM indicates a communication error for this animation. If the I/O device associated with the data is off line, #COM is displayed.

**Q: I'm getting communication errors with my PLC (hardware alarms, #COMS, missing symbols or missing trend data on my pages). What do I do now?**

**A:** Use the CitectSCADA Kernel window. (See [Using the CitectSCADA Kernel](#) for details.)

- 1 Once the project is started, invoke the Kernel window. In the Kernel Main window, type **debug <port name> error**. The CitectSCADA Kernel will put the port into debug mode and log errors.

- 2 Next invoke the probe window by typing probe. Inside the probe window type 'E' and then 'L' (to log the errors)
- 3 Run the project until the error(s) occur. For example, go to the problem page, then shut down the project.
- 4 The CitectSCADA Kernel, in debug mode, will log some low level communications information into the syslog.dat file, located in the windows directory. Probe will log extra information (tag address, for example) that is useful in debugging communication problems.
- 5 Go to the end of the syslog file, to where you last started your CitectSCADA project - this information is required if you need to contact Citect Support for analysis. You can look up the error codes in the online help or the Citect knowledge base and start debugging your PLC set up, cabling and CitectSCADA communications configuration.

**Q: I get the error "Citect low on Physical memory" when I startup CitectSCADA. What can I do about this error?**

**A:** On startup, CitectSCADA checks that you have enough available physical memory (real physical memory not virtual memory) to run your system. If CitectSCADA starts to use lots of virtual memory your system performance will be seriously degraded. Under some conditions CitectSCADA cannot correctly detect the amount of physical memory and this error message displays when in fact you do have enough memory. See the error message [Low physical memory](#) for details.

**Q: Why is my menu in VGA on an XGA resolution?**

**A:** The default menu is a simple menu that puts buttons to all your pages on a VGA size page. If you want a better menu, configure your own page using the menu templates.

**Q: I have a spare dongle but I do not know what type, point count or how many users it supports.**

**A:** When running CitectSCADA, start the kernel and type **PAGE GENERAL** and view the bottom of the screen. There is information on all of the above questions.

**Q: I have configured a few Events that are not running, but run on another machine with exactly the same project. What am I doing wrong?**

**A:** CitectSCADA computers will only run Events if they are set up to do so. Use the Computer Setup Wizard to enable or disable events on each computer.

**Q: How do I reset accumulators. I tried writing to the tag directly but it just keeps counting up?**

**A:** Use the [AccControl](#) function.

**Q: I have deleted a lot variable tags from my project but the number of records remain the same. It seems that even though I have deleted them they still exist?**

**A:** This is true. You need to [pack](#) your database by selecting **Pack** from the File menu in the Project Editor. This deletes all records marked for deletion and reindexes those that remain. You should pack regularly if you have been deleting or editing the Variables database file using third-party database editors (like MS-Excel).

**Q: I want to have a file server, but I'm worried that if the file server crashes I'll lose all the clients connected. How do I make this failsafe?**

**A:** Use the Computer Setup Wizard - run in Custom mode. Enter a standby location in the Backup project path field of the General Options Setup page. See [General Options Setup](#) for details.

**Q: How do I quickly set up communications to an I/O device?**

**A:** Use the Express Communications Wizard to select the I/O Server, manufacturer, then the I/O device, then the communication method. This quickly sets up the basic options required, but does not set up advanced features. See [Using the Communications Express Wizard](#) for details.



# Appendix 1: Windows Language Codes

code	Windows locale setting	code	Windows locale setting
af	Afrikaans	hu	Hungarian
sq	Albanian	is	Icelandic
ar-sa	Arabic (Saudi Arabia)	id	Indonesian
ar-iq	Arabic (Iraq)	it	Italian (Standard)
ar-eg	Arabic (Egypt)	it-ch	Italian (Switzerland)
ar-ly	Arabic (Libya)	ja	Japanese
ar-dz	Arabic (Algeria)	ko	Korean
ar-ma	Arabic (Morocco)	ko	Korean (Johab)
ar-tn	Arabic (Tunisia)	lv	Latvian
ar-om	Arabic (Oman)	lt	Lithuanian
ar-ye	Arabic (Yemen)	mk	FYRO Macedonian
ar-sy	Arabic (Syria)	ms	Malaysian
ar-jo	Arabic (Jordan)	mt	Maltese
ar-lb	Arabic (Lebanon)	no	Norwegian (Bokmal)
ar-kw	Arabic (Kuwait)	no	Norwegian (Nynorsk)
ar-ae	Arabic (U.A.E.)	pl	Polish
ar-bh	Arabic (Bahrain)	pt-br	Portuguese (Brazil)
ar-qa	Arabic (Qatar)	pt	Portuguese (Portugal)
eu	Basque	rm	Rhaeto-Romanic
bg	Bulgarian	ro	Romanian
be	Belarusian	ro-mo	Romanian (Moldavia)
ca	Catalan	ru	Russian
zh-tw	Chinese (Taiwan)	sz	Sami (Lappish)
zh-cn	Chinese (PRC)	sr	Serbian (Cyrillic)
zh-hk	Chinese (Hong Kong SAR)	sr	Serbian (Latin)
zh-sg	Chinese (Singapore)	sk	Slovak
hr	Croatian	sl	Slovenian
cs	Czech	sb	Sorbian
da	Danish	es	Spanish (Traditional)
nl	Dutch (Standard)	es-mx	Spanish (Mexico)
nl-be	Dutch (Belgium)	es-gt	Spanish (Guatemala)
en	English	es-cr	Spanish (Costa Rica)
en-us	English (United States)	es-pa	Spanish (Panama)
en-gb	English (United Kingdom)	es-do	Spanish (Dominican Republic)

code	Windows locale setting	code	Windows locale setting
en-au	English (Australian)	es-ve	Spanish (Venezuela)
en-ca	English (Canada)	es-co	Spanish (Colombia)
en-nz	English (New Zealand)	es-pe	Spanish (Peru)
en-ie	English (Ireland)	es-ar	Spanish (Argentina)
en-za	English (South Africa)	es-ec	Spanish (Ecuador)
en-jm	English (Jamaica)	es-cl	Spanish (Chile)
en	English (Caribbean)	es-uy	Spanish (Uruguay)
en-bz	English (Belize)	es-bo	Spanish (Bolivia)
en-tt	English (Trinidad)	es-sv	Spanish (El Salvador)
et	Estonian	es-hn	Spanish (Honduras)
fo	Faeroese	es-ni	Spanish (Nicaragua)
fa	Farsi	es-pr	Spanish (Puerto Rico)
fi	Finnish	sx	Sutu
fr	French (Standard)	sv	Swedish
fr-be	French (Belgium)	sv-fi	Swedish (Finland)
fr-ca	French (Canada)	th	Thai
fr-ch	French (Switzerland)	ts	Tsonga
fr-lu	French (Luxembourg)	tn	Tswana
gd	Gaelic (Scotland)	tr	Turkish
gd-ie	Gaelic (Ireland)	uk	Ukrainian
de	German (Standard)	ur	Urdu
de-ch	German (Switzerland)	vg	Valley Girl
de-at	German (Austria)	ve	Venda
de-lu	German (Luxembourg)	vi	Vietnamese
de-li	German (Liechtenstein)	xh	Xhosa
el	Greek	ji	Yiddish
he	Hebrew	zu	Zulu
hi	Hindi		

# Glossary

---

[A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [Q](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#) | XYZ

**10baseT** Ethernet implementation on unshielded twisted pair. Typically uses as RJ45 connection.

**10base2** Ethernet implementation on thin coaxial cable. Typically uses a BNC connection.

**10base5** Ethernet implementation on thick coaxial cable.

A

**Accredited - Level 1** Drivers developed under the CiTDriversQA96 CitectSCADA Driver Quality and Accreditation System, which ensures the driver was designed, coded, and tested to the highest possible standards.

**Accredited - Level 2** Drivers developed using the CiTDriversQA92 CitectSCADA Driver Quality and Accreditation System.

**accumulator** A CitectSCADA facility that allows you to track incremental runtime data such as motor run hours, power consumption, and downtime.

**active alarm** An active alarm is an alarm in one of the following states: ON and unacknowledged; ON and acknowledged; OFF and unacknowledged.

**advanced alarms** Triggered when the result of a Cicode expression changes to true. Use advanced alarms only when alarm functionality cannot be obtained with the other alarm types. If you configure too many advanced alarms, your system performance can be affected.

**alarm categories** You can assign each alarm to a category, and then process each category as a group. For example, for each category, you can specify the display characteristics, the action to be taken when an alarm in the category is triggered, and how data about the alarm is logged. You can also assign a priority to the category, which can be used to order alarm displays, filter acknowledgments, and so on.

**alarms server** Monitors all alarms and displays an alarm on the appropriate display client(s) when an alarm condition becomes active.

**alarm display page** The alarm display page displays alarm information in the following format: Alarm Time, Tag Name, Alarm Name, Alarm Description.

**alarm summary page** Displays alarm summary information in the following format: alarm name, time on, time off, delta time, comment.

**analog alarms** Triggered when an analog variable reaches a specified value. CitectSCADA supports four types of analog alarms: high and high high alarms; low and low low alarms; deviation alarms; and rate of change alarms.

**animation number files (.ANT)** ASCII text files that contain a list of animation points (ANs) and the coordinate location (in pixels) of each point.

**animation point** The points on a graphics page where an object displays. When you add an object to your page, CitectSCADA automatically allocates a number (AN) to the animation point, (i.e., the location of the object).

**area** A large application can be visualized as a series of discrete sections or areas. Areas can be defined geographically (where parts of the plant are separated by vast distances) or logically (as discrete processes or individual tasks).

