

CitectSCADA CitectHMI

User Guide

Citect Pty. Limited

3 Fitzsimons Lane
PO Box 174
Pymble NSW 2073
Australia

Telephone: 61 2 9496 7300
Fax: 61 2 9496 7399

DISCLAIMER

Citect Corporation 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 Corporation reserves the right to revise this publication at any time without incurring an obligation to notify any person of the revision.

COPYRIGHT

© Copyright 2004 Citect Corporation. All rights reserved.

TRADEMARKS

Citect Pty. Limited has made every effort to supply trademark information about company names, products and services mentioned in this manual. Trademarks shown below were derived from various sources.

Citect, CitectHMI, and CitectSCADA are registered trademarks of Citect Corporation.

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

MS-DOS, Windows, Windows 95, Windows NT, Windows 98, Windows 2000, Windows for Workgroups, LAN Manager, Microsoft Windows XP, Excel and MSMAIL are trademarks of Microsoft Corporation.

DigiBoard, PC/Xi and Com/Xi are trademarks of DigiBoard.

Novell, Netware and Netware Lite are registered trademarks of Novell Inc.

dBASE is a trademark of Borland Inc.

GENERAL NOTICE

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

<insert month and year> edition for CitectSCADA Version 6.0

Manual Revision Version 6.0.

Printed in Australia.

Contents

Introducing CitectSCADA

Using CitectSCADA	1
Configuring CitectSCADA	2
The CitectSCADA Environment	2
Configuration environment	2
Runtime system	2

Chapter 1

Understanding the CitectSCADA Environment

Citect Explorer	4
Project Editor	5
Citect Graphics Builder	7
Cicode Editor	7
CitectSCADA Help	8

Chapter 2

Configuring CitectSCADA Projects

Configuring a Project	10
Project Design Considerations	10
Project Design Standards	11
The CitectSCADA Database Structure	12
Using Other Database Editors	12
Special considerations for Microsoft Excel	13
Creating New Projects	16
New Project dialog properties	17
Working with Existing Projects	18
Linking and Unlinking Projects	19
Editing Project Properties	19
Project Properties dialog box	20
Linking to CitectSCADA Projects (on the same network)	22
Including Projects	23
Included Projects dialog box	24
The CitectSCADA Include project	25
Backing Up Projects (Archiving)	25
Backup Project dialog properties	26
Backup/Restore – Password Encryption dialog properties	27
Restoring Projects	28
Restore Project dialog properties	28

Using the Backup Utility from the Command Line	29
CtBackup and CtBackup32 utility command line options	30
Copying Projects	31
Copy Project dialog properties	32
Printing Project Details	33
Print (project details) dialog properties	34
Project Editor Options	35
Project Editor Options Dialog Properties	35
Insert Tag dialog box	37
Insert Function dialog box	37
Find User Function dialog properties	38

Chapter 3

Tagging Process Variables

Configuring variable tags	39
Variable Tag Properties	41
Formatting numeric variables	44
Using arrays	47
Using structured tag names	51
Linking, Importing, and Exporting Tags	54
Linking tags	54
Refresh Linked Tags properties	56
Importing tags	56
Import variable tags properties	58
FastLinx for Mitsubishi Tag Browser Properties	59
Defining Variable Tag Names for Mitsubishi FastLinx	60
Reserved Names for Mitsubishi FastLinx Variable Tags	60
OPC Data Access Server Tag Browser Properties	78
Exporting tags	79
Export Variable Tags properties	80
External data source	80
Format file	82
Format file layout	83
Field conversion	87
Having CitectSCADA recognize format files	92

Chapter 4

Understanding Object Types

Using Free Hand Line Objects	95
Freehand Line Properties - Appearance (General)	96
Using Straight Line Objects	97
Straight Line Properties - Appearance(General)	98
Using Rectangle Objects	99
Rectangle Properties - Appearance (General)	100
Using Ellipse Objects	101

Ellipse Properties - Appearance (General)	103
Using Polygon Objects	106
Polygon Properties - Appearance (General)	107
Using Pipe Objects	109
Pipe Properties - Appearance (General)	110
Using Text Objects	111
Text Properties - Appearance (General)	112
Text Properties - Appearance Display Value (On/Off)	113
Text Properties - Appearance Display Value (Multi-state)	114
Text Properties - Appearance Display Value (Array)	116
Text Properties - Appearance Display Value (Numeric)	118
Text Properties - Appearance Display Value (String)	119
Using Button Objects	120
Button Properties - Appearance (General)	121
Using Symbol Set Objects	122
Symbol Set Properties - Appearance General (On/Off)	123
Symbol Set Properties - Appearance General (Multi-state)	124
Symbol Set Properties - Appearance General (Array)	126
Symbol Set Properties - Appearance General (Animated)	128
Using Trend Objects	129
Trend properties	130
Insert Trend dialog box	132
Using Cicode Objects	132
Cicode Object Properties - Cicode (General)	133
Using animation points	134
Using Pasted Symbol Objects	134
Paste Symbol dialog box	135
Symbol Properties - Appearance (General)	136
Using Pasted Genie Objects	136
Using ActiveX Objects	137
ActiveX Object Properties	137
Tag Association	138
ActiveX Object Properties - Appearance (Tag Association)	138
Object Identification	141
Object Properties - Access (Identification)	141

Chapter 5

Defining Common Object Properties

3D Effects	143
Object Properties - Appearance (3D Effects)	144
Visibility	147
Object Properties - Appearance (Visibility)	148
Movement	148
Object Properties - Movement (Horizontal)	149
Object Properties - Movement (Vertical)	151

Object Properties - Movement (Rotational)	152
Group and object movement - examples	154
Scaling	156
Object Properties - Scaling (Horizontal)	157
Object Properties - Scaling (Vertical)	160
Fill Color	163
Object Properties - Fill Color (On/Off)	164
Object Properties - Fill Color (Multi-state)	166
Object Properties - Fill Color (Array)	167
Object Properties - Fill Color (Threshold)	169
Object Properties - Fill Color (Gradient)	171
Fill Level	173
Object Properties - Fill (Level)	174
Touch Commands	177
Object Properties - Input (Touch)	178
Keyboard Commands	180
Object Properties - Input (Keyboard Commands)	181
Sliders	183
Object Properties - Slider (Horizontal)	184
Object Properties - Slider (Vertical)	185
Object Properties - Slider (Rotational)	186
Access	188
General Access to Objects	188
Object Properties - Access (General)	189
Disable Access to Objects	191
Object Properties - Access (Disable)	191

Chapter 6

Defining Commands and Controls

Defining Commands and Controls	193
System Keyboard Commands	194
System keyboard command properties	195
Keyboard Keys	196
Keyboard keys properties	196
Keyboards	197
Defining Key Sequences for Commands	198
Using a hot key	199
Using variable data input	199
Passing multiple arguments	201
Passing keyboard arguments to functions	201

Chapter 7

Configuring and Processing Alarms

Configured alarms	203
Using alarm delay	204

Using custom alarm filters	204
Alarm Categories	207
Alarm Category Properties	208
Digital Alarms	212
Digital Alarm Properties	212
Multi-digital Alarms	215
Multi-digital Alarm Properties	216
Time-stamped Alarms	220
Time-stamped Alarm Properties	221
Analog Alarms	224
Analog Alarm Properties	224
Advanced Alarms	229
Advanced Alarm Properties	229
Time-stamped Digital Alarms	231
Time-stamped Digital Alarm Properties	231
Time-stamped Analog Alarms	234
Time-stamped Analog Alarm Properties	235
Formatting an Alarm Display	239
Including CitectSCADA data	239
Including fixed text	240
Displaying lists and tables	240
Variable data in alarm messages	240
Alarm display fields	241
Alarm summary fields	243
Changing the Order of the Alarm Summary Display	245
Using Alarm Properties as Tags	245
Supported alarm properties	245
Writing to alarm properties	247
Setting up alarms	248
Handling Alarms at Runtime	248
Using CitectSCADA Fonts	250
Fonts properties	251

Chapter 8

Configuring Events

Events Properties	255
Running Events	257
Specifying times and periods	257
Using triggers	258

Chapter 9

Using Accumulators

Accumulator Properties	260
----------------------------------	-----

Chapter 10	Logging and Trending Data	
	Trending Data	263
	Configuring trend tags	264
	Trend Tag Properties	264
	Trend Graphs	269
	Creating trend pages	270
	Trend interpolation	271
	Printing Trend Data	272
	Exporting Trend Data	272
	Using Trend History Files	273
	Storage method	274
	Calculating disk storage	275
	Reconfiguring history files	276
	Using Path Substitution	276
	Default path definitions	276
	Debugging Trending	277
 Chapter 11	 Using Objects	
	Using groups	280
	Reshaping objects	280
	Using bitmaps	281
	Importing graphics	281
	Object Properties	281
	Appearance	282
	Movement	283
	Scaling	284
	Fill	285
	Input	286
	Slider	287
	Access	288
	Manipulating Objects	289
	Selecting objects	290
	Moving objects	291
	Resizing objects	291
	Deleting objects	292
	Locking/unlocking objects	293
	Grouping objects	293
	Copying and pasting objects	294
	Send to Back and Send Backwards	294
	Bring to Front and Bring Forwards	295
	Aligning objects	296
	Align dialog box	296
	Rotating objects	297
	Rotate dialog box	298

Mirroring objects	298
Mirror dialog box	298
Locate an object	298
Goto Object dialog box	299

Chapter 12 Understanding Statistical Process Control

Process variation	301
Statistical control	302
Process capability	303
XRS control charts	303
Capability charts	305
Pareto charts	305
Using Statistical Process Control (SPC) with CitectSCADA	306
SPC Tags	306
SPC tag properties	307
SPC Control Charts	312
XRS control chart	312
Configuring XRS charts	313
Capability charts	313
Configuring capability charts	313
Pareto Charts	313
Configuring Pareto charts	313
SPC Alarms	314
SPC Formulas and Constants	315
Control Chart Line Constants	320

Chapter 13 Defining and Drawing Graphics Pages

Creating a New Graphics Page	323
New Dialog Box	323
Working with pages	323
Use Template (new page/template) dialog box	326
Open/Save As dialog box	327
Find dialog box	328
Using Page Templates	328
Choosing a page style	329
Linking templates	329
Creating your own templates	329
New Style dialog box	330
Using a Browse Sequence	330
Specifying a Startup Page	331
Sizing the Page	332
Page (screen) resolution	332
Page size at runtime	333

Defining Page Properties	333
Page Properties - General	334
Page Properties - Appearance	336
Page properties - Keyboard Commands	337
Page Properties - Events	339
Page Properties - Environment	340
Setting Default Page Settings	341
Page defaults	341
Understanding the Drawing Environment	342
Grids	342
Grid Setup dialog box	343
Guidelines	343
Guidelines Setup dialog box	345
Options	345
Options dialog box	346
Colors	348
Edit Favorite Colors dialog box	349
Swap Color dialog box	352
Adjust colors dialog box	353
Zooming	355
Using libraries	356
Using symbols	357
Bitmaps	358
Import dialog box	360

Chapter 14

Reporting Information

Configuring reports	363
Reports dialog box	364
Running Reports	366
Running a report on startup	366
Specifying times and periods	366
Using triggers	367
Using commands	367
Report Format File	368
Report example	370
Handling Communication Errors in Reports	371
Reporting errors in I/O device data	371
Suppressing reports	372

Chapter 15

Using Security

Maintaining user records	373
Adding user records	374
User properties	374

Defining User Privileges	376
Using hierarchical privilege	378
Defining Areas	378
Using areas for security	380
Using labels to name areas	380
Using groups of areas	381
Groups properties	382
Using areas with privileges	383
Specifying security requirements	383
Viewing areas of the plant	384

Chapter 16 Using Labels

Using Arguments in Labels	389
Converting Values into Strings	389
Substituting Strings	390
Defining Labels	390

Chapter 17 Using Devices

Using groups of devices	395
Using devices to read data	396
Configuring Devices	396
Devices Properties	397
Formatting Data in the Device	400
Printer and ASCII devices format	401
dBASE and SQL database devices format	402
Using a database device	403
Using Device History Files	406
Using Command Fields	409
About Print Management	409

Chapter 18 Exchanging Data with Other Applications

Using DDE (Dynamic Data Exchange)	411
DDE conversations and client syntax	412
Setting up DDE conversations	413
CitectSCADA DDE function types	415
Exchanging CitectSCADA data via DDE	415
Connecting to the CitectSCADA tag database using DDE	416
Posting select CitectSCADA data using DDE	416
Writing values to a DDE application	417
Reading values from a DDE application	418
Using DDE with Microsoft Office applications	419
Network DDE	420
Starting network DDE services	420

Setting up network DDE shares	422
DDE Shares	424
Using DDE Trusted Shares	425
Using network DDE	426
Connecting to a network DDE shared application	426
Using the Citect Tags Excel macros	427
Using External Databases	429
dBASE databases	429
SQL databases	429
Using Structured Query Language	430
Connecting to an SQL database	430
Executing SQL commands	431
Using a transaction	432
Expressing dates and times in SQL	432
Using ODBC drivers	433
Installing the ODBC driver	434
About the ODBC driver	435
Setting up ODBC	435
Getting the correct syntax with ODBC	436
Programming style with ODBC	436
Comparing DDE with ODBC	437
ODBC compatibility	438
Using CitectSCADA as an ODBC server	439
Reading data from an access table with ODBC	441
Appending data with ODBC	442
Editing data with ODBC	442
Deleting rows from an Access table	443
Calling action queries with ODBC	443
Parameter queries	444
Access and Cicode date/time conversions	445

Chapter 19

Using Genies and Super Genies

Understanding Genies	448
Creating Genies	449
Opening a Genie	449
Saving a Genie	450
Defining Substitutions for Genies	450
Using Genies	451
Paste Genie dialog box	453
Genies properties	453
Using Genie Substitutions in Templates	454
Using Super Genies	454
Defining Substitutions for Super Genies	457
Using Super Genies without Genies	458

Using Constants and Arrays with Super Genies	459
Creating a Genie controller	461
Attach Super Genie dialog box	461
Select Super Genie dialog box	462
Nesting Super Genies	462
Super Genie areas	462
Super Genie environment variables	462
Using Structured Tag Names with Genies and Super Genies	463
Using structured tags with Genies	463
Using structured tags with Super Genies	464
Hiding Graphics Objects	465
IFDEF format	465

Chapter 20

Working with Multi-Language Projects

Changing Languages	467
Marking text for language change	467
Language databases	468
Multiple languages	469
Multiple projects	470
Changing languages at runtime	470
Logging data in different languages	471
ASCII and ANSI character sets	471
OEM character sets	472

Chapter 21

Using the Computer Setup Wizard

Computer Setup Wizard - introduction	474
Computer Setup Wizard flow diagram	475
Computer Setup Wizard - Computer role	475
Computer Setup Wizard - Project	476
Computer Setup Wizard - I/O Server	476
Computer Setup Wizard - Internet server	476
Computer Setup Wizard - Alarm	476
Computer Setup Wizard - Advanced Alarms	476
Computer Setup Wizard - Reports	477
Computer Setup Wizard - Advanced reports	478
Computer Setup Wizard - Trends	478
Computer Setup Wizard - Advanced trends	478
Computer Setup Wizard - Server	478
Computer Setup Wizard - Network	479
Computer Setup Wizard - Events	479
Computer Setup Wizard - Time	480
Computer Setup Wizard - Menu security	480
Computer Setup Wizard - Keyboard security	481

Computer Setup Wizard - Miscellaneous security	481
Computer Setup Wizard - General options setup	481
Computer Setup Wizard Finish	482

Chapter 22

Communicating with I/O Devices

How CitectSCADA Communicates with I/O Devices	483
How CitectSCADA reads from the I/O device	485
How CitectSCADA writes to the I/O device	486
Performance Considerations	488
Caching data	488
Grouping registers	489
Remapping variables in an I/O device	491
Remapping properties	493
Remapping example	494
Serial Communications	495
Using a COM port	495
COMX driver special options reference	495
TCP/IP driver special options reference	497
Using a Serial Board	498
Serial board setup	499
Using Digiboards with Windows	499
Using Proprietary Boards	500
Proprietary board setup	500
Serial Port Loop-Back Test	501
Setting Up Communications	503
Manually Configuring Communications	503
I/O Server Properties	504
Boards Properties	504
Ports Properties	505
I/O Devices Properties	507
Wiring Cables	512
Common Serial Communication Standards	514
RS-232C (or EIA-232C or RS-232)	514
RS-422 (or EIA-422)	515
RS-485 (or EIA-485)	516
Using a Transparent I/O Device	517
Configuring transparent devices	518
Citect Driver Accreditation	518
Advanced Driver Information	519
Variable (Digital) Limitations	519
Validating distributed project data for tag-based drivers	520
Generic driver errors	520
Standard driver errors	523
Scheduled Communications	526

Specifying a schedule	526
Writing to the scheduled I/O device.	527
Reading from the scheduled I/O device	527
Communicating with Diallable Remote I/O Devices (Windows NT/2000 Only) . . .	529
Modems at the I/O server	530
Modems at the I/O device	531
Example configurations for modems at the I/O Server	531
I/O device constraints for multi-dropping	535
Configuring multidrop remote I/O devices	536
Modems Properties	540
I/O server redundancy for diallable remote I/O devices	540
Trouble-shooting diallable remote I/O device communications	541
Alternative (backward compatibility) method of permanent connection . . .	542

Chapter 23 Using Memory and Disk I/O Devices

Memory I/O Devices	545
Memory I/O Device Setup	546
Disk I/O devices	547
Disk I/O device setup	547
Redundant Disk I/O Devices	549

Chapter 24 Using the Communications Express Wizard

Express Communications Wizard - introduction	551
Express Communications Wizard - Server selection.	552
Express Communications Wizard - Device selection	552
Express Communications Wizard - I/O device type	552
Express Communications Wizard - I/O device communications selection . .	552
Express Communications Wizard - TCP/IP address.	552
Express Communications Wizard - I/O device address	553
Express Communications Wizard - I/O device connection schedule	553
Caller ID and commands (Windows NT only)	554
Express Communications Wizard - Link to external database	555
Express Communications Wizard - Serial device	556
Express Communications Wizard - Summary	557

Chapter 25 Building Your Citect Project

Compiling the Project	559
Incremental compilation.	560
Debugging the compilation	560
Compile Error Properties.	561
Compile Error Messages.	562
Running the System	568
Startup and runtime configuration	568

Running Your System Over the Internet	569
CitectSCADA Internet Display Client.	569
CitectSCADA Internet server.	570
Startup and runtime configuration	570
Server - client file updates.	570
Citect Internet Client setup properties	572
Providing Runtime Security	573
Running CitectSCADA as a service under NT	573
Running CitectSCADA as the shell under NT	573
Disabling Windows keyboard commands	573
Disabling control menu commands	574
Removing the Cancel button	574
Using an Alternative INI File	574
Debugging the Runtime System	575
Hardware alarms	575
SysLog.DAT file.	575
Debugging I/O Devices and Protocols	576
Creating a communications test project	576
Debugging a COMx driver.	578
Debugging a TCP/IP driver	581
Debugging a protocol driver using serial communications	582
Debugging proprietary board drivers.	584
Contacting Citect support	584
Restarting the System Online	584
Restarting a networked system online.	585
CitactSCADA Software Protection	588
Updating your hardware key	588
CiUSAFE dialog properties	589
Citact license point count.	590
Demo mode.	590
Using the CitactSCADA Kernel	591
Displaying the kernel window	591
Inside the kernel	592
Using Kernel Commands.	596
Kernel commands	597
Gathering Runtime Information	597
System tuning	598

Chapter 26

Using CitactSCADA on a Network

Setting up a Network	602
Using TCP/IP for network communications	606
Using Distributed Processing	607
Splitting the processing load for multiple I/O servers	610
Using Distributed Servers	610

Switching between clusters	612
Configuring projects for distributed servers	612
Configuring CitectSCADA to communicate over a WAN	613
Building Redundancy into Your System	615
I/O server redundancy	615
Redundancy and persistence	617
Data path redundancy	618
Alarms, reports, and trends server redundancy	621
How CitectSCADA handles alarms server redundancy	622
How CitectSCADA handles reports server redundancy	622
How CitectSCADA handles trends server redundancy	623
How CitectSCADA handles file server redundancy	623
How CitectSCADA handles FTP server redundancy	624
LAN redundancy	625
NetBIOS Errors	626
CiNet	628

Appendix A Parameters

Using Parameters	629
Rules for using parameters	629
Using parameters on a network	630
Parameters Dialog	631
Parameter Properties	631

Appendix B CitectSCADA Reference Information

Specifications	633
Graphics	633
Projects	634
I/O device data types	634
Reserved ANs	635
Predefined Templates	636
Predefined Commands	638
Predefined Character Sets	639
Predefined Fonts	640
Predefined Devices	641
Predefined Cicode Files	642
Predefined Color Names and Codes	642
Predefined Keyboard Key Codes	643
Predefined Labels	647
ASCII/ANSI Character Code Listings	653
Format Fields	659
Alarm Display Fields	660
Alarm Summary Fields	661

Command Fields	663
Error Messages	663
Protocol Generic Errors	664
Protocol-Specific Errors	668
NetBIOS Errors	671

Glossary	673
-----------------	------------

Index	689
--------------	------------

Introducing CitectSCADA

CitectSCADA systems are used in various applications in many different industries. Whatever your application, CitectSCADA will help you deliver an effective plant monitoring and control system.

CitectSCADA's client-server architecture provides many benefits. For big and small applications alike, you have the flexibility to choose your own system design, confident that your system will be fast, efficient, and completely scalable. When it comes time to resize your system, you can do so without wasting any of your initial investment.

CitectSCADA provides complete [redundancy](#), tolerating failure anywhere in your system, with no loss of functionality or performance.

See Also [Using CitectSCADA](#)
[Configuring CitectSCADA](#)
[The CitectSCADA Environment](#)

Using CitectSCADA

With CitectSCADA you can:

- Provide your operators with central or local control using clear, concise, resizable graphics pages (screens).
- Add graphical control buttons to your pages to perform single or multiple tasks.
- Design sophisticated animations to display the operating status and performance of your plan
- Display text messages and graphics to show the status of a process or the state of an alarm.
- Configure your [CitectSCADA project](#) in one language and display it in any other language.
- Specify keyboard commands that operate universally (for all pages) or just for individual pages.
- Monitor, control, log, and display (in various formats) all alarms.
- Provide historical and real-time [millisecond trending](#) in graphical format.
- Monitor performance and efficiency as it happens by using trend and data logging facilities.

- Produce periodic and event-driven reports in Rich Text Format (RTF).
- Monitor product quality with Statistical Process Control (SPC) facilities.
- Develop a multi-layered security system that allows your personnel access to the [area](#) or areas of the plant within their control.
- Exchange plant-floor data with other applications for data analysis and post processing, or to control and tune your system.

Configuring CitectSCADA

Features such as templates, Genies, wizards, RAD Graphics, and automatic color swapping make it easy to configure your CitectSCADA system, as well as maximize its performance.

The CitectSCADA Environment

CitectSCADA is conceptually divided into two parts:

- Configuration
- Runtime

Configuration environment

The configuration environment consists of a set of tools (applications) you use to build the runtime system. The configuration environment is centered around the Citect Explorer, which is used to create and manage projects.

You use projects to organize your configuration data into logical, well organized, groups. You can design your system to use one or more projects at a time, depending on the modularity of your plant or system.

The configuration environment consists of the Citect Explorer, Project Editor, Graphics Builder, and Cicode Editor.

Runtime system

The runtime system is the application that you use to control and monitor your plant. You must tailor make the runtime system to suit your requirements, using the configuration tools mentioned above. Once you have configured your [CitectSCADA project](#) (or projects), you compile it to get your runtime system.

It is at runtime that CitectSCADA communicates with your I/O devices, process alarms, animate levels and symbols etc. To use the runtime system your computer requires a protection key (otherwise it runs in demonstration mode).

The runtime system consists primarily of the runtime application (as developed and compiled by you), but also includes the Citect Kernel, and the Cicode Debugger.

Chapter 2: Understanding the CitectSCADA Environment

The CitectHMI/SCADA environment is conceptually divided into two distinct parts:

- the Configuration Environment
- the Runtime System

The Configuration Environment

The configuration environment consists of a set of tools (applications) that are used to build the runtime system. The configuration environment is centred around the Citect Explorer, which is used to create and manage projects.

Projects are used to organise your configuration data into logical, well organised, groups. You can design your system to utilise one or more projects at a time, depending on the modularity of your plant or system.

The configuration environment consists of the following components:

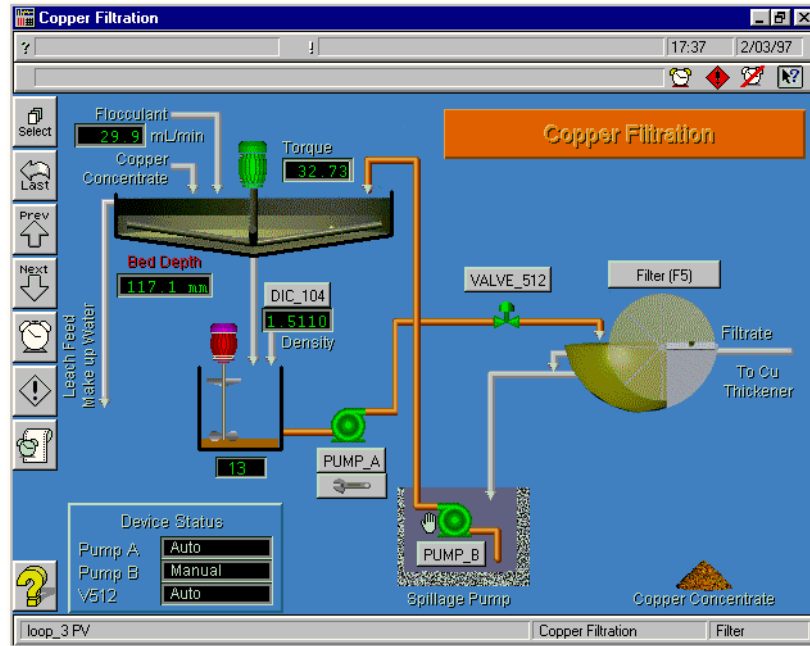
- [Citect Explorer](#)
- [Project Editor](#)
- [Citect Graphics Builder](#)
- [Cicode Editor](#)
- [CitectSCADA Help](#)

The Runtime System

CitectSCADA runtime is the application that you will use to control and monitor your plant. You must tailor make the runtime system to suit your requirements, using the configuration tools mentioned above. Once you have configured your project (or projects), they must be compiled to build your runtime system.

It is at runtime when CitectHMI/SCADA will communicate with your I/O devices, process alarms, animate levels and symbols etc. The runtime system is

the graphical interface (of your design) that you use to control and monitor your plant. A typical runtime screen might look like this:



The runtime system consists primarily of the runtime application, but also includes the CitectHMI/SCADA Kernel and the Cicode Debugger, used to debug systems and check performance.

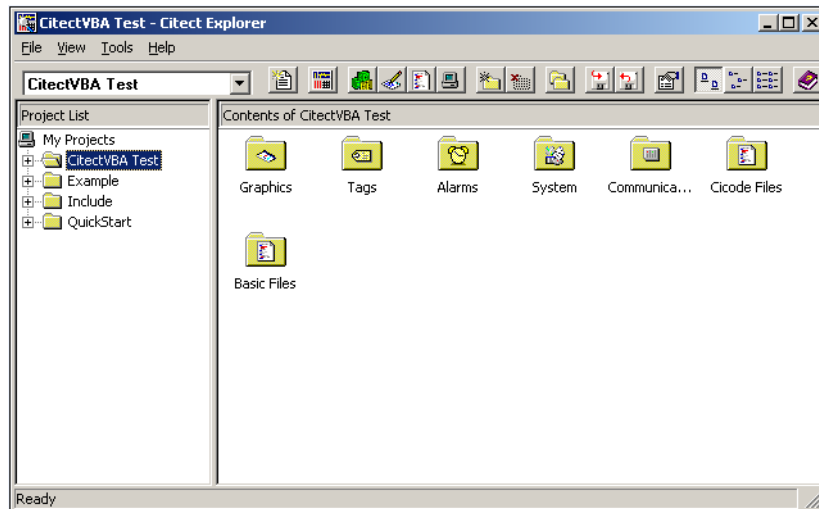
To use the runtime system, your computer requires a protection key (otherwise it will run in demonstration mode).

See Also [Configuring CitectSCADA Projects](#)

Citect Explorer

You use Citect Explorer to create and manage your CitectSCADA projects. It is also the controlling configuration application from which you can run the

Project Editor, Graphics Builder, and Cicode Editor. Citect Explorer looks like this:



When you start Citect Explorer, the Project Editor and Graphics Builder automatically start and are minimized. When you close Citect Explorer, the other CitectSCADA applications are shut down.

See Also [Understanding the CitectSCADA Environment](#)

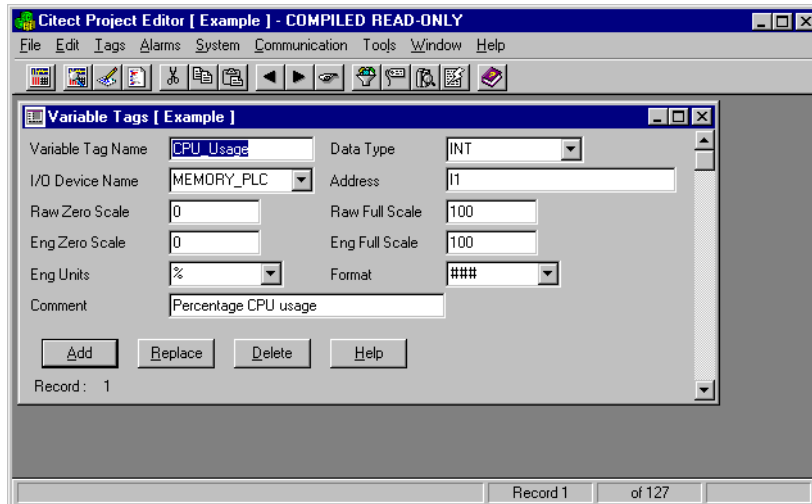
Project Editor

You use the Citect Project Editor to create and manage the CitectSCADA database containing the configuration information for your [CitectSCADA project](#), which is not related to graphics pages. You can view all CitectSCADA project database records in the Citect Project Editor.

The Project Editor has some specific commands that you can access by using the menu system or buttons, as indicated below:

- **Report Selection** button: Choose a CitectSCADA report.
- **Find User Function** button: Search for a user-defined [Cicode](#) function.
- **Insert Function** menu: Insert a pre-defined Cicode function.

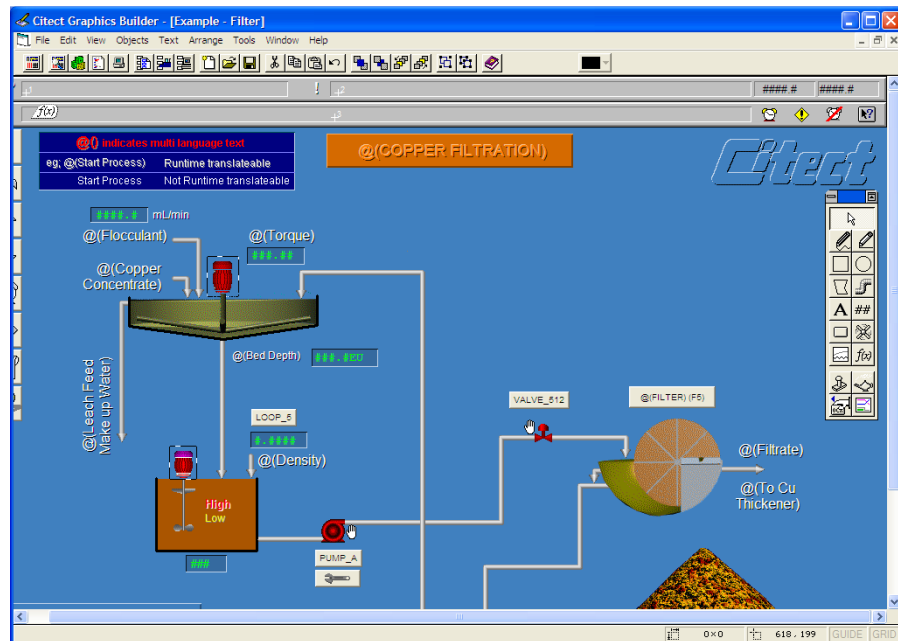
- Paste Tag menu: Insert a pre-defined variable tag.



See Also [Understanding the CitectSCADA Environment](#)
[Configuring CitectSCADA Projects](#)

Citect Graphics Builder

You use the Graphics Builder to create and edit your graphics pages, including the objects that comprise the graphics pages. Graphics Builder starts automatically when you double-click a graphic object in Citect Explorer.

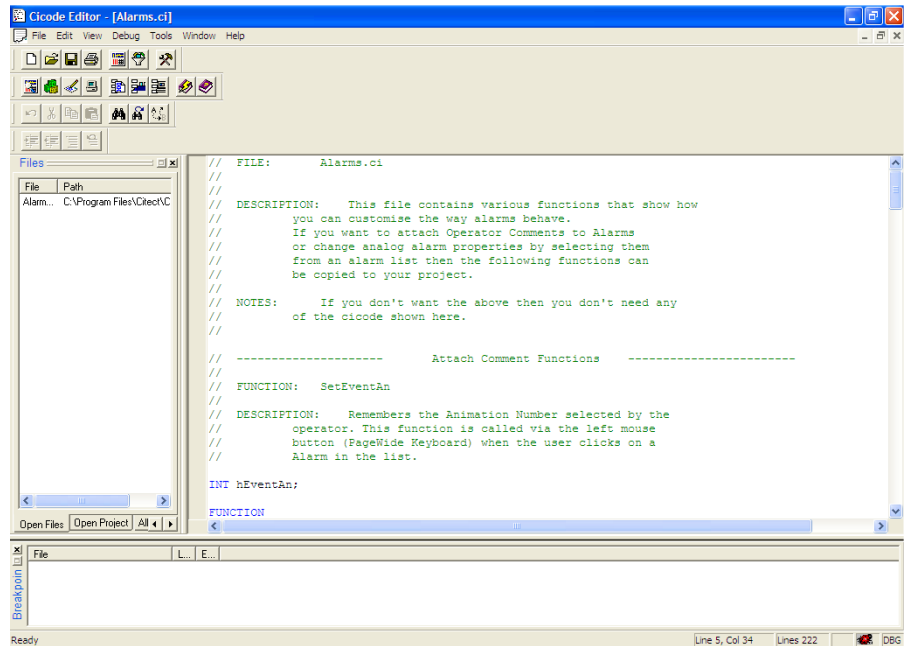


See Also [Understanding the CitectSCADA Environment](#)
[Defining and Drawing Graphics Pages](#)

Cicode Editor

You use the Cicode Editor to write and edit your [Cicode](#) programs. Within the Cicode Editor window, you can get help for any Cicode function. Right-click the function name and choose **Help** from the menu.

You can use the Cicode Editor as a debugger at runtime to help you trace through your running Cicode and track down programming errors. You can also debug your Cicode programs from a remote computer.



Refer to the *Cicode Reference* manual for details about writing and debugging Cicode programs, Cicode files, Cicode libraries and functions, Cicode commands and expressions, and the Cicode Editor. This information is also available in the CitectSCADA Help.

See Also [Understanding the CitectSCADA Environment](#)
[Introducing Cicode](#)

CitectSCADA Help

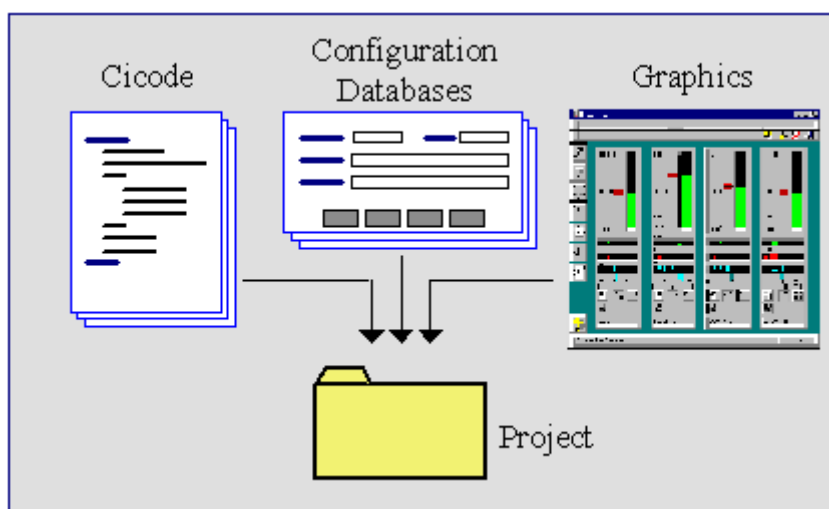
The CitectSCADA Help contains useful information to help you develop your CitectSCADA system. You can access Help from Citect Explorer, Graphics Builder, Project Editor, and Editor/Debugger.

See Also [Understanding the CitectSCADA Environment](#)

Chapter 3: Configuring CitectSCADA Projects

A CitectSCADA project consists of three major elements:

- Graphics pages
- Configuration databases
- [Cicode](#) files



Your **graphics pages** that you view on your computer screen display the status or condition of the plant. Graphics pages can also contain controls and command buttons that enable an operator to control plant processes.

Databases store configuration information about the plant that is used by the runtime system to control and monitor the plant. Some databases are linked to specific graphics pages.

Cicode files store your custom Cicode functions. Cicode is used to perform commands and actions and extend the functionality of your system.

All CitectSCADA Projects are created, selected, opened, closed, deleted, linked to, and have their properties edited from within the Citect Explorer.

See Also [Configuring a Project](#)
[Working with Existing Projects](#)
[Including Projects](#)
[Backing Up Projects \(Archiving\)](#)
[Copying Projects](#)

[Printing Project Details](#)

Configuring a Project

To completely configure a [CitectSCADA project](#), you need to engineer several different areas:

- 1 Create the project using the Citect Explorer. Once you have created your project, back it up regularly to minimize the amount of lost data should you have a problem (such as hard disk failure).
- 2 Set up communication with a device by following the basic steps given in the [I/O device](#) setup procedure. Often, the details of communication are not known when first creating a project, in which case a 'dummy' I/O device can be used, defined as a memory device.
- 3 Define the data that CitectSCADA needs to read, write and use, by defining variable tags. If you adopt a structured tagging convention, you can define most of your variable tags without knowing the physical address.
- 4 Create your graphics pages using the Graphics Builder. Once you have created the basic pages, you can add the graphic objects for indication and user interaction.
- 5 Configure any features that are not page-based in the Project Editor. This includes alarms, reporting, events, logging, etc.
- 6 Create and write custom Cicode functions using the Cicode Editor.

These steps are listed in a logical order, but not necessarily in the order that you must follow. For example, you will most likely develop your Cicode at the same time as the pages, reports, and so on.

Before doing any of these steps, first consider your requirements and design your system.

See Also [Project Design Considerations](#)

Project Design Considerations

The first and most important step in any system development is design. Good design ensures that your system:

- 1 Performs the control and monitoring tasks that are required.
- 2 Is implemented with minimal interruption to the application.
- 3 Achieves the best possible performance.

Poor design often results in substantial rework, major disruption to the organization, poor performance, or all three. With CitectSCADA you can easily

configure a system to do whatever you want; there are no restrictions on how your system will operate, or how your operators will interact with it.

Some issues to consider are:

- How the plant is graphically represented to the operator.
- How the operator navigates the system.
- What plant-floor data will be displayed on the screen.
- What operator controls are required and where they are presented on the page.
- What plant conditions need to be monitored for alarm conditions.
- What data logging is required for maintenance and performance monitoring purposes.
- What reports management will require.
- What level of security (if any) is required in the runtime system.

See Also [Project Design Standards](#)

Project Design Standards

Design standards promote consistency and clarity. Consistency and clarity reduce your development time, and reduce the time that your operators need to learn your system.

You should, for instance, choose a common screen location for all control buttons of a certain type, keyboard keys that always perform the same operation, and standard colors for displaying similar types of information (e.g. alarms).

Naming standards are recommended throughout your configuration, use a naming convention for pages, alarms, commands (and all database records). A standard naming convention can:

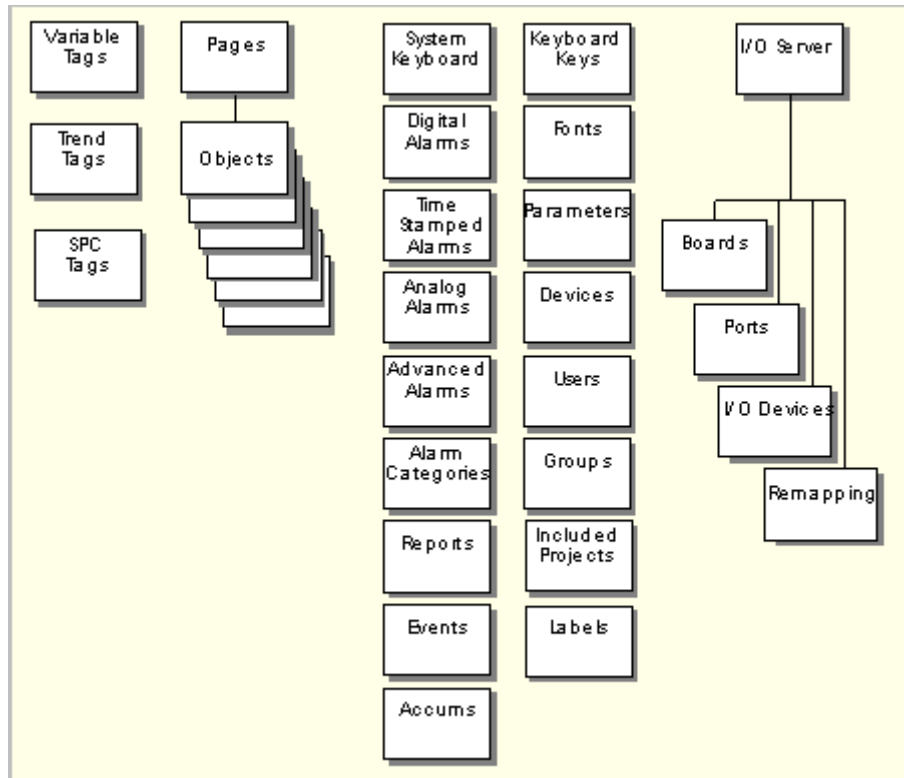
- Reduce database search time.
- Reduce data entry.
- Reduce time and effort when configuring future changes and enhancements.

There are many conventions you can use to standardize the names of your database records. The most common method is to include as much (abbreviated) information as possible in the name (up to 16 characters, 79 for variable tag names). For instance, you can include the [area](#) and the process, or the machine and the device with which the record is associated.

See Also [The CitectSCADA Database Structure](#)

The CitectSCADA Database Structure

CitectSCADA uses several databases to store configuration data. The following diagram shows the basic structure of these databases:



You can display and edit the information in each database using a database form of the same name.

See Also [Using Other Database Editors](#)

Using Other Database Editors

All CitectSCADA databases are stored on disk in a standard dBASE III format. You can therefore edit any CitectSCADA database using any database editor that reads dBASE III files (e.g., Access, dBASE, Clipper, or FoxPro).

When using other database editors, remember the following:

- 1 All key fields must be uppercase. The CitectSCADA compiler only recognizes upper case key fields.
- 2 You can add or change (expand or reduce) fields in a CitectSCADA database, but you must not remove existing fields, or problems might occur during compilation.

- 3 Some databases are indexed. If you add new records or edit the index key field, the index must be rebuilt. The easiest way to do this is to pack the database in the Project Editor after editing. Packing the database deletes all records marked for deletion, and re-indexes all the databases.
- 4 All CitectSCADA database fields are left-justified. Make sure that any key fields entered by another database editor are also left-justified.
- 5 CitectSCADA databases only support string data types.
- 6 Be careful when adding records to the page animation records, such as numbers, colors, strings, buttons, and so on. These databases are linked into the background graphic image; only add records to a page database where the AN already exists. (If an AN does not exist, the record is deleted when the page is edited.)
- 7 You should pack regularly if you have been deleting or editing the Variables database file using third-party database editors (such as Microsoft Excel). To pack the database, choose **File | Pack** in the Project Editor. This deletes all records marked for deletion and reindex those that remain.

See Also [Special considerations for Microsoft Excel](#)

Special considerations for Microsoft Excel

When using Excel, you cannot change the width of the fields: all the fields are truncated to the new size, effectively destroying the entire database. In addition, if Excel finds only numbers in a field, it tries to change the type of the field into the number format, also destroying the database. (CitectSCADA only supports the string format.)

Format specifications (dBASE III)

The table below shows the format specifications for dBase III.

Maximum record size	4000 bytes
Maximum number of fields	128
Maximum length of field	254 bytes
Maximum length of field name	10 characters

Saving database files in Microsoft Excel

If you add records (rows) to a database file using Microsoft Excel, you must redefine the size of the database range before saving. If you don't, the new rows are not considered records in the database. Because it is easy to forget to redefine the size of the database range, CitectSCADA comes with a macro (Save_DBF) which does it for you as part of the save process. The operation of this macro has been verified on Excel 95 and 97.

Note: CitectSCADA does not recognize the added records (even if the database range has been resized) until the project database is Packed.

To use the Save_DBF macro, open **Save_DBF.xls** from the CitectSCADA `bin` directory; the macro loads automatically. (Leave Save_DBF.xls open. When closed, the macro stops working.) When the macro is loaded, it adds the following options to the workbook shortcut (right mouse) menu:

Option	Description
Cell Length	Reports the number of characters in the selected cell on the status bar (clears after 5 seconds). If a cell range is selected, the cell length of the top left cell is evaluated.
Save DBF	Re-defines the database range and saves the file.
Save/Close DBF	Performs the same actions as Save DBF , then closes the workbook. Using this option, you are not prompted with the “Do you want to save” dialog that normally displays when a workbook which has not been saved in native Excel format is closed.
Save DBF As	Allows the file to be saved under a different file name and/or in a different location.

Option	Description
Database Settings	<p>Allows various options to be set as follows.</p> <p>Number of Records Based on:</p> <p>First Column: The first column is searched upwards from the bottom row. The row number of the first non-empty cell found is defined as the number of rows in the database.</p> <p>Longest Column: Starting at the left column, multiple columns are searched upwards from the bottom. The largest row number of the first non-empty cell in these columns is assumed to be the numbers of rows in the database. The number of columns searched is set by the Number of Columns field.</p> <p>Region (Records & Fields): The current region (i.e., based on the active cell) is defined as the database region. This is equivalent to using CTRL + * to select a range.</p> <p>Active Cell: The row number of the active cell is assumed to be the number of rows.</p> <p>Number of Columns: Limits the number of columns searched when Longest Column is selected.</p> <p>Number of Fields Based on:</p> <p>First Row: The first row is searched leftward from the right most column (256). The column number of the first non-empty cell found is assumed to be the number of fields.</p> <p>Number of Columns: The value in the Number of Columns field (above) is used. NOTE: If this number is incorrect, you might experience problems with Filter and Extract.</p> <p>Active Cell: The column number of the active cell is assumed to be the number of columns.</p> <p>Show Form: Information about the range being saved appears in the status bar and on a form.</p> <p>Save Settings: The current settings are saved when quitting Excel. If you do not choose this option, the default settings apply the next time the macro is loaded.</p> <p>Filter Options: Adds/removes the Filter/Extract options to the Shortcut Menu.</p>

Note: The macro first attempts to find row 65,536 when searching for the bottom of a work sheet. If this fails it tries row 16,384.

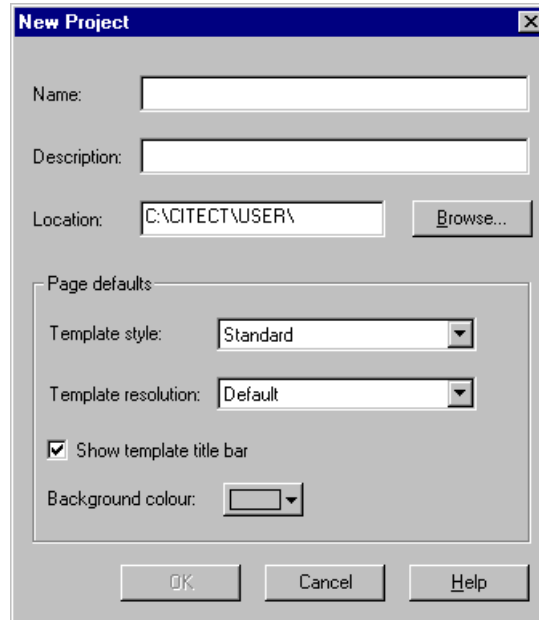
Option	Description
Set Criteria/Extract	<p>Copies the database range column headings to two locations below the database range in preparation for the Filter and Extract operations. The first is 5 rows below the database range and is named "Criteria Titles". The second is 15 rows below the database range and is named "Extract".</p> <p>You can type up to ten criteria under the "Criteria Titles", choose the "Criteria Titles" and criteria you want to use, and right click to choose either Filter or Extract from the shortcut menu.</p> <p>Empty criteria columns have no effect if selected, but if any empty criteria rows are selected, all records are Filtered/Extracted.</p> <p>When either Filter or Extract is chosen from the shortcut menu, the cells that are selected at the time are named as the CRITERIA range (see above) and the desired operation is performed.</p> <p>Filter A Filter in Place is performed using the criteria defined above. To cancel the operation, choose Filter/Show All from the Excel Data menu.</p> <p>Extract Data is extracted to the Extract range using the criteria defined above.</p>

Creating New Projects

To create a new project:

- 1 Start Citect Explorer.
- 2 Click the **New Project** button, or choose **File | New Project**.
- 3 Complete the New Project dialog box. You must at least complete the **Name** field.

- 4 Click **OK** to create the project, or **Cancel**.



See Also [New Project dialog properties](#)
[Working with Existing Projects](#)

New Project dialog properties

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.

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 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.

[Page defaults] Background color

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

Working with Existing Projects

To open an existing project:

- 1 Open Citect Explorer.
- 2 Choose a project, or click a project in the Project List area.

To delete an existing project:

- 1 Open Citect Explorer.
- 2 Choose a project from the list.
- 3 Choose **File** | **Delete Project**, or click **Delete**.

- 4 Click **OK** to delete the project, or click **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.

Warning! You cannot recover a deleted project that hasn't been backed-up.

See Also [Linking and Unlinking Projects](#)

Linking and Unlinking Projects

CitectSCADA installations on different computers can share the same project. After a project has been created on one computer, other computers can *link* to the same project, but only if the project location is on a shared or [network](#) drive.

To link a project:

- 1 Open Citect Explorer.
- 2 Click the **Add Link** button, or choose **File | Add Project Link**.
- 3 Use the Select Project Directory dialog box 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. Use the **Help** button for more information about the fields.

To remove a link to a project:

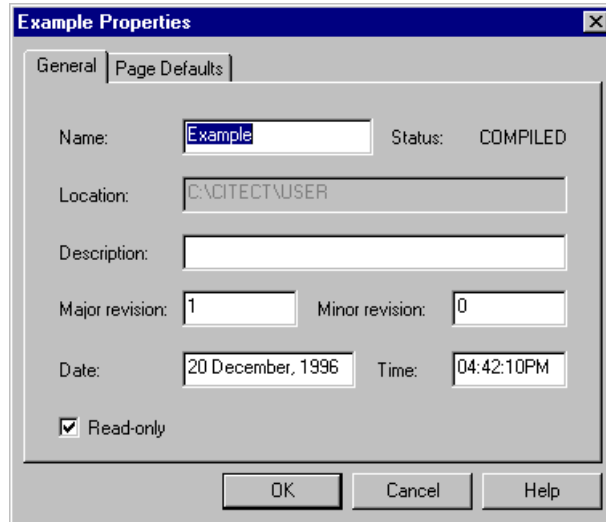
- 1 Open Citect Explorer.
- 2 Select a project from the list.
- 3 Click the **Remove Link** button, or choose **File | Remove Project Link**.
- 4 A message box asks you if you want to proceed. Click **Yes** to remove the link, or **No** to cancel.

See Also [Editing Project Properties](#)

Editing Project Properties

To edit the properties of a project:

- 1 Open Citect Explorer.
- 2 Select a project from the list.
- 3 Click the **Properties** button, or choose **File | Project Properties**.
- 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 box](#)

Project Properties dialog box

Use this dialog for [Editing Project Properties](#).

Projects have **General** properties and **Page** properties.

General project properties

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.

Status

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

Location

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

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.

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.

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.

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.

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.

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 Project Properties

[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

Screen Type	Screen Width (pixels)	Screen Height (pixels)
User	****	****

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

[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.

[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.

Background color

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

If you change properties, you must click OK to save the changes to the project.

Linking to CitectSCADA Projects (on the same network)

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.

Warning! 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 choose **File | Add Project Link**.
- 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 choose **File | Remove Project Link**.
- 4 You are prompted if you want to proceed. Click **Yes** to remove the link, or **No** to cancel.

See Also [Including 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.

Note: Be careful not to confuse include files with included projects:

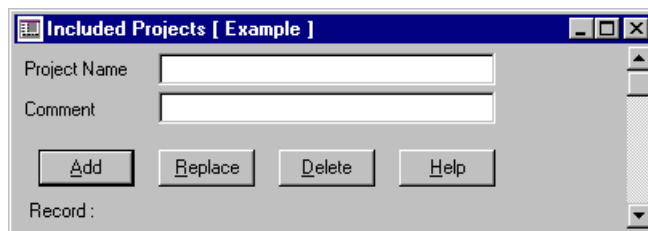
- **Include Files** contain CitectSCADA commands and/or expressions and are used as substitutions in a CitectSCADA command or [expression](#) property field.
- **Included Projects** are separate and (usually smaller) CitectSCADA projects that can be included in another CitectSCADA project so that they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

Each CitectSCADA system is supplied with an include project. This project contains pre-defined database records and is automatically included in each of your projects.

To include another project (in the current project):

- 1 Open the Citect Explorer.
- 2 Choose **System | Included Projects**.
- 3 Complete the Included Projects form that appears.
- 4 Click **Add** to append a record you have created, or **Replace** if you have modified a record.

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



See Also [Included Projects dialog box](#)
[The CitectSCADA Include project](#)

Included Projects dialog box

This dialog box lets you [include another project](#) in the current CitectSCADA project. With large systems, it might be better to 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).

The CitectSCADA Include project

Project Name

The name of the project to include in this project.

Comment

Any useful comment.

Note: Each CitectSCADA system is supplied with an include project. This project contains pre-defined database records and graphics libraries, and is automatically included in each of your projects.

Each CitectSCADA project is automatically supplied with a predefined Include project designed to help you develop your CitectSCADA project faster. The Include project contains pre-defined database records and graphics libraries.

Note: Do not modify the Include project. Your changes to the Include project are lost when you reinstall CitectSCADA or upgrade to a new version.

The Include project is hidden from the project tree in Citect Explorer by default.

To show/hide the CitectSCADA Include project:

- 1 Open the Citect Explorer.
- 2 Choose **View | Show Include Project**.

Backing Up Projects (Archiving)

After you have configured your CitectSCADA system, back up (or archive) the project onto disk. With your project backed up, you will not lose any valuable data if the hard disk on your computer becomes damaged. CitectSCADA lets you back up a project to a local (floppy disk, hard drive) or network location.

Note: When you are developing a project, adopt a regular backup strategy. This allows you to archive versions which you can then refer back to if required. Before performing a backup, ensure that you have refreshed any linked tags in your project.

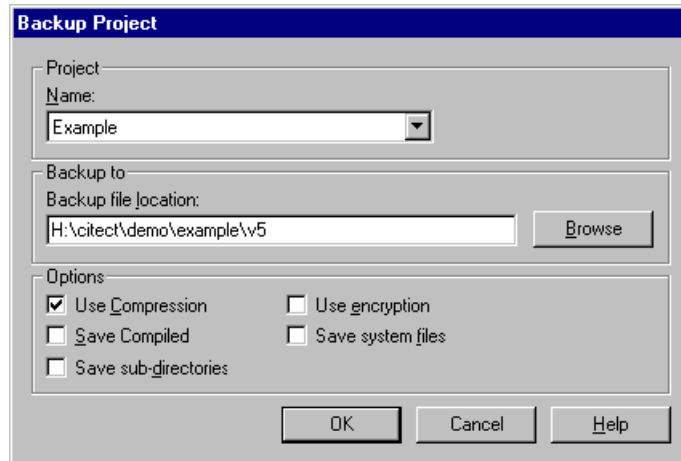
The Citect Backup project utility archives files using a standard compression routine, producing PKZip® v2.04g compatible files. The default extension for CitectSCADA backup files is .CTZ, though any extension (including .ZIP) can be used. This means you can also use the PKZip® utility to extract files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

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

To back up a project:

- 1 Open Citect Explorer.
- 2 Click **Backup**, or choose **Tools | Backup**. The Backup Project dialog box appears.

- 3 Specify a source project and destination, and any options.
- 4 Click **OK**.



See Also [Backup Project dialog properties](#)
[Restoring Projects](#)
[Using the Backup Utility from the Command Line](#)

Backup Project dialog properties

This dialog box lets you [back up a project](#) to a local (floppy disk, hard drive) or network location. Backing up your project ensures that you do not lose any valuable data if the hard disk on your computer becomes damaged. To back up a project, specify a source and destination, and any option, then click **OK**.

[Project] Name

The name of the project to backup.

[Backup to] Backup file location

The path to the backup file location, including the backup file name. You can either type the path in directly or use the **Browse** button. When browsing, you might want to use the **Network** button to map a drive letter to a destination directory (if it were a UNC path for example).

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

Note: When you backup a project to a floppy disk, the backup program deletes all files on the floppy disk before backing up the project onto the disk. The backup program always warns you before it deletes any files on the floppy disk.

[Options] Use compression

You can use data compression when you are backing up a project, to preserve space on your floppy disk.

[Options] Save compiled

When you back up a project, CitectSCADA normally saves it in UNCOMPILED mode. If you choose this option, CitectSCADA backs up both the COMPILED and UNCOMPILED projects, resulting in a larger backup file.

[Options] Save sub-directories

If you choose this option, CitectSCADA also backs up all data in any sub-directories (below the project directory). The directory structure is maintained in the backup, so the sub-directories are recovered when restoring.

[Options] Use encryption

If security is important, you can backup your project in an encrypted format. If you choose this option, CitectSCADA displays a dialog box requesting a password, before the project is backed up. See [Backup/Restore – Password Encryption dialog properties](#).

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 when the password is used.

Warning! If you forget the password, you cannot restore the project.

[Options] Save system files

If you choose this option, CitectSCADA also backs up your AUTOEXEC.BAT, CONFIG.SYS, and all Windows configuration (.INI) files. Only use this option if advised to do so by Citect Support.

Note: If you restore a project that has been archived with this option, the AUTOEXEC.BAT and CONFIG.SYS files are placed in the project directory (the system files are not overwritten).

**Backup/Restore –
Password Encryption
dialog properties**

This dialog box lets you set a password for your project. 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 when the password is used.

Enter Password

When you enter your password, asterisks display in its place.

Re-Enter Password

When you re-enter your password, CitectSCADA checks that you have typed the same password both times.

Warning! If you forget the password, you cannot restore the project.

See Also [Backing Up Projects \(Archiving\)](#)

Restoring Projects

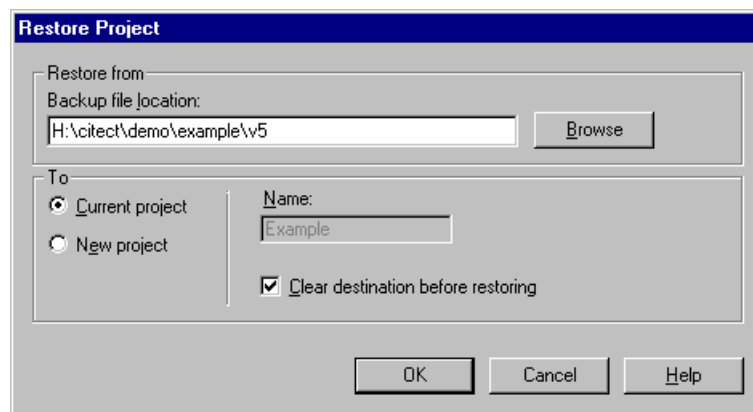
You can restore backed-up and archived projects by using the Restore Project utility. This utility allows you to overwrite any current project with a backed-up version, or restore a backed-up project as a completely new project.

To restore a project:

- 1 Open Citect Explorer.
- 2 Click **Restore**, or choose **Tools | Restore**.
- 3 In the Restore dialog specify a source directory and a destination project.

Note: When restoring to the current project directory, check that you have set the *Clear destination before restoring* option correctly

- 4 Click **OK** to restore the project, or click **Cancel**.



See Also [Restore Project dialog properties](#)

Restore Project dialog properties

This dialog box lets you [restore](#) a previously backed up project from a local (floppy disk, hard drive) or network location. You can restore to a new project (and directory) or an existing project. To Restore a project, specify a source [Restore from] and destination [To] project, then click the **OK** button.

[Restore from] Backup file location

The path to the backup file. You can either type the path in directly or use the **Browse** button.

Note: When browsing, you might want to use the **Network** button to map a drive letter to a destination directory (if it were a UNC path for example).

[To] Current Project

Specifies that you want to restore the backup to the currently selected project. When this option is selected, the fields below are shown in the group box to the right.

Name

The name of the currently selected project: where the backup is restored to. This field cannot be edited. To change the Current Project Name, you must close the form and choose a different project.

Clear destination before restoring

Specifies to delete the contents of the currently selected (existing) project first. This ensures that no residual files are left behind to interfere with the restored project.

[To] New Project

Specifies that you want create new project from the backup. When this option is selected, the following fields are shown in the group box to the right:

Name

A unique name for the project. The project name is restricted to 64 characters, and can contain any alphanumeric characters (A - Z, a - z, or 0 - 9), and the underscore '_' character. Because the project name is a unique identifier, CitectSCADA will not permit you to create or restore a project with the same name.

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

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 might override this by manually entering the location or clicking **Browse**.

Note: The project properties are stored in the backup and are restored also. If the project Name already exists, you are prompted to enter a new one.

Using the Backup Utility from the Command Line

You can use the CitectSCADA backup utility from the command line to back up and restore files other than CitectSCADA projects if required.

CtBackup.exe was shipped with CitectSCADA versions earlier than version 5, and CtBackup32 is shipped with CitectSCADA version 5 and later. It is installed to the Citect project 'Bin' folder by default.

Note: The Citect Backup Project utility 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 (if you have it) to extract files from a compressed CitectSCADA backup if you prefer.

CtBackup and
CtBackup32 utility
command line options

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

Note: Be careful when configuring the backup utility to restore files as it will first delete all files in the destination directory and all subdirectories before restoring. If you accidentally set your restore path to the root directory of the drive, the utility will delete your entire disk drive.

[CtBackup and CtBackup32 utility command line options](#)

The table below describes the utility command line options.