**arguments** Values (or variables) passed in a key sequence to a keyboard command in runtime (as operator input). Arguments can also be the values (or variables) passed to a Cicode function when it executes.

**attachment unit interface (AUI)** Typically used to interface to a transceiver through what is often known as a drop cable.

**automation component (ActiveX object)** ActiveX objects typically consist of a visual component (which you see on your screen) and an automation component. The automation component allows the interaction between the container object and the ActiveX object.

## B

**baud rate** The number of times per second a signal changes in a communication channel. While the baud rate directly affects the speed of data transmission, the term is often erroneously used to describe the data transfer rate. The correct measure for the data rate is bits per second (bps).

**BCD variable (I/O device)** BCD (Binary Coded Decimal) is a two-byte (16-bit) data type, allowing values from 0 to 9,999. The two bytes are divided into four lots of four bits, with each lot of four bits representing a decimal number. For example the binary number 0010 represents decimal 2. Thus the BCD 0010 0010 0010 0010 represents 2,222.

**bottleneck** A bottleneck occurs when too many requests are being sent to a PLC communication link/data highway. It can occur with all types of protocols, and is dependent on several factors, including the frequency of requests, the number of duplicated (and hence wasteful) requests, whether the protocol supports multiple outstanding requests, as well as other network traffic.

**browse sequence** A series of graphics pages linked by a browse sequence, which is a linear navigation sequence within your runtime system that uses Page Previous and Page Next commands.

**byte variable (I/O device)** Byte is a one-byte data type, allowing values from 0 to 255. One byte consists of 8 bits. Each ASCII character is usually represented by one byte.

## C

**cache (I/O device data cache)** When caching is enabled, all data read from a I/O device is stored temporarily in the memory of the I/O server. If another request is made (from the same or another display client) for the same data within the cache time, the CitectSCADA I/O server returns the value in its memory, rather than read the I/O device a second time.

**callback function** A function that is passed as an argument in another function. Callback functions must be user-written functions.

**Cicode** Programming language designed for plant monitoring and control applications. Similar to languages such as Pascal.

**Cicode blocking function** A Cicode function that blocks, or waits, for an asynchronous event to complete before returning.

**CiNet** CiNet is no longer supported. CiNet was designed as a low speed wide area network (for remote monitoring applications). If you have a widely-distributed application where CitectSCADA computers are separated by vast distances, using a LAN to connect your display clients can be expensive. To connect display clients in this instance, use Microsoft's remote access server (RAS) or a Microsoft-approved solution, such as Shiva LanRover.

**CitectSCADA project** The elements of a CitectSCADA monitoring and control system, such as graphics pages, objects, and so on. These elements are stored in files of various types; for example, graphics files for graphics pages, databases for configuration records, and so on. You use the CitectSCADA compiler to compile the project into a runtime system.

**CitectSCADA computer** A computer running CitectSCADA. Other common industry terms for this computer could be node, machine or workstation.

**CitectSCADA server** A computer connected to an I/O device (or number of I/O devices). When CitectSCADA is running, the server exchanges data with the I/O device(s) and distributes information to the other display clients as required.

**`citect.ini` file** A text file that stores information about how each CitectSCADA computer (servers and display clients) operates in the CitectSCADA configuration and runtime environments. The `Citect.INI` file stores parameters specific to each computer and therefore cannot be configured as part of the project.

**client** A computer that accesses shared network resources provided by another computer called a server. CitectSCADA's client-server based architecture is designed to distribute the processing tasks and optimize performance.

**cluster** A discrete group of alarms servers, trends servers, reports servers, and I/O servers. It would usually also possess local CitectSCADA display clients. For a plant comprising several individual sections or systems, multiple clusters can be used, one cluster for each section.

**command** A command performs a particular task or series of tasks in your runtime system. A command is built from Cicode and can consist of just a function or a statement.

**communications link** A connection between computers and peripheral devices, enabling data transfer. A communications link can be a network, a modem, or simply a cable.

**communications port** PC port used for sending and receiving serial data (also called serial or COM ports).

**custom alarm filters** Custom alarm filters provide a way to filter and display active alarms. Up to eight custom filter strings can be assigned to a configured alarm. In conjunction with a user-defined query function, the custom filters enable operators to identify and display active alarms of interest.

## D

**data acquisition board** Data acquisition boards communicate directly with field equipment (sensors, controllers, and so on). You can install a data acquisition board in your CitectSCADA server to directly access your field equipment.

**data bits** Group of binary digits (bits) used to represent a single character of data in asynchronous transmission.

**data transfer** Transfer of information from one location to another. The speed of data transfer is measured in bits per second (bps).

**data type (I/O device)** Type of I/O device variable. I/O devices may support several data types that are used to exchange data with CitectSCADA. You must specify the correct CitectSCADA data type whenever I/O device variables are defined or referenced in your CitectSCADA system.

**DB-15** Often called a 'D' type connector due to the vague D shape of the casing. Has 15 pins arranged in two rows of 8 and 7 pins. While not as common as DB-9 or DB-25 they may be found on some computers and data communication equipment. Comes in both male (pins protruding) and female (pin sockets) configurations.

**DB-25** Often called a 'D' type connector due to the vague D shape of the casing. Has 25 pins arranged in two rows of 13 and 12 pins. This kind of connection is a part of the standard for RS-232-D and is found on many computers, modems and other data communication equipment. Comes in both male (pins protruding) and female (pin sockets) configurations.

**DB-9** Often called a 'D' type connector due to the vague D shape of the casing. Has 9 pins arranged in two rows of 5 and 4 pins. This kind of connection is common and is often used as the serial (com) port in computers. Often used in modems and other data communication equipment. Comes in both male (pins protruding) and female (pin sockets) configurations.

**data communications equipment (DCE)** Devices that establish, maintain, and terminate a data transmission connection. Normally referred to as a modem.

**dynamic data exchange (DDE)** A Microsoft Windows standard protocol set of messages and guidelines that enables communication between Windows applications on the same Windows computer.

**dynamic data exchange (DDE) Server** A Windows standard communication protocol supported by CitectSCADA. The CitectSCADA I/O server communicates with the DDE server using the Windows standard DDE protocol. DDE servers are appropriate when data communication is not critical as DDE servers are not designed for high-speed data transfer.

**deviation alarm** Triggered when the value of a variable deviates from a setpoint by a specified amount. The alarm remains active until the value of the variable falls (or rises) to the value of the deadband.

**dial back** Only returns calls from remote I/O devices.

**dial-in modem** Only receives calls from remote I/O devices, identifies the caller, then hangs up immediately so it can receive other calls. CitectSCADA then returns the call using a dial-back modem.

**dial-out modem** Makes calls to remote I/O devices in response to a CitectSCADA request; e.g., scheduled, event-based, operator request, and so on. Also returns calls from remote I/O devices.

**Digiboard** A high-speed serial board manufactured by the Digiboard Corporation.

**digital alarms** Triggered by a state change in a digital variable. Use these alarms when a process has only one of two states. You can use either the on (1) state or off (0) state (of a digital variable) to trigger the alarm.

**digital variable (I/O device)** Usually associated with discrete I/O in your I/O device, a digital variable can only exist in one of two states: on (1) or off (0). Allowed values for the digital data type are therefore 0 or 1. Discrete inputs (such as limit switches, photoelectric cells, and emergency stop buttons) and discrete outputs are stored as digital variables.

**disk I/O device** A disk file that resides on the hard disk of a computer and emulates a real I/O device. The value of each variable in the disk I/O device is stored on the computer hard disk. The disk I/O device is not connected to any field equipment in the plant.

**display client** The interface between the runtime system and an operator. If you are using CitectSCADA on a network, all CitectSCADA computers (on the network) are display clients.

**display period** Defines the rate at which trend data is displayed on the trend page.

**distributed processing** For large applications with large amounts of data, you can distribute the data processing to reduce the load on individual computers.

**distributed servers** If your plant consists several sections or systems, you can assign a cluster to each individual section, and then monitor all sections using one CitectSCADA display client.

**Note:** Don't use distributed servers to split up a single section or process into discrete areas. A single cluster system with distributed processing would be better used here since it would not be hampered by the maintenance overhead of a distributed server system (such as extra project compilations, and so on).

**dither (imported bitmaps)** A method of approximating colors in imported or pasted bitmaps that involves combining pixels of different or colors from a color palette.

**domain name server (DNS)** Database server that translates URL names into IP addresses.

**dot notation** Used for Internet addresses. Dot notation consists of four fields (called octets), each containing a decimal number between 0 and 255 and separated by a full stop (.).

**data terminal equipment (DTE)** Devices acting as data source, data sink, or both.

**driver** Used by CitectSCADA to communicate (with control and monitoring devices) in a device-independent way, allowing the run-time system to interact directly with different types of devices. Communication with an I/O device requires a device driver which implements the communication protocol(s).

**duplex** The ability to send and receive data over the same communication line.

## E

**empty value** Indicates that the variant has not yet been initialized (assigned a value). Variants that are empty return a VarType of 0. Variables containing zero-length strings (" ") aren't empty, nor are numeric variables having a value of 0.

**ethernet** Widely used type of local area network based on the CSMA/CD bus access method (IEEE 802.3).

**expression** A statement (or group of statements) that returns a value. An expression can be a single variable, a mathematical formula, or a function.

**Event data displayed by time** As an alternative to viewing event trend data by event number, it is possible to see event trends across a timeline. When event trends



are shown by time, the trend graph includes a start and end time and enables operators to see both the time of a triggered event, and the elapsed period between events. This data can also be displayed on the same graph as a periodic trend.