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

TechSupport=0/1 ! backup tech support data

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.

```

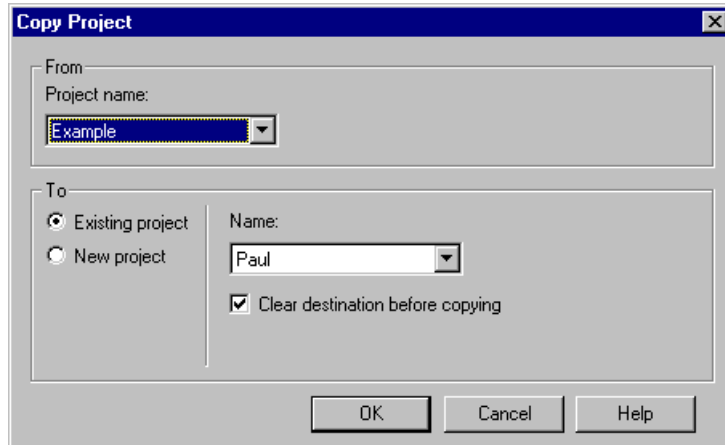
Copying Projects

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

To copy a project:

- 1 Open Citect Explorer.
- 2 Click **Copy**, or choose **File | Copy To**.
- 3 In the Copy dialog specify a source project and destination project.

- 4 Click **OK** to copy the project, or click **Cancel**.



See Also [Copy Project dialog properties](#)

Copy Project dialog properties

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.

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.

Clear location before copying

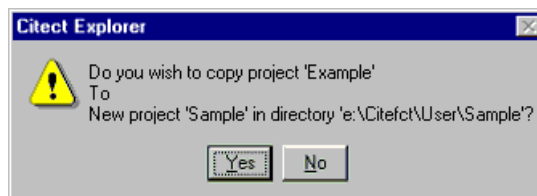
Specifies to delete the contents of the existing destination project before copying the source project to the destination location. This ensures that no residual files are left behind to interfere with the copied project.

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.

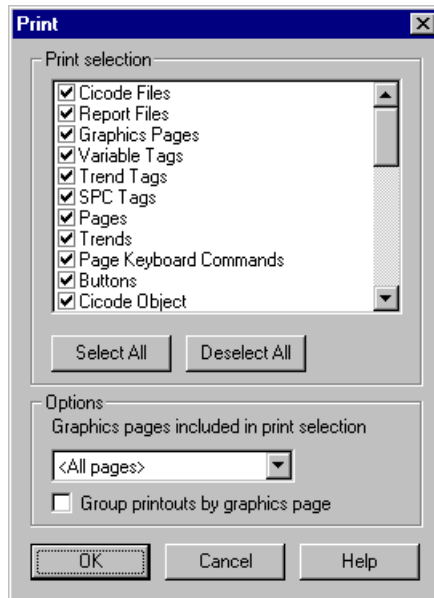
Printing Project Details

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 Choose **File** | **Print**.
- 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 **Tools | Options**. For other print options, refer to your Windows documentation.



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

Print (project details) dialog properties

This dialog box lets you [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.

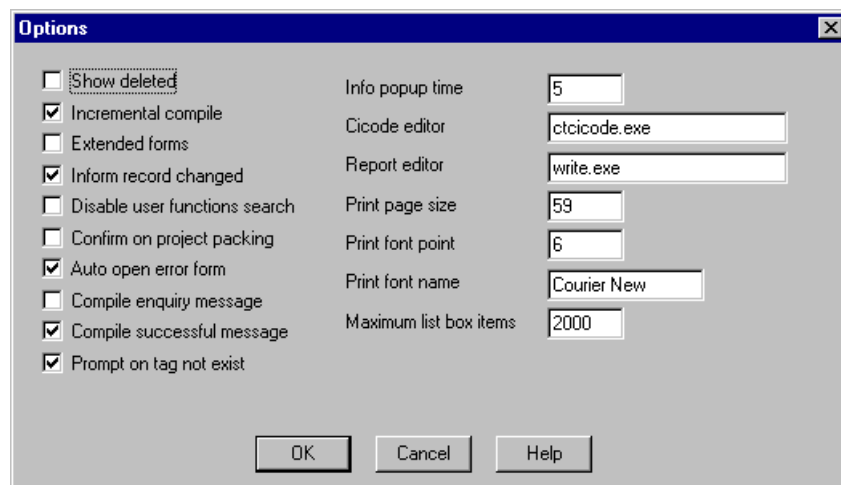
[Options] Group printouts by graphics page

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).

Project Editor Options

The Project Editor has options that allow you to change the configuration environment. These options are available through the Tools menu.



See Also [Project Editor Options Dialog Properties](#)

Project Editor Options Dialog Properties

This dialog box lets you set (and change) the [Project Editor Options](#).

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 when you add (or change) data in a database form and then try to close the form - before you add or replace the record.

Warning! 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, 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 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 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 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.

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

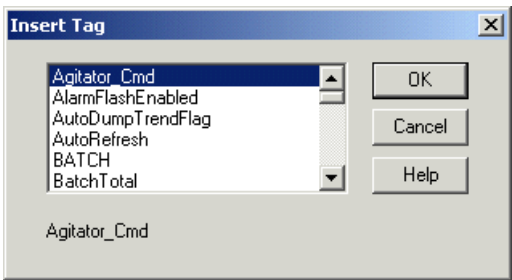
The name of the font used when printing database records.

Maximum list box items

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

Insert Tag dialog box

Use the Insert Tag dialog box to insert a variable tag into a tag or expression field. To insert a variable tag, select the tag name, then click **OK**. The tag is inserted in the tag or expression field at the location of the cursor.

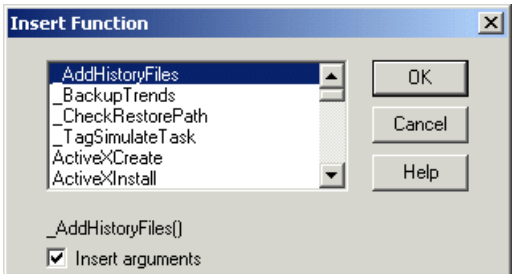


Insert Function dialog box

Use the Insert Function dialog box to insert into the current field. To insert a function, select the function name, then click **OK**. To insert the function with its [arguments](#), 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 properties**

This dialog box lets you find a [Cicode](#) function in your Cicode source files. To locate a function, enter the function name (or part of the function name) and click the OK button.

A list of functions that match your search criteria displays in the large dialog box window.

To edit the Cicode file that contains the function, select the function name (from the large window) and select the Edit button.

Chapter 4: 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](#).)

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 [Configuring variable tags](#)
[Formatting numeric variables](#)
[Using arrays](#)
[Using structured tag names](#)

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 appears.

Variable Tags [Example]

Variable Tag Name	<input type="text" value="CPU_Usage"/>	Data Type	<input type="text" value="INT"/>
I/O Device Name	<input type="text" value="MEMORY_PLC"/>	Address	<input type="text" value="I1"/>
Raw Zero Scale	<input type="text" value="0"/>	Raw Full Scale	<input type="text" value="100"/>
Eng Zero Scale	<input type="text" value="0"/>	Eng Full Scale	<input type="text" value="100"/>
Eng Units	<input type="text" value="%"/>	Format	<input type="text" value="###"/>
Comment	<input type="text" value="Percentage CPU usage"/>		

Record: 1

- 3 Enter the properties of the variable tag.
- 4 Click **Add** to append a new record, or **Replace** to modify a record.

Note: You must at least 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.

Note: 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.

Note: 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). If you have many tags, use a naming convention. This makes it easier to find and debug your tags.

Note: 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
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)

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.

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).

Note: 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 memory or 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.

Note: 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 you are 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 depend on the I/O device you are using.

Raw Zero Scale / Raw Full Scale

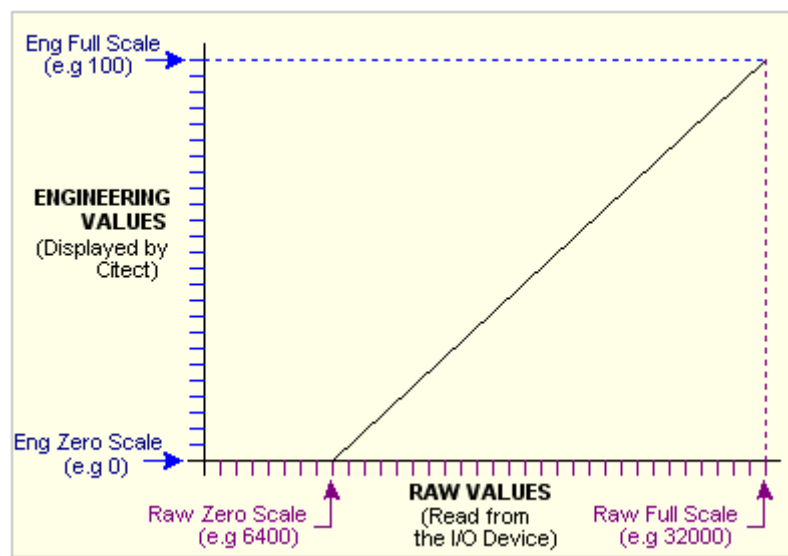
The unscaled (raw) values (of the variable) that represent the zero point and full scale point for the data (10 characters). The raw values are the values that CitectSCADA reads from the I/O device.

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

(8 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

(10 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.

Comment

Any useful comment (32 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.

Formatting numeric variables

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 (e.g. 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.

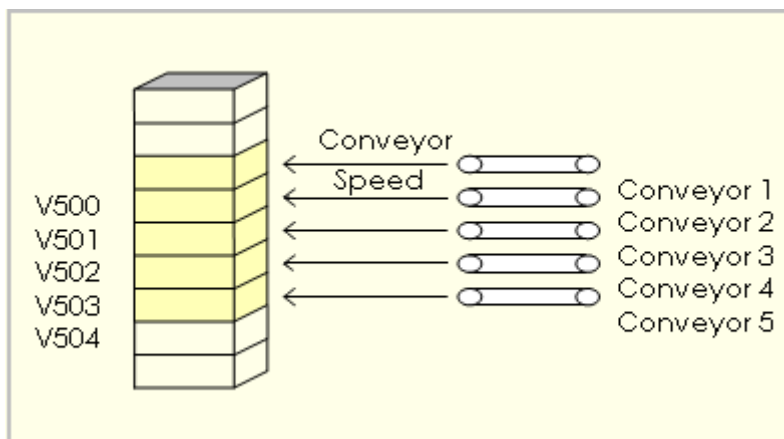
See Also [Using arrays](#)

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:



In the above example, 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]

In this example, 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 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:

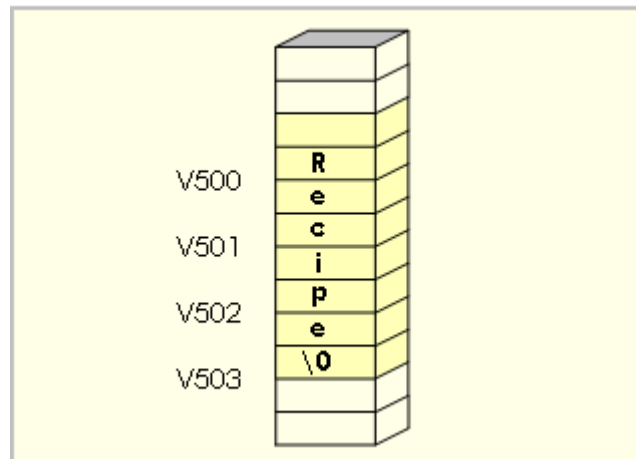
$$\text{Size of Array} = \frac{(\text{number of characters} + 1)}{2}$$

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:



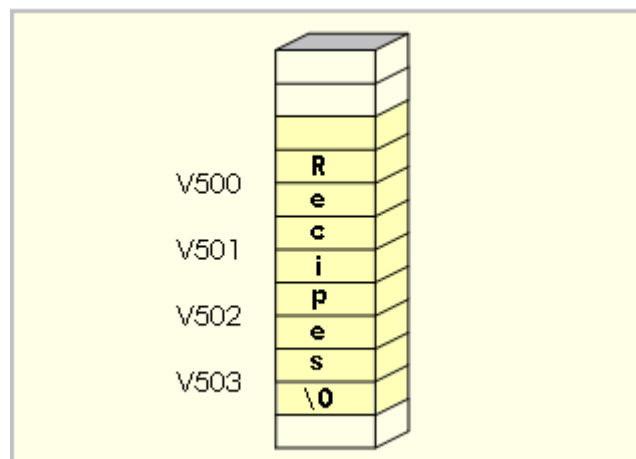
You can then refer to the entire string by specifying the tag:

<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 does support 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](#)

Using structured tag names

CitectSCADA puts no restrictions on the names of variable tags, but you will benefit from using a tag naming convention. By using a tag naming convention, your project will be easier and faster to design, configure, and commission, and will require less time for future maintenance.

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

Mnemonic	Process Alarms	Data Type	Range
_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 Hysterisis	Analog	
_Hhyst	High Alarm Hysterisis	Analog	
_Lhyst	Low Alarm Hysterisis	Analog	
_LLhyst	Low Low Alarm Hysterisis	Analog	
_LDhyst	Low Deviation Alarm Hysterisis	Analog	
_HDhyst	High Deviation Hysterisis	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	
_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	

Mnemonic	Analog Control / Monitoring	Data Type	Range
_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.

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](#)

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 Citect 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.

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

Similarly, 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 fields:

Select Linked I/O Devices

Every linked I/O device in your project (and included projects) will display here. To refresh the tags for an I/O device, click the I/O device, then click **Refresh**. This will update your project with the latest tag values for the selected I/O device.

If you used CitectSCADA to modify any of the I/O device's 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.

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 MXChange 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:

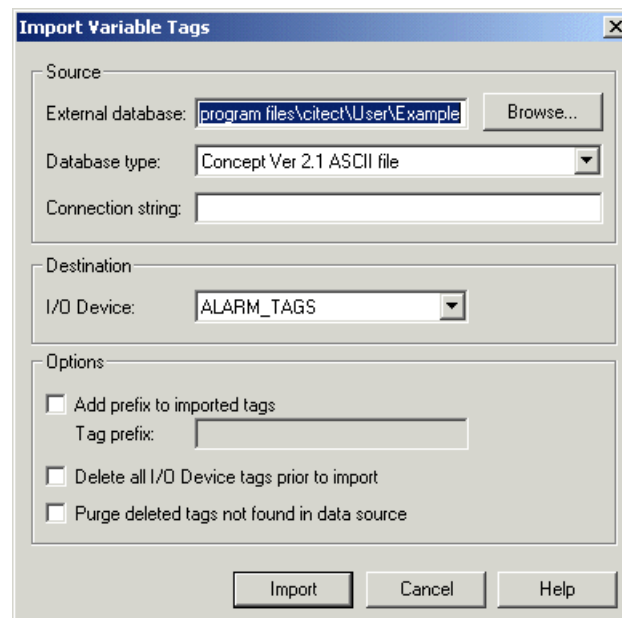
- Delete all tags associated with that I/O device prior to the import.
- Update existing tags on import. Tags found in CitectSCADA and in the external data source are updated in CitectSCADA. Tags found in CitectSCADA but not in the external data source will remain untouched. All new tags are appended.

If you import a data source that is already linked, all of the tags in the data source are duplicated; i.e., you will have two copies of each tag: one local and one linked.

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 datasource. The Import Variable Tags dialog has the following fields:

External database (128 Chars.)

A reference to the external data source to be imported. 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)

Database type

The format of the data referenced by the external data source.

Connection string (128 Chars.)

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.

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.

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.

FastLinx for Mitsubishi Tag Browser Properties

Purge deleted tags not found in data source

Tick this box if you want to delete any 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.

Use the CitectSCADA FastLinx tag browser to:

- Browse the available Mitsubishi servers on your network.
- Generate the connection strings and server parameters that the MxChange 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 MxChange 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 MxChange database with the same name as the current Citect project:

- If an MxChange database exists with the same name as the Citect project and the database contains a GID node with the same name as the Citect 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 MxChange database exists with the same name as the Citect 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 Citect 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](#)
[Reserved Names for Mitsubishi FastLinx Variable Tags](#)

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. Using these reserved words may cause a syntax error in Mitsubishi FastLinx when you export tags to the MxChange tag name database. You also cannot use the name of certain [Cicode](#) function names; you should review these function names before defining your variable tag names.

When defining variable tag names, do not use the following reserved words:

BOOL	TIME	VAR_GLOBAL	DO
FALSE	DATE	VAR_EXTERNAL	EXIT
TRUE	STRING	VAR_INPUT	IF
INT	DINT	VAR_OUTPUT	END_IF
TIME	WORD	REAL	END_FOR
DATE	DWORD	END_VAR	WHILE
STRING	DUT	VAR_IN_OUT	END_WHILE
TRUE	ARRAY	FOR	UNTIL
INT	ANY_NUM	TO	REPEAT
ACTION	OF	CONFIGURATION	FUNCTION
CASE	ELSE	CONSTANT	PROGRAM
END_CASE	EN, ENO	END_PROGRAM	TASK
END_CONFIGURATION			

******, NOT, *, /, MOD, +, -, <, >, <=, >=, =, <>, &, AND, XOR, OR

See Also [FastLinx for Mitsubishi Tag Browser Properties](#)
[Reserved Names for Mitsubishi FastLinx Variable Tags](#)

Reserved Names for Mitsubishi FastLinx Variable Tags

When defining names for Mitsubishi FastLinx variable tags, do not use names of any of the following Cicode

Name	Op_Code	Type	ArgC
_AlarmDsp	113	INT	(INT, INT, INT, INT)
_AlarmGetFieldRec	374	STRING	(LONG,STRING,LONG)

Name	Op_Code	Type	ArgC
_AlarmQueryFirstRec	176	LONG	(INT,INT,INT,INT)
_AlarmQueryNextRec	177	LONG	(LONG,INT,INT,INT,INT)
_AlarmSetQuery	650	INT	(INT,STRING,STRING)
_CreateControlObject	627	OBJECT	(STRING,STRING,INT,INT,INT,INT,STRING)
_DDERead	136	STRING	(STRING,STRING,STRING,INT)
_DevClose	155	INT	(INT,INT)
_DevOpen	154	INT	(STRING,INT)
_DspAnCreateControlObject	628	OBJECT	(INT,STRING,INT,INT,STRING)
_DspButton	109	INT	(INT, INT, STRING, INT, INT, INT,INT,INT,INT)
_DspButtonFn	322	INT	(INT,FUNCTION,STRING,INT,INT,INT,FUNCTI ON,FUNCTION,INT)
_DspChart	440	INT	(INT,STRING,INT,INT,INT,INT,INT,INT,INT,INT)
_DspExec	425	INT	(STRING, STRING)
_DspGetAnFromPoint	551	INT	(INT,INT,INT)
_DspSetTooltipFont	652	VOID	(STRING,INT,STRING)
_DspSym	73	INT	(INT,STRING,INT)
_DspSymAnm	74	INT	(INT,STRING,STRING,STRING,STRING,STRI NG,STRING,STRING,STRING,INT,STRING)
_DspTrend	72	INT	(INT,STRING,INT,INT,INT,INT,INT,INT,INT,INT)
_DspTrendInfo	375	INT	(STRING,INT,INT)
_ErrGetHw	171	INT	(INT,INT)
_ErrSetHw	170	INT	(INT,INT,INT)
_Exec	85	INT	(STRING,INT)
_ExecuteDTSPkg	651	LONG	(STRING,STRING,STRING,STRING,STRING, STRING,STRING,STRING,STRING,STRING,S TRING,STRING)
_FormComboBox	461	INT	(INT,INT,INT,INT,VAR STRING,INT)
_FormGroupBox	459	INT	(INT,INT,INT,INT,STRING)
_FormListBox	460	INT	(INT,INT,INT,INT,VAR STRING,INT)
_FormSaveAsFile	546	STRING	(STRING,STRING,STRING,STRING)
_KeyReplay	18	INT	(INT)
_ObjectAssociatePropertyWit hTag	633	INT	(OBJECT,STRING,STRING,STRING)
_ObjectCallMethod	629	VARIANT	(OBJECT,STRING,VARARG)
_ObjectGetProperty	631	VARIANT	(OBJECT,STRING)
_ObjectServerInvoke	672	STRING	(STRING,STRING,STRING,STRING,STRING, STRING,STRING,STRING,STRING,STRING,S TRING,STRING)
_ObjectSetProperty	630	INT	(OBJECT,STRING,VARIANT)
_PageDisplay	165	INT	(STRING)
_PageGoto	1	INT	(STRING)

Name	Op_Code	Type	ArgC
_PlotInfo	502	STRING	(INT,INT,STRING)
_Shutdown	92	INT	(STRING,STRING,LONG)
_TableMath	263	REAL	(var REAL,INT,INT,INT)
_TagRead	517	STRING	(STRING,INT)
_TagWrite	516	LONG	(STRING,STRING,INT)
_TaskHnd	332	INT	(STRING)
_TimeSub	119	LONG	(LONG,INT)
_TmGetTable	266	INT	(STRING,LONG,REAL,INT,var REAL,LONG,LONG)
_TmNew	199	INT	(INT,STRING,STRING,STRING,STRING,STR ING,STRING,STRING,STRING,STRING)
_TmScroll	198	INT	(INT,INT,INT,LONG)
_TmSetTable	270	INT	(STRING,LONG,REAL,INT,var REAL, LONG)
_UserCreate	519	INT	(STRING,STRING,STRING,STRING,STRING, STRING,STRING,STRING,STRING,STRING,S TRING,STRING,STRING,STRING)
_UserPassword	521	INT	(STRING,STRING,STRING)
_UserPasswordExpiryDays	666	INT	(STRING,STRING)
_VbCallRun	663	LONG	(OBJECT)
_VbCicodeCallReturn	665	VOID	(OBJECT,VARIANT)
_VbExpressionOpen	662	OBJECT	(STRING)
_Wave	67	INT	(INT,INT,INT,INT,INT)
_WinCopy	321	INT	(REAL,REAL,STRING)
_WinFile	320	INT	(STRING,REAL,REAL,STRING)
_WinPrint	319	INT	(STRING,REAL,REAL,STRING)
_WinPrintFile	385	INT	(STRING,STRING,REAL,REAL,STRING)
_WinTitle	103	INT	(STRING)
Abs	53	REAL	(REAL)
AccControl	529	LONG	(STRING,INT)
AlarmAck	114	INT	(INT, INT)
AlarmAckRec	178	INT	(LONG)
AlarmActive	234	INT	(INT)
AlarmClear	398	INT	(INT,INT)
AlarmClearRec	581	INT	(LONG)
AlarmComment	233	INT	(STRING)
AlarmDelete	316	INT	(INT,INT)
AlarmDisable	128	INT	(INT,INT)
AlarmDisableRec	179	INT	(LONG)
AlarmDspNext	126	INT	(INT)
AlarmDspPrev	127	INT	(INT)
AlarmEnable	129	INT	(INT,INT)

Name	Op_Code	Type	ArgC
AlarmEnableRec	180	INT	(LONG)
AlarmFirstTagRec	181	LONG	(STRING,STRING,STRING)
AlarmGetDelay	680	LONG	(INT)
AlarmGetDelayRec	681	LONG	(LONG,INT)
AlarmGetDsp	359	STRING	(INT,STRING)
AlarmGetInfo	232	LONG	(INT,INT)
AlarmGetOrderbyKey	671	STRING	(INT)
AlarmGetThreshold	483	REAL	(INT)
AlarmGetThresholdRec	482	REAL	(LONG,INT)
AlarmHelp	242	INT	()
AlarmNextTagRec	182	LONG	(LONG,STRING,STRING,STRING)
AlarmSaveType	443	INT	(INT)
AlarmSetDelay	678	INT	(INT,STRING)
AlarmSetDelayRec	679	INT	(LONG,INT,STRING)
AlarmSetInfo	231	INT	(INT,INT,STRING)
AlarmSetPriority	455	INT	(INT)
AlarmSetPriorityRec	456	INT	(LONG,INT)
AlarmSetThreshold	278	INT	(INT, STRING)
AlarmSetThresholdRec	345	INT	(LONG,INT,STRING)
AlarmSplit	267	INT	()
AlarmSumAppend	372	INT	(STRING)
AlarmSumCommit	528	INT	(INT)
AlarmSumDelete	362	INT	(INT)
AlarmSumFind	480	INT	(LONG, LONG)
AlarmSumFirst	355	INT	()
AlarmSumGet	357	STRING	(INT,STRING)
AlarmSumLast	427	INT	()
AlarmSumNext	356	INT	(INT)
AlarmSumPrev	428	INT	(INT)
AlarmSumSet	358	INT	(INT,STRING,STRING)
AlarmSumSplit	361	INT	(INT)
AlarmSumType	657	INT	(INT)
AnByName	638	LONG	(STRING)
ArcCos	48	REAL	(REAL)
ArcSin	47	REAL	(REAL)
ArcTan	49	REAL	(REAL)
AreaCheck	563	INT	(INT)
Ass	512	INT	(INT,INT,STRING,INT)
AssChain	557	INT	(INT,INT,INT)
AssInfo	556	STRING	(INT,INT)

Name	Op_Code	Type	ArgC
AssScaleStr	565	STRING	(INT,INT,INT)
Beep	93	INT	(INT)
CallEvent	106	INT	(INT,INT)
CAPIPost	626	INT	(STRING,INT)
ChainEvent	354	INT	(INT)
CharToStr	29	STRING	(INT)
CitectColourToPackedRGB	639	LONG	(LONG)
CitectInfo	236	LONG	(STRING,STRING,STRING)
ClipCopy	390	INT	(STRING)
ClipPaste	391	STRING	()
ClipReadLn	393	STRING	()
ClipSetMode	544	INT	(INT)
ClipWriteLn	392	INT	(STRING)
ClusterGetName	602	INT	(var STRING, var STRING, INT)
ClusterSetName	601	INT	(STRING,STRING,INT)
CodeSetMode	367	INT	(INT,INT)
CodeTrace	541	INT	(INT,INT)
ComClose	432	INT	(INT)
ComOpen	431	INT	(STRING,INT)
ComRead	433	INT	(INT,var STRING,var LONG,INT)
ComReset	435	INT	(INT)
ComWrite	434	INT	(INT,var STRING,var LONG,INT)
Cos	45	REAL	(REAL)
CreateObject	634	OBJECT	(STRING)
DateAdd	123	LONG	(LONG,LONG)
DateInfo	642	STRING	(INT,INT)
DateSub	124	LONG	(LONG,LONG)
DDEExec	150	INT	(STRING,STRING)
DDEhExecute	489	LONG	(LONG,STRING)
DDEhGetLastError	515	LONG	(LONG)
DDEhInitiate	485	LONG	(STRING,STRING)
DDEhPoke	488	LONG	(LONG,STRING,STRING)
DDEhReadLn	583	STRING	(INT,STRING)
DDEhRequest	487	STRING	(LONG,STRING)
DDEhSetMode	584	INT	(INT,LONG)
DDEhTerminate	486	LONG	(LONG)
DDEhWriteLn	582	INT	(INT,STRING,STRING)
DDEPost	138	STRING	(STRING,STRING)
DDEWrite	137	STRING	(STRING,STRING,STRING,STRING)
Debug	328	INT	(STRING,STRING,STRING,STRING)

Name	Op_Code	Type	ArgC
DebugBreak	603	VOID	()
DegToRad	55	REAL	(REAL)
DevAppend	296	INT	(INT)
DevControl	394	INT	(INT,INT,STRING)
DevCurr	132	INT	()
DevDelete	297	INT	(INT)
DevDisable	153	INT	(STRING,INT)
DevEOF	160	INT	(INT)
DevFind	163	INT	(INT,STRING,STRING)
DevFlush	164	INT	(INT)
DevGetField	143	STRING	(INT,STRING)
DevHistory	366	INT	(INT)
DevInfo	235	STRING	(INT,INT)
DevModify	530	INT	(STRING,STRING,STRING,STRING,INT)
DevNext	156	INT	(INT)
DevOpenGrp	585	INT	(INT,INT)
DevPrev	157	INT	(INT)
DevPrint	133	INT	(INT,STRING,INT)
DevRead	162	STRING	(INT,INT)
DevReadLn	295	STRING	(INT)
DevRecNo	159	LONG	(INT)
DevSeek	158	INT	(INT, LONG)
DevSetField	142	INT	(INT,STRING,STRING)
DevSize	352	LONG	(INT)
DevWrite	161	INT	(INT,STRING)
DevWriteLn	294	INT	(INT,STRING)
DevZap	373	INT	(INT)
DLLCall	371	STRING	(INT,STRING)
DLLCallOEM	567	STRING	(INT,STRING)
DLLClose	370	INT	(INT)
DLLOpen	369	INT	(STRING,STRING,STRING)
DriverInfo	674	STRING	(STRING,INT)
DspAnFree	636	INT	(INT)
DspAnGetArea	615	INT	(INT)
DspAnGetPos	94	INT	(INT, var LONG, var LONG)
DspAnGetPrivilege	614	INT	(INT)
DspAnInfo	262	STRING	(INT,INT)
DspAnInRgn	111	INT	(INT,INT,INT)
DspAnMove	77	INT	(INT,INT,INT)
DspAnMoveRel	78	INT	(INT,INT,INT)

Name	Op_Code	Type	ArgC
DspAnNew	95	INT	(INT, INT)
DspAnNewRel	104	INT	(INT,INT,INT)
DspAnWrite	241	INT	()
DspBar	71	INT	(INT,STRING,INT)
DspBarLoad	466	INT	(STRING)
DspBmp	540	INT	(INT,STRING,INT)
DspCol	75	INT	(INT,INT)
DspDel	76	INT	(INT)
DspDelayRenderBegin	568	INT	()
DspDelayRenderEnd	569	INT	()
DspDirty	125	INT	(INT)
DspError	88	INT	(STRING)
DspFile	212	INT	(INT,INT,INT,INT)
DspFileGetInfo	216	INT	(INT,INT)
DspFileGetName	214	STRING	(INT)
DspFileScroll	215	INT	(INT,INT,INT)
DspFileSetName	213	INT	(INT,STRING)
DspFlushObj	476	INT	(INT)
DspFont	69	INT	(STRING,INT,INT,INT)
DspFontHnd	80	INT	(STRING)
DspFullScreen	333	INT	(INT)
DspGetAnCur	317	INT	()
DspGetAnExtent	552	INT	(INT, var LONG, var LONG, var LONG, var LONG)
DspGetEnv	545	STRING	(STRING)
DspGetMouse	79	INT	(var LONG, var LONG)
DspGetNearestAn	81	INT	(INT,INT)
DspGetParentAn	612	INT	(INT)
DspGetSlider	559	INT	(INT)
DspGetTip	511	STRING	(INT,INT)
DspGrayButton	507	INT	(INT,INT)
DspInfo	130	STRING	(INT,INT,INT)
DspInfoDestroy	145	INT	(INT)
DspInfoField	131	STRING	(INT,STRING,STRING)
DspInfoNew	139	INT	(INT)
DspInfoValid	144	INT	(INT)
DspIsButtonGray	509	INT	(INT)
DspIsVisible	590	INT	(INT)
DspKernel	378	INT	(INT)
DspMarkerMove	442	INT	(INT,INT,INT)

Name	Op_Code	Type	ArgC
DspMarkerNew	441	INT	(INT,INT,INT)
DspMCI	438	STRING	(STRING)
DspPage	68	INT	(STRING,STRING,STRING)
DspPlaySound	437	INT	(STRING,INT)
DspRichText	586	INT	(INT,LONG,LONG,INT)
DspRichTextEdit	596	INT	(INT,INT)
DspRichTextEnable	595	INT	(INT,INT)
DspRichTextGetInfo	592	STRING	(INT,INT)
DspRichTextLoad	587	INT	(INT,STRING)
DspRichTextPgScroll	591	INT	(INT,INT)
DspRichTextPrint	594	INT	(INT,STRING)
DspRichTextSave	593	INT	(INT,STRING)
DspRichTextScroll	588	INT	(INT,INT,INT)
DspRubEnd	533	INT	(var LONG,var LONG,var LONG,var LONG)
DspRubMove	532	INT	(INT,INT)
DspRubSetClip	534	INT	(INT,INT,INT,INT)
DspRubStart	531	INT	(INT,INT,INT)
DspSetSlider	560	INT	(INT,INT)
DspSetTip	510	INT	(INT,STRING)
DspShow	589	INT	(INT,INT)
DspStatus	550	INT	(INT,INT)
DspStr	245	INT	(INT,STRING,STRING)
DspSymAtSize	561	INT	(INT,STRING,INT,INT,INT,INT,INT)
DspSymLoad	467	INT	(STRING)
DspText	70	INT	(INT,INT,STRING)
DspTipMode	514	INT	(INT)
DspTrnLoad	468	INT	(STRING)
DspVerbose	110	INT	(STRING)
DumpKernel	508	VOID	(INT,STRING)
EngToGeneric	122	LONG	(REAL,REAL,REAL)
ErrCom	348	INT	()
ErrDrv	492	STRING	(STRING,STRING,VAR LONG)
ErrHelp	553	INT	(STRING)
ErrInfo	368	STRING	(INT)
ErrLog	148	INT	(STRING)
ErrMsg	346	STRING	(INT)
ErrSet	175	VOID	(INT)
ErrSetLevel	174	INT	(INT)
ErrTrap	172	INT	(INT,INT)
Exp	43	REAL	(REAL)

Name	Op_Code	Type	ArgC
Fact	100	LONG	(LONG)
FileClose	282	INT	(INT)
FileCopy	292	INT	(STRING,STRING,INT)
FileDelete	279	INT	(STRING)
FileEOF	285	INT	(INT)
FileExist	444	INT	(STRING)
FileFind	477	STRING	(STRING,INT)
FileFindClose	649	INT	()
FileGetTime	445	LONG	(INT)
FileMakePath	479	STRING	(STRING,STRING,STRING,STRING)
FileOpen	281	INT	(STRING,STRING)
FileRead	288	STRING	(INT,INT)
FileReadBlock	286	INT	(INT,VAR STRING,INT)
FileReadLn	289	STRING	(INT)
FileReName	280	INT	(STRING,STRING)
FileRichTextPrint	597	INT	(STRING,STRING)
FileSeek	283	LONG	(INT,LONG)
FileSetTime	446	INT	(INT,LONG)
FileSize	284	LONG	(INT)
FileSplitPath	478	INT	(STRING,VAR STRING, VAR STRING, VAR STRING, VAR STRING)
FileWrite	290	INT	(INT,STRING)
FileWriteBlock	287	INT	(INT,var STRING,INT)
FileWriteLn	291	INT	(INT,STRING)
FmtClose	308	INT	(INT)
FmtFieldHnd	313	INT	(INT,STRING)
FmtGetField	310	STRING	(INT,STRING)
FmtGetFieldHnd	312	STRING	(INT,INT)
FmtOpen	307	INT	(STRING,STRING,INT)
FmtSetField	309	INT	(INT,STRING,STRING)
FmtSetFieldHnd	311	INT	(INT,INT,STRING)
FmtToStr	314	STRING	(INT)
FormActive	260	INT	(INT)
FormAddList	462	INT	(STRING)
FormButton	254	INT	(INT,INT,STRING,FUNCTION,INT)
FormCheckBox	458	INT	(INT,INT,STRING,VAR STRING)
FormCurr	258	INT	(var LONG, var LONG)
FormDestroy	261	INT	(INT)
FormEdit	252	INT	(INT,INT,var STRING,INT)

Name	Op_Code	Type	ArgC
FormField	250	INT	(INT,INT,INT,INT,INT,var STRING,STRING,FUNCTION)
FormGetCurrInst	259	INT	(var LONG, var STRING)
FormGetData	349	INT	(INT)
FormGetInst	257	INT	(INT,INT,var LONG, var STRING)
FormGetText	247	STRING	(INT,INT)
FormGoto	376	INT	(INT)
FormInput	253	INT	(INT,INT,STRING,var STRING,INT)
FormListAddText	490	INT	(INT,INT,STRING)
FormListDeleteText	613	INT	(INT,INT,STRING)
FormListSelectText	491	INT	(INT,INT,STRING)
FormNew	249	INT	(STRING,INT,INT,INT)
FormOpenFile	209	STRING	(STRING,STRING,STRING)
FormPassword	255	INT	(INT,INT,STRING,var STRING,INT)
FormPosition	364	INT	(INT,INT,INT)
FormPrompt	251	INT	(INT,INT,STRING)
FormRadioButton	457	INT	(INT,INT,STRING,VAR STRING)
FormRead	246	INT	(INT)
FormSelectPrinter	547	STRING	()
FormSetData	363	INT	(INT)
FormSetInst	256	INT	(INT,INT,LONG,STRING)
FormSetText	248	INT	(INT,INT,STRING)
FormWndHnd	377	INT	(INT)
FtpClose	618	INT	(INT)
FtpFileCopy	619	INT	(INT,STRING,STRING)
FtpFileFind	616	STRING	(INT,STRING)
FtpOpen	617	INT	(STRING,STRING,STRING)
FullName	141	STRING	()
FuzzyClose	605	INT	(LONG)
FuzzyGetCodeValue	609	LONG	(LONG,INT,var LONG)
FuzzyGetShellValue	607	REAL	(LONG,INT,var LONG)
FuzzyOpen	604	LONG	(STRING)
FuzzySetCodeValue	608	INT	(LONG,INT,LONG)
FuzzySetShellValue	606	INT	(LONG,INT,REAL)
FuzzyTrace	610	INT	(LONG,INT)
GetArea	108	INT	()
GetEnv	387	STRING	(STRING)
GetEvent	353	INT	(INT)
GetGlbBool	21	INT	(INT)
GetGlbFlt	20	REAL	(INT)

Name	Op_Code	Type	ArgC
GetGlbInt	19	LONG	(INT)
GetGlbStr	22	STRING	(INT)
GetPriv	386	INT	(INT,INT)
GetWinTitle	677	STRING	()
GraphBox	417	INT	(INT,INT,INT,INT,INT,INT,INT)
GraphClose	420	INT	()
GraphGetInfo	421	STRING	(INT,STRING)
GraphGrid	418	INT	(INT,INT,INT,INT,INT,INT)
GraphLine	416	INT	(INT,INT,INT,INT,var REAL,REAL,REAL,INT)
GraphMarker	424	INT	(INT,INT,INT,INT,INT)
GraphOpen	415	INT	(STRING,INT,INT)
GraphScaleMarker	423	INT	(INT,INT,INT,INT)
GraphText	419	INT	(INT,INT,INT,INT,STRING)
GrpClose	335	INT	(INT)
GrpDelete	337	INT	(INT,INT)
GrpFirst	339	INT	(INT)
GrpIn	338	INT	(INT,INT)
GrpInsert	336	INT	(INT,INT)
GrpMath	341	INT	(INT,INT,INT,INT)
GrpName	344	STRING	(INT)
GrpNext	340	INT	(INT,INT)
GrpOpen	334	INT	(STRING,INT)
GrpToStr	343	STRING	(INT)
Halt	169	INT	()
HexToStr	470	STRING	(LONG,INT)
HighByte	63	LONG	(LONG)
HighWord	65	LONG	(LONG)
Input	207	STRING	(STRING,STRING,STRING)
IntToReal	493	REAL	(LONG)
IntToStr	30	STRING	(LONG)
IODeviceControl	397	INT	(STRING,INT,STRING)
IODeviceInfo	396	STRING	(STRING,INT)
IsError	173	INT	()
KerCmd	473	INT	(STRING,STRING)
KeyAllowCursor	210	INT	(INT,INT)
KeyBS	12	INT	()
KeyGet	16	INT	()
KeyGetCursor	204	INT	()
KeyMove	13	INT	(INT)
KeyOEM	484	INT	(STRING,INT)

Name	Op_Code	Type	ArgC
KeyPeek	107	INT	(INT)
KeyPut	15	INT	(INT)
KeyPutStr	17	INT	(STRING)
KeySetCursor	14	INT	(INT)
KeySetSeq	116	INT	(STRING,INT,FUNCTION)
KeySetType	117	INT	(INT)
LanguageFileTranslate	611	INT	(STRING,STRING)
LineAnswer	623	INT	(INT)
LineClose	621	INT	(INT)
LineDrop	624	INT	(INT)
LineInfo	625	STRING	(INT, LONG)
LineMakeCall	622	INT	(INT, STRING)
LineOpen	620	INT	(STRING, LONG)
Ln	50	REAL	(REAL)
Log	51	REAL	(REAL)
Login	89	INT	(STRING, STRING)
Logout	90	INT	()
LowByte	62	LONG	(LONG)
LowWord	64	LONG	(LONG)
MailError	454	INT	()
MailLogoff	451	INT	()
MailLogon	450	INT	(STRING, STRING, INT)
MailRead	453	INT	(VAR STRING, VAR STRING, VAR STRING, VAR STRING, INT)
MailSend	452	INT	(STRING, STRING, STRING, STRING, INT)
Max	60	REAL	(REAL, REAL)
Message	208	INT	(STRING, STRING, INT)
Min	59	REAL	(REAL, REAL)
MsgBrdcst	276	INT	(STRING, INT, STRING)
MsgClose	272	INT	(STRING, INT)
MsgGetCurr	293	INT	()
MsgOpen	271	INT	(STRING, INT, FUNCTION)
MsgRead	273	INT	(var LONG, var STRING)
MsgRPC	275	STRING	(INT, STRING, STRING, INT)
MsgState	667	INT	(INT)
MsgWrite	274	INT	(INT, LONG, STRING)
Name	140	STRING	()
ObjectAssociateEvents	632	INT	(STRING, OBJECT)
ObjectByName	635	OBJECT	(STRING)
ObjectSaveState	641	INT	(OBJECT, STRING)

Name	Op_Code	Type	ArgC
ObjectServerInvokeEx	673	STRING	(STRING,STRING)
OnEvent	105	INT	(INT,FUNCTION)
PackedRGBToCitectColour	640	LONG	(LONG)
PageFileInfo	645	INT	(STRING,INT)
PageGetInt	447	LONG	(INT)
PageGetStr	554	STRING	(INT)
PageInfo	102	STRING	(INT)
PageLast	4	INT	()
PageNext	2	INT	()
PagePeekLast	168	STRING	(INT)
PagePopLast	166	STRING	()
PagePrev	3	INT	()
PagePushLast	167	INT	(STRING)
PageSetInt	448	INT	(INT,LONG)
PageSetStr	555	INT	(INT,STRING)
ParameterGet	28	STRING	(STRING,STRING,STRING)
ParameterPut	27	INT	(STRING,STRING,STRING)
PathToStr	472	STRING	(STRING)
Pi	66	REAL	()
PlotClose	495	INT	(INT)
PlotDraw	499	INT	(INT,INT,INT,INT,INT,INT,INT,INT,INT,INT)
PlotGetMarker	505	INT	(STRING)
PlotGrid	496	INT	(INT,INT,INT,INT,INT,INT,INT,INT,INT,INT,INT,INT,INT,INT,LONG)
PlotLine	497	INT	(INT,INT,INT,INT,INT,INT,INT,INT,INT,var REAL,REAL,REAL,LONG)
PlotMarker	500	INT	(INT,INT,INT,INT,INT,INT,INT)
PlotOpen	494	INT	(INT,STRING,LONG)
PlotScaleMarker	503	INT	(INT,INT,INT,INT,INT,INT,INT)
PlotSetMarker	504	INT	(INT,STRING)
PlotText	501	INT	(INT,INT,INT,INT,INT,STRING)
PlotXYLine	498	INT	(INT,INT,INT,INT,INT,INT,INT,INT,INT,var REAL,REAL,REAL,var REAL,REAL,REAL,LONG)
PointData	384	STRING	(INT,INT)
PointFree	380	INT	(INT)
PointNew	379	INT	(STRING,INT,INT,LONG,LONG,INT)
PointRead	382	INT	(INT,INT)
PointStatus	383	INT	(INT,INT)
PointWrite	381	INT	(INT,STRING,INT)
PointWriteArrayLong	526	INT	(INT,var LONG,INT)

Name	Op_Code	Type	ArgC
PointWriteArrayReal	527	INT	(INT,var REAL,INT)
Pow	54	REAL	(REAL,REAL)
PrintFont	481	INT	(STRING)
ProjectRestartGet	523	STRING	()
ProjectRestartSet	522	INT	(STRING)
ProjectSet	524	INT	(STRING)
Prompt	87	INT	(STRING)
QueClose	299	INT	(INT)
QueLength	302	INT	(INT)
QueOpen	298	INT	(STRING,INT)
QuePeek	469	INT	(INT,VAR LONG,VAR STRING,INT)
QueRead	301	INT	(INT,var LONG, var STRING, INT)
QueWrite	300	INT	(INT,LONG,STRING)
RadToDeg	56	REAL	(REAL)
Rand	61	INT	(INT)
RdbClose	220	INT	(INT)
RdbEOF	221	INT	(INT)
RdbFind	230	LONG	(INT,STRING,STRING)
RdbFirstRec	224	LONG	(INT)
RdbGet	223	STRING	(INT,STRING)
rdbGetPath	185	STRING	()
RdbLastRec	225	LONG	(INT)
RdbNextRec	226	LONG	(INT)
RdbNoRec	229	LONG	(INT)
RdbOpen	217	INT	(STRING)
RdbOpenPage	218	INT	(STRING)
RdbOpenSub	219	INT	(INT,STRING)
RdbPosRec	228	LONG	(INT,LONG)
RdbPrevRec	227	LONG	(INT)
RdbSet	222	INT	(INT,STRING,STRING)
rdbSetPath	184	INT	(STRING)
RealToStr	31	STRING	(REAL,INT,INT)
RepGetControl	146	LONG	(STRING,INT)
Report	115	INT	(STRING)
RepSetControl	147	LONG	(STRING,INT,LONG)
ReRead	325	VOID	(INT)
Round	58	REAL	(REAL,INT)
RunTest	675	LONG	(STRING)
RunTests	676	INT	()
SemClose	304	INT	(INT)

Name	Op_Code	Type	ArgC
SemOpen	303	INT	(STRING,INT)
SemSignal	305	INT	(INT)
SemWait	306	INT	(INT, LONG)
SendKeys	513	INT	(STRING, STRING)
ServerControl	449	INT	(STRING, INT, INT)
ServerInfo	350	STRING	(STRING, INT)
SetArea	82	INT	(INT)
SetEvent	360	INT	(INT, INT)
SetGlbBool	25	VOID	(INT, INT)
SetGlbFlt	24	VOID	(INT, REAL)
SetGlbInt	23	VOID	(INT, LONG)
SetGlbStr	26	VOID	(INT, STRING)
SetLanguage	599	INT	(STRING, INT)
ShutdownMode	525	LONG	()
Sign	52	REAL	(REAL)
Sin	44	REAL	(REAL)
Sleep	183	INT	(LONG)
SleepMS	566	INT	(LONG)
SPCAlarms	580	INT	(STRING, INT)
SPCClientInfo	573	REAL	(STRING, INT)
SPCClientTableGet	598	INT	(STRING, INT, INT, INT, var REAL, INT)
SPCGetHistogramTable	574	INT	(STRING, INT, var REAL)
SPCGetSubgroupTable	575	INT	(STRING, INT, var REAL)
SPCProcessXRSGet	577	INT	(STRING, var REAL, var REAL, var REAL)
SPCProcessXRSSet	576	INT	(STRING, REAL, REAL, REAL)
SPCSetLimit	436	INT	(INT, INT, REAL, INT)
SPCSpecLimitGet	579	INT	(STRING, var REAL, var REAL)
SPCSpecLimitSet	578	INT	(STRING, REAL, REAL)
SPCSubgroupSizeGet	571	INT	(STRING, var LONG)
SPCSubgroupSizeSet	570	INT	(STRING, INT)
SQLAppend	464	INT	(INT, STRING)
SQLBeginTran	407	INT	(INT)
SQLCommit	408	INT	(INT)
SQLConnect	399	INT	(STRING)
SQLDisconnect	400	INT	(INT)
SQLEnd	403	INT	(INT)
SQLErrMsg	413	STRING	()
SQLExec	401	INT	(INT, STRING)
SQLFieldInfo	406	INT	(INT, INT, var STRING, var LONG)
SQLGetField	404	STRING	(INT, STRING)

Name	Op_Code	Type	ArgC
SQLInfo	414	STRING	(INT,INT)
SQLNext	402	INT	(INT)
SQLNoFields	405	INT	(INT)
SQLNumChange	410	LONG	(INT)
SQLRollBack	409	INT	(INT)
SQLSet	463	INT	(INT,STRING)
SQLTraceOff	412	INT	()
SQLTraceOn	411	INT	(STRING)
Sqrt	42	REAL	(REAL)
StrClean	98	STRING	(STRING)
StrFill	41	STRING	(STRING,INT)
StrFormat	121	STRING	(REAL,INT,INT,STRING)
StrGetChar	429	INT	(var STRING,INT)
StrLeft	32	STRING	(STRING,INT)
StrLength	36	INT	(STRING)
StrLower	40	STRING	(STRING)
StrMid	34	STRING	(STRING,INT,INT)
StrPad	347	STRING	(STRING,STRING,INT)
StrRight	33	STRING	(STRING,INT)
StrSearch	35	INT	(INT,STRING,STRING)
StrSetChar	430	INT	(var STRING,INT,INT)
StrToChar	96	INT	(STRING)
StrToDate	135	LONG	(STRING)
StrToFmt	315	INT	(INT,STRING)
StrToGrp	342	INT	(INT,STRING)
StrToHex	471	INT	(STRING)
StrToInt	37	LONG	(STRING)
StrToLines	668	STRING	(STRING,LONG,var LONG)
StrToLocalText	600	STRING	(STRING)
StrToPeriod	211	LONG	(STRING)
StrToReal	38	REAL	(STRING)
StrToTime	134	LONG	(STRING)
StrToValue	57	REAL	(STRING)
StrTrim	97	STRING	(STRING)
StrUpper	39	STRING	(STRING)
StrWord	99	STRING	(var STRING)
SwitchConfig	548	INT	(INT)
SysTime	83	LONG	()
SysTimeDelta	84	LONG	(var LONG)
TableLookUp	323	INT	(var REAL,INT,REAL)

Name	Op_Code	Type	ArgC
TableShift	324	INT	(var REAL,INT,INT)
TagInfo	562	STRING	(STRING,INT)
TagScaleStr	564	STRING	(STRING,INT,INT)
Tan	46	REAL	(REAL)
TaskGetSignal	656	INT	(INT)
TaskKill	329	INT	(INT)
TaskNew	149	INT	(STRING,STRING,INT)
TaskResume	331	INT	(INT)
TaskSetSignal	655	INT	(INT,INT)
TaskSuspend	330	INT	(INT)
TimeCurrent	118	LONG	()
TimeInfo	643	STRING	(INT)
TimeSet	474	INT	(LONG,INT)
TimeToStr	120	STRING	(LONG,INT)
TimeToStrFmt	475	STRING	(LONG,STRING)
TraceMsg	86	INT	(STRING)
TrnAddHistory	237	INT	(STRING)
TrnClientInfo	572	STRING	(INT,INT,INT,STRING,var LONG)
TrnDelete	200	INT	(INT)
TrnDelHistory	238	INT	(STRING)
TrnEcho	201	INT	(INT,INT)
TrnEventGetTable	538	INT	(STRING,LONG,LONG,INT,var REAL,var LONG,INT)
TrnEventGetTableMS	647	INT	(STRING,LONG,LONG,INT,var REAL,var LONG,INT,var LONG)
TrnEventSetTable	539	INT	(STRING,LONG,LONG,INT,var REAL,var LONG)
TrnEventSetTableMS	648	INT	(STRING,LONG,LONG,INT,var REAL,var LONG,var LONG)
TrnFlush	277	INT	(STRING)
TrnGetBufEvent	537	LONG	(INT,INT,INT)
TrnGetBufMSTime	637	LONG	(INT, INT, INT)
TrnGetBufTime	240	LONG	(INT,INT,INT)
TrnGetBufValue	239	REAL	(INT,INT,INT)
TrnGetCursorEvent	536	LONG	(INT,INT)
TrnGetCursorMSTime	646	LONG	(INT,INT)
TrnGetCursorPos	269	INT	(INT)
TrnGetCursorTime	196	LONG	(INT,INT)
TrnGetCursorValue	197	REAL	(INT,INT)
TrnGetCursorValueStr	203	STRING	(INT,INT,INT)
TrnGetDefScale	422	INT	(STRING,var REAL, var REAL)

Name	Op_Code	Type	ArgC
TrnGetDisplayMode	654	LONG	(INT,INT)
TrnGetEvent	535	LONG	(INT,INT,INT)
TrnGetFormat	192	INT	(INT,INT,var LONG,var LONG)
TrnGetGatedValue	659	REAL	()
TrnGetInvalidValue	660	REAL	()
TrnGetMode	243	INT	(INT,INT)
TrnGetMSTime	644	LONG	(INT,INT,INT)
TrnGetPen	206	STRING	(INT,INT)
TrnGetPenFocus	244	INT	(INT)
TrnGetPenNo	426	INT	(INT,STRING)
TrnGetPeriod	187	REAL	(INT)
TrnGetScale	191	REAL	(INT,INT,INT)
TrnGetScaleStr	202	STRING	(INT,INT,INT,INT)
TrnGetSpan	543	LONG	(INT)
TrnGetTime	189	LONG	(INT,INT,INT)
TrnGetUnits	193	STRING	(INT,INT)
TrnInfo	558	STRING	(STRING,INT)
TrnIsValidValue	658	INT	(REAL)
TrnSelect	549	INT	(INT,STRING,INT)
TrnSetCursor	195	INT	(INT,INT)
TrnSetCursorPos	268	INT	(INT,INT)
TrnSetDisplayMode	653	INT	(INT,INT,INT)
TrnSetEvent	506	INT	(INT,INT,INT)
TrnSetPen	205	INT	(INT,INT,STRING)
TrnSetPenFocus	194	INT	(INT,INT)
TrnSetPeriod	186	INT	(INT,REAL)
TrnSetScale	190	INT	(INT,INT,INT,REAL)
TrnSetSpan	542	INT	(INT,INT)
TrnSetTime	188	INT	(INT,INT,INT)
UserDelete	520	INT	(STRING)
UserInfo	395	STRING	(INT)
UserSetStr	669	INT	(STRING,STRING,STRING)
UserUpdateRecord	670	INT	()
VbCallOpen	661	OBJECT	(STRING,VARARG)
VbCallReturn	664	VARIANT	(OBJECT)
Version	0	STRING	(INT)
WhoAml	91	INT	()
WinFree	9	INT	()
WinGetFocus	465	INT	()
WinGetWndHnd	11	INT	()

Name	Op_Code	Type	ArgC
WinGoto	5	INT	(INT)
WinMode	318	INT	(INT)
WinMove	10	INT	(INT,INT,INT,INT)
WinNew	8	INT	(STRING)
WinNewAt	351	INT	(STRING,INT,INT,INT)
WinNext	6	INT	()
WinNumber	101	INT	()
WinPos	264	INT	(INT,INT)
WinPrev	7	INT	()
WinSelect	112	INT	(INT)
WinSize	265	INT	(INT,INT)
WndFind	152	INT	(STRING)
WndGetFileProfile	389	STRING	(STRING,STRING,STRING,STRING)
WndGetProfile	327	STRING	(STRING,STRING,STRING)
WndHelp	439	INT	(STRING,INT,STRING)
WndInfo	365	INT	(INT)
WndPutFileProfile	388	INT	(STRING,STRING,STRING,STRING)
WndPutProfile	326	INT	(STRING,STRING,STRING)
WndShow	151	INT	(INT,INT)
WndViewer	518	INT	(STRING,INT,STRING)

See Also [FastLink for Mitsubishi Tag Browser Properties](#)

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

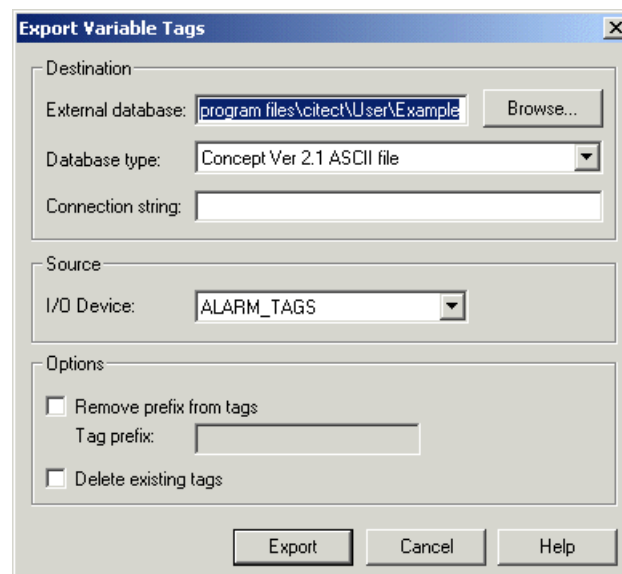
The Export feature allows you to export I/O device data to an external data source, specifying the destination and format of your choice (e.g. RSLOGIX driver). This file might then be imported into a third party I/O device programming package database. Alternatively, you might use it as a backup.

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.

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.

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
- CSV (comma-separated values)
- Concept Ver 2.1 ASCII file
- Mitsubishi MxChange (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.

- 1 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

- 1 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 MxChange
 - **Connection string:** UserID=Citect;Password=Citect

The Connection String must be in the format which the specialized driver (in this case Mitsubishi MxChange) 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. 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”, “Totype”) 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, you must 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), you must 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 else in the file you need to 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.

Character in format file	Matches what string in external data source
\	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, there are two special characters that 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.
\	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 “BJ6542:78”. 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:\Citect\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.

Tagdriv.ini is divided into sections. The first section is the [External data sources] section, and it has 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 MxChange
```

```
[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 causes the following menu to be generated when importing or exporting tags:



The 4 entries in the menu match the strings on the right of the “=” sign in the [External data sources] section.

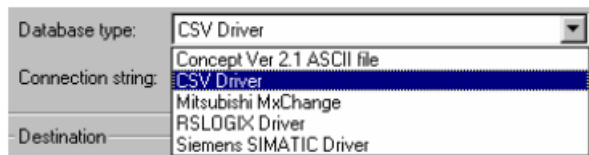
If you add another format file, you will also need to add a matching entry to Tagdriv.ini. For example, if you add a new format file for a Simatic data source, you will need to add a line similar to the following to the [External data sources] section:

```
SIMATIC = Siemens SIMATIC Driver
```

You will also need to 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 as follows:



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.

Chapter 5: Understanding Object Types

Citect has fifteen different object types, each with its 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 Button Objects](#)
[Using Symbol Set Objects](#)
[Using Trend Objects](#)
[Using Cicode Objects](#)
[Using Pasted Symbol Objects](#)
[Using ActiveX Objects](#)

Using Free Hand Line Objects

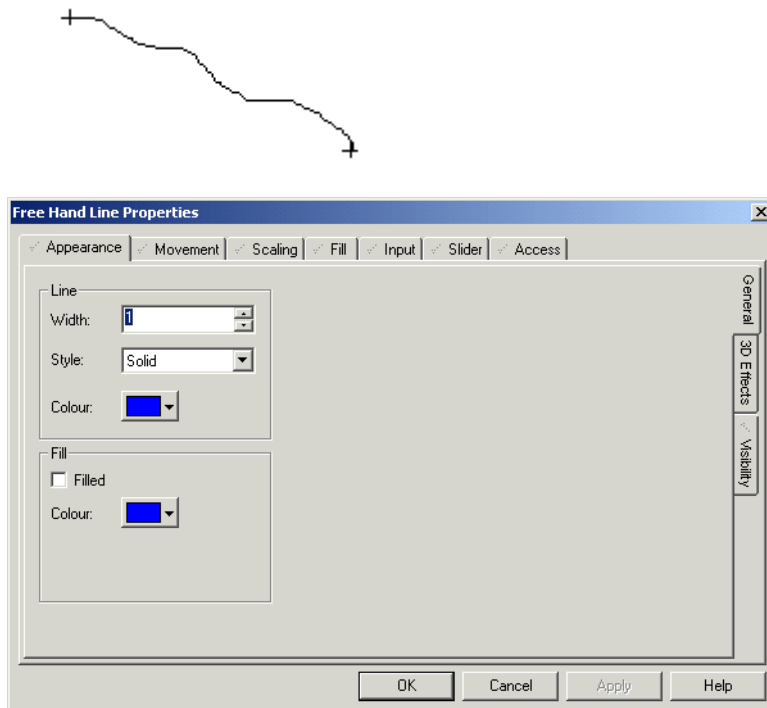
The Free Hand Line tool allows you to draw lines. Lines can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

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.

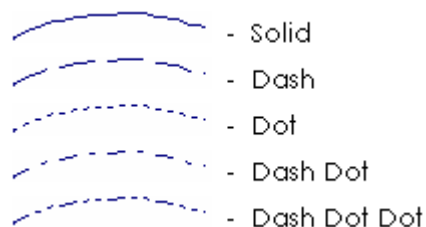
[Line] Width

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.

Note: 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:



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.

**[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

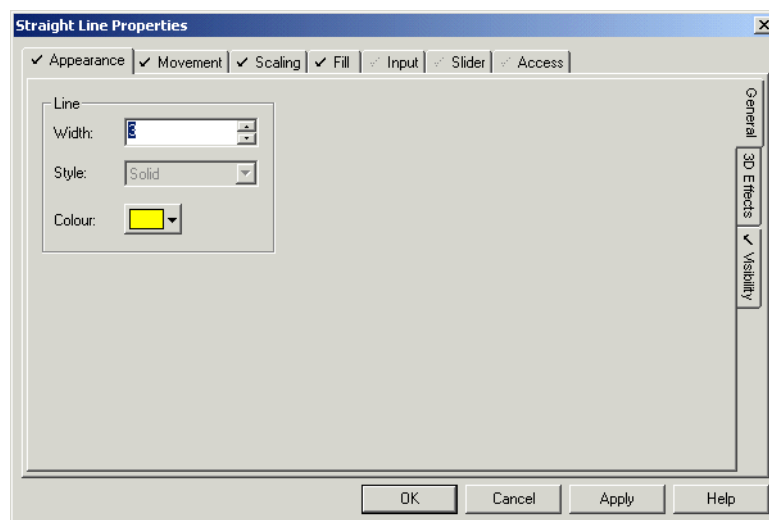
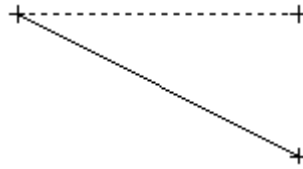
The Straight Line tool allows you to draw straight lines. Straight lines can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like any other type of object.

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.

Note: 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:

- 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.

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.

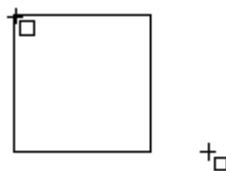


- 2 Move the cursor to where you want the rectangle to start.
- 3 Click and drag the mouse to the opposite corner of the rectangle and release the mouse button. If you hold the **Shift** key before you start drawing the rectangle, it is drawn from its center outwards.

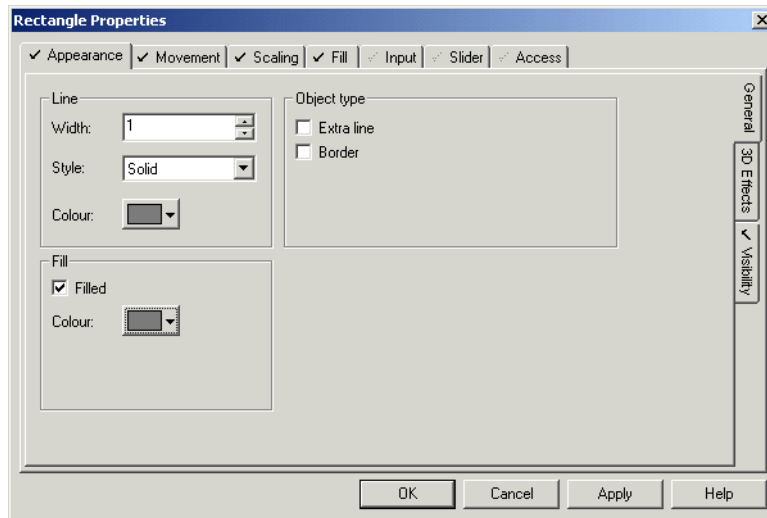


To draw a square:

- 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.

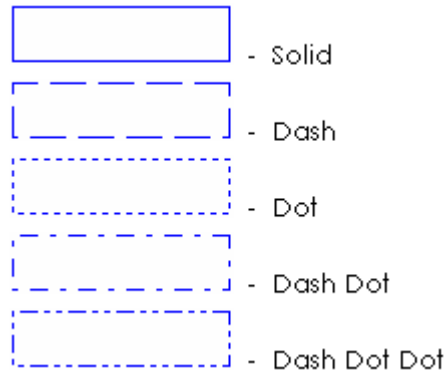
[Line] Width

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.

Note: 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).

[Object type] Border

Adds an extra line (1 pixel width) of black to the perimeter of the rectangle.

For help on the remaining properties tabs, see [Defining Common Object Properties](#).

Using Ellipse Objects

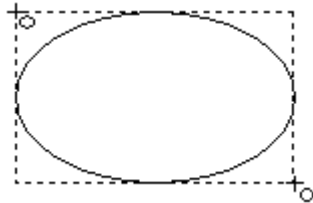
You use the Ellipse tool to draw ellipses, circles, arcs, and pie-slices. Ellipse objects can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

To draw an ellipse:

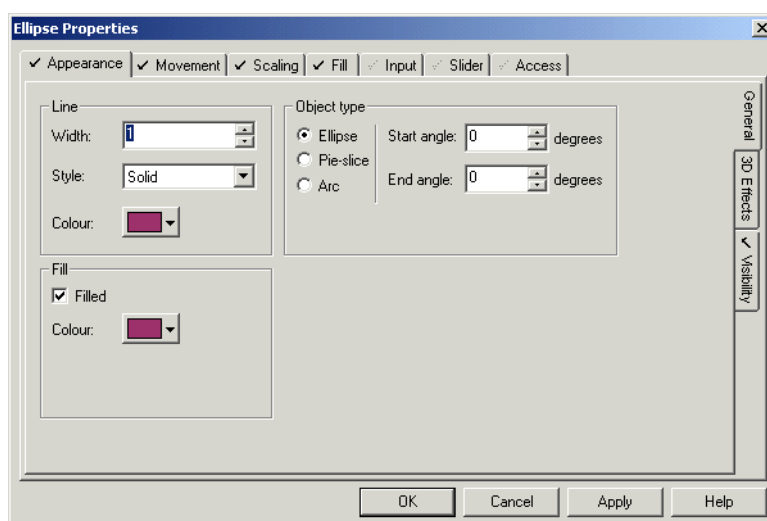
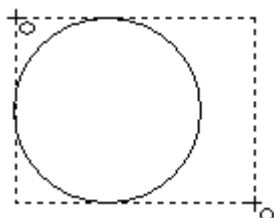
- 1 Click the **Ellipse** tool.



- 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).

[Object type] Ellipse

Select this radio button if you want to 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

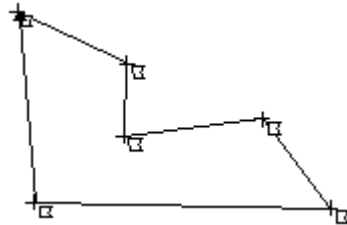
Use the Polygon tool to draw polygons and polylines. Polygons can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

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).



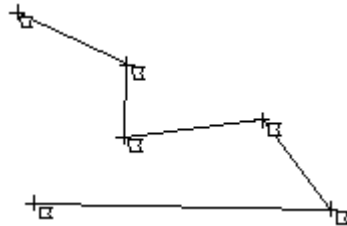
- 5 To complete the polygon, double-click the mouse button.

Note: To draw horizontally or vertically only, hold the Ctrl key down when you are drawing the polygon.

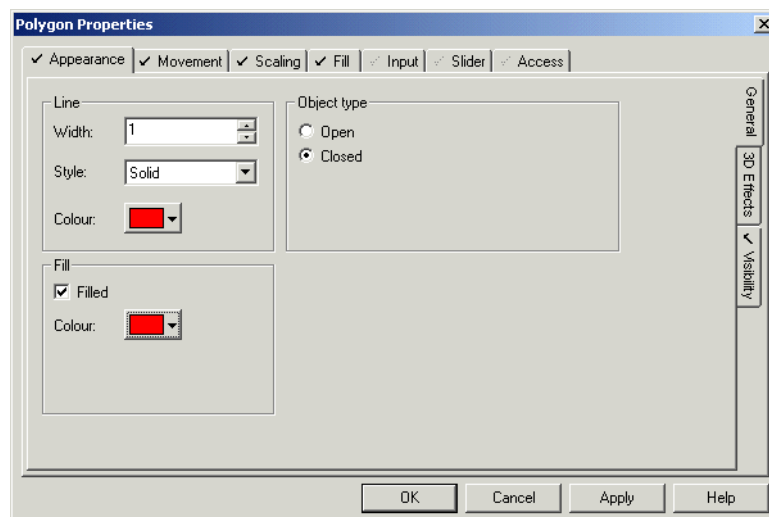
To draw a polyline:

- 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.

Note: 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).

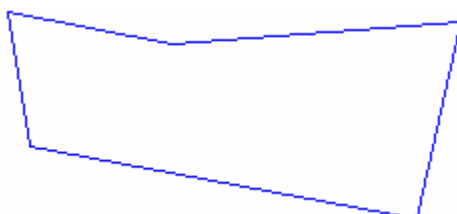
[Object type] Open

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

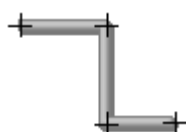
Use the Pipe tool to draw pipes with automatic three-dimensional shading. Pipes can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

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).



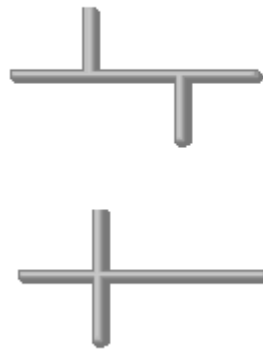
- 5 To complete the pipe, double-click the mouse button.

Hint: To draw horizontally or vertically only, hold the **Ctrl** key down when drawing the pipe.

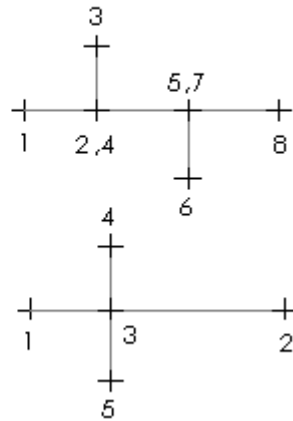
Drawing Complex Pipe Arrangements

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

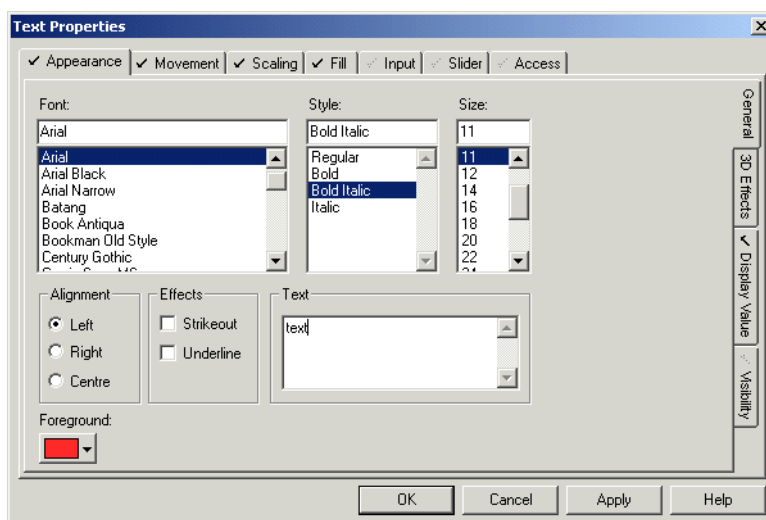
Use the Text tool to type text on the page. Text can be moved, resized, reshaped, brought to the front and so on, and its properties edited just like other types of object.

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](#)

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](#).

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

Check this box to make the text will appear with a line through it.

[Effect] Underline

Check this box to underline the text.

Text

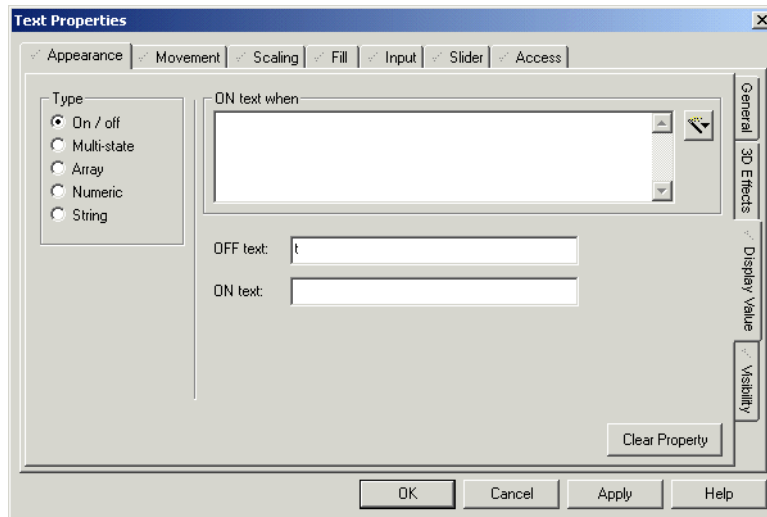
This field contains the text that will display on the page. You can enter any keyboard character(s). It is useful to edit text at this field, as you can apply text changes at the same time as you apply other font and color changes.

Note: This text changes automatically dependig 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.

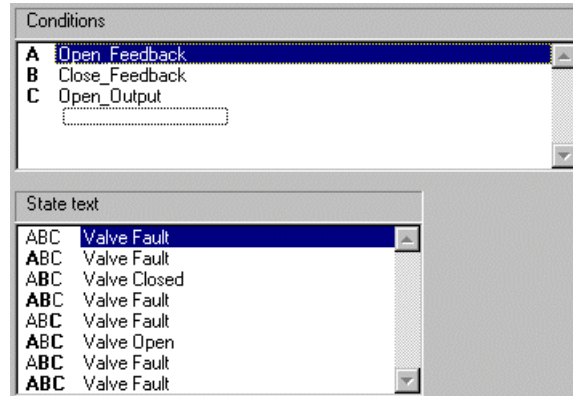
The default number of conditions is 3, but you can add more (to a maximum of 5 conditions, providing 32 combinations), using the **Add** button. You can also delete conditions using the **Delete** button, but there must always be a condition in this field. To enter a condition, click the relevant line (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:

The screenshot shows a configuration window with two main sections. The top section, labeled 'Array expression', contains a text box with the value 'MOTOR_STATUS' and a 'Wizard' button to its right. The bottom section, labeled 'Array text', contains a list box with a scroll bar. The list box has eight items, indexed 0 through 7. Item 0 is 'Running', item 1 is 'Starting', item 2 is 'Stopping', item 3 is 'Stopped', item 4 is 'Faulted', item 5 is 'Isolated', and items 6 and 7 are empty. Item 0, 'Running', is currently selected and highlighted in blue.

Index	Text
0	Running
1	Starting
2	Stopping
3	Stopped
4	Faulted
5	Isolated
6	
7	

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 Button Objects

Use the Button tool to draw buttons on [graphics page](#). You can then assign security rights and attach commands to it.

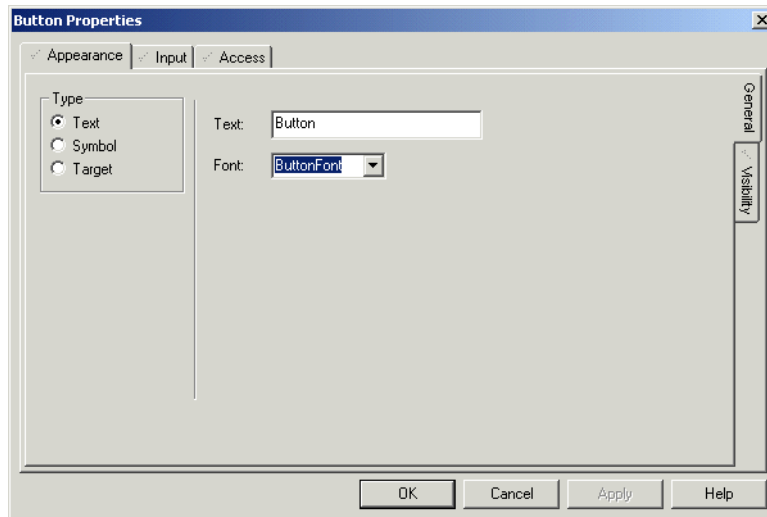
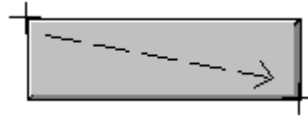
Buttons can be moved, resized, reshaped, brought to the front, and so on, and its properties edited just like other types of object.

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.

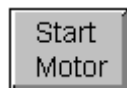
[Type] Text

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.

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.

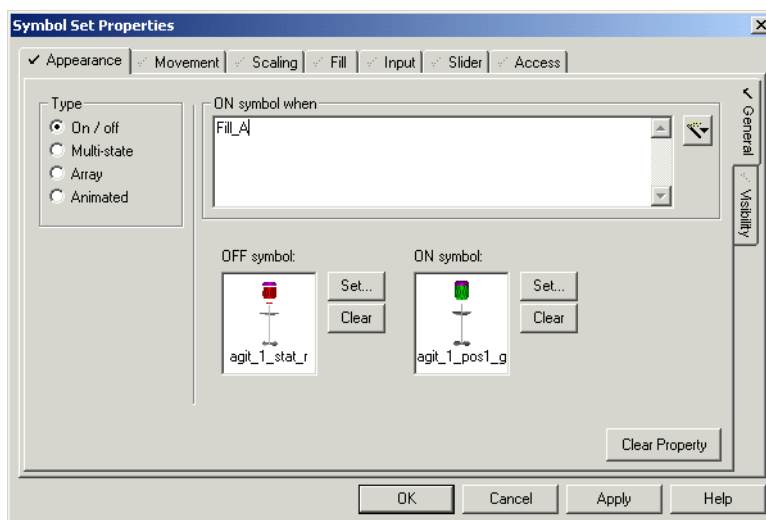
To add a Symbol Set:

- 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**.

Note: 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.

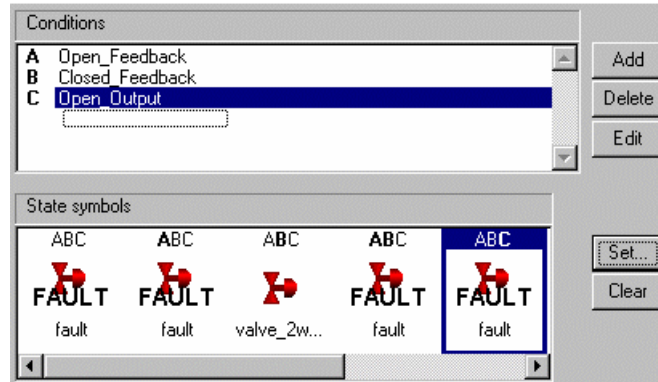
To enter a condition, click the relevant line (A, B, C, etc.), and click **Edit**. You can add more conditions (to a maximum of 5, providing 32 combinations), using the **Add** button. To insert a tag or function, click the **Wizard** button. This button displays two options; **Insert Tag** and **Insert 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:

The screenshot shows a software interface for configuring an array of symbols. At the top, the 'Array expression' field contains the text 'MOTOR_STATUS'. Below this, the 'Array symbols' section displays five numbered slots (0 to 4). Slot 0 is selected and shows a green motor icon with the label 'running'. Slot 1 shows a cyan motor icon with 'starting'. Slot 2 shows a purple motor icon with 'stopping'. Slot 3 shows a red motor icon with 'stopped'. Slot 4 shows a red motor icon with a fault symbol and the labels 'FAULT' and 'fault'. To the right of the slots are two buttons: 'Set...' and 'Clear'.

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.

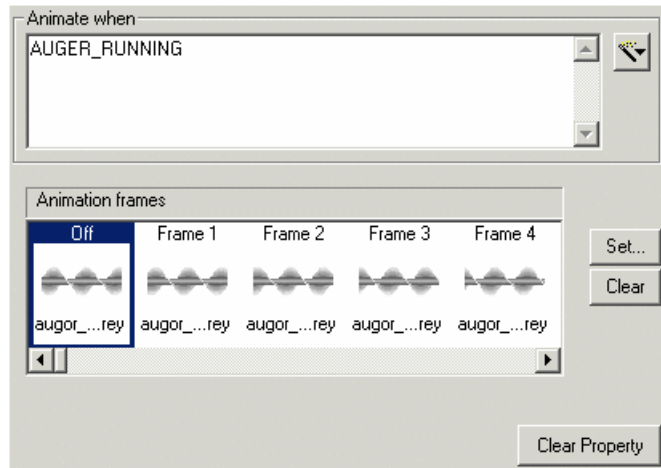
To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options; **Insert tag**, and **Insert Function**.

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.

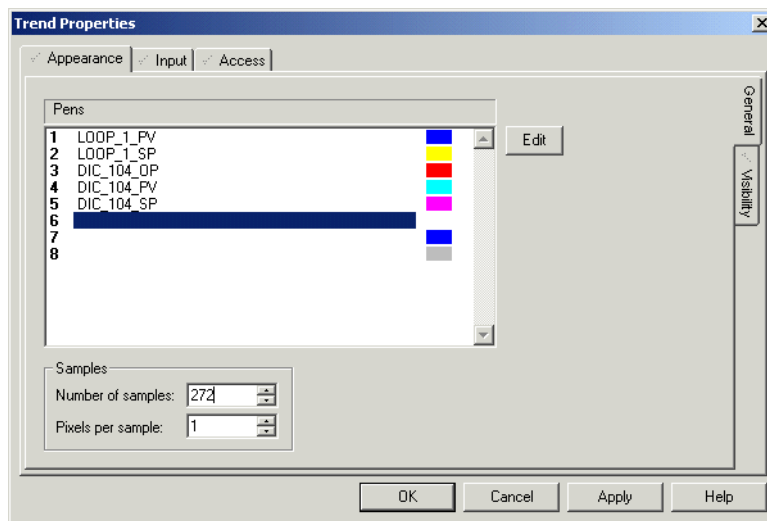
To add a trend to a page:

- 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](#)
[Understanding Object Types](#)

Trend properties

Trends have the following general appearance properties:

Pens (31 Chars.)

The pens (including color) to be displayed on the graph. You can use up to eight pens.

Double-clicking a selected pen or clicking the **Edit** button allows you to change the trend tag and pen color. To insert a trend tag, click the **Wizard** button.

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](#).

Hint: 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. Simply hold down the **Control** (CTRL) key and double-click the object. Alternatively, you can select Goto Object from the Tools menu, select the object, and 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 which are easily configured through Genies. Use the Genies rather than defining these trend tags for yourself. The following table lists all 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
<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

Note: If you are using the PageTrend() function to display this trend page, leave these fields blank.

Display all Trend Types as Periodic

When ticked, 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 unchecked, event and periodic pens will have different styles and must be displayed on separate graphs.

Note: This option is set by default in the predefined CitectHIM/SCADA 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 the resolution of your display. The width of a trend object is equal to **Pixels per sample** x **Number of samples**.

Hint: For a meaningful trend graph, **Pixels per sample** x **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** x **Number of samples**. The default is 1 pixel.

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](#).

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 etc.). 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 etc., and its properties can be edited, just like any other type of object.

To add a Cicode Object to a page:

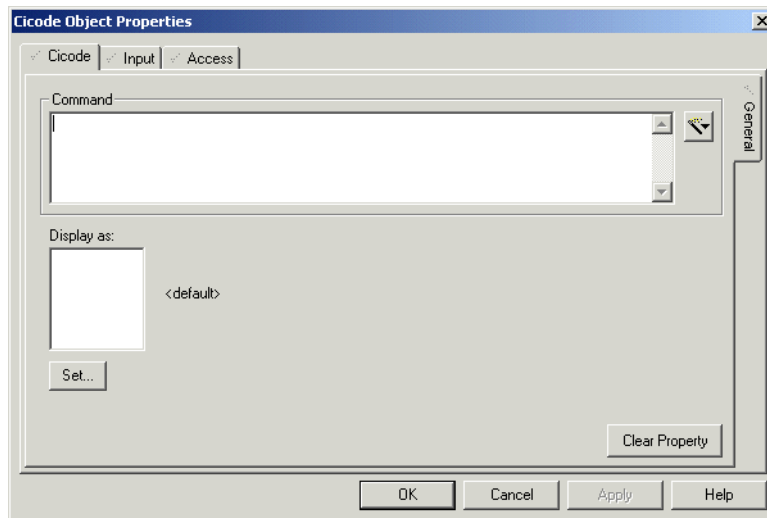
- 1 Click the **Cicode Object** tool,



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).

Note: 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 further 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, re-sized, re-shaped, brought to the front etc., and its properties can be edited, just like any other type of object

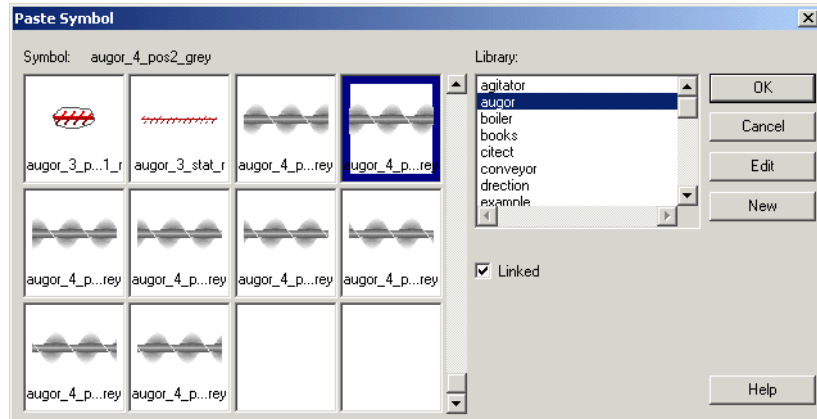
Pasted [library] symbols can be linked to their source, so that any changes made to the original are also made to the pasted symbol.

Note: 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, you can select Goto Object from the Tools menu, select the object, and 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).

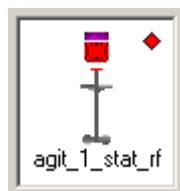
The Paste Symbol dialog box has the following properties:

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

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.

Note: If you have selected **Paste Symbol as Flashing**, two dialog boxes will appear in sequence, allowing you to choose two images that you'd like to implement as a flashing symbol. The **Primary Select Symbol** dialog allows you to select the initial image used, the **Flashing Select Symbol** dialog allows you to choose 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.

Symbol Properties - Appearance (General)

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 re-sized, rotated, moved, copied, duplicated, pasted, brought to the front etc.

Note: 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 CiRecipe 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 is dependent on the individual design of each ActiveX object. This is determined by the amount of flexibility and support its creator has included.

For example, the ActiveX Calendar Control included in the Example project will include "General", "Font", and "Color" tabs when its Appearance properties are displayed. In comparison, the ActiveX Shortcut Control will include "Button", "Shortcut", and "Fonts".

For instructions on how to configure these additional tabs, refer to the documentation provided with the ActiveX object.

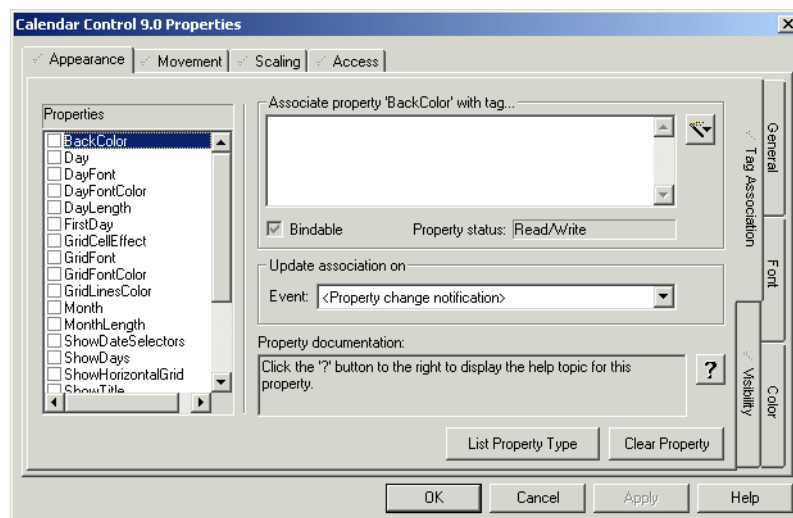
See Also [Tag Association](#)

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 will 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, you will be able to 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 see a list of available events that can be used to trigger a tag association update.

The available events that can be used with a particular property are pre-defined 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 come with some form of documentation to explain the object's controls and functionality. Some have a separate Help file included, others might have simple text prompts that briefly explain each property. Again, this depends entirely on what an object's creator has included.

The Property documentation field will display help information for a selected property, or will give appropriate instructions to obtain the Help required for a selected property. In most cases, the following message will appear:

The **Help** button displays the ActiveX object Help file (if one is included), usually with the topic displayed that relates to the selected property. It should also provide information about the settings provided on any 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**. If you want to clear a tag association for a particular object property, you just need to uncheck the box to the left of the property.

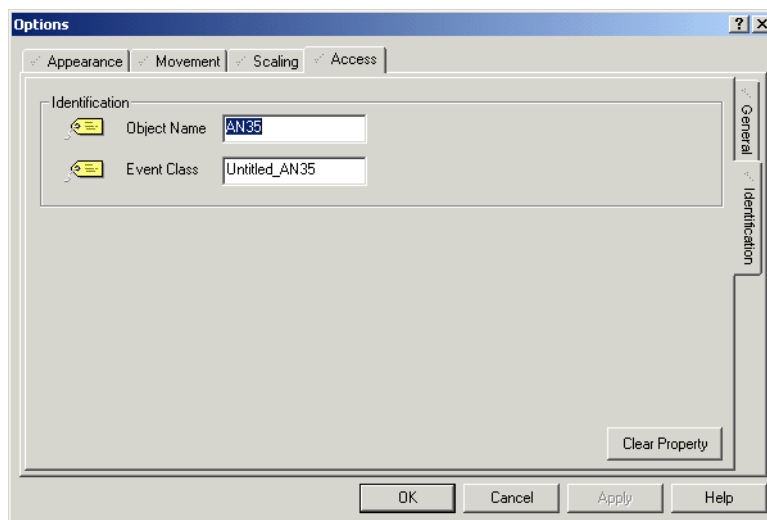
If you accidentally click the Clear Property button, you can restore your tag associations by clicking the Cancel button and reopening the ActiveX properties dialog.

Object Identification

You can identify your ActiveX object.

To identify your ActiveX object:

- 1 From Graphics Builder, double-click the Active X object you want to identify. 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 (32 Chars.)

This field allows you to assign a name to your ActiveX object. It will be used to identify 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 (16 Chars.)

Allocate a name for the event class of your ActiveX object. You can then use this name to create a Cicode function to trap an event.

Note: Don't change the default value if you want to access the ActiveX object using CitectVBA. If you do, CitectVBA won't be able to 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

Check this box to allow changes made to the control to be persisted between pages. For example, you can check this 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 Citect runtime is shutdown and restarted, the updated data will no longer be available.

Note: The data persisted is dependent on the ActiveX controls persistence implementation. Some controls will not persist certain data, therefore that data cannot be saved away and restored.

Chapter 6: Defining Common Object Properties

See Also

The properties described here are those that are common to several object types. Some are also common to object groups.

[3D Effects](#)

[Visibility](#)

[Movement](#)

[Scaling](#)

[Fill Color](#)

[Fill Level](#)

[Touch Commands](#)

[Keyboard Commands](#)

[Sliders](#)

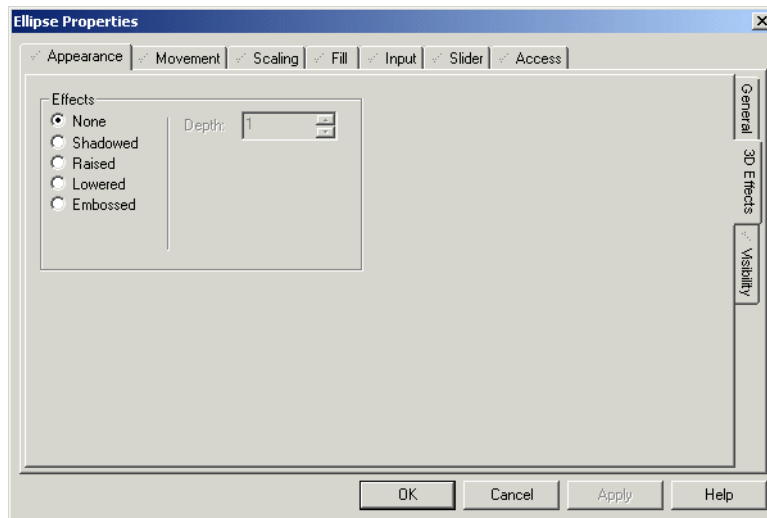
[Access](#)

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:



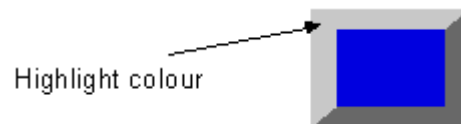
- Depth = 5



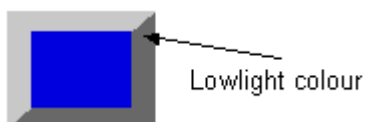
- Depth = 15

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

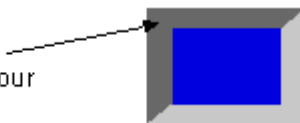
The color of the directly illuminated “edges” of the object.



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.

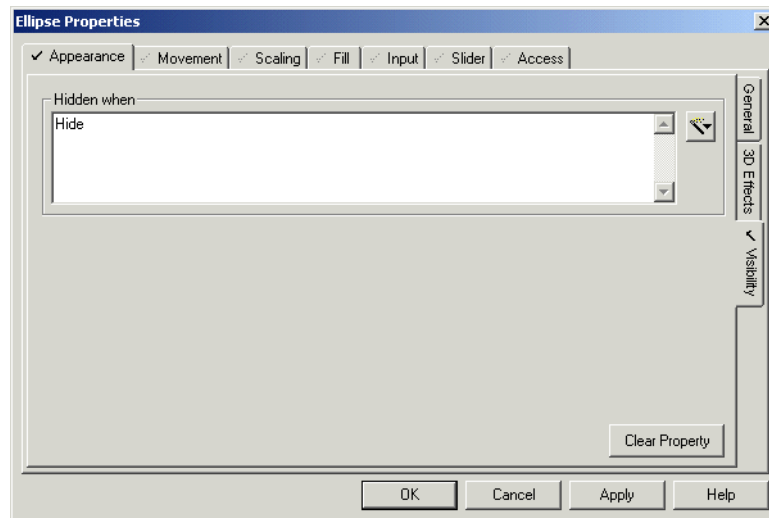
Visibility

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.

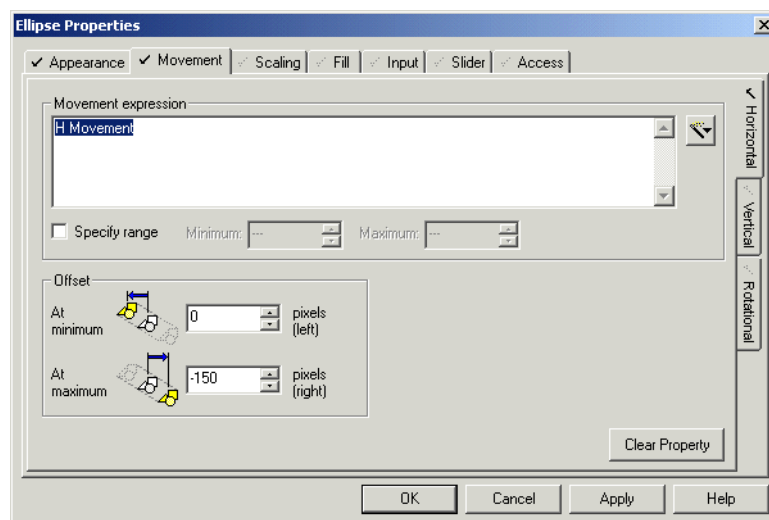
Click **Clear Property** to clear property details and disable the property.

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 **Movement 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 **Movement expression** returns its minimum 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.

Hint: 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.

Hint: 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.

Moving

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° to 60° , 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° , 'up' is actually 15° (not 0°), and 'down' is actually 195° (not 180°). When the group has rotated 50° , 'up' is 50° , and 'down' is 230° , 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° – 60° , and one of its objects is configured to

rotate from 90°–0°, 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° to the right, and the object rotates 90° to the left, the object has only rotated 30° 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](#)

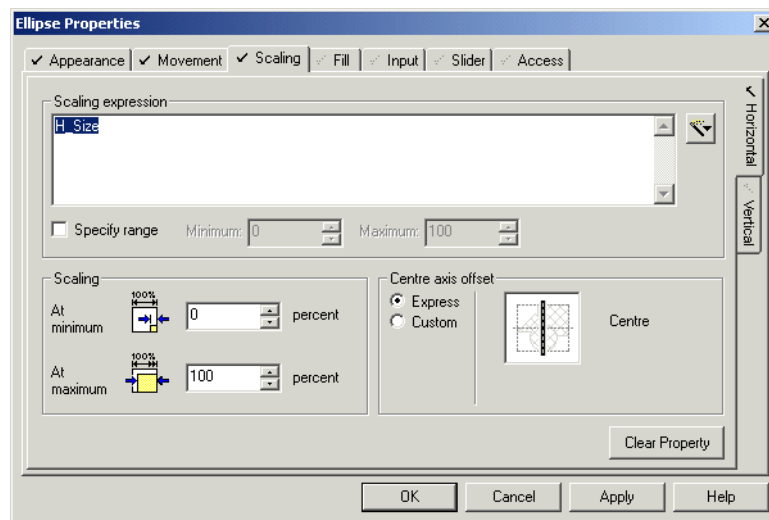
Scaling

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**.

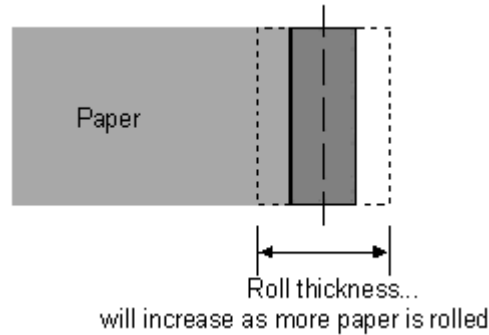


See Also [Object Properties - Scaling \(Horizontal\)](#)
[Object Properties - Scaling \(Vertical\)](#)
[Defining Common Object Properties](#)

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.

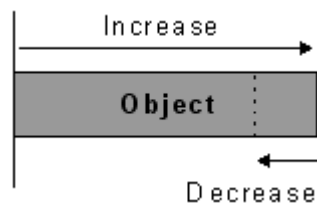
You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.

Hint: 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).

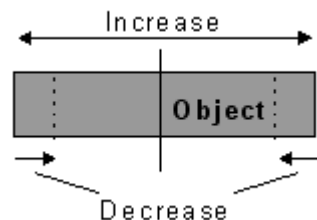
[Center axis offset] Express

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.

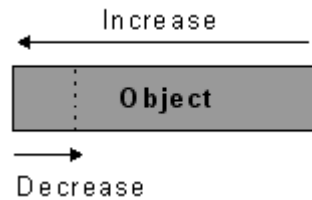
If you choose **Left**, all width changes occur to the right of the object. (i.e., the left edge remains anchored):



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.

Hint: Enter a negative value to move the center axis left instead of right.

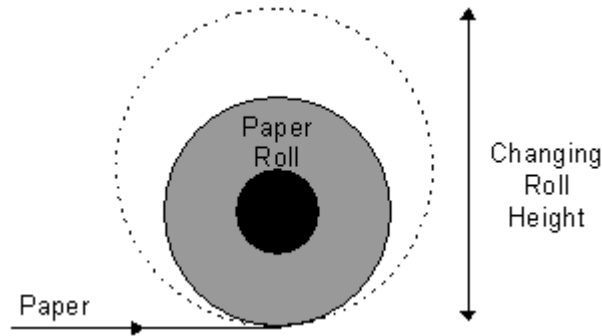
Note: If a group and its objects are configured to change size during runtime, the group scaling effects will be combined with the object scaling effects. For example, if a group is configured to double in size at runtime, and one of its object is configured to halve in size, the object will appear to 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.

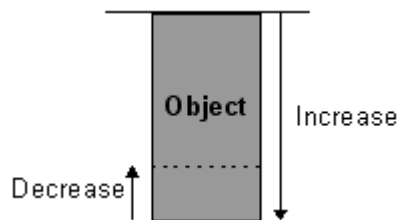
You can change the percentage by pressing the up and down arrows to the right of the field, or by entering another value in this field. Percentages of greater than 100% can be entered.

Hint: 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).

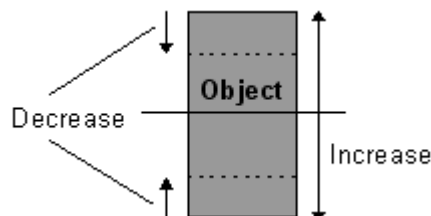
[Center axis offset] Express

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.

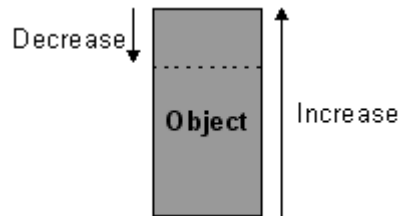
If you choose the top, all height changes will occur from the top of the object down. (i.e. the top edge will remain anchored):



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.

Hint: Enter a negative value to move the Center axis up instead of down.

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.

Note: There are several radio buttons in Object Properties - Fill Color (On/Off, Multi-state and so on). When selected, these radio buttons change the appearance of the right hand side of the dialog.

See Also [Object Properties - Scaling \(Horizontal\)](#)

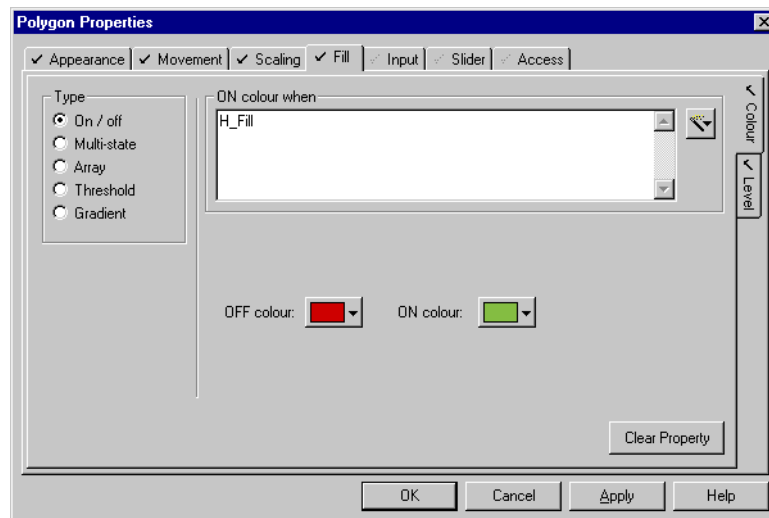
Fill Color

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.

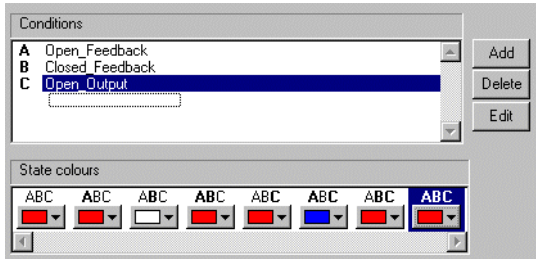
State colors

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).

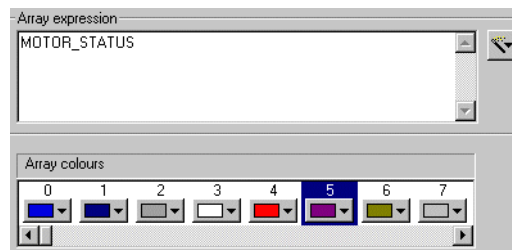
To insert a tag or a function, click the **Wizard** button to the right of this field. This button displays two options: Insert Tag and Insert Function.

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 less than 10 , the color fill will be Bright Blue .
< 15%	Blue	When the expression returns less than 30 , the color fill will be Blue .
> 85%	Red	When the expression returns greater than 170 , the color fill will be Red .
> 95%	Bright Red	When the expression returns greater than 190 , the color fill will be Bright Red .

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 \(Threshold\)](#)

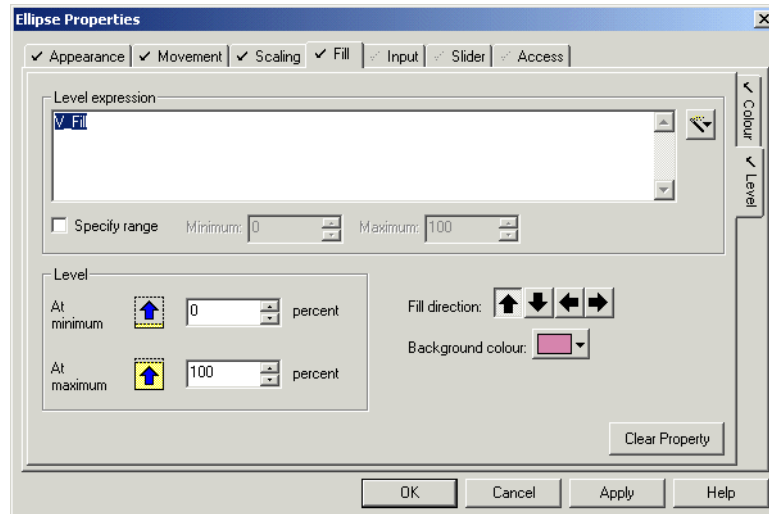
Fill Level

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

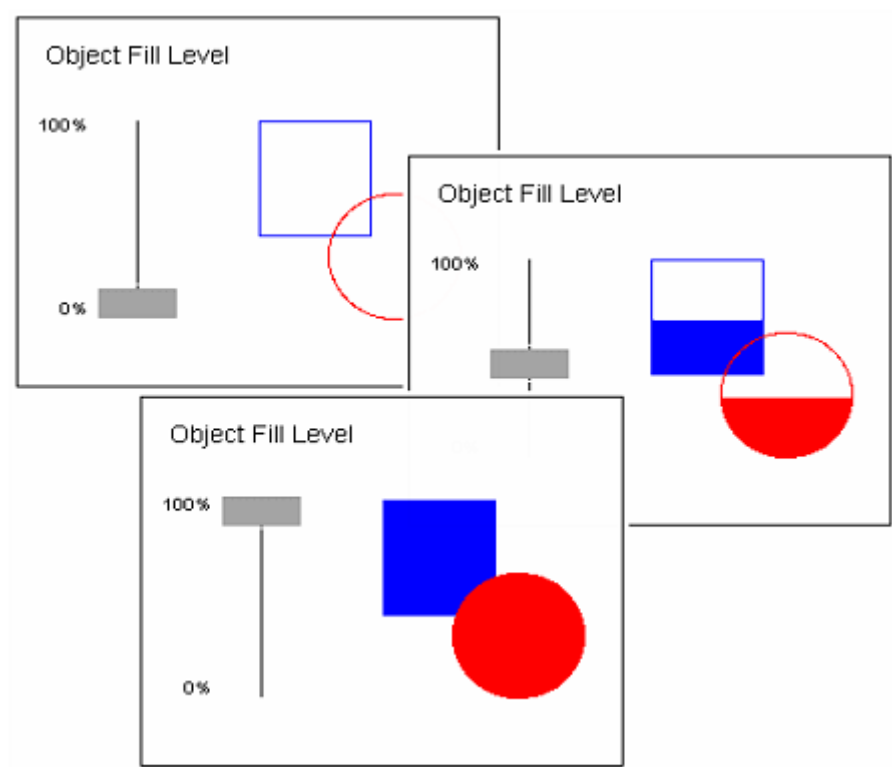
background is often made transparent. Using transparent, you would see the outline of the object/group, and anything behind the object/group on the page.

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.

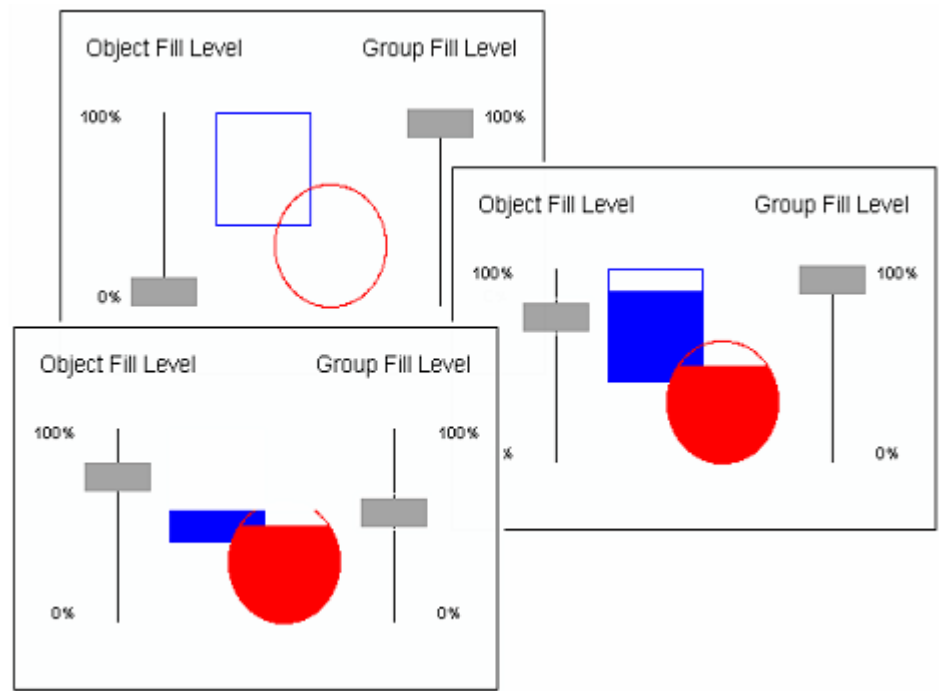
Group and Object Fill Level: Examples

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).

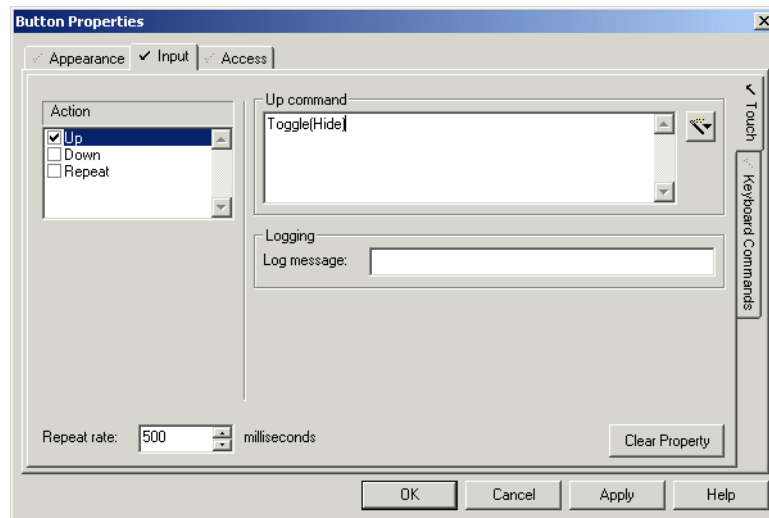
Touch Commands

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 commands(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.



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.

Hint: If the object is part of a Genie or symbol, this property can be defined after the Genie/symbol is pasted onto a page. (Simply 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.

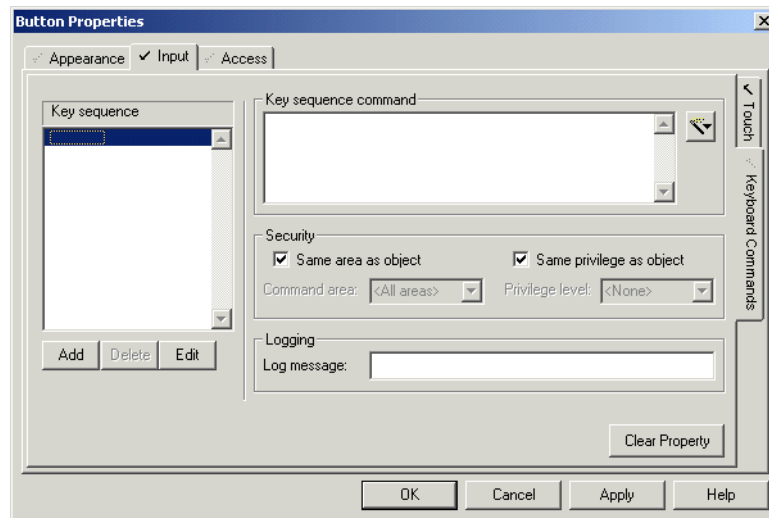
Keyboard Commands

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.

Hint: 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.

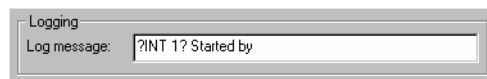
Click the menu to the right of this field to select a privilege, or type in an area number directly.

Hint: 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).

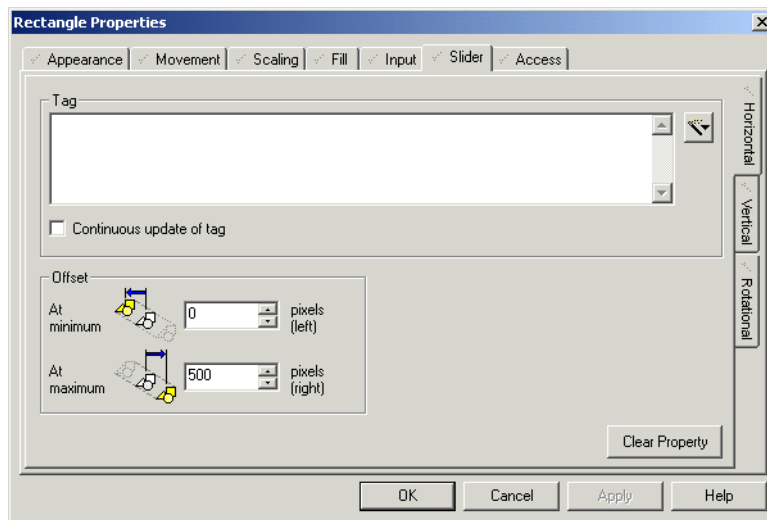
Sliders

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.

Hint: 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.

Hint: 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.

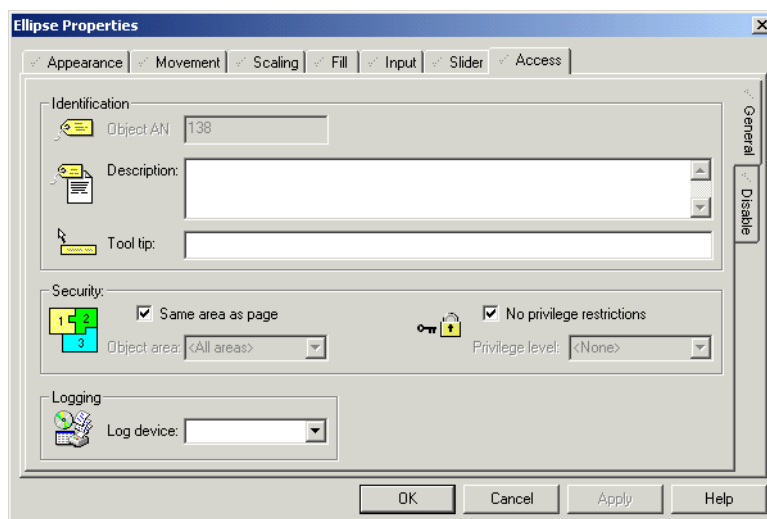
- [General Access to Objects](#)
- [Disable Access to Objects](#)

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 \(General\)](#)

Object Properties - Access (General)

Defining Common Object Properties

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.

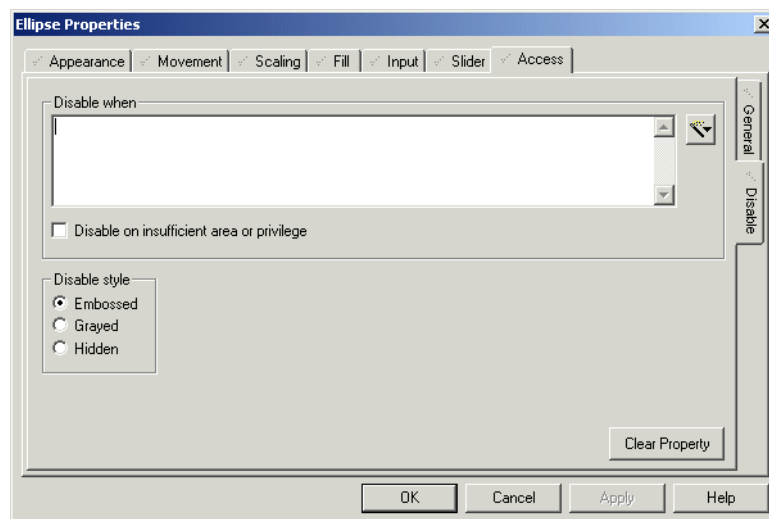
See Also [Disable Access to Objects](#)

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 of the objects in that group will also be disabled.

Chapter 7: 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 on 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.

Note: 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.

[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.

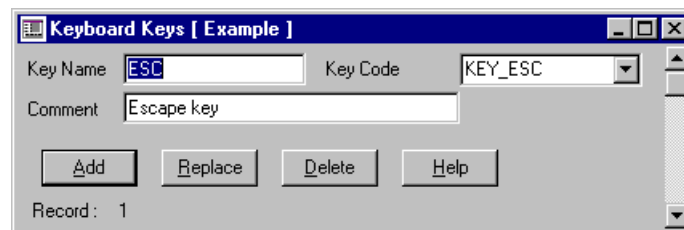
Note: You must include the *MsgLog* field in the format of the log device for the message to be sent.

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 different types of keyboard to control CitectSCADA.

Using non-standard keyboards

There are many types of keyboards that you can use 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

In some situations, you might need to use keyboards of different types. For example, you might use an IBM compatible keyboard in your control room and a sealed-membrane keyboard on the plant floor. If you do 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 the mouse buttons, you must 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.

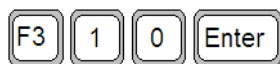
See Also [Using variable data input](#)

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:



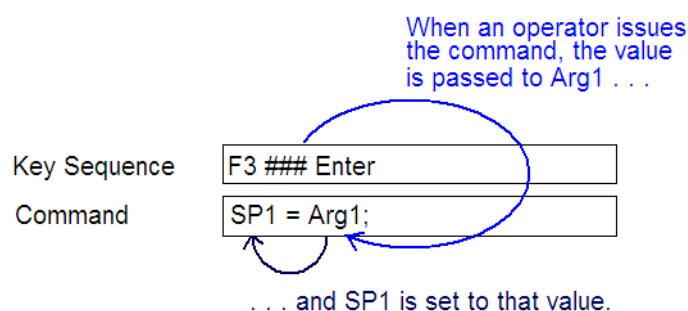
or



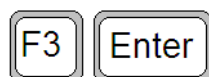
or



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;

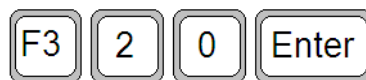
In this case, an operator can set two variables with the same command, for example:



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.

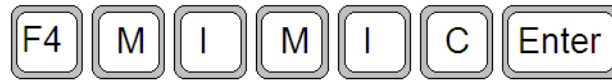
See Also [Passing keyboard arguments to functions](#)

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 8: 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 careful 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.

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 queries that use custom alarm filters

Implementing a query based on the custom alarm filters you have defined is a two step process:

- 1 You firstly need to create your own 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 AlarmGetFieldRec ; 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 You then need to set this query by calling it via the function [AlarmSetQuery](#).

Make sure the custom filters have been defined appropriately for your alarm tags before you proceed.

Example

Say you wanted to create a query that called all the 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 need to 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

Say you would then like to create a button on a graphics page that called up a list of 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, "", "")
```

Hint: 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.