**event trend/SPC** To construct an event trend/SPC, CitectSCADA takes a sample when a particular event is triggered (in the plant). This sample is displayed in the window. The event must then reset and trigger again, before the next sample is taken. Events are identified by the event number.

## F

**file server** A computer with a large data storage capacity dedicated to file storage and accessed by other client computers via a network. On larger networks, the file server runs a special network operating system. On smaller installations, the file server may run a PC operating system supplemented by peer-to-peer networking software.

**full duplex** Simultaneous two-way (in both directions) independent transmission (4 Wires).

## G

**generic protocol** A pseudo-protocol supported by disk I/O devices that provides a convenient way to represent disk data. The generic protocol is not a real protocol (communicates with no physical I/O device).

**Genie** If you have numerous devices of the same type (e.g., 100 centrifugal pumps), the display graphics for each will behave in much the same way. Using Genies, you only have to configure common behavior once. The graphics can then be saved as a Genie and pasted once for each device.

**Genie controller** A Genie that references a Super Genie (using a Super Genie function). Using a Genie controller, the same Super Genie can be used over and over for different applications.

**global Cicode variable** Can be shared across all Cicode files in the system (as well as across include projects).

**global client** A display client used to monitor information from several systems or sections (using clusters).

**graphics bounding box** A faint (grayed) dotted rectangular box outline defining the exterior boundary region of a graphic object. Only visible and active when the graphics object is selected and being resized. Contains sizing handles in each corner and (if sized large enough to display) one in the centre of each side.

**graphics page** A drawing (or image) that appears on a workstation to provide operators with control of a plant, and display a visual representation of conditions within the plant.

**group (of objects)** CitectSCADA allows you to group multiple objects together. Each group has a unique set of properties, which determine the runtime behavior of the group as a whole.

## H

**half duplex** Transmission in either direction, but not simultaneously.

**histogram** A bar graph that shows frequency of occurrence versus value. Quite often the data is fitted to a distribution such as a normal distribution.

## I

**include file (.CII)** There is a maximum number of characters that you can type in a Command or Expression field (usually 128). If you need to include many commands (or expressions) in a property field, you can define a separate include file that contains commands or expressions. An include file is a separate and individual ASCII text file containing only one sequence of CitectSCADA commands or expressions that would otherwise be too long or complicated to type into the command or expression field within CitectSCADA. The include file name is entered instead, and the whole file is activated when called.

**I/O device** An item of equipment that communicates with plant-floor control or monitoring equipment (sensors, controllers, and so on). The most common I/O devices are PLCs (programmable logic controllers); however, CitectSCADA supports a wide range of I/O devices, including loop controllers, bar code readers, scientific analyzers, remote terminal units (RTUs), and distributed control systems (DCS). CitectSCADA can communicate with any I/O device that has a standard communications channel or data highway.

**I/O device address** The (logical) location of the I/O device in the system. Each I/O device must have a unique address in the CitectSCADA system, unless the I/O device is defined in other servers (to provide redundancy). If redundancy is used, the I/O device must then have the same I/O device name, number, and address for each server.

**I/O device variable** A unit of information used in CitectSCADA. Variables are stored in memory registers in an I/O device. CitectSCADA exchanges information with an I/O device by reading and writing variables. CitectSCADA refers to I/O device variables by their register addresses. I/O devices usually support several types of variables; however, the most common are digital variables and integer variables.

**I/O server** A dedicated communications server that exchanges data between I/O devices and CitectSCADA display clients. No data processing is performed by the I/O server (except for its local display). Data is collected and passed to the display clients for display, or to another server for further processing. All data sent to an I/O device from any CitectSCADA computer is also channelled through the I/O server. If data traffic is heavy, you can use several I/O servers to balance the load.

**IP address** A unique logical address used by the Internet Protocol (IP). Contains a network and host ID. The format is called dotted decimal notation, and is written in the form: w.x.y.z

**integer variable (Cicode)** A 4-byte (32-bit) data type allowing values from 2,147,483,648 to 2,147,483,647.

**integer variable (I/O device)** A 2-byte data type, allowing values from -32,768 to 32,767, that is used to store numbers (such as temperature or pressure). Some I/O devices also support other numeric variables, such as real (floating point) numbers, bytes, and binary-coded decimals.

**Internet display client** Allows you to run CitectSCADA projects over the Internet from a remote location. It is basically a “runtime-only” version of CitectSCADA: you can run your project from that computer, just as you would from any normal display client.

**interrupt** An external event indicating that the CPU should suspend its current task to service a designated activity.

## K

**keyboard command** Consist of a key sequence that an operator enters on the keyboard, and an instruction (or series of instructions) that executes when the key sequence is entered. Keyboard commands can be assigned to an object or page, or they can be project-wide.

**knowledge base** Provides high-level technical information beyond the scope of standard CitectSCADA technical documentation that is updated regularly and available at <http://www.citect.com>.

**kurtosis** An index indicating the degree of peakedness of a frequency distribution (usually in relation to a normal distribution). Kurtosis < 3 indicates a thin distribution with a relatively high peak. Kurtosis > 3 indicates a distribution that is wide and flat topped.

## L

**local area network (LAN)** A system that connects computers to allow them to share information and hardware resources. With real-time LAN communication, you can transfer data, messages, commands, status information, and files easily between computers.

**language database** When a project is compiled, CitectSCADA creates a language database (dBASE III format) consisting of two fields: native and local. Any text marked with a language change indicator is automatically entered in the native field. You can then open the database and enter the translated text in the local field.

**link** A copy of a library item, possessing the properties of the library original. Because it is linked, the copy is updated whenever the original is changed.

**local Cicode variable** Only recognized by the function within which it is declared, and can only be used by that function. Local variables must be declared before they can be used. Any variable defined within a function (i.e., after the function name) is a local variable, therefore no prefix is needed. Local variables are destroyed when the function exits and take precedence over global and module variables.

**local language** The language of the end user. Runtime display items such as alarm descriptions, button text, keyboard/alarm logs, graphic text, Cicode strings and so on can be displayed in the local language, even though they may have been configured in the language of the developer (native language).

**long BCD variable (I/O device)** A 4-byte (32-bit) data type, allowing values from 0 to 99,999,999. The four bytes are divided into eight lots of four bits, with each lot of four bits representing a decimal number. For example the binary number 0011 represents decimal 3. Thus the BCD 0011 0011 0011 0011 0011 0011 0011 0011 represents 33,333,333.

**long variable (I/O device)** A 4-byte (32-bit) data type allowing values from 2,147,483,648 to 2,147,483,647.

**low and low low alarms** Defined by specifying the values of the variable that trigger each of these alarms. As a low alarm must precede a low low alarm, the low alarm no longer exists when the low low alarm is triggered. Note that the variable must rise above the deadband before the alarm becomes inactive.

## M

**manager client** A computer configured with manager-only access to the runtime system. No control of the system is possible, but full access to data monitoring is permitted.

**maximum request length** The maximum number of data bits that can be read from the I/O device in a single request. For example, if the maximum request length is 2048 bits, the maximum number of integers that can be read is:  $2048/16 = 128$ .

**millisecond trending** Allows you to use a trends sample period of less than one second.

**mimic** A visual representation of a production system using an organised set of graphical pages.

**module Cicode variable** Specific to the file in which the variable is declared. This means that it can be used by any function in that file, but not by functions in other files. By default, Cicode variables are defined as module, therefore prefixing is not required (though a prefix of MODULE could be added if desired). Module variables should be declared at the start of the file.

**multi-digital alarms** Use combinations of values from three digital variables to define eight states. For each state, you specify a description (e.g., healthy or stopped), and whether or not the state triggers an alarm.

## N

**native language** Generally the language of the project developer. Display items such as alarm descriptions, button text, keyboard/alarm logs, graphic text, Cicode strings and so on can be configured in the native language, and displayed, at runtime, in the language of the end-user (local language).

**Network Dynamic Data Exchange (NetDDE)** Enables communication between Windows applications on separate computers connected across a common network.

**network** A group of computers and peripheral devices, connected through a communications link. Data and services (e.g., printers, file servers, and modems) can be shared by computers on the network. A local network of PCs is called a LAN.

**network computer** A computer running CitectSCADA that is connected to a LAN through a network adaptor card and network software.

**nodes** A structural anchor point for a graphic object, usually visible as a small square box superimposed over a graphic. Nodes are always located separately at the start, at the end, and at every change in direction within a graphic object.

**normal distribution** Also known as a 'bell' curve, the normal distribution is the best known and widely applicable distribution. The distribution is symmetrical and popularly represents the laws of chance. 68.27% of the area lies between -1 sigma and +1 sigma, 95.45% between -2 sigma and +2 sigma, and 99.73% between -3 sigma and +3 sigma. The values of skewness and kurtosis are used to provide quantitative measures for normality. Assuming that at least 20 samples are used to construct a distribution, a good rule of thumb is to accept the data as a normal distribution when,  $-1.0 = \text{skewness} = 1.0$   $2 = \text{kurtosis} = 4$

**null value** Indicates that a variant contains no valid data. Variants that are *null* return a VarType of 1. Null is not the same as *empty*, which indicates that a variant has not yet been initialized. It is also not the same as a zero-length string (" "), which is sometimes referred to as a null string. Null is not equivalent to zero or blank. A value of null is not considered to be greater than, less than, or equivalent to any other value, including another value of null. A boolean comparison using a null value will always return false.

## O

**object** Basic building blocks of a graphics page. Most objects possess properties that allow them to change dynamically under user-definable runtime conditions allowing them to provide animated display of conditions within the plant.