Efficiency considerations

Cicode takes longer to implement than CitectSCADA's pre-defined 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.

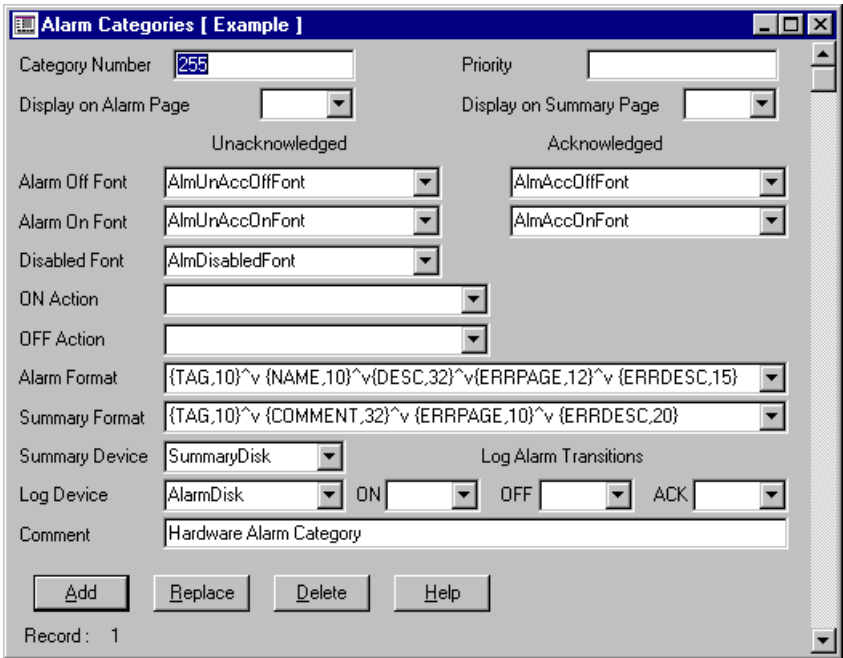
You can also customize the order in which alarms will be displayed on the alarm summary page using the SummarySort and SummarySortMode parameters. (This order will override the alarm category priority order.)

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.

You 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.

Note: 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 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.

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). If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g., you cannot have the same alarm tag name in more than one cluster).

Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters).

Alarm Desc

The description of the alarm (maximum of 254 characters). This can include variable data.

Alarm Tag, **Alarm Name**, and **Alarm Desc** are three separate strings that you can associate with the alarm. These are optional properties. CitectSCADA only uses them when details of the alarm are displayed on the screen or logged to a device. You could use these properties to define the alarm name, physical device, and description of the alarm.

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).

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.

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](#)

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). If you are using [distributed servers](#), the name must be unique to the cluster (e.g., you cannot have the same alarm name in more than one [cluster](#)).

Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters).

Alarm Desc

The description of the alarm (maximum of 254 characters). This can include variable data.

The **Alarm Tag**, **Alarm Name**, and **Alarm Desc** are three separate strings that you can associate with the alarm. These are optional properties. CitectSCADA only uses them when details of the alarm are displayed on the screen or logged to a device. You could use these properties to define the alarm name, physical device, and description of the alarm.

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 to be recorded when the states change. With a value of 0, only the time 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.

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 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.

The screenshot shows a Windows-style dialog box titled "Time Stamped Alarms [Example]". It has a standard title bar with minimize, maximize, and close buttons. The dialog contains several input fields and buttons. The fields are: "Alarm Tag" (a single-line text box), "Alarm Name" (a single-line text box), "Alarm Desc" (a single-line text box), "Variable Tag" (a text box with a dropdown arrow), "Timer" (a text box with a dropdown arrow), "Category" (a single-line text box), "Help" (a text box with a dropdown arrow), and "Comment" (a single-line text box). Below these fields are four buttons: "Add", "Replace", "Delete", and "Help". At the very bottom, there is a label "Record :" followed by a vertical scrollbar on the right side of the dialog.

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). If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g. you cannot have the same alarm tag name in more than one cluster).

Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters).

Alarm Desc

The description of the alarm (maximum of 254 characters). This can include variable data.

The **Alarm Tag**, **Alarm Name**, and **Alarm Desc** are three separate strings that you can associate with the alarm. These are optional properties. CitectSCADA only uses them when details of the alarm are displayed on the screen or logged to a device. You could use these properties to define the alarm name, physical device, and description of the alarm.

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 st	Hours	00–24	(most significant byte)
2 nd	Minutes	00–60	
3 rd	Seconds	00–60	
4 th	100 th /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 it is now OK for the PLC to 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 will be called. On calling the function, CitectSCADA reads the AlmTimer1 register. The function resets the trigger bit (handshaking), and the value of AlmTimer1 will be 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

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](#)

Analog Alarms

Analog alarms are triggered when an analog variable changes beyond one or more specific limits. Each alarm may 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). If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g., you cannot have the same alarm tag name in more than one cluster).

Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters).

The **Alarm Tag** and **Alarm Name** are two separate strings that you can associate with the alarm. These are optional properties. CitectSCADA only uses them when details of the alarm are displayed on the screen or logged to a device. You

could use these properties to define the alarm name and the physical device, or the alarm name and description of the alarm.

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.

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).

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](#)

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). If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g. you cannot have the same alarm name in more than one cluster).

Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters).

Alarm Desc

The description of the alarm (maximum of 254 characters). This can include variable data.

The **Alarm Tag**, **Alarm Name**, and **Alarm Desc** are three separate strings that you can associate with the alarm. These are optional properties. CitectSCADA only uses them when details of the alarm are displayed on the screen or logged to a device. You could use these properties to define the alarm name, physical device, and description of the alarm.

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 will 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 **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). If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g., you cannot have the same alarm tag name in more than one cluster).

Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters).

Alarm Desc

The description of the alarm (maximum of 254 characters). This can include variable data.

Alarm Tag, **Alarm Name**, and **Alarm Desc** are three separate strings that you can associate with the alarm. These are optional properties. CitectSCADA only uses them when details of the alarm are displayed on the screen or logged to a device. You could use these properties to define the alarm name, physical device, and description of the alarm.

Var Tag A/Var Tag B

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:

```
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 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). If you are using [distributed servers](#), the name must be unique to the cluster (e.g., you cannot have the same alarm tag name in more than one cluster).

Alarm Name

The name of the physical device associated with the alarm (maximum of 79 characters).

The **Alarm Tag** and **Alarm Name** are two separate strings that you can associate with the alarm. These are optional properties. CitectSCADA only uses them when details of the alarm are displayed on the screen or logged to a device. You could use these properties to define the alarm name and the physical device, or the alarm name and description of the alarm.

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.

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**).

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 of fields) 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
{Name,n}	Alarm Name
{Native_Name,n}	Alarm Name in the native language
{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: DISABLED ACKNOWLEDGED 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.

Field Name	Description
{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. It is recommended you 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, <i>n</i> }	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, <i>n</i> }	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 of the fields listed below (or a combination of fields) 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,n}	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,n}	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,n}	The text related to the user event
{OnDate,n}	The date when alarm was activated
{OnDateExt,n}	The date (in extended format) when the alarm was activated (dd/mm/yyyy)
{OffDate,n}	The date when the alarm returned to its normal state
{OffDateExt,n}	The date (in extended format) when the alarm returned to its normal state (dd/mm/yyyy)
{OnTime,n}	The time when the alarm was activated
{OffTime,n}	The time when the alarm returned to its normal state
{DeltaTime,n}	The time difference between OnDate/OnTime and OffDate/OffTime, in seconds
{OnMilli,n}	Adds milliseconds to the time the alarm was activated.
{OffMilli,n}	Adds milliseconds to the time the alarm returned to its normal state.
{AckTime,n}	The time when the alarm was acknowledged
{AckDate,n}	The date when the alarm was acknowledged
{AckDateExt,n}	The date (in extended format) when the alarm was acknowledged (dd/mm/yyyy)
{SumState,n}	Describes the state of the alarm when it occurred
{SumDesc,n}	A description of the alarm summary
{Native_SumDesc,n}	A description of the alarm summary, in the native language
{AlmComment,n}	The text entered into the Comment field of the alarm properties dialog.
{Comment,n}	A comment the operator adds to an Alarm Summary entry during runtime. The comment is specified using the AlarmComment() function.
{Native_Comment,n}	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 will be displayed on the alarm summary page. You can do this 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

Property	Description	Return Type
.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

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

Note: 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 will only work 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 the screen)

changes. Acknowledged alarms remain on the screen until their state changes to OFF.

To overcome a situation where an alarm might be faulty or unnecessary, an operator can disable an alarm. CitectSCADA ignores disabled alarms until they are re-enabled. (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 the Page Up and Page Down commands to scroll through the list.

To create an alarm page:

- 1 In **Citect Explorer**, double-click the **Create New Page** icon 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

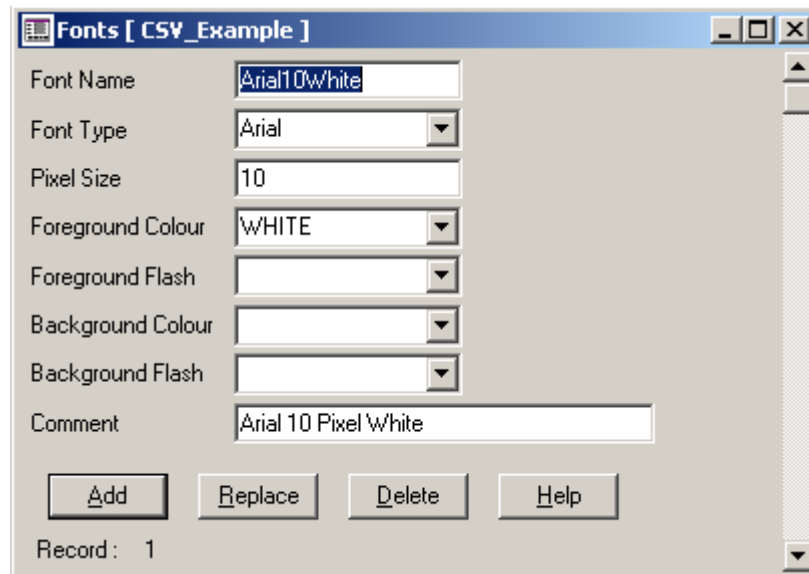
Alarm categories, Cicode functions, and button objects allow you to use predefined fonts, known as CitectSCADA fonts, to display text. You can also configure your own CitectSCADA fonts.

Note: If any animation is associated with an I/O device that fails at startup or goes off-line 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 (that can use any standard Windows font), you must define a CitectSCADA font for animated text (numbers, strings, alarms, etc.).

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:

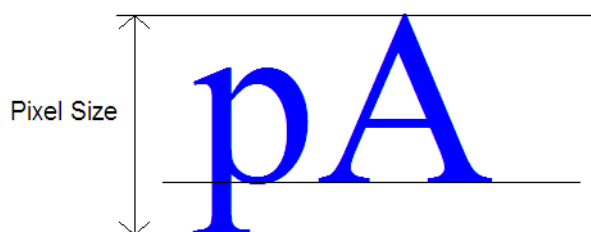
Font Type Courier,B,I,U

Specifies bold, italic, and underlined characters.

Note: If you want to use a font that is not displayed in the drop box, you must install the font on your computer before you can use it in your CitectSCADA system. You can view, install, and remove Windows fonts using the Windows Control Panel (Fonts Option). Refer to your Windows documentation for information on installing fonts. (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.

Note: If you use a point size, the size remains constant across all screen resolutions. On low resolution screens, the font displays larger than on high resolution screens. This 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 pre-defined color label (accessible via the drop-down list), a user-

defined custom label, or the RGB encoded number generated by the function `MakeCitectColour` (see the *Cicode Reference Guide*).

Note: Pre-defined labels should not be confused with the color [Name](#) feature associated with the Color Picker. You cannot use color names with this dialog, doing so will generate a compile error.

Foreground Flash

The secondary color applied to the font if using flashing color for your text characters. You can use a pre-defined color label (accessible via the drop-down list), 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 pre-defined color label (accessible via the drop-down list), 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 pre-defined color label (accessible via the drop-down list), 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).

Chapter 9: Configuring Events

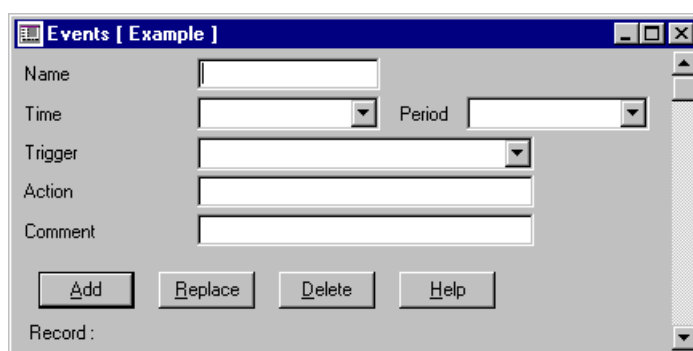
You can use an event to trigger an action, such as a command or set of commands. For example, an operator can be notified when a process is complete, or a series of instructions can be executed when a process reaches a certain stage.

Events must be enabled for events to run. Use the *Citect 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 Citect 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.

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 (i.e. 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, e.g. Monday, Tuesday, Wednesday, etc.
- A monthly period by entering the day of the month to check for the event, e.g. 1st, 2nd, 3rd, 4th, 5th, etc.
- A yearly period by entering the day and the month to check for the event, e.g. 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	1:00:00
Comment	Run the event every hour
Period	6:00:00
Comment	Run the event every six hours
Period	72:00:00
Comment	Run the event every three days
Period	Monday
Comment	Run the event each Monday
Period	15th
Comment	Run the event on the 15th of each month
Period	25th June
Comment	Run the event on the 25th of June

You can also specify the time of day to synchronize the event, for example:

Time	6:00:00
Comment	Synchronize the event at 6:00 am
Time	12:00:00
Comment	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, 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 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 (i.e. if the digital tag RCC1_MC is ON and the analog tag RCC1_SPEED is less than 10).

See Also [Running Events](#)

Chapter 10: 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).

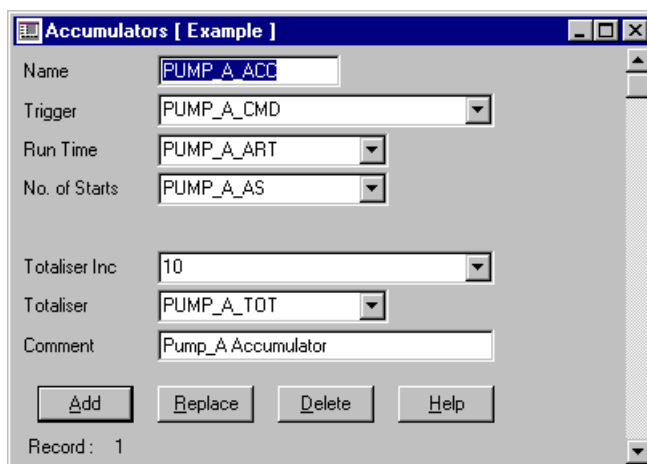
The accumulated data is stored as variable tags in an I/O device. Variable tags are read at CitectSCADA startup and updated regularly while the trigger is active. You can monitor and display accumulated data by animating, trending, or logging the variable tags.

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.



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.

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 totalized 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 11: Logging and Trending Data

CitectSCADA version 6.0 includes the Process Analyst, a new, more powerful trend visualization tool that supersedes 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](ProcessAnalyst.chm).

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 provide a better understanding of the performance of your plant and equipment. It 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.

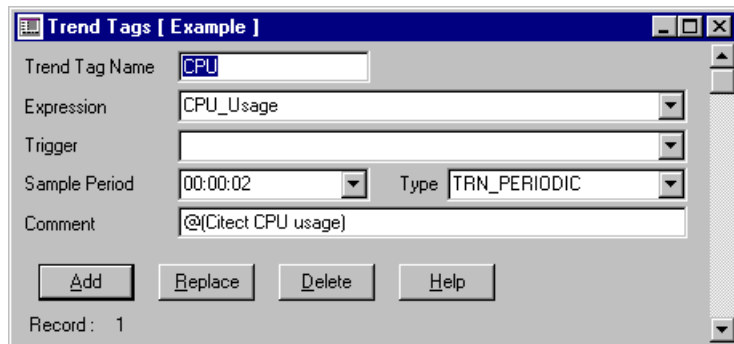
See Also [Configuring trend tags](#)

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 Tags have the following 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.

If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g. you cannot have the same trend tag name in more than one cluster). The first 8 characters of your trend tag names must not be the same as the first 8 characters of your SPC tag names.

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 $1/0.3 = 3.333 \dots$

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), it is recommended that you 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	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

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):

- 1 TRN_PERIODIC
- 2 TRN_EVENT
- 3 TRN_PERIODIC_EVENT

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 \CITECT\DATA directory on the hard disk where you installed CitectSCADA. 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'.

Note the following:

- 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 \CITECT\DATA\<Name> on the hard disk where you installed CitectSCADA. <Name> is the Trend Tag Name. If you do use this

property, ensure that no other trend tags have the same name, otherwise the data might be lost.

Note the following:

- 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 either **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 (32 Chars.)

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 (32 Chars.)

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

A trend graph is a visual representation of past and current activity of plant-floor data. It builds a picture over time of how a variable (such as product output, level, temperature, etc.) is changing or how a device or process is performing. You can monitor current activity as it happens and scroll back through time to view the 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.

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 pre-defined trend templates for your trend pages, or use a pre-defined 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 either:

- 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 sync 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 have to 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
Average (<i>default</i>) - this displays the average of the samples within the previous display period	Step (<i>default</i>) - This method simply displays the value of the most recent sample.
Minimum - This displays the lowest value that occurred during the previous display period.	Ratio - This method uses the ratio of sample times and values immediately before and after the requested time to interpolate a "straight line" value.

Condense methods

Maximum - This displays the highest value that occurred during the previous display period.

Stretch methods

Raw Data - 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
<code>TrnPrint</code>	Prints a trend that is displayed on the screen.
<code>TrnPlot</code>	Prints a plot of one or more trend tags.
<code>TrnComparePlot</code>	Prints two trends (one overlaid on the other), each of up to four trend tags.
<code>WinPrint</code>	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
<code>TrnGetTable</code>	Retrieves trend information and stores it in a Cicode array
<code>TrnExportClip</code>	Copies trend data to the clipboard
<code>TrnExportCSV</code>	Copies trend data to a CSV file

Function	Purpose
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](#)

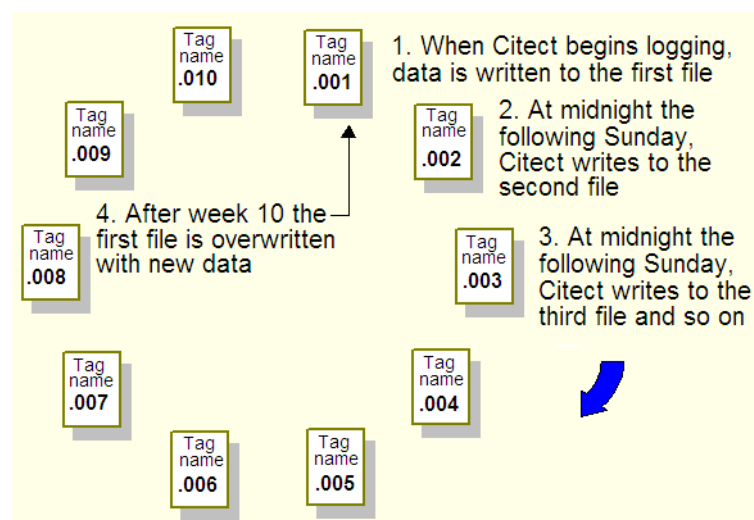
Using Trend History Files

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 th
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:

$$\begin{array}{l} \text{Bytes reqd} \\ \text{for each} \\ \text{trend} \end{array} = 464 \times \text{No. Files} + 176 + \left(\frac{\text{File Period (secs)} \times (\text{No. Files}) \times 2}{\text{Sample Period (secs)}} \right)$$

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:

$$\begin{array}{l} \text{Bytes reqd} \\ \text{for each} \\ \text{trend} \end{array} = 704 \times \text{No. Files} + 160 + \left(\frac{\text{Period (secs)} \times (\text{No. Files}) \times 8}{\text{Sample Period}} \right)$$

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 following debugging options to help you while logging and trending data:

- `#define TREND_LOG`, which has a value of 0x01.
- `#define TREND_BACKFILL`, which has a value of 0x02.
- `#define TREND_SETTABLE`, which has a value of 0x04.
- `#define TREND_BACKFILL_SUMMARY`, which has a value of 0x08.

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 12: 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 they can be moved, reshaped, and 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.

There are 15 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		

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.

Note: 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.

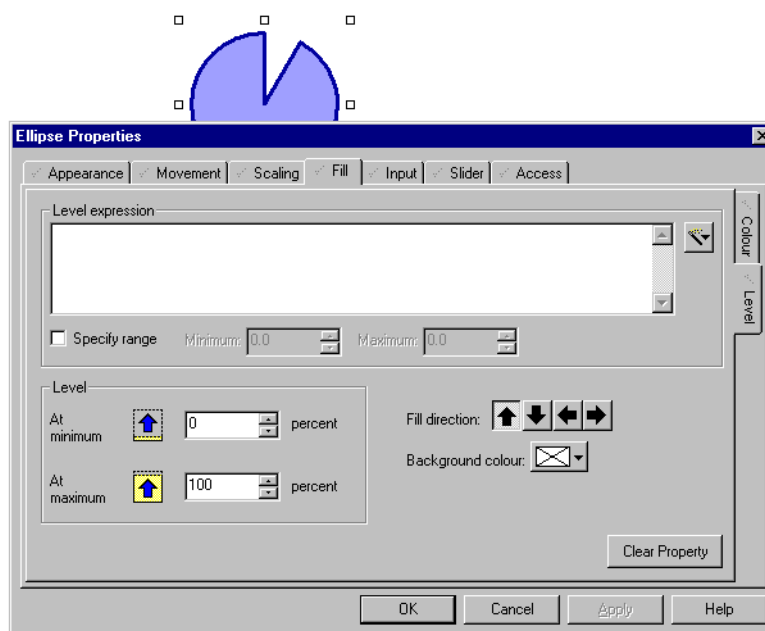
Note: By default, unavailable colors in an imported bitmap are dithered. To disable this feature, select Options from the Tools menu in the Graphics Builder, and remove the tick from the **Dither bitmaps on paste** option. If dithering is disabled, unavailable colors are replaced by the closest match 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.



Hint: 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 the CTRL key and double-clicking the object. Alternatively, choose **Tools | Goto Object**, select the object, and then click **OK**.

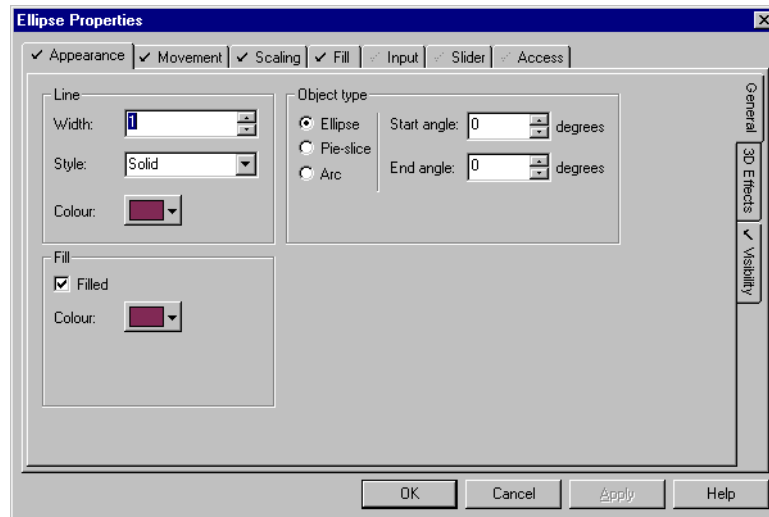
See Also [Appearance](#)
[Movement](#)
[Scaling](#)
[Fill](#)
[Input](#)
[Slider](#)
[Access](#)

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 checkmark to the left of the Appearance tab tells you when an appearance property has been configured. The checkmarks in the tabs down the right of the dialog tell you exactly 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.



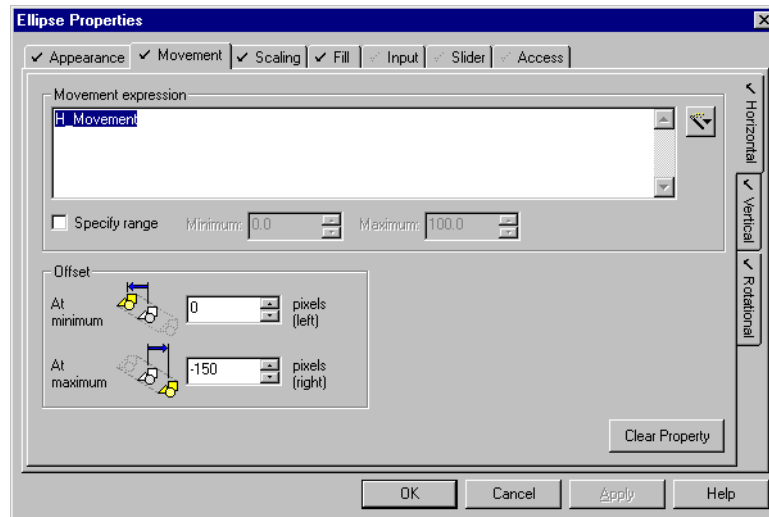
See Also [Object Properties](#)
[Movement](#)

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 checkmark to the left of the Movement tab tells you when a Movement property has been configured. The checkmarks in the tabs down the right of the dialog tell you exactly 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.



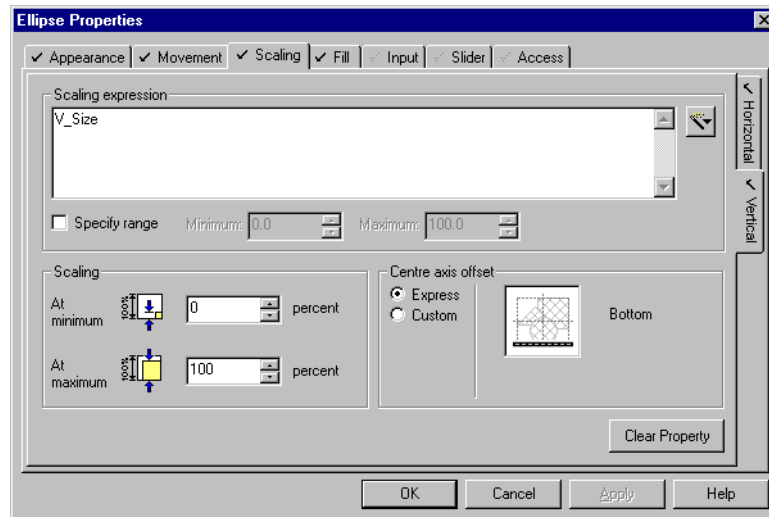
See Also [Object Properties](#)
[Scaling](#)

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 checkmark to the left of the Scaling tab tells you when a Scaling property has been configured. The checkmarks in the tabs down the right of the dialog tell you exactly 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.



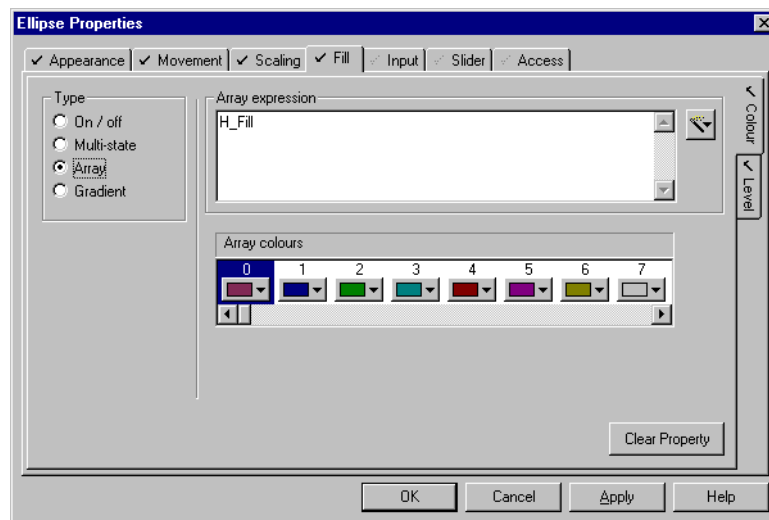
See Also [Object Properties](#)
[Fill](#)

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 checkmark to the left of the Fill tab tells you when a Fill property has been configured. The checkmarks in the tabs down the right of the dialog tell you exactly 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.



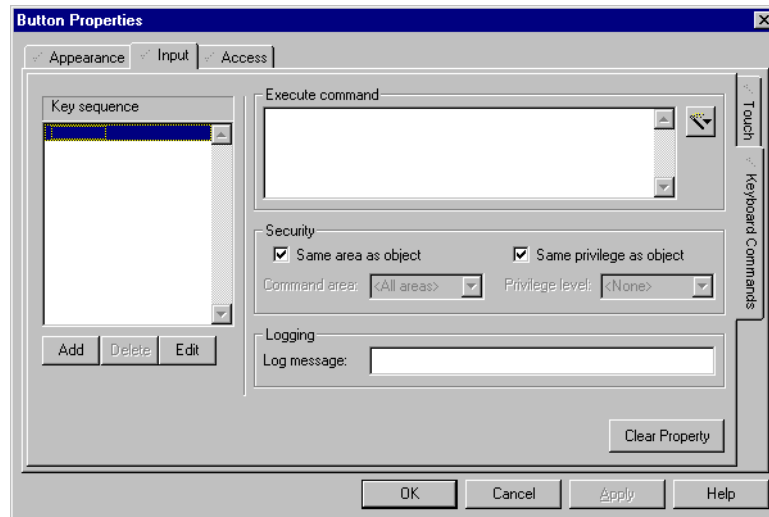
See Also [Object Properties](#)
[Input](#)

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 checkmark to the left of the Input tab tells you when an Input property has been configured. The checkmarks in the tabs down the right of the dialog tell you exactly 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.



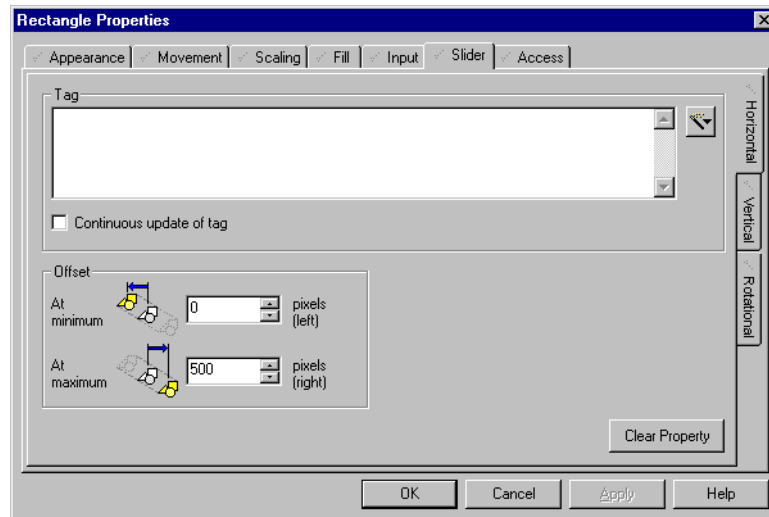
See Also [Object Properties](#)
[Slider](#)

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 checkmark to the left of the Slider tab tells you when an Slider property has been configured. The checkmarks in the tabs down the right of the dialog tell you exactly 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.



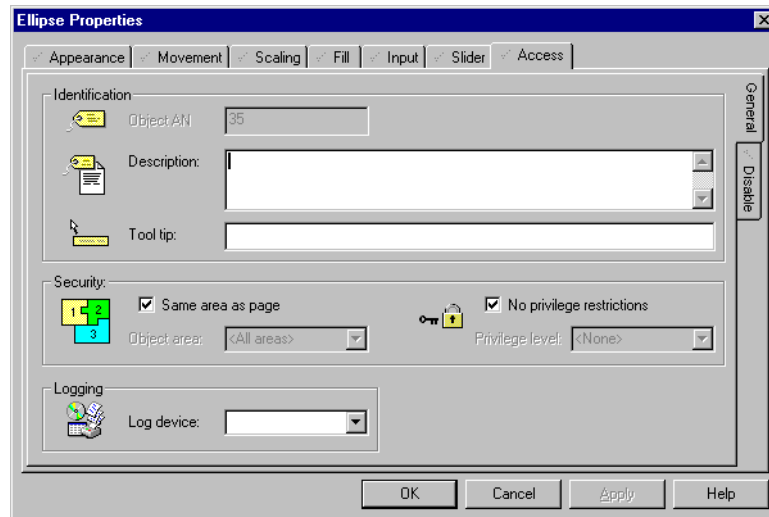
See Also [Object Properties](#)
[Access](#)

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 checkmark to the left of the Access tab tells you when an Access property has been configured. The checkmarks in the tabs down the right of the dialog tell you exactly 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

Note: Groups, Symbols, and Genies can all be manipulated in the same way as objects.

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](#)

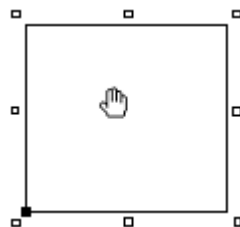
[Mirroring objects](#)

[Locate an object](#)

Selecting objects

To select a single object:

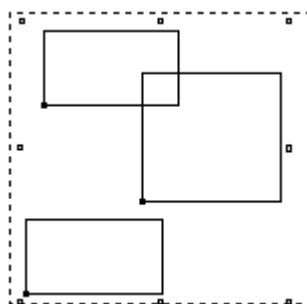
- 1 In Graphics Builder, click **Select**.
- 2 Click the object. The object's sizing handles appear, and the cursor changes from an arrow to a hand while on the object.



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.

Note: 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).

Note: If you have a node selected and then click another node within the same object, the first node will deselect. To select more than one node, hold down the

Ctrl key and click each of the nodes you would like to select. Note that while the Ctrl key remains held down, you can click a previously selected node to deselect it. Further clicking on the same node will toggle 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.

Note: 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.

Note: 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.

Note: 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 objects you no longer want.

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**.

Note: 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**.

Note: 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](#)

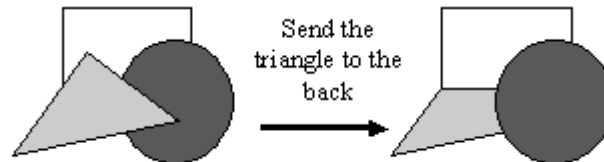
Send to Back and Send Backwards

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).

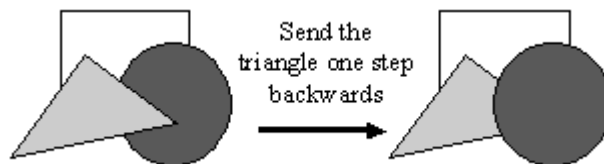
- 3 Click **Send to Back** or choose **Arrange | Send to Back**.



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

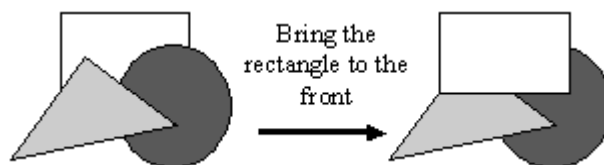
[Manipulating Objects](#)
[Bring to Front and Bring Forwards](#)

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**.

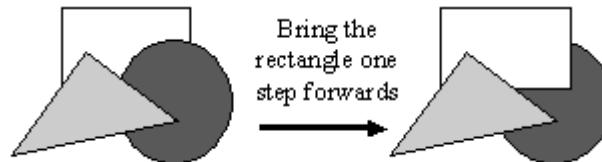


An object can be moved backwards and forwards one step at a time (rather than all the way to the back, or all the way to the front).

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.



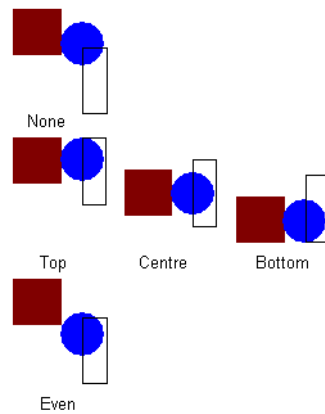
See Also [Align dialog box](#)
[Manipulating Objects](#)

Align dialog box

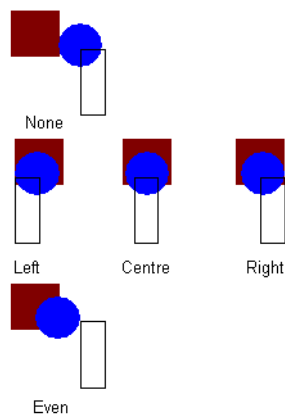
You use the Align dialog box to align a group of objects vertically, horizontally, or both.

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).

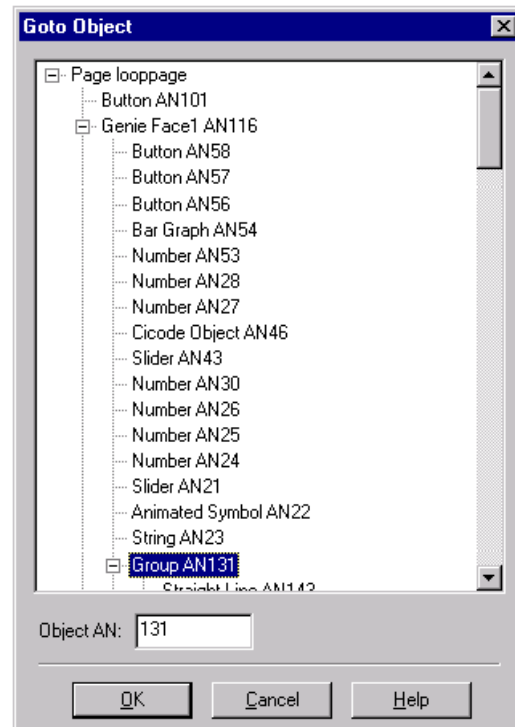
See Also [Manipulating Objects](#)

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, intuitive method of locating 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), so they have a plus sign (+) next to them; click the plus 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, etc.) 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 13: 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.

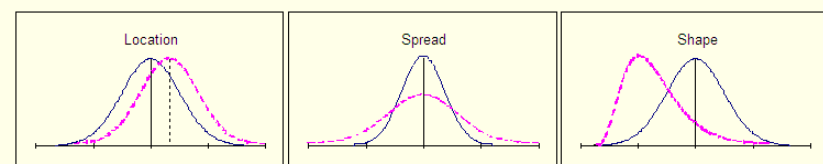
See Also

[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.

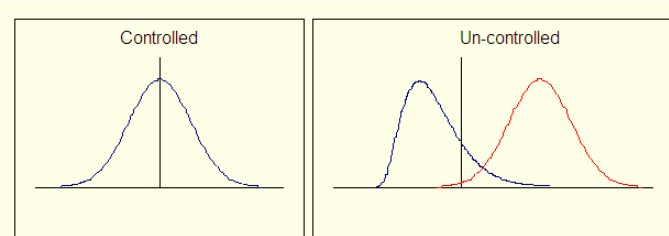
Statistics indicate that common variations account for about 85% of a process problems. Usually these problems require solution at the management level.
- **Special:** refers to variation that is not always present. When special variation occurs it will tend to change the distribution characteristics. 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

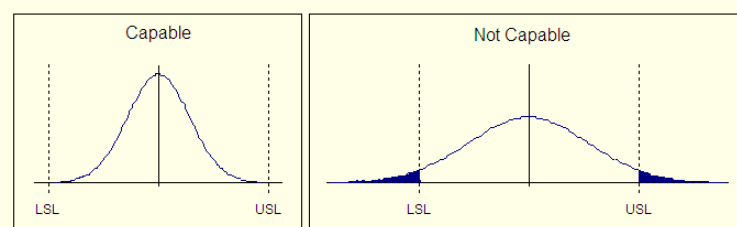
A process is said to be capable when the percentage of samples outside the specification limits is less than a predefined value. The following assumptions must also be true:

- 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:

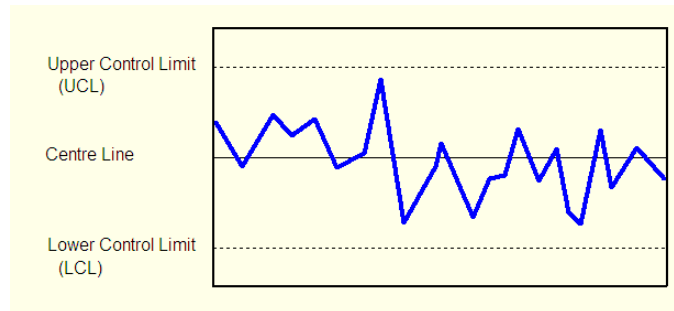


See Also [XRS control charts](#)

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 (X bar, 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 Cp, Cpk, [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 (Cp) 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:

- $Cp > 1.0$ - Indicates the process variation will fit within the specified limits (USL and LSL) and therefore, is capable.
- $Cp < 1.0$ - Indicates the process is not capable.

Cpk Index

The process capability based on the worst case (Cpk) is similar Cp. This index, however, indicates where the mean lies in relation to the USL and LSL limits. It is used to mathematically clarify Cp. Its meaning is defined as follows:

- $Cpk < 0$ - Indicates the process mean is outside the specified limits (USL and LSL)
- $Cpk = 0$ - Indicates the process mean is equal to one of the specified limits.
- $Cpk > 0$ - Indicates the process mean is within the specified limits.
- $Cpk = 1.0$ - Indicates that one side of the 6-sigma limits falls on a specification limit.
- $Cpk > 1.0$ - 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.

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.

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.

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 Alarmsform. You use the SPCAlarms() Cicode function to check for the condition of the alarms:

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 (± 2 sigma) from the center line.

Name	Description
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 ± 1 sigma of) the center line.
XMixture	Several consecutive points are far from (outside ± 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:

$$MR_i = X_i - X_{i-1}$$

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:

$$s_i = \frac{MR_i}{D_2}$$

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}):

$$\bar{R} = \frac{MR_1 + MR_2 + \dots + MR_m}{m}$$

Where MR_1 , MR_2 , 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 center) 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_x and LCL_x):

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 A_2 is a constant (given in the Control Chart Line Constant table).

Range Control Limits (UCL_R and LCL_R):

Specify the approximated 3-sigma boundaries. For a normal distribution 99.74% of the samples will fall within this boundary.

$$UCL_R = \bar{R} + D_4 \bar{R}$$

$$LCL_R = \bar{R} - D_3 \bar{R}$$

Where \bar{R} is the Process Range and D_3 and D_4 are constants (given in the Control Chart Line Constant table).

Standard Deviation Control Limits (UCL_s and LCL_s):

Specify the approximated 3-sigma boundaries. For a normal distribution 99.74% of the samples will fall within this boundary.

$$UCL_s = \bar{s} + B_4 \bar{s}$$

$$LCL_s = \bar{s} - B_3 \bar{s}$$

Where \bar{s} is the Process Standard Deviation and B_3 and B_4 are constants given in the Control Chart Line Constant table).

Process Capability (Cp):

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:

$$C_p = \frac{(USL - LSL)}{6\sigma}$$

$C_p > 1.0$ - 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:

$$C_{pk} = \frac{\min\left(\frac{USL - \bar{X}}{3\sigma}, \frac{\bar{X} - LSL}{3\sigma}\right)}{1}$$

$C_{pk} < 0$ - 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).

$$S_k = \frac{\sum (X - \bar{X})^3}{N\bar{s}^3}$$

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).

$$Ku = \frac{\sum (x_i - \bar{x})^4}{N \bar{s}^4}$$

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	

Samples in	Averages	Ranges	Standard Deviations			
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 14: 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](#)
[Defining Page Properties](#)
[Understanding the Drawing Environment](#)

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:

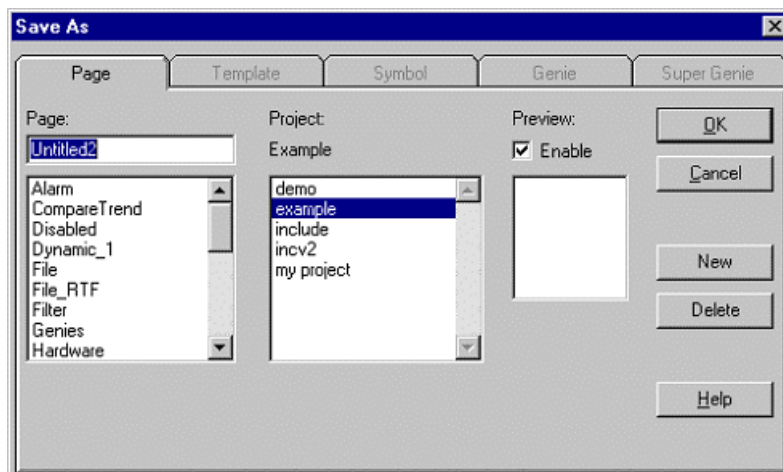
- 1 Click **Save**, or choose **File | Save**.
- 2 Select the **Project** in which to store the page. (The first eight characters of the name must be unique to this page.)
- 3 Click **OK**.

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

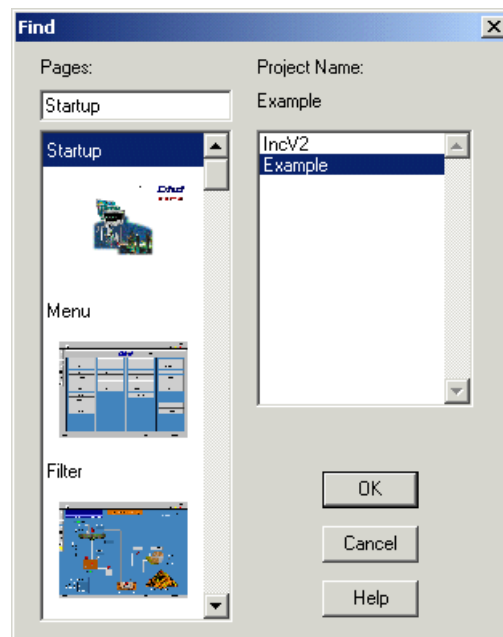
- Choose **File | Save All**. (The first eight characters of each page name must be different.)



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 [Find dialog box](#)

To print a graphics page on your printer:

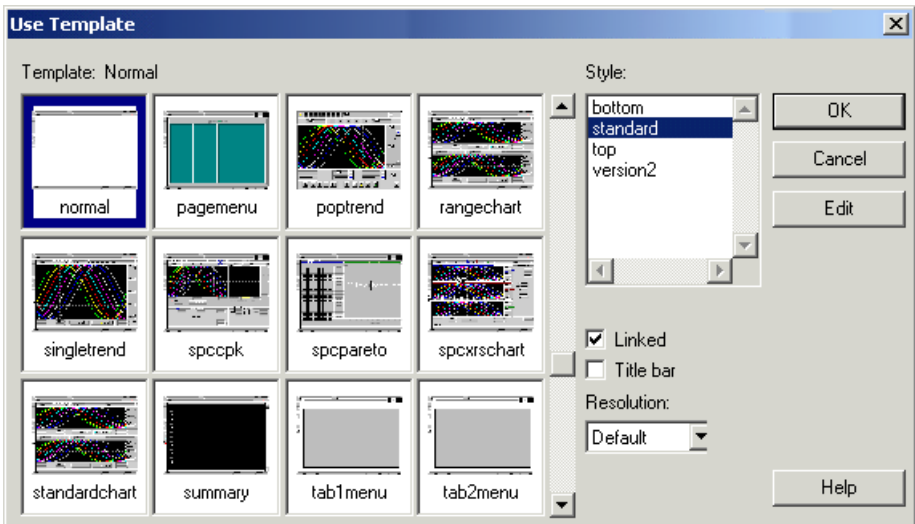
- Choose **File** | **Print**. (This prints without a confirmation dialog.)

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 checkbox. 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.

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

Open/Save As dialog box

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

Note: To delete a page, template, symbol, Genie, or Super Genie, select it and click **Delete**.

Find dialog box

This dialog box lets you locate a graphics page.

Pages
A list of graphics pages in the project.

To open a page, use the scroll bar to locate the thumbnail image of the page, select the page, and click the **OK** button (or double-click the thumbnail image).

Project Name
The project (or project library) where the graphics page is stored.

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:

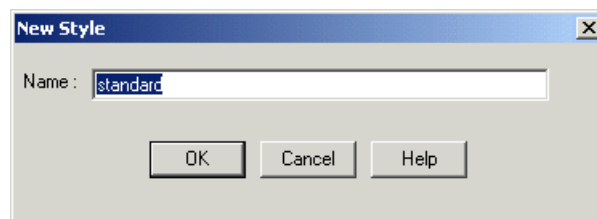
- 1 Click **Open**, or choose **File | Open**.
- 2 Select **Type: Template**.
- 3 Select the **Project** where the template is stored.
- 4 Select the **Template**.
- 5 Click **OK**.

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](#)

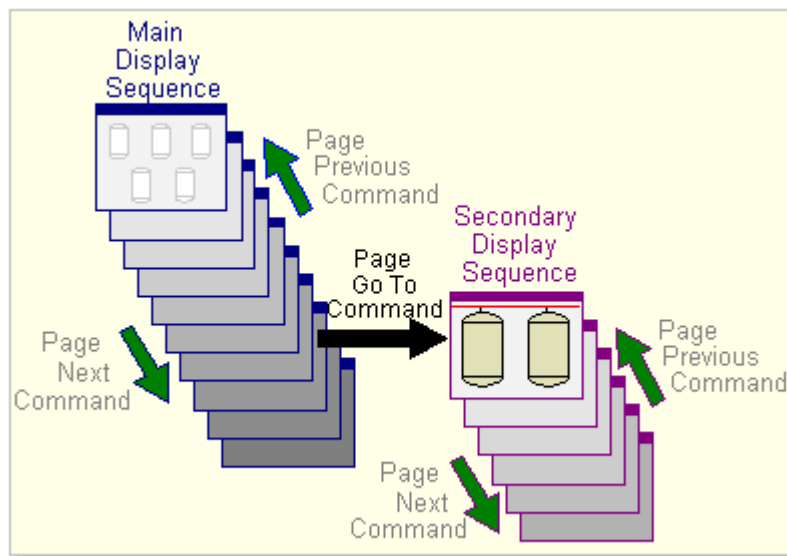
Using a Browse Sequence

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](#)

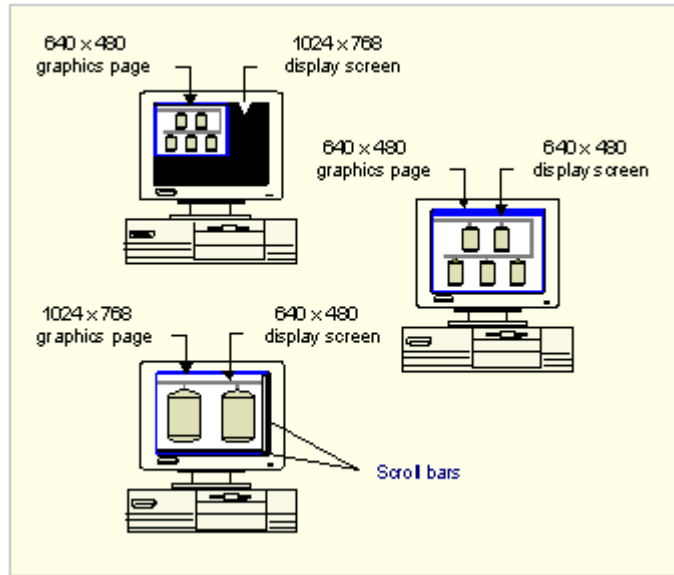
Page (screen) resolution

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](#)

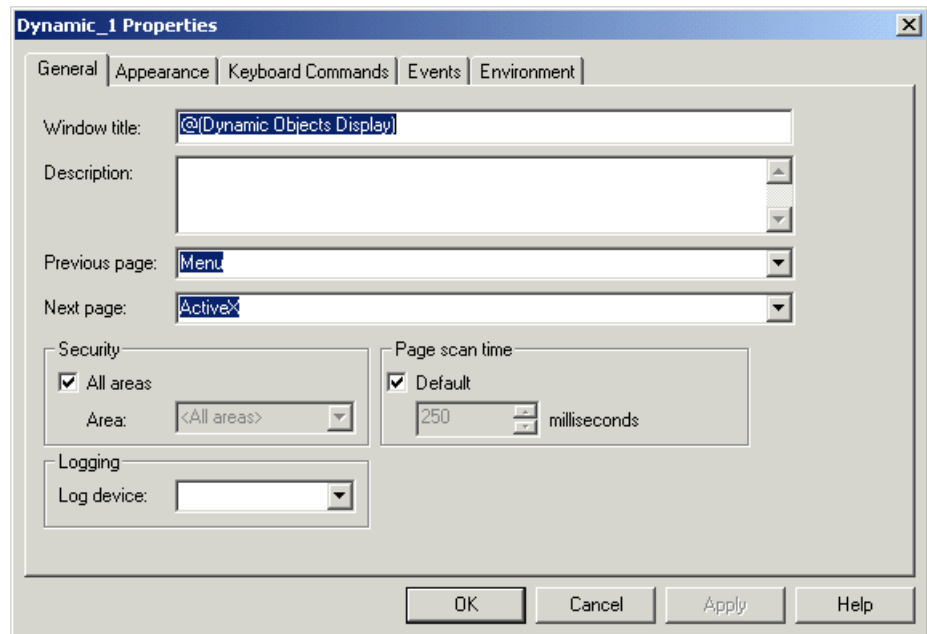
Defining Page Properties

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 checkbox 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. When the page is updated, all relevant data (such as variable tags) represented on the graphics page is scanned to determine if field conditions have changed.

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. However, if CitectSCADA cannot read all of the data from the relevant I/O devices within 200ms, the page is processed more slowly. For example, if it takes 800ms to read all the data from the I/O devices, CitectSCADA will process the page every 800ms.

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.

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 checkbox 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](#)).

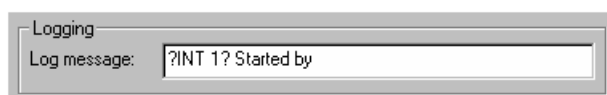
[Security] Privilege level

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.

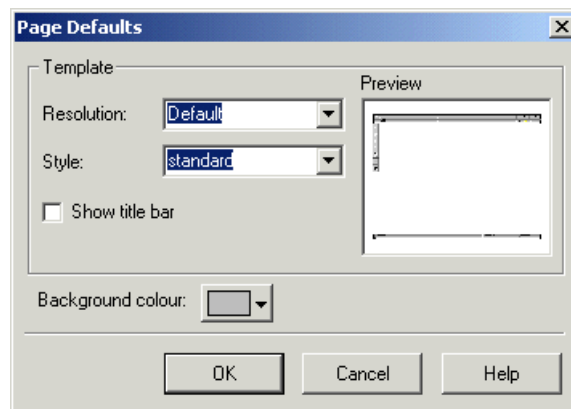
To set the properties to be used for all new graphics pages:

- 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.



Note: 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 Citect 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.

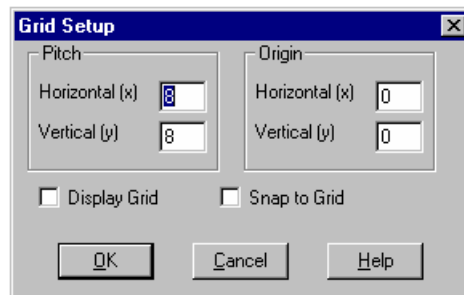
To snap to the grid:

- 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)

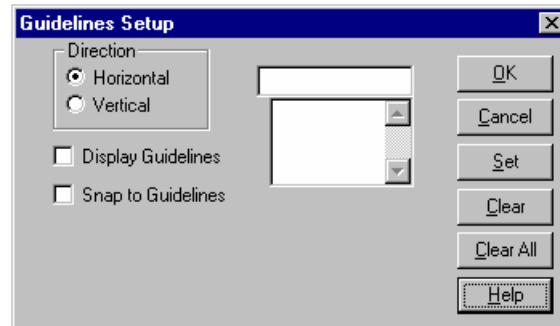
Activates the grid. When the grid is active, any object or group of objects that you create, move, or re-size snaps to the nearest grid intersection.

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**.

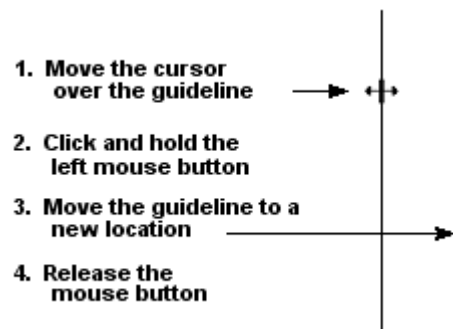


- 2 To run the guideline horizontally across your page, click **Horizontal**.
- 3 Enter a position (distance from the top of your page) for the guideline, and click **Set**.
- 4 To run the guideline vertically down your page, click **Vertical**.
- 5 Enter a position (distance from the left edge of your 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**.

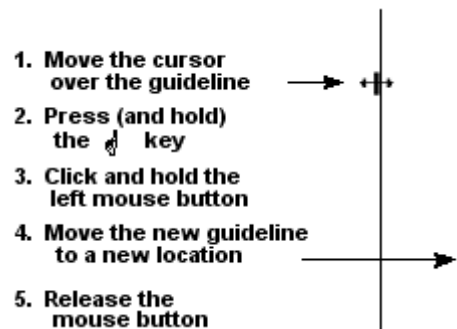
To turn off guidelines:

- 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.

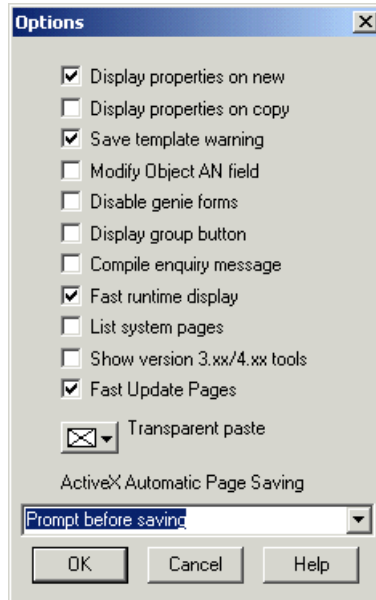
Options

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

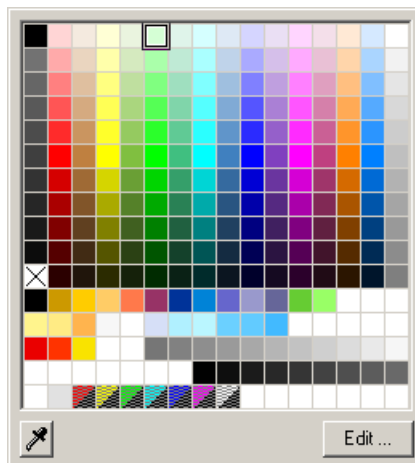
CitectSCADA supports True Color (16.7 million colors) for all static and animated objects, including page backgrounds, imported images, symbols, metafiles and bitmaps.

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.

The 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.

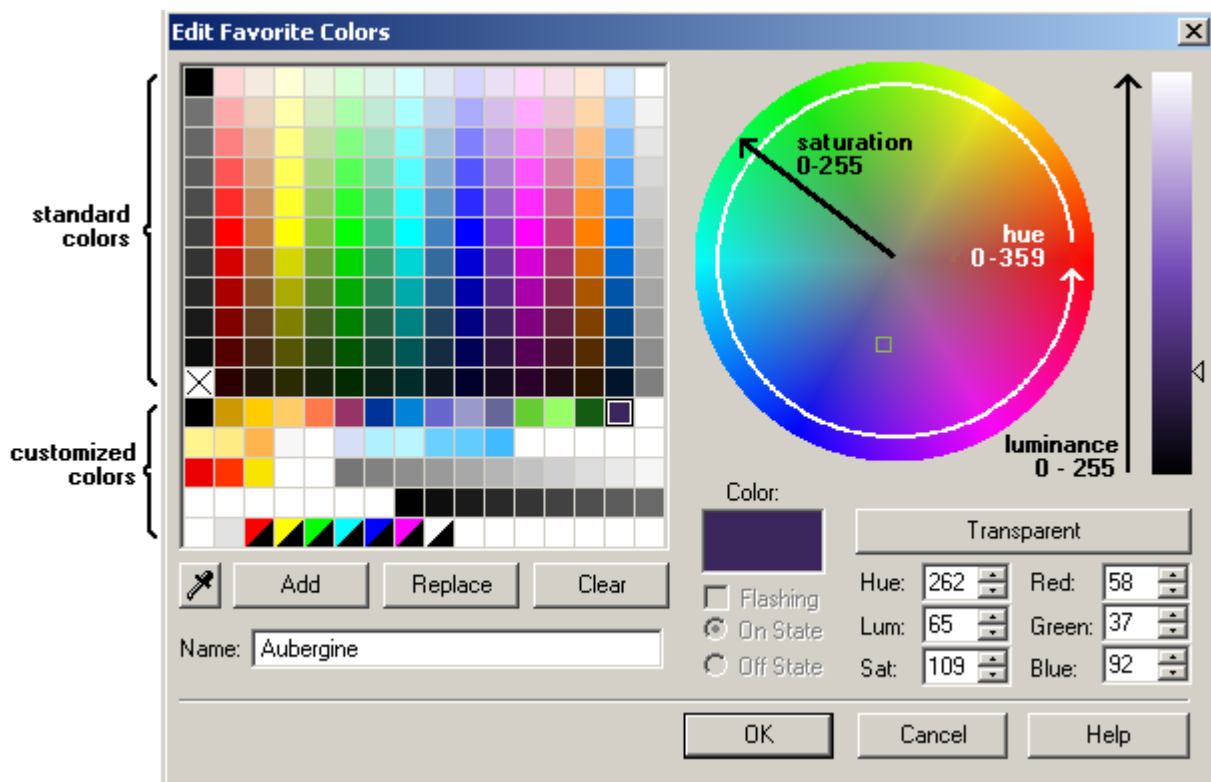
See Also [Edit Favorite Colors dialog box](#)
[Swap Color dialog box](#)
[Adjust colors dialog box](#)

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.

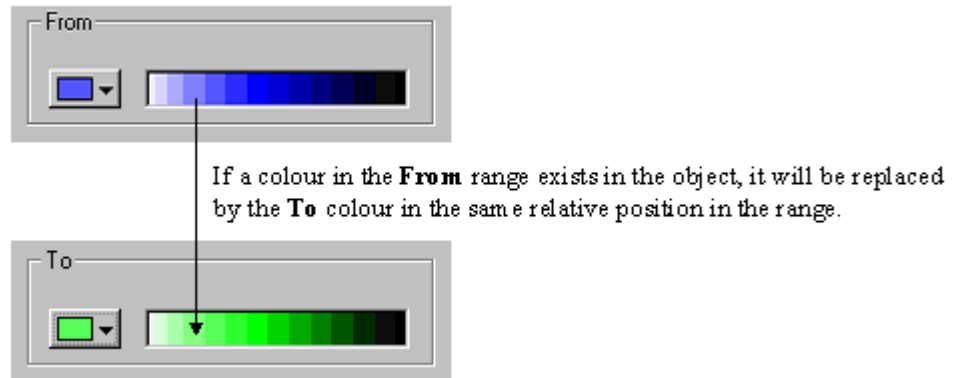
Swap Color dialog box

You use the Swap Color dialog box to swap the colors of an object (or group of objects, or a bitmap) to new colors.

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.

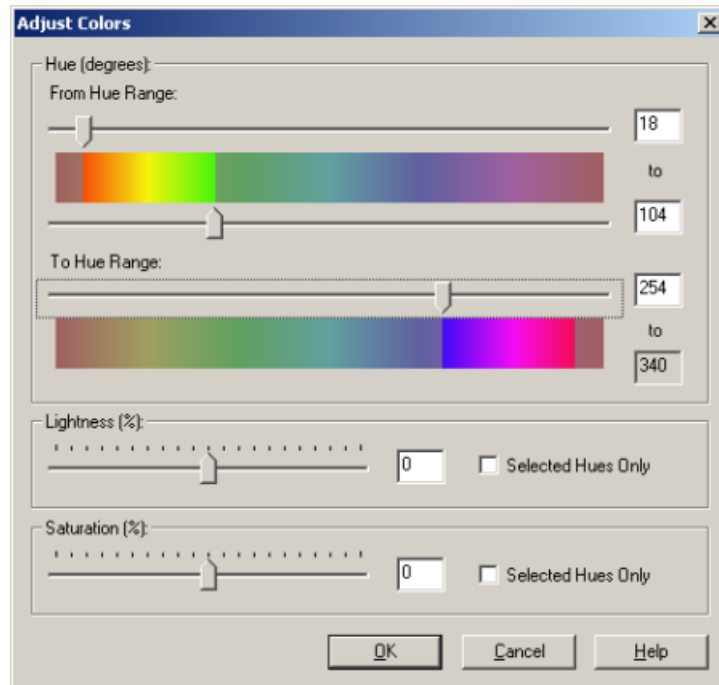
Swap range

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 zoom 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.

To display the Zoom box:

- Choose **View | Show Zoom**.

To hide the Zoom Box:

- Double-click the control menu box (on the Zoom box) or choose **View | Show Zoom**.

You can change the cursor into cross hairs 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.

To toggle the cursor between a cross hair cursor and a pointer:

- Choose **View | Cross Hair Cursor**.

To hide the status bar:

- Choose **View | Show Status Bar**.

To redisplay the status bar:

- Choose **View | Show Status Bar**.

To hide the Toolbox:

- Double-click the control menu box (on the Toolbox) or choose **View | Show Tools**.

To redisplay the Toolbox:

- Choose **View | Show Tools**.

To hide the Toolbar:

- Choose **View | Show Tool Bar**.

To redisplay the Toolbar:

- Choose **View | Show Tool Bar**.

To change the speed of the cursor when you are using the mouse:

- Choose **View | Mouse Slow Speed**.

To change the speed of the cursor when you are using the cursor keys:

- Choose **View | Cursor Keys Slow Speed**.

Hint: Move the cursor on screen 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.

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.

Name Enter a 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:

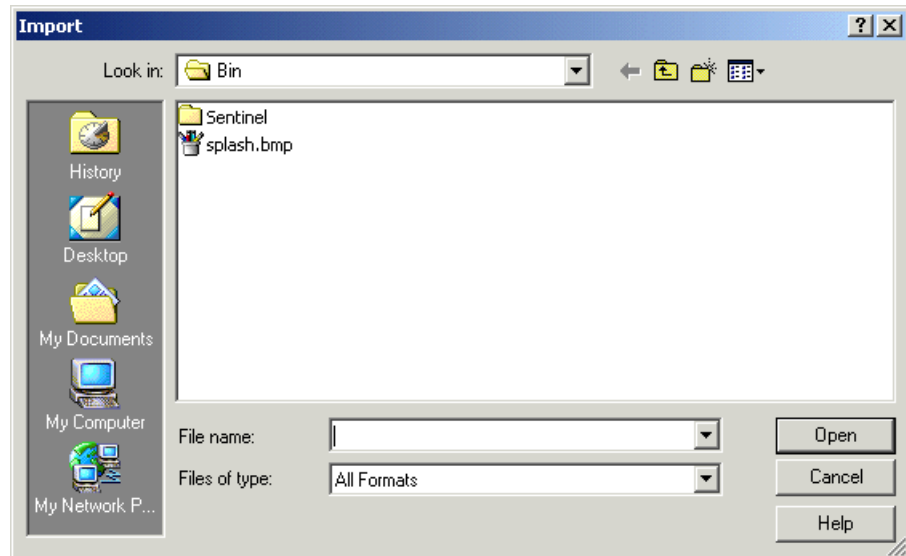
- 1 Choose **File | Import as Flashing**. The Primary Import dialog box appears.
- 2 Select the first file you want to use for your flashing image.
- 3 Click **OK**. The Flashing Import dialog box appears.
- 4 Select the second file you want to use for your flashing image.

Import dialog box

You use the Import dialog box to import a graphic produced with a different application.

Note: If you have selected **Import as Flashing**, two Import dialog boxes will appear in sequence, allowing you to choose two images that you'd like to implement as a flashing symbol. The **Import Primary** dialog, allows you to select

the initial image used, while 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 15: 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

Note: If report data is associated with an I/O device that fails at startup or that goes off line 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, then 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 a .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** if you need 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 [Example]". It contains several input fields and buttons. The "Name" field is set to "Example". The "Time" and "Period" fields are empty dropdown menus. The "Trigger" field is an empty dropdown menu. The "Report Format File" field contains "example.rtf". The "Output Device" field is a dropdown menu set to "replug". 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 ('). If you are using [distributed servers](#), the name must be unique to the [cluster](#) (e.g., you cannot have the same report name in more than one cluster).

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 th
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 all 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, etc..).

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 (i.e. the 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}
```