**object variable (Cicode)** An ActiveX control that can only be declared with local, module, or global scope.

**open database connectivity (ODBC)** Allows applications to access data in database management systems using structured query language (SQL) to access data.

**object ID (OID)** An object ID associated with every tag in a CitectSCADA project that uniquely identifies the tag for use by tag-based drivers, automatically generated at compile. It is used instead of the actual address of the register (which is what most other drivers use to read from and write to I/O devices).

## P

**periodic trend** A trend that is sampled continuously at a specified period. You can also define a trigger (an event) to stop and start the trend (when a specified condition occurs in the plant).

**point limit** An individual digital (or analog) variable read from an I/O device. CitectSCADA only counts physical points (and counts them only once, no matter how many times they are used).

The point limit is the maximum number of I/O device addresses that can be read and is specified by your CitectSCADA license. When you run CitectSCADA the point count of your project is checked against the point limit specified by your Hardware Key.

**properties, object** Describes the appearance of an object (size, location, color, and so on.) and its function (the command or expression executed by the object, the privilege required to gain access to the object, and so on).

**pack** Packing a database re-indexes database records and deletes records marked for deletion. If you edit your CitectSCADA databases externally to CitectSCADA, you should pack the database afterwards.

**page environment variable** A read-only variable associated with a particular page. When you make the association, you name the variable, and assign it a value. When the page is opened during runtime, CitectSCADA creates the variable. Its value can then be read. When the page is closed, the environment variable memory is freed (discarded).

**parity** A communications error-checking procedure. The number of 1's must always be the same (even or odd) for each group of bits transmitted without error.

**persistence cache** Cache data saved to a computer hard disk that allows an I/O server to be shut down and restarted without having to re-dial each I/O device to get its current values. This cache consists of all the I/O device's tag values.

**PLC interface board** You can sometimes install a PLC interface board in your CitectSCADA server. A proprietary interface board is usually supplied by your PLC manufacturer, and you can connect it to a PLC or a PLC network. You can only use proprietary interface boards with the same brand of PLC.

**port(s)** Provide the communication gateway to your I/O device(s).

**primary alarms server** The server that normally processes alarms.

**primary reports server** The server that normally processes reports.

**primary trends server** The server that normally processes trends.

**protocol** Messaging format consisting of a set of messages and guidelines used for communication between the CitectSCADA server and an I/O device. The communication protocol determines how CitectSCADA and the I/O device communicate; the type of data to exchange; rules governing communication initiation and termination; and error detection.

**proxi/proxy server** Caches internet transactions to improve performance by reducing the average transaction times by storing query and retrieved information for re-use when the same request is made again. When an Internet display client (IDC) connects to a proxy server, that server provides the TCP/IP addresses necessary to access report server session information.

## R

**rate of change alarms** Triggered when the value of the variable changes faster than a specified rate. The alarm remains active until the rate of change falls below the specified rate. Deadband does not apply to a rate of change alarm.

**real variable (Cicode)** Real (floating point) is a 4-byte (32-bit) data type allowing values from  $3.4E38$  to  $3.4E38$ . Use a real variable to store numbers that contain a decimal place.

**real variable (I/O device)** Real (floating point) is a 4-byte (32-bit) data type, allowing values from  $3.4E38$  to  $3.4E38$ . Use a real variable to store numbers that contain a decimal place.

**record name** Usually the primary property of a database record, referenced in CitectSCADA system through its name. Database record names must be unique for each type of database record. Sometimes you can use identical names for different record types. However, to avoid confusion, you should use a unique name for each database record in your application.

When you specify a name for a database record, the name must begin with an alphabetic character (A-Z, a-z) and can only include alphanumeric characters (A-Z, a-z, 0-9) and the underscore character (\_). For example, "Pressure," "Motor\_10," and "SV122\_Open" are all valid database record names. Each database record name can contain up to 16 characters.

Database record names are not case-sensitive, so "MOTOR\_1," "Motor\_1" and "motor\_1" are all identical database record names. Always use a meaningful name for any database record as well as the necessary naming conventions.

**redundancy** A method of using the hardware in a CitectSCADA system such that if one component in the system fails, control of the system is maintained, and no data is lost.

**remote communications** Interaction between two computers through a modem and telephone line.



**remote terminal** A terminal remote from the computer that controls it. The computer and remote terminal communicate via a modem and telephone line.

**report** A statement or account of plant-floor conditions. CitectSCADA reports can be requested when required, on a periodic basis, or when an event occurs.

**report format file** Controls the layout and content of reports. The format file is edited using a text editor and can be in either ASCII or RTF format.

**reports server** Controls report processing. You can request reports at any time or when specific events occur.

**reserved words** Words that cannot be used as a name for any database record or Cicode function.

**RJ11** A type of IDC plug commonly used in data communications. Recognizable as the style of data plug used in phone line and handset connectors. RJ11 is a 6/4 plug with 6 contacts but only 4 loaded.

**RJ12** A type of IDC plug commonly used in data communications. Recognizable as the style of data plug used in phone line and handset connectors. RJ12 is a 6/6 plug with 6 contacts.

**RJ45** A type of IDC plug commonly used in data communications. Recognizable as the style of data plug used in phone line and handset connectors. RJ45 is often used with 10baseT and is an 6/8 plug with 8 contacts.

**runtime system** The CitectSCADA system that controls and monitors your application, process, or plant. The runtime system is sometimes called the Man-Machine Interface (MMI), and is compiled from a CitectSCADA project

**RS-232C standard** An industry standard for serial communication. The standard specifies the lines and signal characteristics that are used to control the serial transfer of data between devices.

**RS-422** An industry standard for serial communication. The standard specifies the lines and signal characteristics that are used to control the serial transfer of data between devices. RS-422 uses balanced voltage interface circuits.

**RS-485** An industry standard for serial communication. The standard specifies the lines and signal characteristics that are used to control the serial transfer of data between devices. RS-485 uses balanced voltage interface circuits in multi-point systems.

## S

**scalable architecture** A system architecture that can be resized without having to modify existing system hardware or software. CitectSCADA lets you re-allocate tasks as more CitectSCADA computers are added, as well as distribute the processing load.



**serial communication** Uses the communication port on your computer or a high speed serial board (or boards) installed inside your computer.

**schedule period** Determines how often the I/O server contacts a scheduled I/O device to read data from it.

**server** A local area network (LAN) computer that perform processing tasks or makes resources available to other client computers. In CitectSCADA, client-server architecture distributes processing tasks to optimize performance.

**simplex transmission** Data transmission in one direction only.

**skewness** An index indicating the degree of asymmetry of a frequency distribution (usually in relation to a normal distribution). When a distribution is skewed to the left (for example), that the tale is extended on that side. Skewness  $> 0$  indicates the distribution's mean (and tale) is skewed to the right. Skewness  $< 0$  indicates the distribution's mean (and tale) is skewed to the left.

**soft PLC** A pure software (virtual) PLC created by software and existing only within the computer memory. Usually provides a software interface for communication (READ and WRITE) operations to take place with the soft PLC. Also known as a 'virtual field unit' or 'virtual I/O device'.

**software protection** CitectSCADA uses a hardware key that plugs into the printer port of your computer to safeguard against license infringement. The hardware key contains the details of your user license. When you run CitectSCADA, the point count in your project is checked against the point limit specified in the hardware key.

**slider control** Allow an operator to change the value of an analog variable by dragging an object (or group) on the graphics page. Sliders also move automatically to reflect the value of the variable tag.

**standby alarms server** The CitectSCADA Server that processes alarms if the primary alarms server is unavailable.

**standby reports server** The CitectSCADA server that processes reports if the primary reports server is unavailable.

**standby trends server** The CitectSCADA server that processes trends if the primary trends server is unavailable.

**stop bits** The number of bits that signals the end of a character in asynchronous transmission. The number is usually 1 or 2. Stop bits are required in asynchronous transmissions because the irregular time gaps between transmitted characters makes it impossible for the server or I/O device to determine when the next character should arrive.

**symbol** An object (or group of objects) stored in a library for later retrieval and use. By storing common objects in a library, you reduce the amount of disk space

required to store your project, and reduce the amount of memory required by the run-time system.

## T

**task** Includes operations such as I/O processing, alarm processing, display management, and Cicode execution. Any individual 'instance' of Cicode is also a 'task'.

**template** A base drawing or time-saving pattern used to shape a graphics page. Each template contains base information for the page, such as borders and common control buttons. CitectSCADA provides templates for all common page types.

**text box** When text is added to a graphics page, it is placed in a text box. A text box has a number of handles, which can be used to manipulate the text object.

**thread** Used to manage simultaneous execution of tasks in multitasking operating systems, enabling the operating system to determine priorities and schedule CPU access.

**timeout** The period of time during which a task must be completed. If the timeout period is reached before a task completes, the task is terminated.

**time server** Periodically synchronizes the time clock on all display clients (and other CitectSCADA servers) to the time and date of the computer configured as the time server.

**time-stamped alarms** An alarm triggered by a state change in a digital variable. Time-stamped alarms have an associated register in the I/O device to record the exact time when the alarm changes to active. Use time-stamped alarms when you need to know the exact order in which alarms occur.

**time-stamped analog alarms** Time stamped analog alarms work in the same way as analog alarms except that they are time stamped (with the Alarm On and Alarm Off times) using millisecond precision from the time kept by the field device (i.e. the RTU or PLC). The configuration details for time stamped analog alarms are exactly the same as for analog alarms.

**time-stamped digital alarms** Time stamped digital alarms work in the same way as digital alarms except that they are time stamped (with the Alarm On and Alarm Off times) using millisecond precision from the time kept by the field device (i.e. the RTU or PLC). The configuration details for time stamped digital alarms are exactly the same as for digital alarms.

**tool tip** A help message that displays in a pop-up window when an operator holds the mouse stationary over an object.

**touch (object at runtime)** An object is considered touched if an operator clicks it.

**Touch command** Can be assigned to objects on graphics pages. Touch commands allow you to send commands to the runtime system by clicking an object.

**trend** A graphical representation of the changing values of a plant-floor variable (or expression), or a number of variables.

**trend line** The actual line on a trend that represents the changing values of a plant-floor variable (or expression).

**trend plot** Consists of a trend (or a number of trends), a title, a comment, scales, times and so on.

**trends server** Controls the accumulation and logging of trend information. This information provides a current and historical view of the plant, and can be processed for display on a graphics page or printed in a report.

## U

**unsigned integer variable (I/O device)** A 2-byte (16 bit) data type, representing an integer range from 0 to 65,535. This is supported for all I/O devices that can use INT types. This means you can define any integer variable as an unsigned integer to increase the positive range.

## V

**variable type (Cicode)** The type of the variable (INT (32 bits), REAL (32 bits), STRING (256 bytes), OBJECT (32 bits)).

**virtual** Behavioural identification rather than a physical one. For example, Windows 95 is a virtual desktop.

## W

**wizard** A CitectSCADA facility that simplifies an otherwise complex procedure by presenting the procedure as a series of simple steps.



# Index

## Numerics

- 10base2, defined, 893
- 10base5, defined, 893
- 10baseT, defined, 893
- 9-to-25-pin converter, 713

## A

- access
  - disabling, 422
  - object, 419
- access properties, 420
- access property, object, 308
- access table, reading data from, 618
- accredited - level 1, defined, 893
- accredited - level 2, defined, 893
- accumulator, defined, 893
- accumulators, 497
  - configuring, 497
- action queries, calling, 620
- active alarm, defined, 893
- Active Alarms button, 868
- ActiveX objects, 367
  - identifying, 370
  - managing associated data sources, 367
- ActiveX.zip, 832
- adding, user records, 552
- address forwarding, 770
- Admin Tools page template, 864
- advanced alarms, 463, 893
- Advanced Security Settings dialog, 248
- alarm categories, defined, 893
- alarm display fields, 478
- alarm display page, defined, 893
- alarm filters, 436, 896
- alarm page template, 858
- Alarm Silence button, 868
- alarm summary fields, 480
- alarm summary page, defined, 893
- alarm tag name, 197
- alarms, 435
  - advanced, 463, 893
  - advanced alarm properties, 463
  - alarm category properties, 439
  - analog, 457, 894
  - analog alarm properties, 457
  - categories of, 439
  - configured, 435
  - creating pages for, 486
  - custom filters, 436
  - debugging hardware, 769
  - delay, 436
  - digital, 443
  - digital alarm properties, 444
  - display order, 481
  - displaying variable data in, 477
  - filters, 436
  - formatting display, 476
  - multi-digital, 447
  - multi-digital alarm properties, 448
  - pages, displaying, 486
  - properties supported as tags, 482
  - runtime, 485
  - setting up, 485
  - time-stamped, 453
  - time-stamped alarm properties, 453
  - time-stamped analog, 470
  - time-stamped analog alarm properties, 470
  - time-stamped digital, 466
  - time-stamped digital alarm properties, 466
  - using properties as tags, 482
  - writing to properties of, 484
- alarms group, 878
- alarms server, defined, 893
- Alarms Summary button, 868
- Alarms toolbar, 868
- alarms, audible, 872
- Align dialog box, 316
- aligning objects, 279, 280, 316
- alternative .ini file, 768
- AN (animation point), 894
- analog alarms, 457, 894
- animation number files, defined, 894
- Animation Point (AN), 283, 320
- animation point (AN), 894
- ANSI character sets, 647
- appearance

- graphics page, 272
  - of objects, 302
- appending data with ODBC, 619
- archiving
  - data, 585
  - projects, 97
- area prefix in tag names, 198
- areas, 308, 894
  - and security, 558
  - defining, 556
  - groups, 559, 560
  - labels for, 558
  - privileges, 561, 562
  - Super Genies and, 638
  - viewing, 562
- arguments
  - defined, 894
  - in labels, 567
  - values for, 567
- arrangements, pipe, 339
- array size, 212
- arrays, 211
  - Super Genies and, 635
- ASCII character sets, 647
- ASCII devices, 571
  - format, 579
- assigning variable tags, 197
- Attach Super Genie dialog box, 637
- attachment unit interface (AUI), defined, 894
- attribute section in tag names, 199
- attributes for tag names, 200
- audible alarms, implementing, 872
- automation component, 894
- Autoscale button, 863

## B

- Back button, 866
- backup utility, 102
- backups, project, 97
- baud rate, defined, 894
- BCD variable, 894
- Before You Begin, 163
- bitmaps, 295, 301
- boards
  - proprietary, 700

- serial, 698
- bottleneck, defined, 894
- breaking link to data source, 216
- Bring Forwards command, 315
- Bring to Front command, 315
- browse sequence
  - defined, 894
  - pages, 267
- Button Properties dialog box, 351
- buttons (objects), 350
- byte variable, defined, 895

## C

- cables, wiring, 713
- cache
  - data, 688
  - defined, 895
  - tuning, 813
- cache kernel command, 792
- calculating
  - array size, 212
  - disk storage (trends), 513
- callback functions, 783, 895
- calling action queries, 620
- capability charts, 521
- capability, process, 519
- categories, alarm, 439
- center limit, 520
- Changes
  - Abandoning changes, 195
  - Saving a backup, 194
  - Saving changes, 194
- changing
  - alarm summary display, 481
  - languages, 643
- character sets, 647
- charts
  - capability, 521
  - control, 520
- Cicode blocking function, defined, 895
- Cicode blocks in reports, 545
- cicode kernel command, 793
- Cicode objects, 362
- Cicode, defined, 895
- circles, drawing, 329

- 
- Citect driver accreditation, 722
  - Citect Internet Client Setup dialog box, 768
  - Citect Server Manager
    - interface, 817
    - shutting down, 818
  - Citect support, 779
  - citect.ini
    - About citect.ini, 161
    - Comments, 163
    - Header, 162
    - Opening the configuration file, 173
    - Parameters, 162
  - citect.ini file, defined, 895
  - CiUSAFE dialog box, 784
  - client syntax and DDE, 590
  - client, defined, 895
  - cls kernel command, 794
  - cluster context, 81
  - Clusters
    - cluster properties, 139
  - clusters
    - defined, 896
  - color fonts, defining, 490
  - color, fill, 393, 394, 396, 400
  - color, gradient, 327, 331, 336
  - colors on graphics pages, 285
  - command fields, 586
  - commands
    - defined, 896
    - kernel, 791
    - keyboard, 411
    - SQL, 608
    - touch, 408
  - Comments, 163
    - Commenting a parameter entry, 187
    - Commenting a section entry, 186
    - Entering comments in the header, 188
  - comments in reports, 546
  - common toolbars, 865
  - communications
    - configuring, 703
    - report errors, 547
    - scheduling, 728
    - serial, 695
    - troubleshooting, 743
    - types, 683
    - with I/O devices, 683
  - communications link, defined, 896
  - communications port, defined, 896
  - communications test project, 772
  - Comparison Wizard, 191
    - Interpreting the Report, 190
    - Running the Report, 190
  - compilation
    - debugging, 755
    - incremental, 755
    - project, 753
  - Computer Setup Wizard, 125
    - alarm, 128
    - cluster connections, 133
    - computer role, 126
    - CPU configuration, 130
    - events, 131
    - general options setup, 135
    - internet server, 128
    - keyboard security, 135
    - menu security, 134
    - miscellaneous security, 135
    - network model, 127
    - project, 126
    - reports, 129
    - Startup Functions setup, 132
    - time, 134
    - trends, 130
  - Computer Setup Wizard, about, 125
  - Computer Setup Wizard, running for CSV\_Include, 870
  - COMx driver, 695
    - debugging, 773
  - configuration
    - incorrect, 820
  - Configuration Analysis Report
    - Running a Report, 189
  - configuration, startup/runtime, 763, 765
  - Configure Alarm Groups dialog box, 879
  - Configure Trend Groups dialog box, 880
  - configured alarm types, 435
  - configuring
    - accumulators, 497
    - alarms, 435, 485

- communications, 703
- DDE conversations, 591
- devices, 574
- events, 493
- history files, 514
- network DDE shares, 600
- ODBC drivers, 612
- reports, 539
- trend tags, 502
- variable tags, 203
- Web Client deployment, 835
- connecting
  - database, 594
  - network DDE shared application, 604
  - SQL database, 608
- constants and Super Genies, 635
- control charts, 520
- Control Menu page, CSV\_Include, 870
- control, statistical, 518
- conversations, DDE, 590
- converting
  - date and time, 622
  - values to strings, 567
- Copy Project dialog properties, 94
- copying
  - objects, 313
  - projects, 94
- corners, rectangle, 328
- cp index, 521
- cpk index, 521
- creating
  - alarm page, 486
  - alarms group, 878
  - browse sequence, 267
  - custom menus, 875, 877
  - Genies, 625
  - graphics pages, 259
  - pages, 872
  - projects, 869
  - templates, 266, 267
  - Web Client deployment, 836
- custom alarm filters, 436, 896
- custom fonts, 490