In this case, the variable is formatted using only four characters (i.e. the number of characters between the braces).

Note that 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}
```

Note: 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 several 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
#COM	Communication with the I/O device has failed
#RANGE	The value returned is out of range
#DIV/0	An attempt was made to divide a number by 0 (zero)
#STACK	The value has caused a stack overflow
#ASS	The value is incorrectly associated (with a substitution string or Genie).
#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.

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 off line 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 off line. 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 16: 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, etc.). 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 backup, or install a new project from a compiled offline master, then the user records will be reset to match those originally configured in the project. If the runtime user creation, password change ability, or password expiry functions are used, then the runtime details might be thrown out of synchronization with master offline projects.

In this situation, you must have procedures in place to use the current Users.dbf file (which is running live in the plant) when any offline project compilations are performed. This will minimize 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.

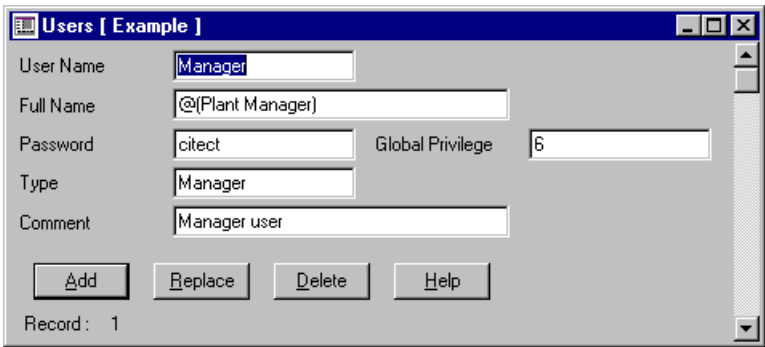
See Also [Adding user records](#)

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.

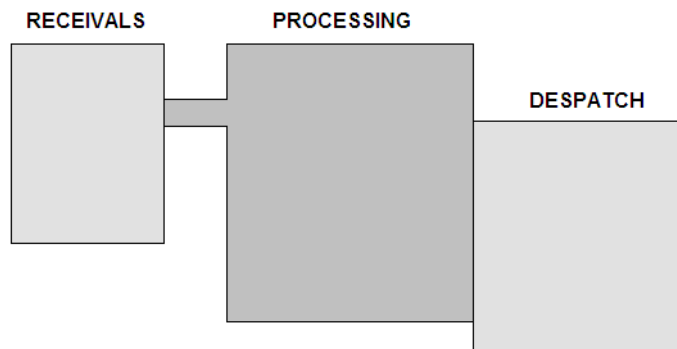
Defining Areas

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).

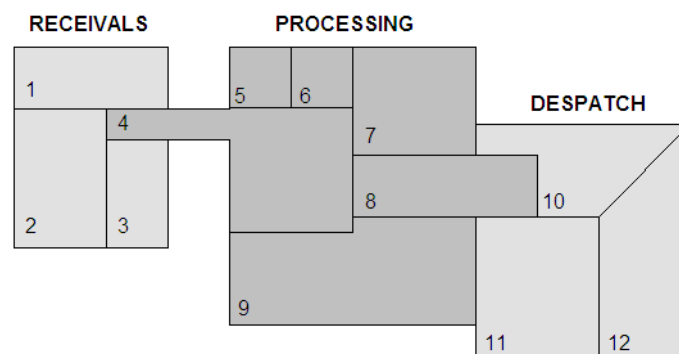
Note: The area facility is implemented with extended properties.

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.

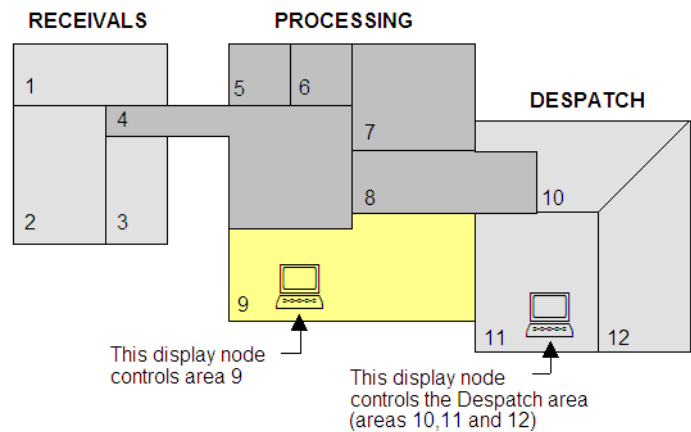


However, with larger or more complex plants you might need to define several areas, like this:



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	<code>CONVEYOR = 1;</code>
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	<code>DespatchAccum</code>
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:

The screenshot shows a Windows-style dialog box titled "Groups [Example]". It contains the following elements:

- Group Name:** A text box containing "Plantwide".
- Associations:** Two columns of five dropdown menus each, labeled "Association 1" through "Association 10". "Association 1" is set to "1" and "Association 2" is set to "2". The others are empty.
- Comment:** A text box containing "PlantWide Group".
- Buttons:** Four buttons at the bottom: "Add", "Replace", "Delete", and "Help".
- Status:** A label at the bottom left says "Record : 1".

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 17: 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.

Note: Labels are similar to the macros used in programming languages such as Basic or 'C'.

You often use the same combination of statements in different commands. For example, each time 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 set of statements each time they are required in a command, you can define a label. You can then use the label instead of the set of statements. When you compile your project, each occurrence of the label is resolved, i.e. the expression in the label is substituted for the label name, for example:

Label	
Label Name	<u>_Log_Alarms</u>
Expression	<code>FileWrite(AlarmFile, Time()); FileWrite(AlarmFile, Date()); . . .</code>

Notice the use of an underscore to identify the label.

Once you have defined a label, you can use it as a statement in a command, for example:

System Keyboard	
Key Sequence	<code>F5 Enter</code>
Command	<code>_Log_Alarms;</code>
Privilege	<code>1</code>

When an operator issues this command, the expression defined in the label is substituted in the command.

Label	
Label Name	<code>_Log_Alarms</code>
Expression	<code>FileWrite(AlrmFile, Time()); FileWrite(AlrmFile, Date()); ...</code>

When an operator issues the command, the statements in the expression execute

System Keyboard	
Key Sequence	<code>F5 Enter</code>
Command	<code>_Log_Alarms;</code>
Privilege	<code>1</code>

You can also use the label in combination with other statements, for example:

Label	
Label Name	<code>_Log_Alarms</code>
Expression	<code>FileWrite(AlrmFile, Time()); FileWrite(AlrmFile, Date()); ...</code>

The label expression is substituted in each occurrence of the label

System Keyboard	
Key Sequence	<code>F5 Enter</code>
Command	<code>AlarmAck(0,0);_Log_Alarms;</code>
Privilege	<code>1</code>

Users	
User Name	<code>Supervisor</code>
Entry Command	<code>PageAlarm();_Log_Alarms;</code>
Privilege	<code>1</code>

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##) )
```

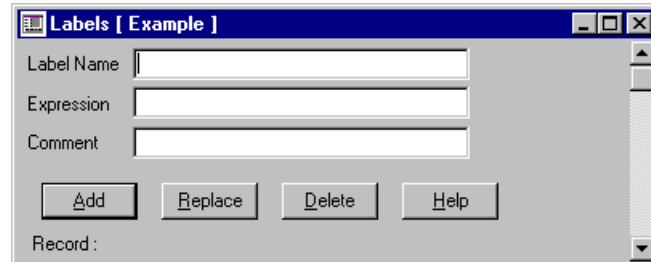
See Also [Defining Labels](#)

Defining Labels

You can define labels to use in your system.

To define a label:

- 1 Choose **System | Labels**. The Labels dialog box appears.

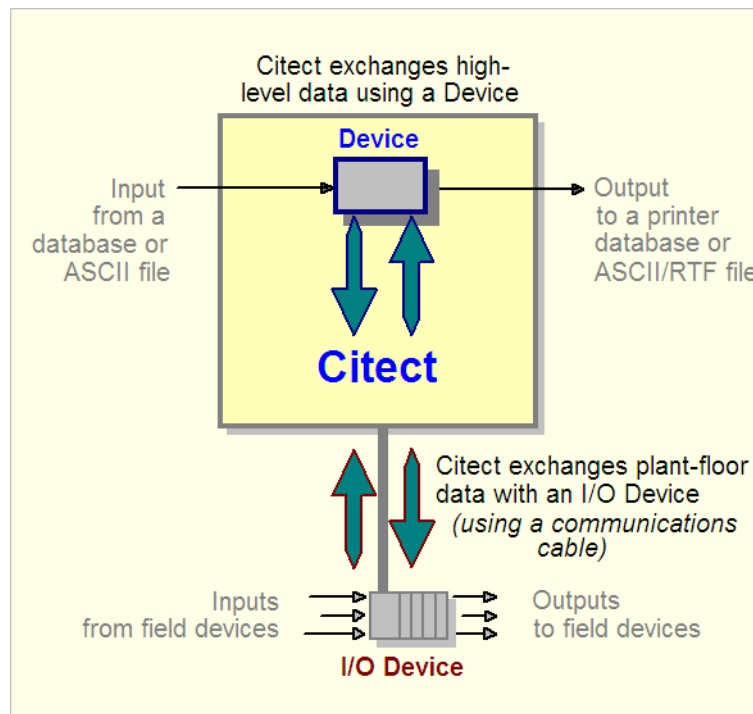


The screenshot shows a Windows-style dialog box titled "Labels [Example]". It has a standard title bar with minimize, maximize, and close buttons. Inside the dialog, there are three text input fields stacked vertically, labeled "Label Name", "Expression", and "Comment". Below these fields are four buttons: "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 list box.

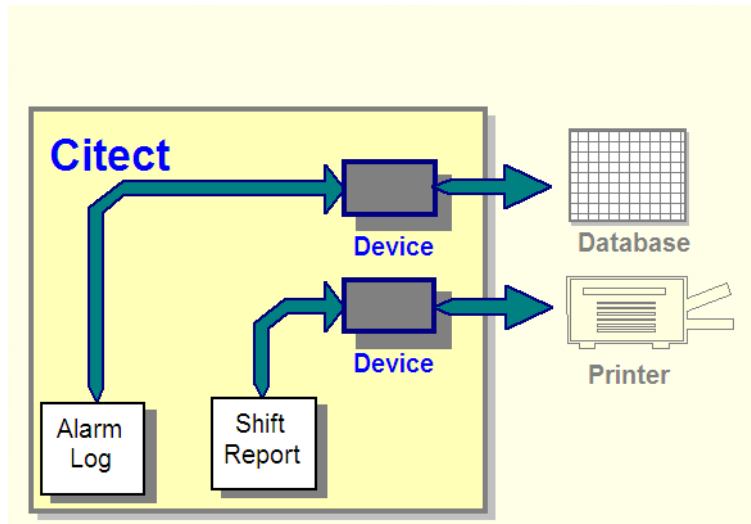
- 2 Enter a **Label Name** of 64 characters or less. Whenever this name is used (i.e., in Cicode or a field), CitectSCADA automatically substitutes the expression below.
- 3 Enter an **Expression** to be substituted for the label. You can use a label to substitute a name for an entity or Cicode expression; 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 18: 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.



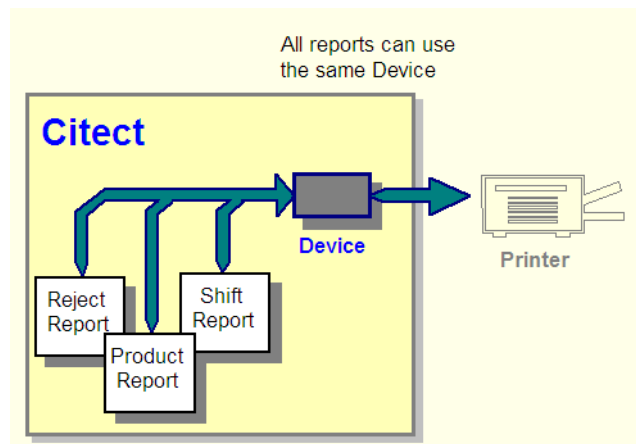
You can use devices for various purposes; for example, to send the output of a report to a printer, or write data to a database.



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)

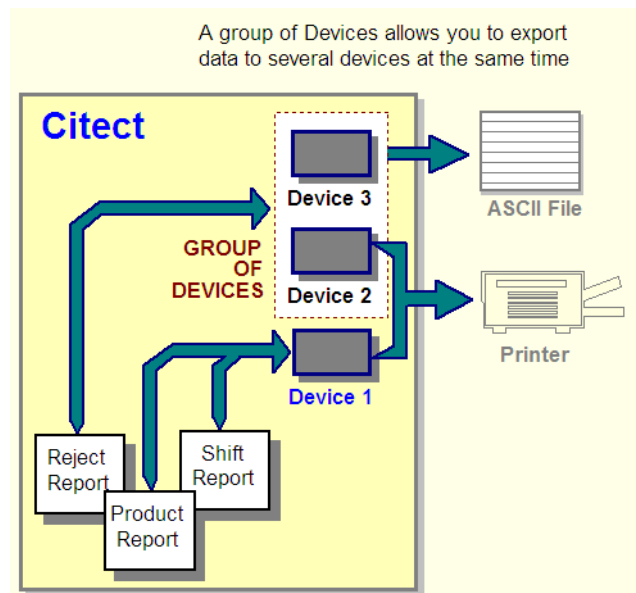
You can configure any number of devices; however, a device is a common resource. You can, for example, configure a single device that sends the output of all your CitectSCADA reports to a printer (when they are requested).



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.

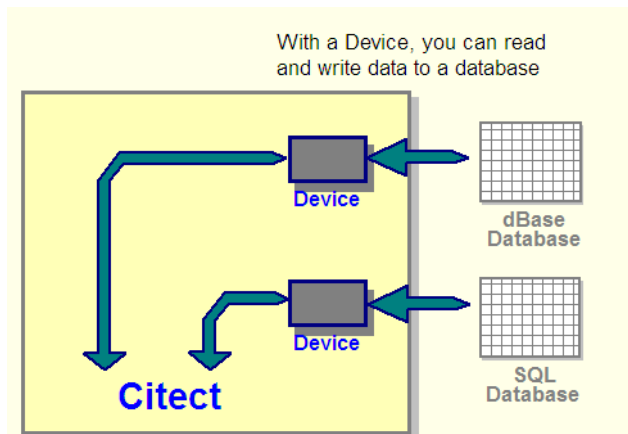


See Also [Using devices to read data](#)

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.

The screenshot shows a Windows-style dialog box titled "Devices [Example]". It contains several input fields and buttons. The "Name" field is filled with "replug". The "File Name" field is filled with "[data]replug.txt". The "Type" field is a dropdown menu showing "ASCII_DEV". The "No. Files" field is filled with "10". There are buttons for "Add", "Replace", "Delete", and "Help" at the bottom. A status bar at the bottom left indicates "Record: 1".

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 \CITECT\BIN\

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.

Warning! 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.

dBASE and SQL database devices format

```
{<field name>, <width>}
```

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);
```

dBASE Devices

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

You can access a device using Cicode functions.

Opening the device

Before you can use a 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, you must 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
```

Note: To delete records from an SQL database, you must use the Cicode SQL functions, discussed in Using SQL.

Closing a CitectSCADA database device

When you have finished with a device, close it to free Cicode system resources. The DevClose() function closes the device:

```
DevClose (hRecipe) ;
```

To define a group of devices:

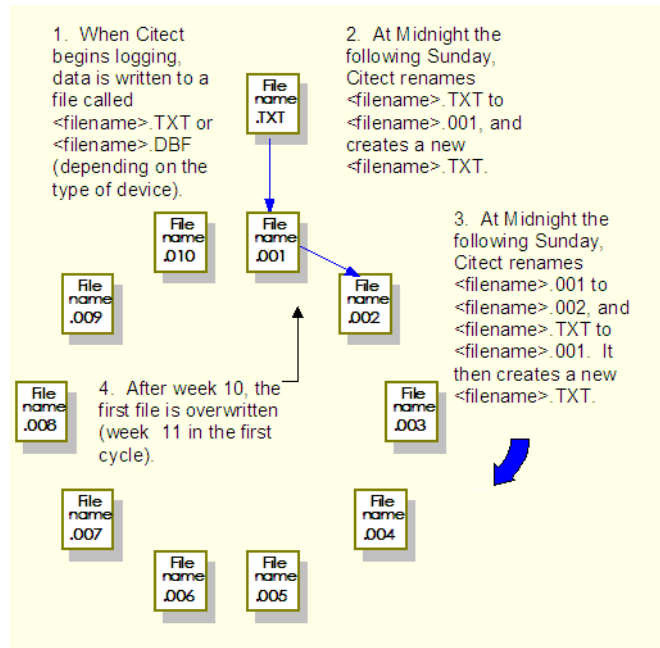
- 1 Choose **System | Groups**. The Groups dialog box appears.
- 2 Complete the Groups form.
- 3 Click **Add** to append a new record, or **Replace** to modify an existing record.

See Also [Using Device History Files](#)

Using Device History Files

CitectSCADA uses a system of rotational history files to store historical data. This makes long-term storage of logged data easier to organize and more accessible. 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

Note: 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

If you want to archive your data for long term storage, you must backup the 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 native language 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 19: 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.

Note: 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](#).

Warning! 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](#).

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 from the following structure:

```
"<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.

Note: 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 for whatever reason is 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 Remote Requests** check box is cleared (the default setting). In Excel 4, use the **Options | Workspace** command, or in Excel 5 and later, use **Tools | Options | General**.
- 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.

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 Remote Requests** check box is cleared (the default setting). In Excel 4, use the **Options | Workspace** command, or in Excel 5 and later, use **Tools | Options | General**.

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](#).

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 Remote Requests** check box is cleared (the default setting). In Excel 4, use the **Options | Workspace** command, or in Excel 5 and later, use **Tools | Options | General**.

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 using the same DDE protocol, however, shares 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 shipped with all versions of

Windows from (and including) Windows For Work Groups (WFWG) Version 3.11, and is used to communicate the shared dynamic data exchange used by Network DDE. NetDDE.EXE has no graphical user interface (it runs as a background Windows service).

Microsoft disabled the automatic startup of the Network DDE Services in all Windows Operating Systems shipped after version 3.1, so therefore with WFWG, WIN9x, Windows NT, and later versions, 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 WFWG and Windows 9x systems, store a shortcut to NetDDE.EXE (located in the Windows directory) in the Windows Startup folder.
- With Windows NT based systems (NT4, WIN2000, 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 when running. To view the Windows Task Manager, press **CTRL+ALT+DEL**. WIN9x systems display NetDDE.exe in the applications list, whilst NT based systems (NT4, WIN2000, and later) display it on the **Processes** tab.
- In Windows NT based systems, 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 to the system32 folder on Windows NT based systems.

- 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.

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 topic uses Chat only as an example to prove 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](#).

Setting up network DDE shares

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 don't have to create a DDE Share if you are attempting to use DDE between applications running on the same machine. You only have to create a DDE Share if you intend to use DDE between two separate applications running on different machines. Then 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 NT based operating systems (NTx, Windows 2000, Windows XP, and later), located in the WINNT\System32 sub-directory.

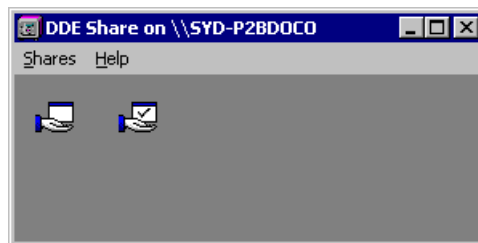
Note: For 16-bit based systems like Windows 95, Windows Me, and Windows For Workgroups, (according to MSDN Knowledge Base article Q181946) the original 16-bit Network DDE Share Manager version, (also named DDEShare.exe) can be found in the Microsoft Windows for Workgroups Resource Kit. As this file proved most difficult to locate, and for your

convenience, a copy is also available from the Citect website toolbox:
www.citect.com/login.

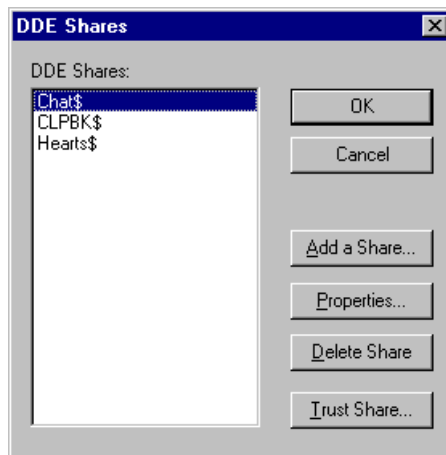
To manually launch the DDE Share utility:

- On the Windows Start menu, click **Start | Run**, type in "ddeshare" (without the quotes) and press the **Enter** key.

When the NT version of 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 tick) 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>\$

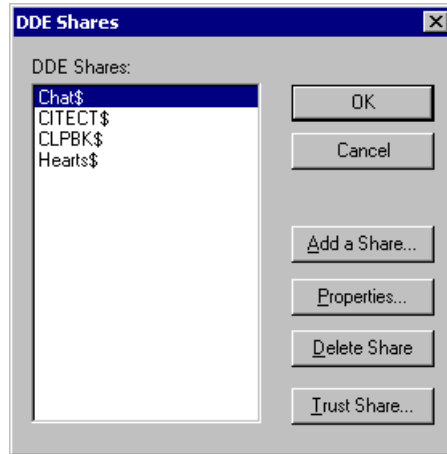
so to set-up a [CitectSCADA server](#) computer as a Network DDE share, use the name "Citect\$" as the sharename on that computer. To expose the CitectSCADA runtime variable tag database for suitable DDE linking, use the word "Variable" as the DDE Share Topic name.

Note: 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 displays. Complete the fields exactly as shown here:

- 2 Click **Permissions**. The DDE Share Name Permissions dialog displays.
- 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

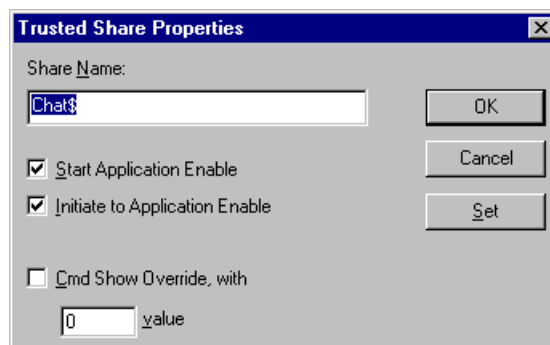
When a network DDE Client user connects to a network DDE Share from a remote computer, Network DDE accepts the request only if both:

- 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 tick) 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](#).

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](#).

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 (*) on Windows NT-based systems.

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 DDEFORUMU.XLS (in the CITECT\BIN

directory), which was updated to support Network DDE with CitectSCADA version 5.41 and later.

Warning! 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 DDEFORUMU.XLS, in Excel 8, from the main menu, choose **Tools | Options** and on the General tab uncheck Macro Virus Protection.

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 DDEFORUMU.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 DDEFORUMU.XLS, Excel will warn you that the file contains macros. To enable the Citect Tags features, you must select **Enable Macros**.

When started, the Citect Tags macros expect that CitectSCADA runtime is operating on the same machine. If not, Excel will produce 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 context sensitive right-click popup 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 DDEFORUMU.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.

Note: Some lines above might have wrapped due to page size limitations. Note that Cicode does not support code written over more than one line and has no line continuation character. Cicode does use the semicolon as the end of line character. If you copy these examples into your project, reassemble any lines that have wrapped and place them back onto the one line in your code.

Refer to the documentation that accompanied your SQL server for information 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')");
```

Note: 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

```

Note: 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

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.

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'.

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 a wide variety of databases. One such supplier is Intersolv Q+E with their DataDirect ODBC Pack. Drivers from this package will give full backward compatibility to the drivers used in CitectSCADA v2.0. In most cases, 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

Below are listed the required and optional ODBC functions that your database driver should support.

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 they 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

Note: 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](#).

How to set up the CitectSCADA ODBC server

Note: 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 the 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**.

How to access the CitectSCADA ODBC server using MS Query (V2.00)

All ODBC capable applications have different ways of constructing queries for accessing CitectSCADA tags. The example instructions for using MS Query, given here, are provided to show a simple implementation. You will find that MS Excel and MS Access follow 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**.

How to access 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 the Program Files\Common Files\ODBC\Data Source folder. File DSN's 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, e.g. Notepad, and create a file containing the following two lines:

```
[ODBC]
DSN=<MyUsrDSN>
```

where <MyUserDSN> 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 without needing to go Browsing.
- 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:
Select Citect Driver and Click Finish button;
Enter "Citect" in the Data Source field (ie <MyUsrDSN>);
Enter Computer Name in the Host field.

When you run MS Query, you will be able to 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

A SELECT query can be used 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 required at the time, if the data needs to be sorted or if there is a requirement 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
    END
EndFunction
```

```

        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, an SQL INSERT statement can be used.

```

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, the standard Cicode Device Functions can be used to append records to an Access table. Firstly configure an SQL Device. If the table has a lot of 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 as simple as possible and reduces the number of DevWrite instructions). DevOpen, DevWrite and DevClose can then be used to add records to the table.

CitectSCADA will accept successive DevWrites until they equal the number of fields in the device definition at which time it will construct an SQL INSERT statement. The DevWrites must contain data for fields in the same order as the device definition. It is best to do a DevOpen followed immediately by successive DevWrites for as many records as are required then a DevClose to avoid the risk of the 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: The ODBC/SQL environment does not provide the facility to edit the "Current Record". There is in fact NO Current record. For this reason DevAppend and DevSetField cannot be used to add or modify records.

See Also [Deleting rows from an Access table](#)

Deleting rows from an Access table

The DELETE keyword is used in conjunction 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}"
```

Note: Note that 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](#)

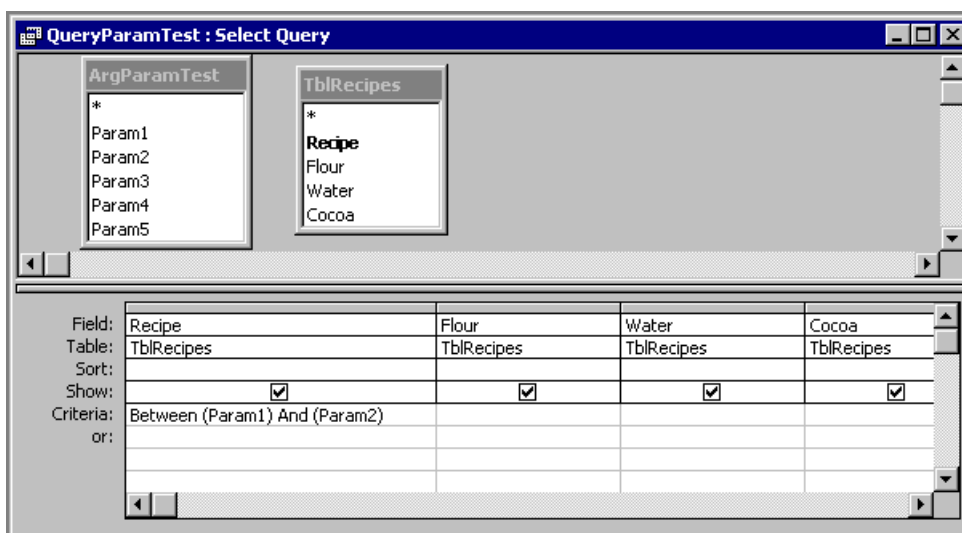
Parameter queries

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. When this has been done, 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");
```

Note: 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.

Note: 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

Note: All date references in an external database should be based on the Gregorian Calendar.

Access and Cicode have different Date/Time variables data types. There are three ways to convert between the two:

Convert to real numbers

In both Cicode and Access it is possible to equate real numbers to Data/Time variables. The conversion between the two systems is as follows:-

```
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 be useful in a WHERE clause:

```
SELECT * FROM qryMyQuery WHERE 'Date' BETWEEN #3/20/96# AND #3/27/96#
```

Note: 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 20: 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).

Note: 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:

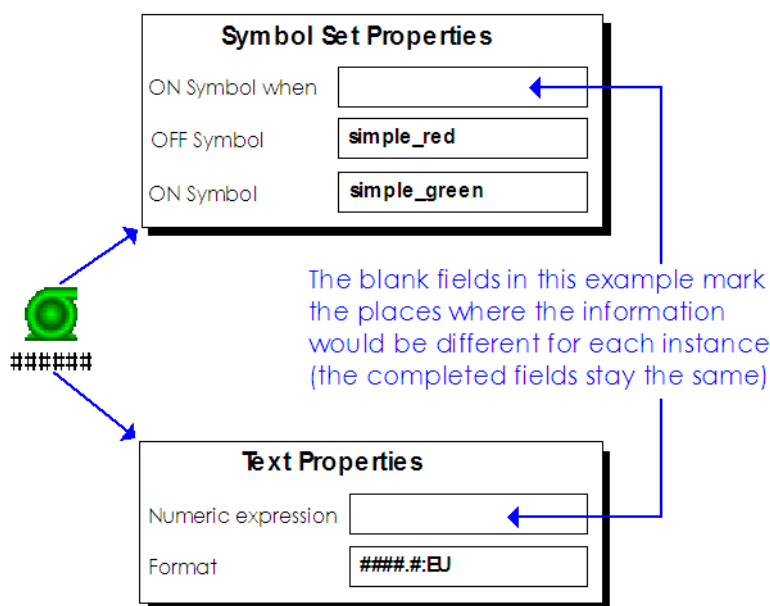
- If you modify a Genie or Super Genie after you have used it in your project, all occurrences of the Genie or Super Genie are automatically updated throughout the project (with the exception of Super Genie Environment Variables).
- If you modify a Genie when the project is running in the background, 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.

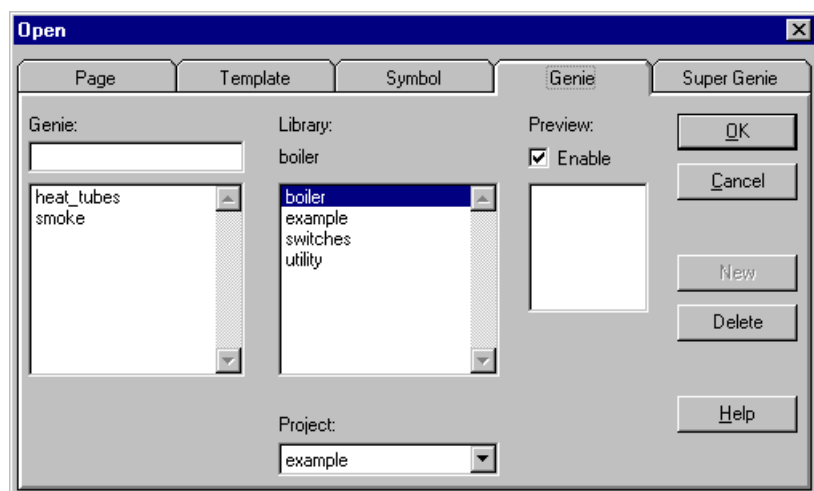
See Also [Creating Genies](#)
[Opening a Genie](#)
[Saving a Genie](#)
[Defining Substitutions for Genies](#)
[Using Genies](#)

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.

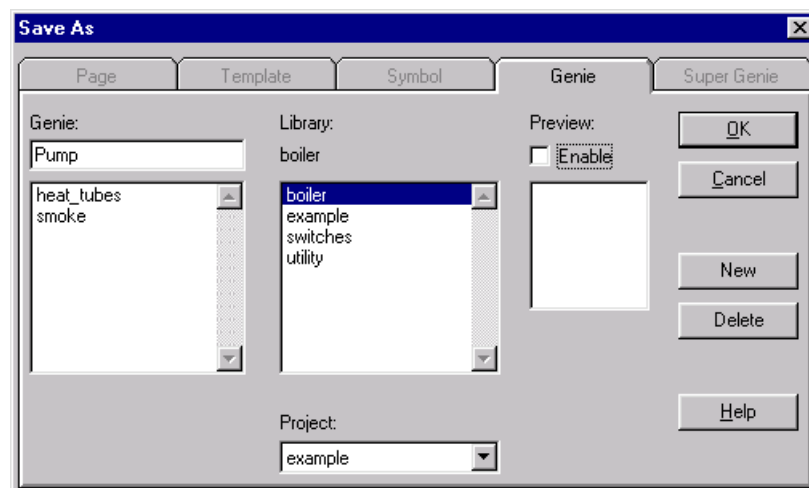
See Also [Saving a Genie](#)

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**.

Note: To create a new library for the Genie, click **New**.

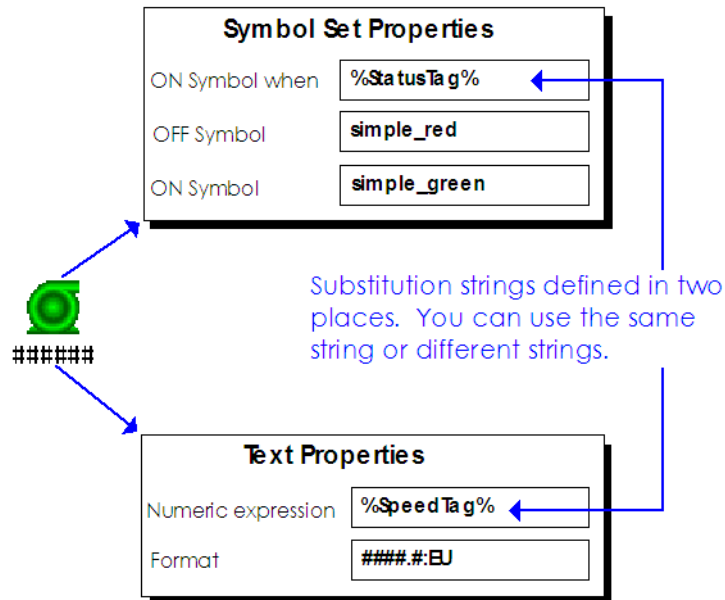


See Also [Defining Substitutions for Genies](#)

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.

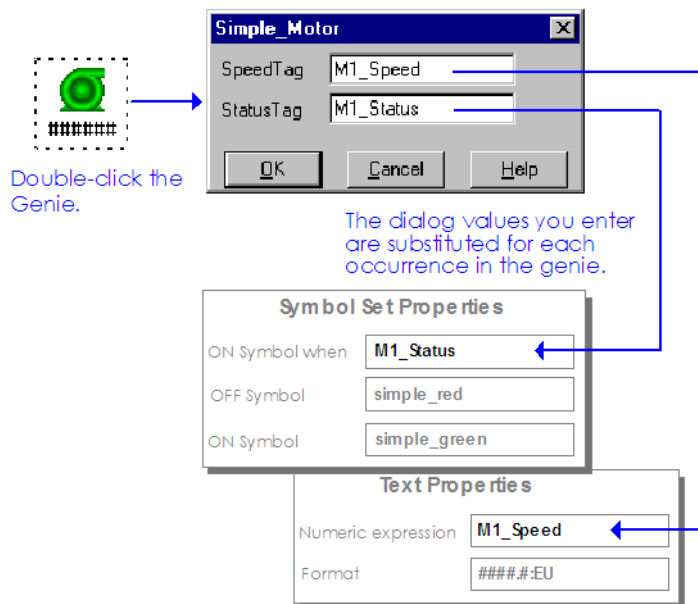
See Also [Using Genies](#)
[Using Genie Substitutions in Templates](#)

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.



Note: 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.

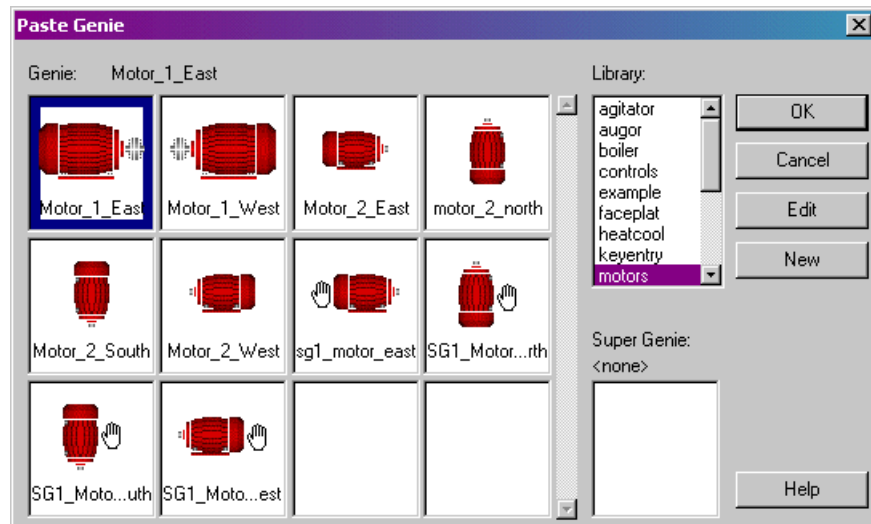
The above example is a simple use of a Genie - it only contains two objects and two substitution strings. You can define Genies that use many objects, with substitution strings for any **text** property (or properties) of an object.

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 object. If, however, a link to the Genie has been retained, most of these properties will be read-only.

See Also [Understanding Genies](#)

Using Genie Substitutions in Templates

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.)

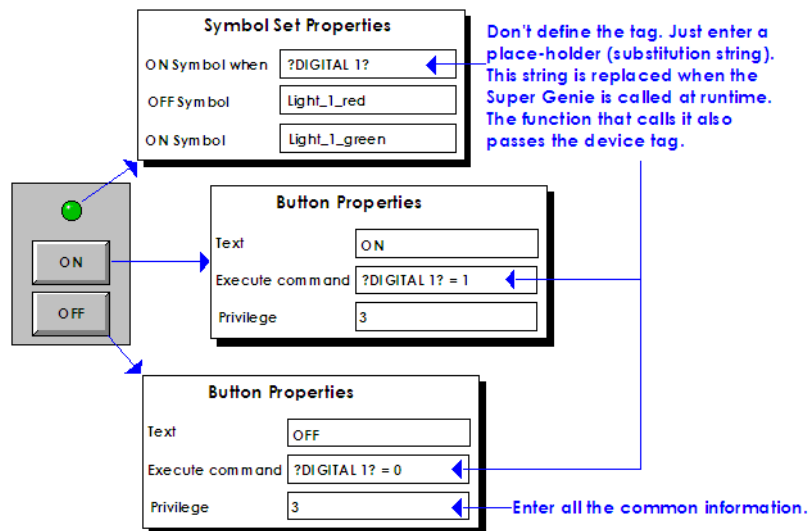
When you subsequently create a new page based on the template, a single dialog prompts you for values for all substitution strings used in the template. This is how the templates for trending and SPC were created.

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.

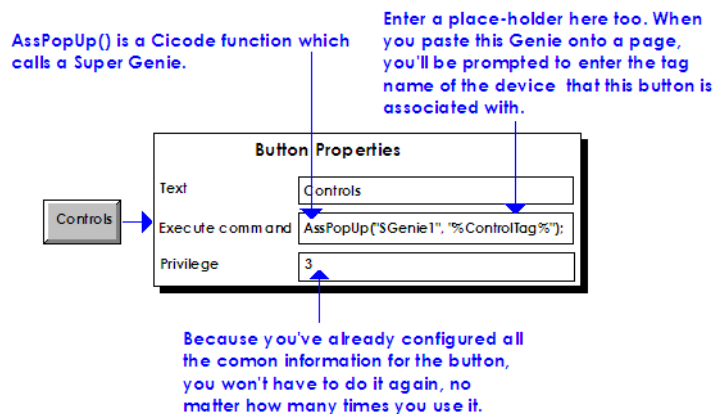
For instance, you might use a Super Genie to configure a single popup page for controlling all electric pumps that have the same functionality. The best way to configure this controller is:

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.

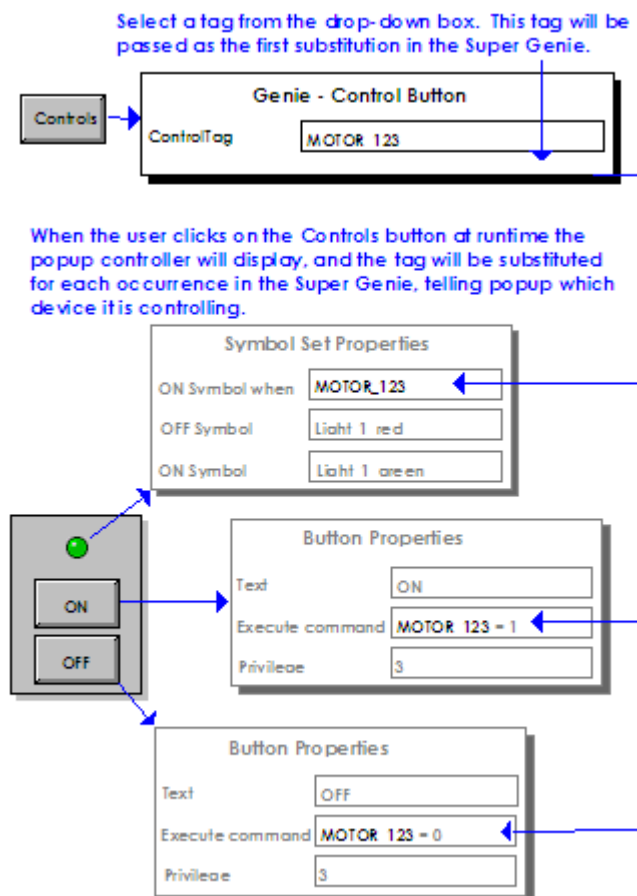
Save your Genie to a Genie Library. Like Genie libraries, Super Genies libraries are global and can be used between CitectSCADA projects.

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: Super Genies are an advanced tool and require careful design. You should be comfortable with Genies and Cicode before attempting to use Super Genies.

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 to make use of 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 provide good performance, 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](#).

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.

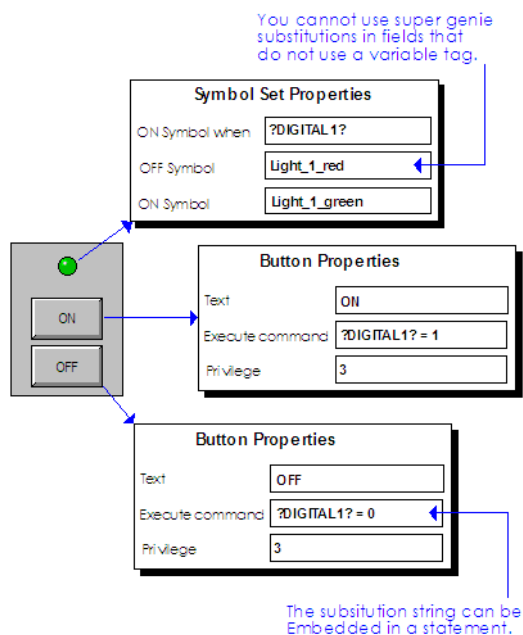
To mark a tag as a substitution string, enclose the tag between question mark (?) characters, in the following format:

?<Data Type> <Substitution String Number>?

where:

- **Data Type** is optional and can be any data type supported by CitectSCADA (BYTE, BCD, DIGITAL, INT, UINT, LONG, LONGBCD, REAL, or 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 will remain 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 a !) 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 is a simple illustration of the power of Genies. The more complex and 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 - it 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.

IFDEF format

```
IFDEF (<"Tag name">, <Hidden When value if tag defined>, <Hidden  
When value if tag undefined>)
```

The IFDEF statement consists of three arguments. 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.

Click **OK**.

Chapter 21: 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. The desired language can be specified before you run the project, or changed dynamically at runtime (using the `SetLanguage()` function) - without affecting any of the project's normal operations.

CitectSCADA distinguishes between what is termed the native language (e.g. the language of the developer), and the local language (e.g. the language of the end user). Language changes are achieved by means of a [language database](#), which has a field for native text, and a field for the translated local text. When the project is run, any native text will be replaced with the equivalent local text.

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 data can be logged to the same device, or to separate devices.

See Also [Changing Languages](#)

Changing Languages

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, any text which is to be changed to another language at runtime must be marked with a language change indicator, in the following format:

```
@( Native Text [,Width [,Justify]])
```

Where *Native Text* is the identifying text which will be displayed when configuring. This text will be replaced by the local equivalent at runtime. Note that the brackets are a necessary part of the indicator. They specify the extent of the native language text. The *Width* and *Justify* are optional (indicated by the square brackets).

For example, if English is the native language, the following alarm description could be entered:

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 lower case. 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.

The *Width* field can be assigned any value from 0 to 254. If the local text is longer than specified, it will be truncated and left justified. If a width is not specified, the field will be the length of the local text, and the text will be left justified.

The *Justify* field specifies the text justification and can only be used in conjunction with the *Width* field. *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, you must 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 will not 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. 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.

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.

Note: For the language change to occur automatically when the project is run, you must specify the language database which is to be used *before* you run the project. This is done 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 will be 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).

Note: For single byte languages (such as French), the database can be edited using Microsoft Excel, but for double byte languages (such as Chinese), it is recommended that Visual FoxPro be used.

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 will be updated each time you compile. Any text which has been marked since the last compile will be appended to the end of the database - the rest of the database will remain unchanged.

See Also [Multiple languages](#)

Multiple languages

Note: When you want 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, it is important that you know what database is set *before* you compile.

Also, if you have several language databases with the same native language, you should remember that newly marked text will only be appended to the *current* local language database (as specified by the **[Language]LocalLanguage** parameter). If you want this text to be added to other databases with the same native language, you should change the **[Language]LocalLanguage** parameter, update pages, and re-compile for each database. Remember that for each database, only the relevant changes made since the last compile will be 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, which is applicable to a whole range of 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 etc. can be changed dynamically at runtime, using the `SetLanguage()` function. All normal operations of the project will continue unaffected.

Note: Forms will not automatically update when the language is changed using the `SetLanguage()` function. They must be closed and re-opened for the change to take place.

Any local translations that are missing from the specified language database will be 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 alpha-numeric character set of the English language. For example, the character capital 'A' has the ASCII value of 65, and the character lowercase 'a' has the ASCII value of 97.

The ASCII character set contains 96 characters, it is commonly used as a standard for protocols and files.

Windows 95 uses ANSI (American National Standards Institute) character sets. ANSI character sets are language based, with each different language version of Windows (French, Korean etc.) requiring a specific ANSI character set. Codes 32 to 127 always contain the standard ASCII characters.

Note: Windows NT uses Unicode, but still supports ANSI character sets. Unicode avoids the problem of multiple character sets by having one 16 bit - worldwide - character encoding standard. To support both Windows 95 and NT, CitectSCADA must use ANSI character sets.

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 will convert 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 that 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 will 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 22: Using the Computer Setup Wizard

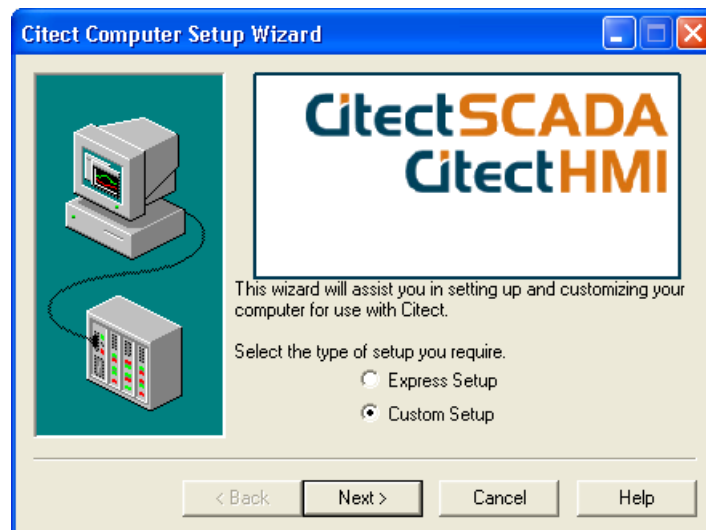
Use the Computer Setup Wizard to set up and customize your PC for use with CitectSCADA. The wizard must be run on each computer running CitectSCADA in your system to define its role and function in relation to CitectSCADA. Run the wizard as the last step in setting up your CitectSCADA system.

Use the Computer Setup Wizard to:

- Define the role your computer will play in your CitectSCADA system (a server and client, a client, or a [manager client](#)).
- Set up alarms, reports, trends and events functionality.
- Select options that affect how the application handles at runtime.

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**. The Citect Computer Setup Wizard appears.



Computer Setup Wizard - introduction

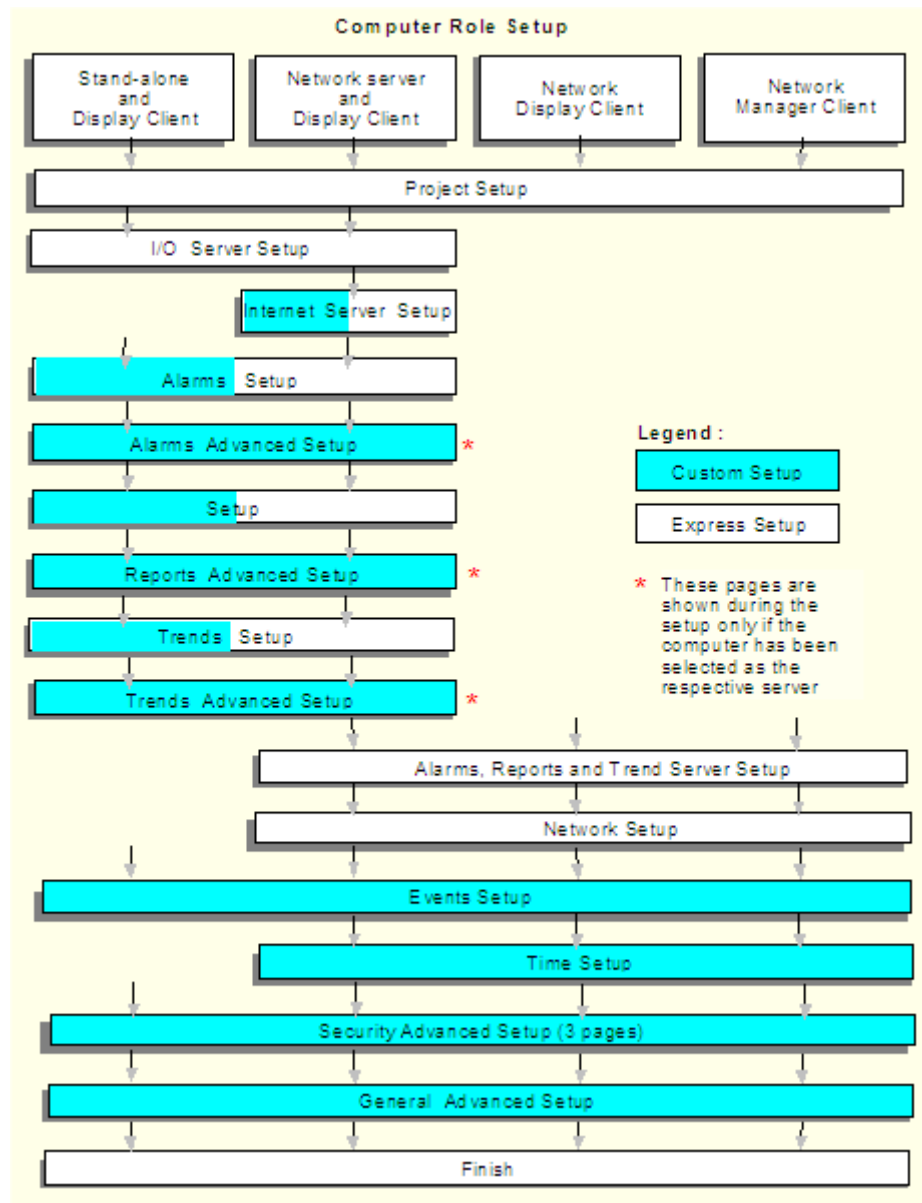
You must run the Computer Setup Wizard on each computer running CitectSCADA in your system to define its role and function in relation to CitectSCADA.

Select **Express Setup** or **Custom Setup**. The [Computer Setup Wizard flow diagram](#) below shows the differences between the two setup procedures.

See Also [Computer Setup Wizard - Computer role](#)

Computer Setup Wizard flow diagram

The illustration below shows the differences between the different types of computer role setup.



Computer Setup Wizard - Computer role

Select if this CitectSCADA computer will be the stand-alone CitectSCADA [I/O server](#) and [display client](#), or this computer is one of several CitectSCADA computers connected together by a network.

If the computer is networked, select if this CitectSCADA computer will be a CitectSCADA I/O server and display client, a display client only, or a manager client only.

See Also [Computer Setup Wizard - Project](#)

Computer Setup Wizard - Project

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

See Also [Computer Setup Wizard - I/O Server](#)

Computer Setup Wizard - I/O Server

Select the I/O server from the selected project to run on this CitectSCADA computer. The Computer Setup Wizard will show you all I/O servers defined in the project. If there is only one, it will be selected automatically.

See Also [Computer Setup Wizard - Internet server](#)

Computer Setup Wizard - Internet server

Select if this computer is to be used as 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).

To determine the TCP/IP address of the Internet server computer:

- For Windows NT4, go to the Command Prompt, type IPCONFIG, and press **Enter**.
- For Windows 95, select Start | Run, type WINIPCFG, and press **Enter**.

You also have the option to enter the IP address of an alternate Internet server. The Internet Display Client will automatically seek connection to this alternate computer should connection to the first fail. Note that this can only happen automatically if an initial connection has previously been made to the first specified Internet server.

See Also [Computer Setup Wizard - Alarm](#)

Computer Setup Wizard - Alarm

Select if this CitectSCADA computer is to function as an alarms server.

If this computer is stand-alone it must be selected as an alarms server in order to display alarms. If this computer is to be an alarms server on a network, then select this computer to be the primary alarms server or the Standby alarms server. Note that for a networked computer to be an alarms server it must also be the I/O server or must be able to communicate with the I/O server on the network.

See Also [Computer Setup Wizard - Advanced Alarms](#)

Computer Setup Wizard - Advanced Alarms

This CitectSCADA computer is to function as an alarms server. CitectSCADA has several options available for alarm processing:

- **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 file. CitectSCADA uses two save files, usually one for each of the two alarms servers. The save primary path is the directory where this alarms server will create its save file. When restoring the file, the most recent (of the primary and secondary) save files will be used.
- **Standby alarms server save path:** The path to the secondary save file.

See Also [Computer Setup Wizard - Reports](#)

Computer Setup Wizard - Reports

Select if this CitectSCADA computer is to function as an [reports server](#).

If this computer is stand-alone it must be selected as a reports server in order to display reports. If this computer is to be a reports server on a network, then select this computer to be the primary reports server or the standby reports server. Note that for a networked computer to be an reports server it must also be the I/O server or must be able to communicate with the I/O server on the network.

See Also [Computer Setup Wizard - Advanced reports](#)

**Computer Setup Wizard
- Advanced reports**

This CitectSCADA computer is to function as an reports server. CitectSCADA has several options available for report processing:

- **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 [Computer Setup Wizard - Trends](#)

**Computer Setup Wizard
- Trends**

Select if this CitectSCADA computer is to function as an trends server.

If this computer is stand-alone it must be selected as an trends server in order to display trends. If this computer is to be a trends server on a network, then select this computer to be the primary trends server or the Standby trends server. Note that for a networked computer to be an trends server it must also be the I/O server or must be able to communicate with the I/O server on the network.

See Also [Computer Setup Wizard - Advanced trends](#)

**Computer Setup Wizard
- Advanced trends**

This CitectSCADA computer is to function as an trends server. 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 [Computer Setup Wizard - Server](#)

**Computer Setup Wizard
- Server**

This is where CitectSCADA alarms server, trends server, and reports server names are defined.

If you have configured your computer to be a server you must specify its name. You will be prompted to enter the following:

- **This Server Name:** Enter a name for this server. All other nodes on the network will reference this server by its name.
- **Other Server Name:** The name of the redundant server (that this server will work in conjunction with if redundancy is being used).

If you have configured your computer as a client and not a server you must specify the name of the server it will use. You will be prompted to enter the following:

- **Primary Server Name:** The name of the primary server that will service this display node. (Only required if all servers are disabled.)
- **Standby Server Name:** The name of the Standby server that will service this display node (if required). (Only required if all servers are disabled and if redundant servers are being used.)

Note: If your computer has multiple servers, all primary servers (primary Alarms, primary Reports, and primary Trends Servers) must have share the same name (e.g. "Central"), and all standby servers must also share a common name (e.g. "Secondary").

See Also [Computer Setup Wizard - Network](#)

Computer Setup Wizard - Network

Enter a name for this CitectSCADA computer. Each computer should have a unique name. This name is used by to individually identify each CitectSCADA computer on a network. It is also used with the MsgOpen(), ClusterSetName(), and ClusterGetName() functions.

Note: You can choose any name. Try to choose a name that is logical but not cryptic.

If you leave the computer name blank, CitectSCADA sets the computer name to the same name as the Windows Computer Name (as set up in the Network Properties section of the Windows Control Panel).

See Also [Computer Setup Wizard - Events](#)

Computer Setup Wizard - 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. Select this option if Events are to be enabled on this CitectSCADA computer.

For specific events to run on a computer they must be individually enabled on that computer.

The event names (defined in your project) allow you to specify which events you want to run on this computer. The event name does not have to be unique, you can specify many events with the same name.

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.

The Custom Setup Wizard only displays named events from the selected project. If you are using Events in included projects you will need to edit your CitectSCADA Configuration file (Citect.ini) to add these under the [Events] section header.

See Also [Computer Setup Wizard - Time](#)

Computer Setup Wizard - Time

Select to synchronize this computer's time with the Time Server or select this CitectSCADA computer to be a Time Server. Choose one of the following options:

- **Synchronize time with the Time Server:** Causes the time on this computer to be updated regularly from the Time Server during run time.
- **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 [Computer Setup Wizard - Menu security](#)

Computer Setup Wizard - Menu security

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.

- **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 result in degraded picture quality. If this is unacceptable, you should re-design the page using the desired resolution.

- **Shutdown on menu:** Allows the operator to use the control menu (top left icon) to shut down CitectSCADA at run time. The shutdown is not password- or privilege-protected. Disabling this provides better security.
- **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 [Computer Setup Wizard - Keyboard security](#)

Computer Setup Wizard - Keyboard security

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 [Computer Setup Wizard - Miscellaneous security](#)

Computer Setup Wizard - Miscellaneous security

The following features might cause security problems and you can disable if you want:

- **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 [Computer Setup Wizard - General options setup](#)

Computer Setup Wizard - General options setup

You have the following options for setting up general options:

- **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 [graphics page](#) to display when CitectSCADA starts up.
- **Page scan time:** 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.

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).

- **Startup Cicode function:** Specifies the Cicode function to run when CitectSCADA starts up. You can only pass constant data to the startup function call; you cannot pass variables or other functions. For example, MyFunc(1, "str") is valid, but MyFunc(PLCTAG, "str") or MyFunc(YourFunc(), "str") is not.

See Also [Computer Setup Wizard Finish](#)

Computer Setup Wizard Finish

- Save the setup (or go back through the forms if an adjustment is required).

Chapter 23: 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 analysers, 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.

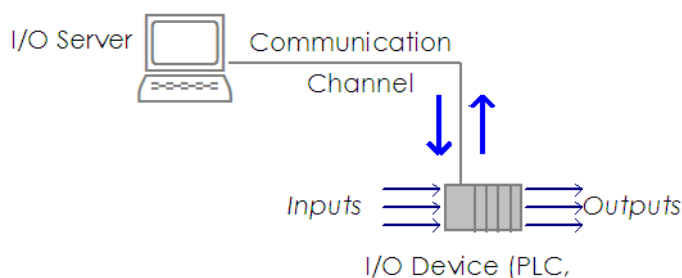
See Also [How CitectSCADA Communicates with I/O Devices](#)

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 (i.e., 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 `ReRead()` after the write and then verify the value of the variable. `ReRead()` forces Cicode to re-read all I/O device variables associated with the current Cicode so that you can check the new value.

```
PLC_VAR = 1234;
ReRead(1);
IF PLC_VAR <> 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 `ReRead`, 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](#).