## D

- data
  - appending, 619
  - archiving, 585
  - caching, 688
  - displaying in alarms, 477
  - editing, with ODBC, 619
  - exchanging, 589, 593
  - exporting trend, 511
  - formatting device, 578
  - logging in different languages, 646
  - printing trend, 510
  - reading data from access table, 618
  - trending, 501
  - using devices to read, 573
- data acquisition board, defined, 896
- data bits, defined, 896
- data communications equipment (DCE), 714, 897
- data sources, external, 226
- data terminal equipment (DTE), 714, 898
- data transfer, defined, 896
- data type (I/O device), defined, 896
- data types
  - DDE, 593
  - variable tag, 204
- database device, 581
- database transactions, 609
- databases
  - external, 606
  - language, 644
- date conversions, 622
- dates in SQL, 610
- DB-15, defined, 896
- DB-25, defined, 896
- DB-9, defined, 897
- dBASE database devices, 571
- dBASE databases, 606
- dBASE devices
  - format, 580
- DDE conversations, 590
  - configuring, 591
- DDE services, 598
- DDE shares, 601
- DDE trusted shares, 603
- debug kernel command, 794



- debugging
  - compilation, 755
  - COMx driver, 773
  - I/O devices, 771
  - proprietary board drivers, 779
  - protocol drivers, 777
  - protocols, 771
  - runtime system, 769
  - server redirection, 770
  - TCP/IP drivers, 776
- decimal notation, 210
- default cluster context, 81
- default languages for Web Client, 843
- defaults, graphics page, 277
- defining
  - access to objects, 419
  - alarm categories, 439
  - areas, 556
  - colors on graphics pages, 285
  - labels, 568
  - page properties, 270
  - paths, 515
  - substitutions for Genies, 626
  - substitutions for Super Genies, 633
  - trend pages, 509
  - user privileges, 554
- delaying alarms, 436
- deleting
  - objects, 312
  - projects, 96
  - rows from access table, 620
- demo mode, 784
- deployment for Web Client, 838
  - configuring, 835
  - creating, 836
  - deleting, 842
  - displaying, 840
  - editing, 841
  - updating, 842
- DevAppend() function, 581
- DevDelete() function, 583
- DevFind() function, 582
- DevGetField() function, 582
- deviation alarm, defined, 897
- device history files, 583
- devices, 571
  - ASCII devices format, 579
  - configuring, 574
  - dBASE devices format, 580
  - formatting data in, 578
  - groups of, 573
  - printer devices format, 579
  - properties, 575
  - reading data, 573
  - SQL devices format, 580
  - using a database device, 581
- DevOpen Cicode function, 581
- DevSetField() function, 581
- DevWrite() function, 582
- dial back, defined, 897
- dial-in modem, defined, 897
- dial-out modem, defined, 897
- Digiboard, defined, 897
- digital alarms, 443, 897
- digital variable (I/O Device), defined, 897
- direction, gradient, 328, 332, 337
- Disabled Alarms button, 868
- disabling object access, 423
- disk I/O devices, 718
  - defined, 897
  - redundant, 721
  - setup, 719
- disk storage, calculating, 513
- display client, defined, 898
- display period, defined, 898
- display, formatting alarm, 476
- displaying
  - alarm pages, 486
  - lists in alarms, 477
  - tables in alarms, 477
- distributed processing, 898
- distributed servers, 898
- dither (imported bitmaps), defined, 898
- domain name server (DNS), defined, 898
- dot notation, defined, 898
- drawing
  - buttons, 350
  - circles, 329
  - ellipses, 328
  - lines, 321, 323

- pipes, 338, 339
  - polygons, 334
  - polylines, 334
  - rectangles, 325
  - squares, 325
- drawing environment, 279
  - options, 282
- driver accreditation, 722
- driver errors, standard, 726
- driver information, 722
- drivers
  - defined, 898
  - generic errors, 724
  - ODBC, 612
  - ODBS, 610
- drivertrace kernel command, 796
- drwFlip\_properties, 318
- DspGetEnv() function, 277
- duplex, defined, 898
- Dynamic Data Exchange (DDE), 589
  - and Office applications, 597
  - connecting to tag database, 594
  - data types, 593
  - posting data, 594
  - reading values from, 596
  - writing values to, 595
- dynamic data exchange (DDE) server, defined, 897
- dynamic data exchange (DDE), defined, 897

## E

- Edit Favorite Colors dialog box, 286
- Edit Item Menu dialog box, 878
- editing
  - project properties, 91
  - using ODBC, 619
- editing menu items, 878
- effects, applying three-dimensional, 373
- elements, array, 212
- ellipse objects, 328
- empty value, defined, 898
- engineering units, 210
- environment
  - drawing, 279
  - graphics pages, 276
- environment variables, 277

- Super Genie, 638
- error messages
  - Find and Replace, 119
- errors
  - generic driver, 724
- ethernet, defined, 898
- event trend/SPC, defined, 899
- events, 493
  - configuring, 493
  - page, 275
  - running, 494
  - times and periods, 495
  - triggers for, 495
- Events Setup page, CSV\_Include project, 870
- Excel macros, 605
- exchanging data, 589, 593
- exit kernel command, 798
- exponential notation, 210
- Export to file button, 863
- Export Variable Tags dialog box, 225
- exporting
  - results, 117
  - tags, 224
  - trend data, 511
- Express Communications Wizard, 745
  - caller ID, 749
  - device selection, 746
  - I/O device address, 747
  - I/O device communications selection, 746
  - I/O device connection schedule, 747
  - I/O device type, 746
  - link to external database, 749
  - serial device, 751
  - server selection, 746
  - summary, 751
  - TCP/IP address, 747
- expression, defined, 898
- expressions (Cicode) in reports, 545
- external data sources, 226
- external databases, using, 606

## F

- FAQs
  - Web Client, 847
- FastLinX for Mitsubishi

- defining variable tag names, 221
  - reserved variable tag names, 221
  - tag browser properties, 220
- FAT32 file system, 247
- field conversion, 232
- fields
  - alarm display, 478
  - alarm summary, 480
  - command, 586
- file names in DDE, 597
- file page template, 863
- file server, defined, 899
- files
  - alternative .ini, 768
  - device history, 583
  - include, 106
  - reconfiguring history, 514
  - report format, 544
  - syslog.dat, 770
  - trend history, 511
  - updating server-client, 766
- fill color, 393, 394, 396, 400
- fill level, 404
- fill property (object), 305
- filters, custom alarm, 436, 896
- Find and Replace dialog box, 113
- finding objects, 318
- finding text, 113
- fixed text
  - alarm displays, 477
  - reports, 544
- fonts
  - custom, 490
  - properties, 488
  - report, 544
- form feed in reports, 546
- format file (import, export, linking), 228
- format files, 229, 237
- formatting
  - alarm display, 476
  - device data, 578
  - numeric variables, 208
- Forward button, 866
- Free Hand Line tool, 321
- freehand line properties, 322

- frequently asked questions
  - Web Client, 847
- full duplex, defined, 899
- functionality limitations, 831

## G

- generic driver errors, 724
- generic protocol, defined, 899
- Genie controller, defined, 899
- Genies, 623, 624
  - and substitutions, 630
  - and Super Genies, 635
  - creating, 625
  - defined, 899
  - opening, 625
  - properties, 629
  - saving, 626
  - substitutions for, 626
  - tag names, 639
  - using, 627
- global Cicode variable, defined, 899
- global client, defined, 899
- Goto Object dialog box, 319
- gradient color, 327, 331, 336
- gradient direction, 328, 332, 337
- graphics bounding box, defined, 899
- graphics objects, hiding, 641
- graphics pages
  - appearance, 272
  - bitmaps, 295
  - color definition, 285
  - creating, 259
  - defaults, 277
  - defined, 899
  - defining colors for, 285
  - environment, 276
  - events, 275
  - keyboard commands for, 274
  - libraries, 292
  - locating, 261
  - opening, 260
  - properties, 270
  - saving, 260
  - screen resolution, 269
  - sizing, 269

- sliders, 414
- symbols, 294
- zooming, 292
- graphics, importing, 301
- graphs, trend, 508
- Grid Setup dialog box, 280
- grid, for alignment, 279
- group (of objects), defined, 900
- grouping
  - objects, 313
  - registers, 689
- groups (areas), 559, 560
- groups (devices), 573
- groups, object, 300
- guidelines (alignment), 280
- Guidelines Setup dialog box, 282

## H

- half duplex, defined, 900
- Hardware Alarms button, 868
- hardware alarms, debugging, 769
- Help button, 867
- help kernel command, 798
- hiding
  - graphics objects, 641
  - objects, 377
- hierarchy, privilege, 556
- histogram, defined, 900
- history
  - device, 583
  - trend, 511
- history files, 514
- History mode, 862
- Home button, 867
- horizontal movement, 379
- hybrid projects, 244

## I

- I/O device address, defined, 900
- I/O device constraints, 737
- I/O device data, errors in, 547
- I/O device properties, 707
- I/O device variable, defined, 900
- I/O device, defined, 900

- I/O devices
  - communication with, 683
  - debugging, 771
  - disk devices, 718
  - properties, 707
  - reading from, 730
  - remote multidrop, 738
  - writing to, 729
- I/O server redundancy, 742
- I/O server, defined, 900
- IFDEF macro, hiding graphics objects, 641
- Import dialog box, 297
- Import Variable Tags dialog box, 219
- importing
  - graphics, 301
  - variable tags, 218, 219
- include files, 106, 900
- Include project
  - securing, 245
- included projects, 106
- Included Projects dialog box, 107
- including projects, 106
- incompatible types error, 759
- incremental compilation, 755
- ini kernel command, 798
- input (keyboard commands), 412
- input property (objects), 306
- Insert Function dialog box, 111
- Insert Tag dialog box, 111
- Insert Trend dialog box, 362
- inserting variable tags, 111
- instant trending, 870
- integer variable (Cicode), defined, 901
- integer variable (I/O device), defined, 901
- Interface
  - Computer Setup Editor Interface, 164
  - Expandable tree pane, 164, 165
  - Help pane, 168
  - Parameter Details pane, 164, 167
  - Parameter Reference pane, 165, 169
  - Parameter status pane, 165, 170
  - Tree search pane, 164
- Internet client setup properties, 768
- Internet display client, 764, 901
- Internet server, 765

interpolation, trend, 509  
 interrupt, defined, 901  
 IP address, defined, 901  
 is, 820

## K

kernel, 785  
 kernel commands, 791  
   cache, 792  
   cicode, 793  
   cls, 794  
   debug, 794  
   drivertrace, 796  
   exit, 798  
   help, 798  
   ini, 798  
   log, 798  
   page driver, 803  
   page general, 799  
   page memory, 806  
   page rdb, 808  
   page table, 807  
   page unit, 808  
   pause, 811  
   shell, 811  
   stats, 812  
   syslog, 812  
 kernel window, 786, 787  
 keyboard commands, 274, 411, 901  
 knowledge base, defined, 901  
 kurtosis, defined, 901

## L

labels, 197, 565  
   arguments in, 567  
   defining, 568  
   for areas, 558  
 language database, defined, 901  
 language databases, 644  
 languages  
   changing, 643  
   changing at runtime, 646  
   logging in different, 646  
   multiple, 645

  Web Client, 843  
 Last Alarms panel, 868  
 layout, format file, 229  
 level, fill, 404  
 libraries, 292  
 limitations  
   Cicode functions, 831  
   functionality, 831  
 line objects, 300  
 link, defined, 901  
 linked symbols, 293  
 linked tags, refreshing, 217  
 linking  
   tags, 215  
   templates, 266  
 links, breaking, 216  
 lists, displaying in an alarm, 477  
 local area network (LAN), defined, 901  
 local Cicode variable, defined, 902  
 local language, 643, 902  
 locating objects, 318  
 locking objects, 312  
 log kernel command, 798  
 Login button, 867  
 login records, 551  
 LogRead field, 710  
 LogWrite field, 710  
 long BCD variable (I/O device), defined, 902  
 long file names (with DDE), 597  
 long variable (I/O Device), defined, 902  
 loop-back test, 701  
 low and low low alarms, defined, 902  
 lower control limit, 520  
 lower specification limit, 521

## M

macros, 605  
 management  
   print, 586  
 manager client, defined, 902  
 manipulating objects, 309  
 manual configuration, 703  
 maximum request length, defined, 902  
 menu configuration tool, 875  
 menus, 877

- creating, 875
- methods, storage, 513
- Migration Tool, 33
- millisecond trending, defined, 902
- mimic, defined, 902
- mirroring objects, 318
- Misc.zip file, 832
- mode, demo, 784
- modems, 732, 733
- module Cicode variable, defined, 902
- monitoring
  - processes, 819
  - startup, 818
- Monitors parameter, 871
- movement, 378
  - horizontal, 379
  - of objects, 311
  - properties, 303
  - vertical, 381
- multi-digital alarms, 447, 902
- multidrop remote I/O devices, 738
- multi-dropping, 737
- multi-language projects, 643
  - , 871, 872
- multiple
  - languages, 645
  - projects, 646
- multiple language support for Web Client, 843
  - non-default languages, 844
- multiple monitors, using, 871
- multi-process system
  - running independent servers, 764

## N

- native language, 643, 903
- navigation toolbar, 866
- nesting Super Genies, 638
- network
  - defined, 903
- network computer, defined, 903
- network DDE, 598, 603, 903
- network DDE shared application, 604
- network DDE shares, 600
- Network Page button, 867
- networked system, restarting, 780

- New Project dialog box, 90
- New Style dialog box, 267
- Next button, 867
- nodes, defined, 903
- normal distribution, defined, 903
- normal page template, 858
- NTFS file system, 247
- null value, defined, 903
- numbers (objects), 349
- numeric variables, 208

## O

- object alignment, 279, 280, 316
- object ID (OID), defined, 904
- object properties, 301
  - disable access, 423
- object types, 321
  - ActiveX, 367
  - buttons, 350
  - Cicode, 362
  - ellipse, 328
  - free hand line, 321
  - numbers, 349
  - pasted symbol, 364
  - polygons, 334
  - rectangles, 325
  - straight line, 323
  - symbol sets, 352
  - text, 340
  - trends, 359
- object variable (Cicode), defined, 903
- objects, 299
  - 3D effects, 374
  - access, 419, 420, 422
  - access property, 308
  - aligning, 316
  - appearance, 302
  - applying effects, 373
  - copying/pasting, 313
  - defined, 903
  - deleting, 312
  - fill color, 393, 394, 396, 400
  - fill level, 404
  - fill property, 305
  - grouping, 313

- groups, 300
- horizontal movement property, 379
- input property, 306
- keyboard commands, 412
- line, 300
- locating, 318
- locking/unlocking, 312
- manipulating, 309
- mirroring, 318
- movement properties, 303
- moving, 311, 378
- pipes, 339
- rectangles, 326
- reshaping, 300
- resizing, 311
- rotating, 317
- rotational property, 417
- scaling, 304, 386, 390
- sliders, 307, 415, 416
- touch commands for, 408
- touch properties, 409
- vertical movement, 381
- visibility, 377, 378
- objects types
  - pipes, 338
- objects, library, 292
- occurrence section in tag names, 199
- ODBC drivers, 610, 612
- ODBC server, using Citect as, 616
- OEM character sets, 647
- OID validation, 723
- OLE objects in reports, 544
- OPC Data Access Server Tag Browser, 223
- open bracket expected error, 760
- open database connectivity (ODBC), defined, 903
- Open/Save As dialog box, 264
- opening
  - Genies, 625
- options, 282

## P

- pack, defined, 904
- page driver kernel command, 803
- page environment variable, defined, 904
- page events, 275
- page general kernel command, 799
- page memory kernel command, 806
- Page Properties dialog box, 270
- page rdb kernel command, 808
- Page recompiled against different Variable.DBF, 723
- page style, graphics, 265
- page table kernel command, 807
- page templates, 265
- page unit kernel command, 808
- pages, 872
  - alarm, 486
  - displaying alarm, 486
  - startup, 268
- pages, preconfigured, 857
- parameter queries, 620
- Parameters
  - Adding a parameter, 178
  - Deleting a parameter, 182
  - Disabling a parameter, 183
  - Editing a parameter, 181
  - Enabling a parameter, 183
  - Overview, 162
  - Viewing undocumented parameters, 184
- Parent Page button, 867
- Pareto charts, 521
- parity, defined, 904
- Paste to clipboard button, 863
- pasted symbol objects, 364
- pasting objects, 313
- path definitions, 515
- path substitution, 514
- pause kernel command, 811
- performance, 688, 813
- periodic trend, defined, 904
- periods, specifying for events, 495
- Permission Entry dialog, 249
- persistence cache, defined, 904
- pipe arrangements, 339
- pipe objects, 338
- PLC interface board, defined, 904
- Plot trend button, 863
- plots, in reports, 546
- point limit, defined, 904
- polygon objects, 334
- polygon properties, 335

- polylines, drawing, 334
  - port(s), defined, 904
  - posting data using DDE, 594
  - preconfigured pages, 857
  - prefixes in tag names, 198
  - preparing a project for Web Client deployment, 831
  - Previous button, 867
  - primary alarms server, defined, 904
  - primary reports server, defined, 904
  - primary trends server, defined, 905
  - print management, 586
  - Print Page button, 867
  - printer devices, 571
    - format, 579
  - printing
    - project details, 95
    - trend data, 510
  - privilege hierarchy, 556
  - privilege/area combinations, 562
  - privileged user, 869
  - privileges, using areas with, 561
  - process capability, 519
  - process variation, 517
  - processes, monitoring, 819
  - Project Properties dialog box, 91
  - project restoration and security, 551
  - Project tag mismatch detected, 723
  - project upgrades, read-only, 253
  - projects
    - backing up, 97
    - compiling, 753
    - copying, 94
    - deleting, 96
    - editing properties, 91
    - hybrid, 244
    - including, 106
    - multi-language, 643
    - multiple, 646
    - printing details, 95
    - read-only, 247
    - restoring, 104
  - projects, creating, 869
  - properties
    - 3D effects, 374
    - access, 420
    - ellipses, 330
    - fill level, 404
    - font, 488
    - freehand line, 322
    - Genies, 629
    - movement, 378
    - object, 301
    - object visibility, 378
    - page, 270
    - pipe, 339
    - polygon, 335
    - rectangles, 326
    - rotational, 417
    - scaling, 386, 390
    - slider, 415
    - straight lines, 324
    - text, 340, 342, 343, 345, 347
    - touch, 409
    - variable tags, 204
    - vertical movement, 381
    - writing to alarm, 484
  - properties, object, defined, 904
  - proprietary board drivers, debugging, 779
  - proprietary boards, 700
  - protection, software, 783
  - protocol drivers
    - debugging, 777
  - protocol, defined, 905
  - protocols, debugging, 771
  - proxi/proxy server, defined, 905
- ## Q
- queries, parameter, 620
- ## R
- range markers, 861
  - rate of change alarms, defined, 905
  - reading from I/O devices, 730
  - read-only projects, 247
    - effects on project, 251
  - real variable (Cicode), defined, 905
  - real variable (I/O device), defined, 905
  - record name, defined, 905
  - rectangle objects, 325, 326



- redundancy, 905
  - disk I/O devices, 721
- refreshing linked tags, 217
- registers, grouping, 689
- Remapping
  - properties, 693
- remapping
  - variables, 691
- remote communications, defined, 905
- remote terminal, defined, 906
- replacing results, 118
- replacing text, 113, 114
- report
  - defined, 906
- report format file, 544
- report format file, defined, 906
- reports, 539
  - Cicode, 545
  - comments in, 546
  - communication errors, 547
  - configuring, 539
  - expressions and variables, 545
  - fixed text, 544
  - fonts, 544
  - form feed, 546
  - I/O device data errors, 547
  - OLE objects in, 544
  - plots, 546
  - report format file, 544
  - running, 542
  - suppressing, 548
  - trend data, 546
  - trend graphs, 546
  - triggers for, 544
- reports server, defined, 906
- reserved words, defined, 906
- reshaping objects, 300
- resizing objects, 311
- restarting the system, 779
- restoring projects, 104
- results
  - exporting, 117
  - replacing, 118
- results list (Find and Replace), 116
- RJ11, defined, 906

- RJ12, defined, 906
- RJ45, defined, 906
- Rotate dialog box, 318
- rotating objects, 317
- rotational property, 417
- rows, deleting, 620
- RS232/485 converter, 714
- RS-232C standard, defined, 906
- RS-422 (or EIA-422), 716
- RS-422, defined, 906
- RS-485, 717, 906
- running
  - a system, 763
  - events, 494
  - reports, 542
- running independent servers, 764
- runtime
  - changing languages at, 646
  - displaying alarm pages at, 486
  - handling alarms at, 485
- runtime configuration, 763, 765
  - running individual processes, 764
- runtime information, 812
- runtime security, 255
- runtime system
  - debugging, 769
- runtime system, defined, 906

## S

- sample period, 503
- saving Genies, 626
- scalable architecture, defined, 906
- scale defaults, 863
- scale markers, 861
- scaling objects, 304, 386, 390
- scan time, page, 271
- schedule period, defined, 907
- schedules, specifying, 728
- scheduling
  - communications, 728
- screen resolution and graphics pages, 269
- search coverage, specifying, 114
- searches, troubleshooting, 123
- searching text, 114
- Sections

- Adding a section, 174
- Deleting a section, 175
- Disabling a section, 176
- Enabling a section, 177
- Maintaining Sections, 174
- securing
  - include projects, 245
  - top-level projects, 244
- security, 551
  - include projects, 245
  - MS Office, 597
  - read-only projects, 247
  - runtime, 255
  - specifying requirements for, 561
  - top-level projects, 244
  - using areas for, 558
- Send Backwards command, 314
- Send to Back command, 314
- serial boards, 698
- serial communications, 695, 907
- serial communication standards, 715
- serial port loop-back cable, 702
- serial port loop-back test, 701
- server redirection, 770
- server, defined, 907
- server-client file updates, 766
- services, starting network DDE, 598
- Set span button, 862
- SetLanguage() function, 643, 646
- shares, DDE, 601
- shell kernel command, 811
- shortform notation, 211
- simplex transmission, defined, 907
- size, calculating array, 212
- sizing
  - graphics pages, 269
  - objects by scaling, 386
- skewness, defined, 907
- slider control, defined, 907
- slider property, 307
- sliders, 414, 417
- Snap to Grid command, 279
- soft PLC, defined, 907
- software protection, 783, 907
- span markers, 861
- specifying
  - schedule, 728
  - search coverage, 114
- SQL database devices, 571
  - format, 580
- SQL databases, 607
- SQLExec() function, 608
- squares, drawing, 325
- standard driver errors, 726
- standards, common serial communication, 715
- standby alarms server, defined, 907
- standby reports server, defined, 907
- standby trends server, defined, 907
- starting network DDE services, 598
- startup configuration, 763, 765
- startup page, 268
- startup, monitoring, 818
- StartupPagen parameter, 872
- statistical control, 518
- statistical process control (SPC), 517
- stats kernel command, 812
- stop bits, defined, 907
- storage method, trend data, 513
- storage, calculating disk, 513
- straight line objects, 323
- straight line properties, 324
- string arrays, 212
- string substitution, 568
- structured query language (SQL), 607
  - commands, 608
  - times and dates, 610
- structured tag names, 198, 639
- substitutions
  - defining for Super Genies, 633
  - path, 514
  - string, 568
- summary display, alarms, 481
- Super Genies, 623
  - and Genies, 635
  - areas, 638
  - arrays and, 635
  - constants and, 635
  - environment variables, 638
  - nesting, 638
  - substitutions for, 633

- tag names, 639
  - using, 630
- support, 13
- suppressing reports, 548
- Swap Color dialog box, 289
- symbol set objects, 352
- symbols, 294, 907
- syntax, ODBC, 613
- syslog kernel command, 812
- syslog.dat file, 770
- system
  - running a, 763
- system tuning, 813

**T**

- tables, displaying, in alarms, 477
- tag
  - naming, 197
- tag browser
  - FastLinux for Mitsubishi, 220
  - OPC Data Access Server, 223
- tag database, connecting to, 594
- tag name syntax, 198
- tag names, 199
- tag-based driver data validation, 723
- tags
  - exporting, 224
  - importing, 218
  - linking, 215
  - names for, 198, 199, 200
  - supported alarm properties, 482
  - syntax for names, 198
  - unused, 110
  - using alarm properties as, 482
- task, defined, 908
- TCP/IP
  - special options for, 697
- TCP/IP drivers, debugging, 776
- technical support, 13
- templates, 90, 908
  - Admin Tools page, 864
  - alarm page, 858
  - creating, 266, 267
  - CSV\_Include, 857
  - file page, 863

- for graphics pages, 263, 265
  - Genie substitutions in, 630
  - linking, 266
  - normal page, 858
  - opening, 266
  - trend page, 859
- text
  - finding and replacing, 113
  - in alarm displays, 477
  - marking for language change, 643
- text box, defined, 908
- text objects, 340
- text properties, 340, 342, 343, 345, 347
- thread, defined, 908
- three-dimensional (3D) effects, 373, 374
- time conversions, 622
- time server, defined, 908
- time stamped analog alarms, 908
- time stamped digital alarms, 908
- timeout, defined, 908
- times
  - in SQL, 610
  - specifying for events, 495
- time-stamped alarms, 453, 908
- time-stamped analog alarms, 470
- time-stamped digital alarms, 466
- tool tip, defined, 908
- toolbar
  - Alarms, 868
  - navigation, 866
- toolbars, common, 865
- Tools button, 867
- top-level projects, 244
- touch (object at runtime), defined, 908
- touch commands, 408, 908
- touch properties, 409
- transactions, database, 609
- trend cursor, 862
- trend data, report, 546
- trend graphs, 546
- Trend group folder, 863
- trend history files, 511
- trend line, defined, 909
- trend objects, 359
- trend page template, 859

- trend plot, defined, 909
- trend tag name, 197
- trend tags
  - configuring, 502
  - properties, 502
- trend, defined, 909
- trend, insert, 362
- trending data, 501
- trending, instant, 870
- trends
  - creating pages for, 509
  - exporting data for, 511
  - graphs of, 508
  - interpolation, 509
  - printing data, 510
  - storage methods, 513
- trends group, 880
- Trends Page button, 867
- trends server, 909
- triggers
  - and reports, 544
  - event, 495
- TrnGetTable() function, 546
- TrnPlot() function, 546
- troubleshooting communications, 743
- troubleshooting searches in projects, 123
- tuning
  - cache, 813
  - system, 813

**U**

- unlinked symbols, 293
- unlocking objects, 312
- unsigned integer variable (I/O device), defined, 909
- unsupported functionality, 831
- unused tags, 110
- updating
  - server-client files, 766
- upper control limit, 520
- upper specification limit, 521
- Use Template dialog box, 263
- user privileges, 553, 554
- user records, adding, 551, 552
- user, privileged, 869
- Users dialog box, 552

- using
  - external databases, 606
  - Genies, 627
  - serial board, 698
  - Super Genies, 630
  - symbols, 294

## V

- validating tag-based driver data, 723
- values
  - argument, 567
  - reading from DDE application, 596
  - string conversion, 567
  - writing to DDE application, 595
- variable data, displaying in alarms, 477
- variable tag name, 197
- variable tags
  - assigning, 197
  - configuring, 203
  - data types, 204
  - importing, 219
  - inserting into field, 111
  - properties, 204
- variable type (Cicode), defined, 909
- variables
  - environment, 277
  - in reports, 545
  - remapping, 691
- variation, process, 517
- vertical movement, 381
- vertical sliders properties, 416
- viewing
  - Include project, 108
  - plant areas, 562
- virtual, defined, 909
- visibility, 377, 378

## W

- Web Client, 825
  - ActiveX.zip file, 832
  - configuring a deployment, 835
  - creating a deployment, 836
  - default languages, 843
  - deleting a deployment, 842

- deploying a project, 838
- displaying a deployment, 840
- editing a deployment, 841
- frequently asked questions, 847
- functionality limitations, 831
- getting started, 827
- implementing non-default languages, 844
- introduction, 825
- Misc.zip file, 832
- multiple language support, 843
- preparing a project for deployment, 831
- preparing user files for delivery, 832
- setting up a system, 827
- system architecture, 825
- updating a deployment, 842
- using a different language to the locale setting, 844

- Web Deployment Preparation tool, 833
- Web Deployment Preparation tool, 833
- Windows user security, 250
- wiring cables, 713
- wizard, defined, 909
- writing
  - to alarm properties, 484
  - to I/O devices, 729

## X

- XP style buttons, 351
- XRS control charts, 519

## Z

- Zoom buttons, 862
- zooming, 292