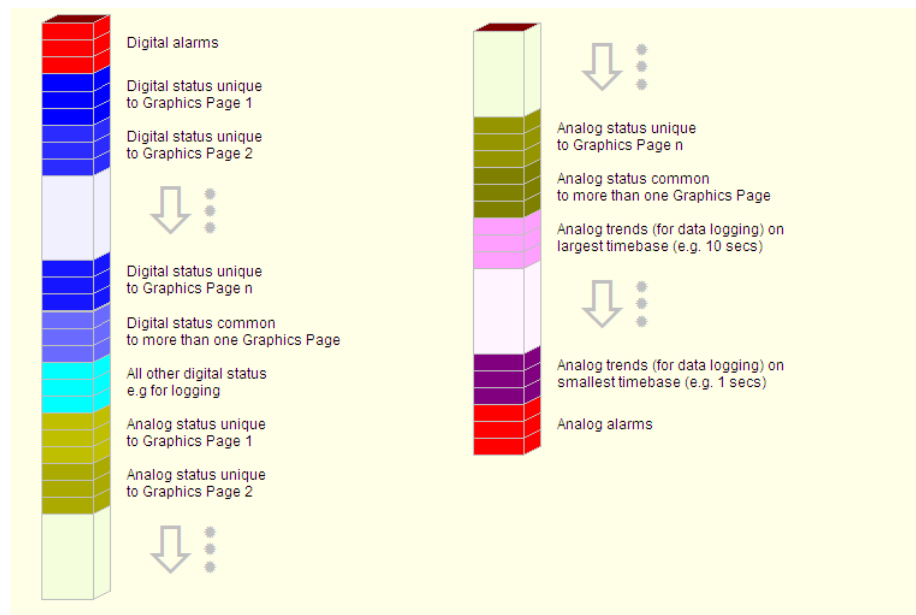
However, reading a block of registers where only a small percentage of the block is actually used is also inefficient. If the registers that your CitectSCADA system will read are scattered throughout the memory of your I/O device, excessive communication will be required. CitectSCADA must either read 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 (i.e., 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](#)

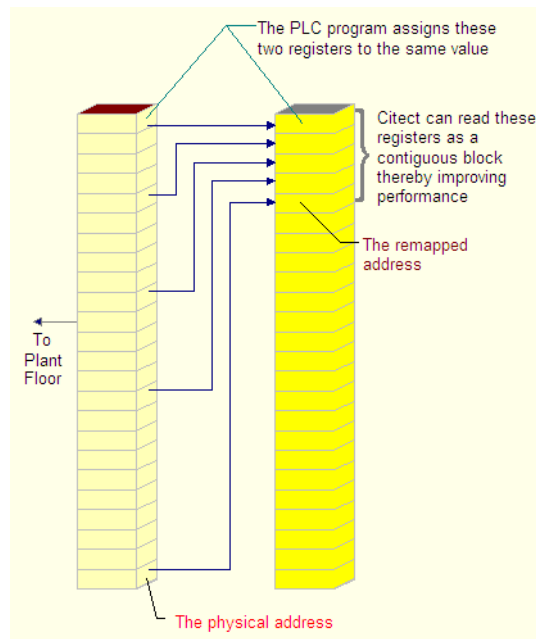
Remapping variables in an I/O device

Some PLCs allow you to remap (or copy) an I/O device variable to another register address. CitectSCADA allows you to remap to:

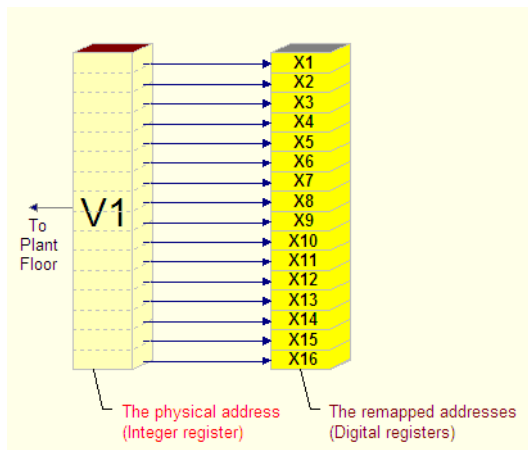
- Group registers more efficiently to increase performance.
- Allow CitectSCADA to interpret a variable type (e.g., an analog variable) as a different variable type (e.g., a digital variable). For example, you can create additional digital addresses if an I/O device has run out of digital addresses.

Note: You cannot remap disk and memory I/O devices.

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 (e.g., an integer) to another type of variable (e.g., a digital variable).

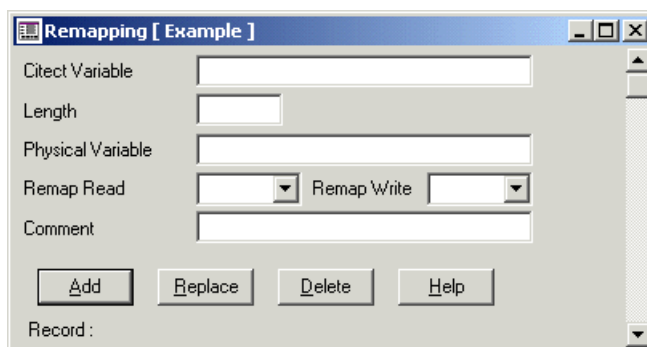


To remap in CitectSCADA, first create the variables in your project as you would normally. Then you can set up the remapping, specifying that any variable with an address in the desired range will be remapped. The I/O server will redirect the addresses at runtime as per the remapping instruction.

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, and 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

Note: Notice that the address M2 is not used. This will not cause any problems.

Remapping database
The remapping is defined as follows.

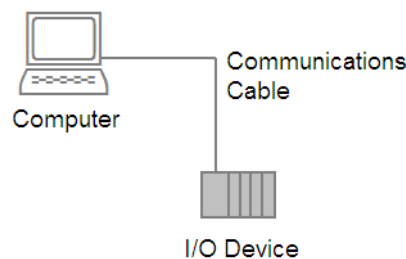
CitectSCADA Variable	Area1 M1
Length	16
Physical Variable	Area1 R10
Remap Read	True
Remap Write	False

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'

- **-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.
- **-sc:** Activates software flow control using XON and XOFF

TCP/IP driver special options reference

- **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_KEEPALIVE 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:

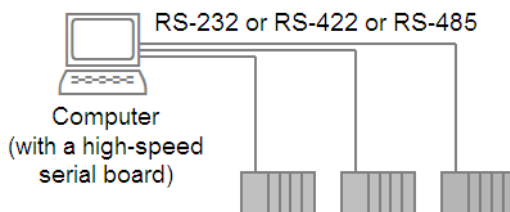
```
COMSetParam((SHORT)ChannelNumber, (UCHAR*)"DO_RECONNECT"),
NULL);
```

- **-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.

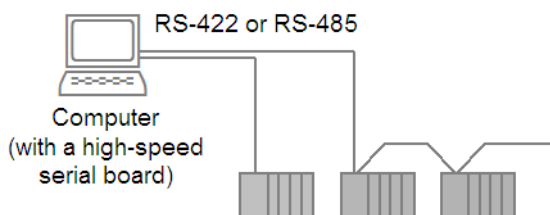
Using a Serial Board

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.

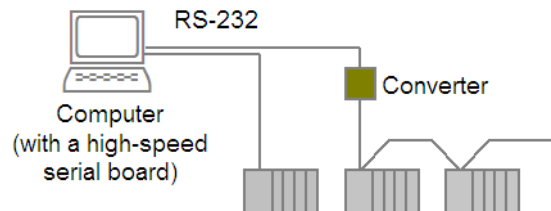


You can use identical I/O devices or I/O devices supplied by different manufacturers; CitectSCADA supports all popular I/O devices. You can connect any number of I/O devices; the only limitation is the size of your computer. High-speed serial boards are available for RS-232, RS-422, or RS-485 communication.

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.



Warning! 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:

Note: If using your computer's COM port, you don't need to install additional software.

- 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 all be left blank.
- 3 Enter the Port Number in the Ports configuration. The COM port number will usually be greater than 2 and is set in the Ports section of the Control Panel. You can use the Special Options 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 [Using Digiboards with Windows](#)
[Using Proprietary Boards](#)

Using Digiboards with Windows

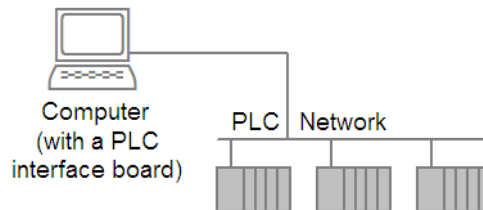
CitectSCADA Version 5 uses the standard Digiboard drivers supplied by Digiboard with Windows. CitectSCADA no longer supplies the drivers for these boards.

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.

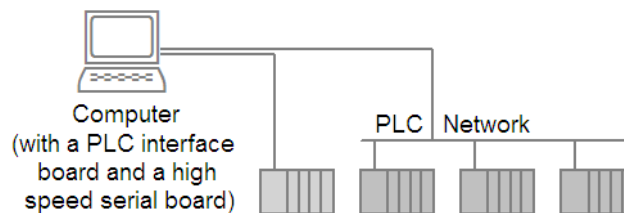
Using Proprietary Boards

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.

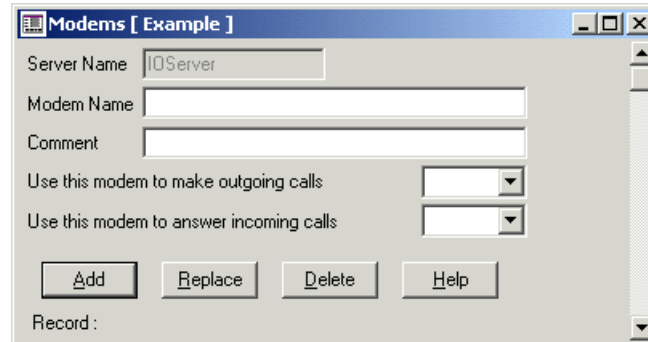
See Also [Proprietary board setup](#)

Proprietary board setup

To set up CitectSCADA to use a proprietary board:

If you are using a proprietary board (i.e., 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 IO device>
- **Number:** <unique network number for the IO 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.

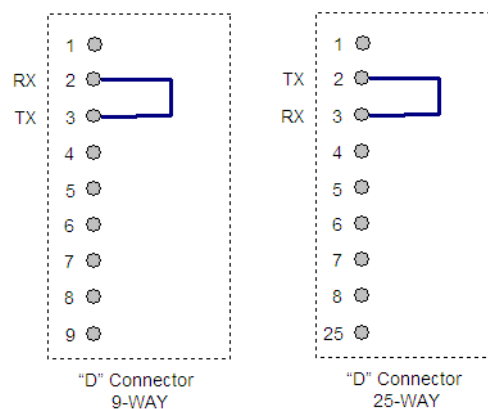
Size = Sets the maximum frame size. The length of each frame transmitted is random between 1 to 'Size'-1. The default size is 512.

- 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.

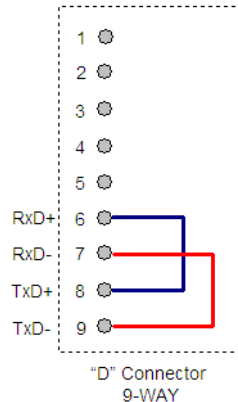
Note: The total number of errors should be 0. If the error is not zero, your serial hardware is faulty.

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.

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 are dependent 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 are dependent 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.

To specify an initialisation string to be sent to the port for modem communication, see [Runtime modem communications](#).

See Also [COMX driver special options reference](#)

Comment

Any useful comment.

I/O Devices Properties

The properties of an I/O device are dependent on the protocol and the I/O device.

I/O devices have the following properties:

Name

A name for your I/O device (PLC). For example PLC_1, OVENS_PLC

Each I/O device must have a unique name 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.

Number

A unique number for the IO Device (0-16383).

Each I/O device must have a unique number 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 name, number and address for each I/O Server.

Address

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.

Protocol

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.

Port Name

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.

Comment

Any useful comment.

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.

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 the Browse button 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 www.citect.com for more information 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, 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.

Startup mode

The type of I/O device redundancy.

Primary	Enable immediate use of this communications channel
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.

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).

Note: Logging all writes to a 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 writes to a 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.

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 shutdown.

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 NT/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 NT/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.

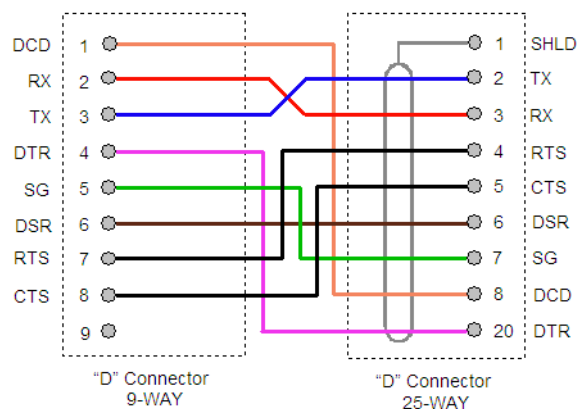
See Also [Communicating with Diallable Remote I/O Devices \(Windows NT/2000 Only\)](#)

Wiring Cables

If using proprietary hardware (i.e., 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 (i.e., 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:

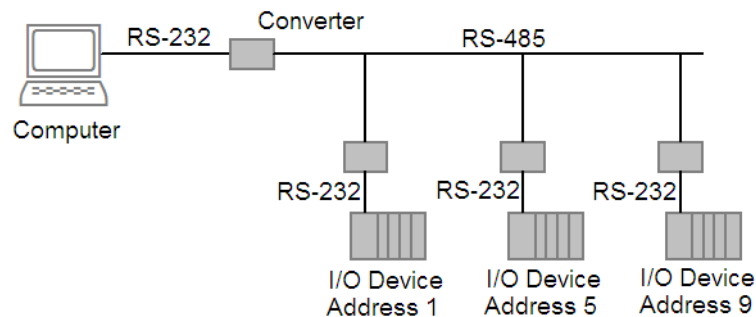
- You want to connect to your computer's com port (ideal for a small system) using RS-232.

- 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. eg 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 (ie 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

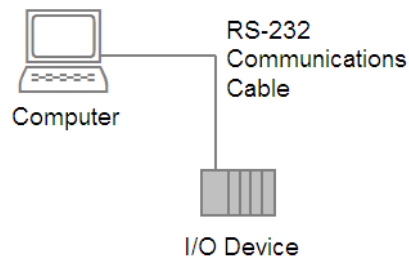
This section describes the following communication standards:

- [RS-232C \(or EIA-232C or RS-232\)](#)
- [RS-422 \(or EIA-422\)](#)
- [RS-485 \(or EIA-485\)](#)

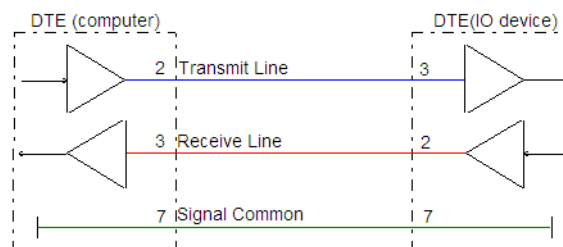
RS-232C (or EIA-232C or RS-232)

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.



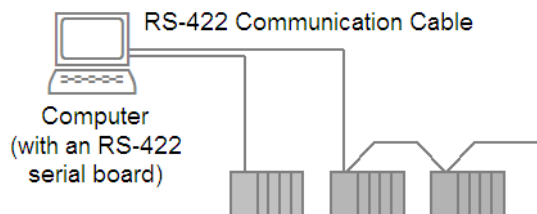
- Less than 75m maximum length at 19.2K maximum Baud rate. Up to 900 meters can be achieved at 900 Baud.

See Also [RS-422 \(or EIA-422\)](#)
[RS-485 \(or EIA-485\)](#)

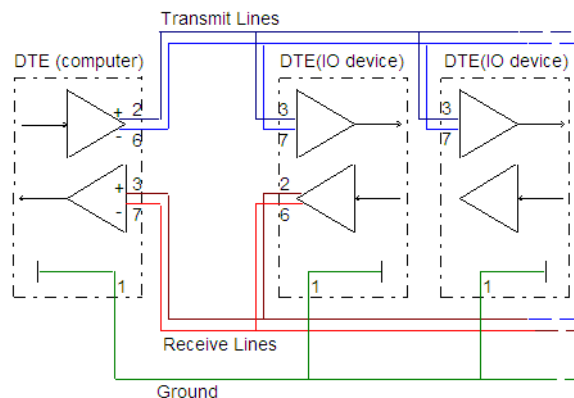
RS-422 (or EIA-422)

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.

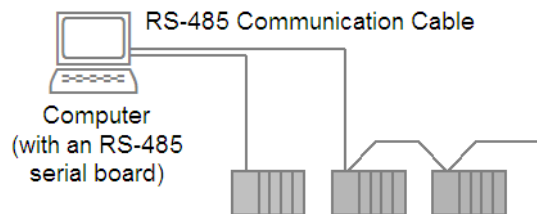
- Distances up to 1200m and transfer rates up to 10Mbps are achievable.
- The protocol used with this standard must take care of who (i.e. which device) is allowed to transmit at any one time. This allows each device to act as a transmitter when requested.

See Also [RS-232C \(or EIA-232C or RS-232\)](#)
[RS-485 \(or EIA-485\)](#)

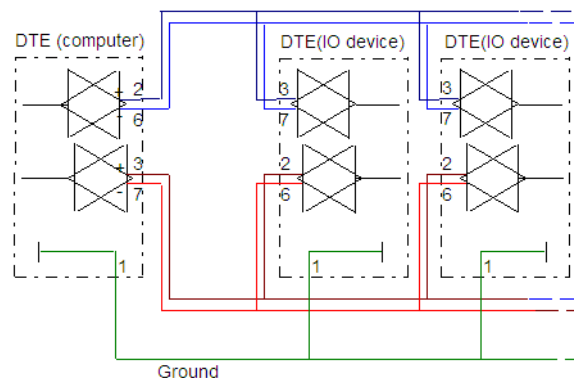
RS-485 (or EIA-485)

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).

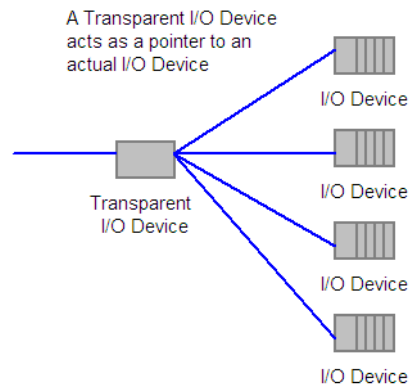
The protocol used with this standard must take care of who (i.e., which device) is allowed to transmit at any one time. This allows each device to act as a transmitter when requested.

See Also [RS-232C \(or EIA-232C or RS-232\)](#)
[RS-422 \(or EIA-422\)](#)

Using a Transparent I/O Device

If you have several identical I/O devices (e.g., controlling identical processes), you can simplify your configuration with a transparent I/O device. A transparent I/O device operates like a pointer to an actual I/O device, and allows you to reassign reads and writes to the actual I/O device.

A transparent I/O device can only be used under special conditions, and you should only use a transparent I/O device if you understand how it works. If your project suits a transparent I/O device, this feature can reduce configuration considerably.



In addition to the I/O device configuration (required to establish communication), you must also set up the transparent I/O device. This I/O device requires no board or port configuration. In the I/O device dialog box, the transparent I/O device must have the same I/O device name as the physical I/O devices referenced by the transparent I/O device, the address may be left blank, and **TRANSPARENT** must be entered into the **Port Name** field.

You can then configure one page, (for example called "TransPg") and define several buttons (on a menu page) that each call the `IODeviceControl()` function followed by `PageDisplay("TransPg")`. For example, to show the data from the physical I/O device No 1 you would use the following command:

```
IODeviceControl(6, 2, 1); PageDisplay("TransPg");
```

and to display data for the physical I/O device No 2,

```
IODeviceControl(6, 2, 2); PageDisplay("TransPg");
```

Alternatively, you can use the following Cicode function:

```
Function
ShowUnit(INT UnitNo)
IODeviceControl(6, 2, UnitNo);
PageDisplay("TransPg");
End
```

The first button calls ShowUnit(1), the second ShowUnit(2) and so on.

Note: In the above example the transparent device is I/O device No 6.

See Also [Configuring transparent devices](#)

Configuring transparent devices

To define a transparent I/O device:

- 1 Configure a new I/O device.
- 2 Enter a unique number for the I/O device.
- 3 Enter **TRANSPARENT** in the **Port Name** field.

To reassign a transparent I/O device to an actual I/O device:

- Use the IIODeviceControl function.

Note the following:

- You can only use this feature if all I/O devices that the transparent I/O device references are identical in every respect.
- Variables read from a transparent I/O device are not compatible with Super Genies or with the advanced DDE features.
- The number of points in a transparent I/O device is counted as follows:

The number of points in the I/O device *multiplied by* the number of physical I/O devices you could reassign to.

For example, if you have 20 points in a transparent I/O device, and 10 physical I/O devices of the same protocol, the point count is $20 \times 10 = 200$ Points: you can read up to 200 physical points from the plant. If you are already reading these points, they are not counted twice.

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

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.

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 is only 200 registers in the device.

GENERIC_CMD_CANCELED (0x0002 | SEVERITY_ERROR) The server cancelled the command while the driver was processing it. This may happen if the driver is taking 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 that is 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, eg. 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) There is a problem with the communication channel 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 standby units to become active. CitectSCADA will attempt to re-initialize the unit.

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. If you are using a NETBIOS driver, increase the number of NETBIOS control blocks.

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.

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.

0 (0x00000000) NO_ERROR No error condition exists.

1 (0x00000001) DRIVER_CHAR_OVERRUN Transmitted characters could not be received fast enough. This error is mapped to Generic Error **GENERIC_INVALID_RESPONSE**.

2 (0x00000002) DRIVER_CHAR_PARITY Parity error in received characters. This error is mapped to Generic Error **GENERIC_INVALID_RESPONSE**.

3 (0x00000003) DRIVER_CHAR_BREAK A break was detected in the receive line. This error is mapped to Generic Error **GENERIC_INVALID_RESPONSE**.

4 (0x00000004) DRIVER_CHAR_FRAMING Framing error. Check the baud rate. This error is mapped to Generic Error **GENERIC_INVALID_RESPONSE**.

5 (0x00000005) DRIVER_MSG_OVERRUN The message received from the device was too long. This error is mapped to Generic Error `GENERIC_INVALID_RESPONSE`.

6 (0x00000006) DRIVER_BAD_CRC The checksum in the received message does not match the calculated value. This error is mapped to Generic Error `GENERIC_INVALID_RESPONSE`.

7 (0x00000007) DRIVER_NO_STX The start of text character is not present. This error is mapped to Generic Error `GENERIC_INVALID_RESPONSE`.

8 (0x00000008) DRIVER_NO_ETX The end of text character is not present. This error is mapped to Generic Error `GENERIC_INVALID_RESPONSE`.

9 (0x00000009) DRIVER_NOT_INIT The driver has not been initialized. This error is mapped to Generic Error `GENERIC_UNIT_OFFLINE`.

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 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**.

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 exist. 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.

Note: Diallable remote I/O device communication (via a modem) is only available on Windows NT/2000 operating systems.

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, etc.
- 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.

Warning! 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`. 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("");

    IODeviceControl(sDevice, 7, 0);

    Sleep(20);

    WHILE bConnected <> 1 AND nRetry > 0 DO
        bConnected = IODeviceInfo(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 (Windows NT/2000 Only)

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 either:

- **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 Citect request; e.g., 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. Citect 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.

[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, check 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 6th 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 (e.g., 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:

Note: 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 (eg. 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.

Note: 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.

Note: 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 | Modem Options (Phone and Modem Options** in Win 2000).
- 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:

Note: Ensure you have set up your modem up 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).

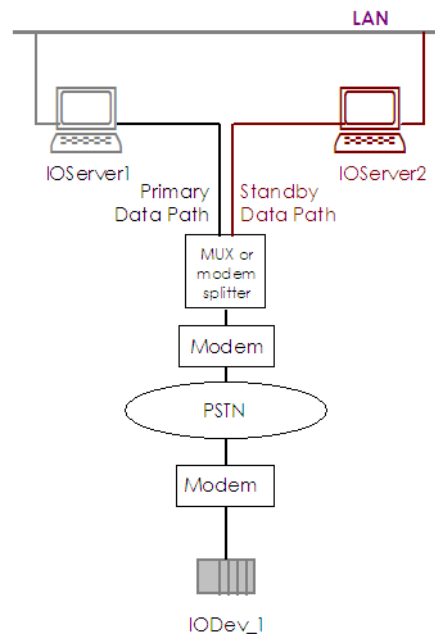
Use this modem to call back I/O Devices

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 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%CO //Compression OFF
AT&D0 //DTR always on
```

If you want to make sure a call from a remote I/O device does not get through while CitectSCADA is shut down (hence losing the data being forwarded), you should set the following parameter to zero:

```
ATS0 = 0 // Auto answer OFF
```

Note, however, that this will also impact any applications that may use the modem other than CitectSCADA, as the modem will not be able to 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%CO //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 \(Windows NT/2000 Only\)](#). This method is available for backward compatibility.

Runtime modem communications

Runtime modem communications are initiated when CitectSCADA starts, and terminated when it shuts down. To communicate using this method, you must enter an initialisation string for the dialling modem (Special Options field on the Ports form). This initialisation string tells the modem which number to dial at startup, etc.

To specify an initialisation string to be sent to the port, use one of the following:

```
-i<STRING>
```

or

```
-i@<filename>
```

where:

- **-i** - Instructs CitectSCADA to send the string to the serial port;

- **<STRING>** - A text string to be sent to the port when the port is initialized (or when all I/O devices on the port fail); and
- **<filename>** - A text file to be sent to the port when the port is initialized (or when all I/O devices on the port fail).

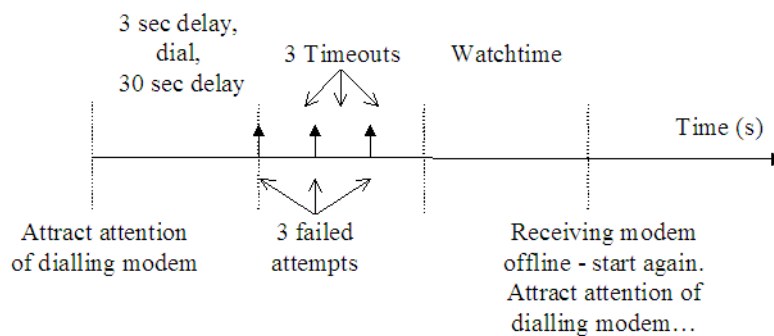
You can use the following special characters in the initialisation string (or file):

- **~** - Delay for 1/2 seconds;
- **~{n}** - Delay for 1/2 seconds x n (Don't forget the {} brackets); and
- **** - The following character is a control character (e.g. \M is <CR> (0xd)). Please refer to your modem's documentation for further information about its control characters.

For example: -i+++~{6}\MATDT123456\M~{60}

- 1 This initialisation string attracts the attention of the dialling modem (using Hayes command +++) , waits 3 seconds, dials the number (123456), and waits 30 seconds before attempting to communicate.
- 2 It then sends a signal, and waits for a response. If a response is received, communications proceed normally. (However, if there is a break in communications at any time, the following steps will be carried out.)
- 3 If no response is received, the dialling modem will wait for the protocol's timeout period (set using the **Timeout** parameter). After this delay, it will attempt to communicate once again.
- 4 The number of attempts depends upon the value of the protocol's **Retry** parameter. For example, if the Retry parameter is set to 3, the modem will make 3 attempts at communication. If all 3 attempts fail, at the end of the third timeout period, the protocol's watchtime will begin (**WatchTime** parameter). At the end of this period, the receiving modem is considered offline, and the whole procedure is repeated (starting at step 1).

This series of steps is illustrated below:



If you use more than one special option, separate each option with a comma.

If you are using serial communication, you may have to enable hardware hand shaking. Refer to the documentation that accompanied your modem for information on modem command strings and to determine if your modem requires hardware hand shaking.

Chapter 24: Using Memory and Disk I/O Devices

Besides supporting actual I/O devices installed in your plant, CitectSCADA supports memory-based and disk-based I/O devices. These I/O devices are "virtual" or pseudo I/O devices—they exist only within your computer. After pseudo I/O devices are configured, they appear exactly as any other I/O device in your CitectSCADA system, but are not connected to any field equipment. Pseudo 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 memory or [disk I/O device](#).

Pseudo I/O devices have several uses :

- When you are configuring a system for the first time, you can configure a pseudo I/O device. You can then design your system, and test it thoroughly without affecting the operation of your plant. When you are satisfied with the design and testing, you can replace your pseudo I/O device in the configuration with the actual I/O device(s).
- You can use pseudo I/O devices together with actual I/O devices - for temporary and permanent (disk) data storage.

See Also [Memory I/O Devices](#)
[Disk I/O devices](#)

Memory I/O Devices

A memory I/O device is created in the memory of your computer when you start your runtime system. The value of each variable in the memory I/O device is stored in your computer's memory.

Memory I/O devices can contain any type of variable supported by CitectSCADA. However, because the memory I/O device is created each time your runtime system starts, these variables are also created at run time; they do not retain their values when you shut down your system.

When a temporary variable is created, it is set to a default value. The default value for numeric and digital variables is 0 (zero) and for strings is "" (empty string). If your system requires initial values other than these defaults, you must set them when you start your system.

Note: Because a memory I/O device is local to an individual computer, you cannot use a memory I/O device across a CitectSCADA network. When you write data to a memory I/O device, the data is stored in the memory of the computer where the memory I/O device is configured. (For example, setting a bit in a memory I/O device only sets that bit on the local computer - it is not set on any other CitectSCADA computers on the network.) If you want to share pseudo I/O device data on a network, use a disk I/O device.

See Also [Memory I/O Device Setup](#)

Memory I/O Device Setup

To set up communications with a device, follow the steps given in the I/O device setup procedure. Sometimes, however, you may need to edit the communications forms directly. They require the following specific information.

When setting up a memory I/O device:

- You do not have to complete a Boards dialog box.
- You do not need to complete a Ports dialog box.
- You must complete the I/O Devices dialog box as follows.

I/O Device name

A name for your memory I/O device, for example: MEMORY_PLC

Note: Each I/O device must have a unique name in the CitectSCADA system.

I/O device number

A unique number for the memory I/O device (0-1023)

Note: Each I/O device must have a unique number in the CitectSCADA system.

I/O device address

Leave this property blank.

I/O device protocol

To use the CitectSCADA [generic protocol](#), enter: GENERIC

- or -

To select a specific protocol supported by Citect, see the I/O devices online Help.

I/O device port name

You must use: MEMORY

I/O device comment

Any useful comment.

I/O device startup mode, log write, log read, cache and cache time

Leave these properties blank.

Disk I/O devices

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.

Use a disk I/O device 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, the disk I/O device has the same performance as a memory 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](#)

Disk I/O device setup

To set up communications with a device, you should follow the basic steps given in the I/O device setup procedure.

Sometimes, however, you may 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.

I/O device name

A name for your Disk I/O device, for example: `DISK_PLC`.

Note: 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)

Note: Each I/O device must have a unique number in the CitectSCADA system.

I/O device address

The path and filename of the disk file, for example:

`[RUN]:DSKDRV.DSK`

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:

`Primary_File, Standby_File`

If this is the Standby Disk I/O device enter:

`Standby_File, Primary File`

Primary_File is the name (and path) of the primary disk I/O device file. You may use path substitution in this part of the field.

Standby_File is the name (and path) of the Standby Disk I/O device file. You may use path substitution in this part of the field.

Note the following:

- If the specified disk I/O device file is not found, CitectSCADA will create a new (empty) file with the specified filename.
- To use redundant disk I/O devices, you must be using 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 `[DiskDrv]UpDateTime` parameter.

I/O device protocol

To use the CitectSCADA generic protocol, enter: **GENERIC**

- or -

To select a specific protocol supported by CitectSCADA, see the I/O devices online Help.

I/O device port name

You must use: **DISKDRV**

I/O device comment

Any useful comment.

I/O device startup mode

If you are not using redundant disk I/O devices, leave this property blank.

If you are using redundant disk I/O devices, use either:

- **Primary:** Enable immediate use of this Disk I/O device
- **StandbyWrite:** 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.

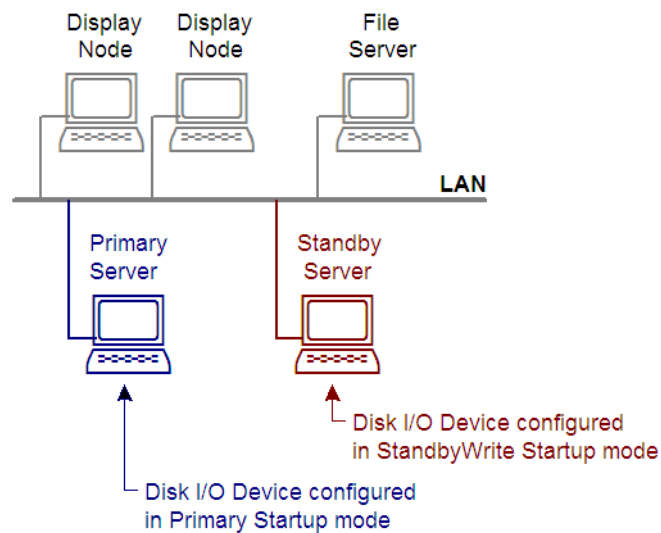
Note: To use StandbyWrite mode, you must also configure an I/O Disk Device in the primary server. Both I/O devices must have the same I/O device name and number.

I/O Device Log Write, Log Read, Cache and Cache Time

Leave these properties blank.

Redundant Disk I/O Devices

If you are using a network, you can configure a redundant disk I/O device to eliminate data loss (in the event of a server failure). The following diagram illustrates the use of redundant Disk I/O devices:



When the system is in operation, 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:

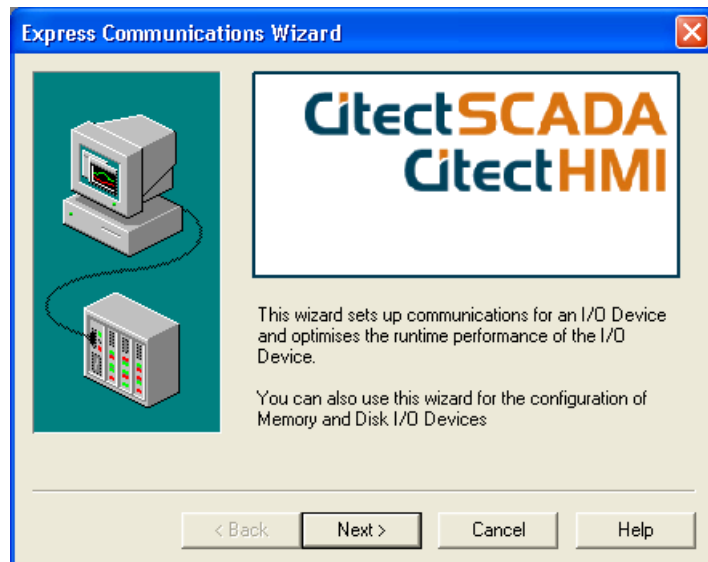
- 1 Configure a new disk I/O device
- 2 Select **StandbyWrite** for the Startup Mode.

Note: 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.

Chapter 25: 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.



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 **IOServerX**. 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

Select to modify one of your existing I/O devices as defined in the current Project, or choose to create a new I/O device.

Only I/O devices associated with the I/O server selected on the previous page are available for editing. To edit I/O devices that are associated with another I/O server, click **Back** to select the I/O server again.

See Also [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**, a **Memory I/O Device**, or a **Disk I/O Device**.

You must also specify the name of the I/O device. CitectSCADA will automatically suggest the name **IODevX** (that you can change if desired).

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 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.

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 \(Windows NT only\)](#)

[Express Communications Wizard - Link to external database](#)

Caller ID and commands (Windows NT only)

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 connection to the I/O device has been established (and before any read or write requests are processed).

Express Communications Wizard - Link to external database

See Also

[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.

[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 MxChange** 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 MxChange Servers found on the network connected to the computer.

Selecting any database type other than Mitsubishi MxChange 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 MxChange** 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 FastLinux products. See www.citect.com for more information 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, 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

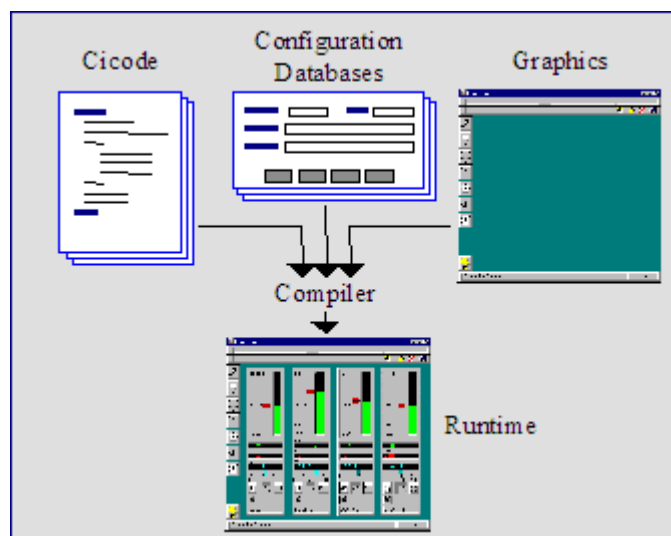
Provided is a summary of the setup of the I/O device. Based on the information you have provided, the summary includes details regarding the CitectSCADA communications setup and the recommended configuration of your I/O device and/or internal boards.

Chapter 26: Building Your Citect Project

See Also [Compiling the Project](#)
[Running the System](#)
[Running Your System Over the Internet](#)
[Providing Runtime Security](#)
[Using an Alternative INI File](#)
[Debugging the Runtime System](#)
[Debugging I/O Devices and Protocols](#)
[Using the CitectSCADA Kernel](#)

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.

Note: 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**.

Note: If there are any compile errors, you must first fix the errors, and then re-compile. CitectSCADA will automatically compile the project (if it is uncompiled) when you try to run it.

See Also [Incremental compilation](#)
[Debugging the compilation](#)
[Compile Error Properties](#)
[Compile Error Messages](#)

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 may 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 further information on an error:

- 1 Click **Help** at the bottom of the Compile Errors dialog box.
- 2 Read through 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](#)

Compile Error Properties

Compiler Errors have the following properties:

Type

The type of error. Three types of errors can occur during compilation. These are:

Type	Meaning
FATAL	The severity of this error is such that it halts the compilation process. The project cannot be compiled until you correct the error.
ERROR	The compilation process continues, however the project will not compile successfully until you have corrected 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

No I/O Devices defined No I/O devices are defined in the project.

Include project not found An included project (specified in the Included Projects database) does not exist. Check the name of the included project.

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 may be incorrect. This can occur if you have edited the variables database using Excel or some other database editor. To re-index the database select the Pack command from the File menu (in the Project Editor).

Database is empty The database does not contain any records.

Too many records in database Too many records have been specified in the database. This error should only occur if the CITECT.FRM file has been changed or become corrupt. Contact Citect Support.

Too many fields in database Too many fields have been specified in the database. This error should only occur if the CITECT.FRM file has been changed or become corrupt. Contact Citect Support.

Bad integer value An integer value cannot be found where one is expected, or the integer value is out of range.

Bad floating point value A floating-point number cannot be found where one is expected, or the floating-point value is out of range.

Tag expected A tag name was not found where one was expected, or an [expression](#) 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.

Unknown I/O Device The I/O device (unit) does not exist in the project. Check that the I/O device name is correct.

Close bracket expected The Cicode statement has a different number of open and close brackets. Another close bracket ')' or ']' is expected in the statement.

Symbol search failed A database record does not exist. Check that the [record name](#) is correct.

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.

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. (A list of appropriate address formats for a particular I/O device can be found in the Help under the "Data Types" topic for each supported I/O device.)

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.

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.

String expected Only strings can be used in database fields.

Protocol expected The protocol field in the I/O devices database is blank. You must select a protocol for the I/O device.

Open bracket expected You must use parentheses () in Cicode functions, even when they have no parameters, e.g. MyFunction().

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.

Incorrect number of arguments for function Too many or too few arguments have been passed to a Cicode function.

Trailing characters in Cicode There are extra trailing characters in a Cicode statement, following the semi-colon.

Incompatible types There is a mismatch of data types in a statement. For example, a string is specified where a number is expected.

Too many arguments Too many arguments are specified in a Cicode function. The maximum number of arguments allowed is 32.

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 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.

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.

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.

FUNCTION expected A function must be declared with the FUNCTION keyword.

THEN expected A THEN statement must be used in an IF statement.

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.

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 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.

Must return value from function If a Cicode function is declared to return a value, it must have a RETURN statement.

Label is defined twice Label names must be unique. Check the Labels form for duplicated names.

Cannot use RETURN outside of functions A RETURN statement can only be used within a function.

Statement expected CitectSCADA is expecting a statement. Check the Cicode for syntax errors.

Operand expected A Cicode operator must be followed by an operand.

Invalid time format 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.

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.

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.

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.

Bad analog format The format is incorrectly specified for an analog variable. Check the Format field in the Variable Tags form.

Maximum report size exceeded The report file size must be less than 63K. Reduce the size of the report or configure two reports.

Bad factor specification A Cicode expression that contains an invalid expression has been used. Check the syntax of the expression.

Semicolon expected All Cicode statements must be separated with semi-colons (;).

Page Name cannot start with underscore A Page Name must start with an alphanumeric character (A - Z, a - z, or 0 - 9).

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.

Cicode data limit reached An array in a Cicode module cannot exceed 60 KB. Reduce the size of the array.

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.

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>.

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.

Trailing characters in Name The database record name contains invalid characters. Remove any invalid characters from the record name.

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.

String too big The string size has been exceeded. The maximum size of the string must not exceed 255 characters.

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.

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 too big The label is too big. The size of a label must not exceed 8 KB.

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.

Out of file handles 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.

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.

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:

CONFIG.SYS

FILES=120

NET.CFG or SHELL.CFG

file handles=120

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 write to file The file cannot accept a write operation. The file has become corrupt or the disk is full.

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.

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.

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.

Out of memory CitectSCADA has run out of memory. Increase the amount of memory in the computer or use smaller databases.

Software error An internal CitectSCADA software error has been detected. Contact Citect Support.

Not database format The database has become corrupt or the file format is unknown. Pack the database.

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.

Unknown DBA error An internal CitectSCADA software error was detected. 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.

Disk full The disk is full. Remove unwanted files from the disk, or replace the existing disk with a larger disk.

File locked A file is in use by another network user.

Database not found The main project database cannot be found.

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.

Index key has changed The database has a corrupted index. Pack the database.

File not indexed The database must be indexed, but the index file associated with the database cannot be found. Pack the database.

Database table full The database is full. If the error persists, contact Citect Support.

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.

File does not exist The file cannot be found. Check that the file name is correct.

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.

File already opened in SINGLE mode The file has been opened by another user. Set the [General] ShareFiles parameter to 1 in the [citect.ini file](#) to open files in shared mode.

Too many Include projects More than 240 Include projects have been defined.

Unknown protocol The protocol does not exist.

Cannot compile all functions Error(s) were detected while compiling the function library.

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

Note: This error is often due to Cicode functions being defined in a number of included projects. Extending this parameter may affect system performance. It should only be set when advised by Citect Customer Service.

Point limit reached The maximum number of points that can be referenced has been reached. The maximum [point limit](#) is determined by your CitectSCADA license. Contact Citect Support.

Bad point limit The incorrect point limit is specified in the CITECT.INI file. The point limit must correspond to your CitectSCADA license.

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.

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.

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.

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.

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.

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.

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.

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 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**.

Note: CitectSCADA automatically compiles the project (if uncompiled) when you try to run it.

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.

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 CitectHMI/SCADA; 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.

Note:

- 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](#).

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.)

Note: 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.

Note: The Citect.INI file is stored in **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 would like 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.

Note: The Internet directory on the Internet Server will only be 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 will have to 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.

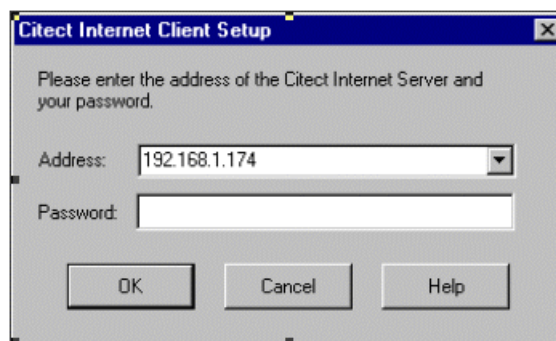
- 4 Go to this temporary folder and double-click **IDC.exe**.
- 5 Follow the prompts to complete the installation.

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 Citect 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.

Providing Runtime Security

The CitectSCADA runtime system is a Windows-based application that runs in the standard Windows 95 or NT environment. The Windows environment allows you to run several applications at the same time. There are several different ways in which anyone accessing the PC can be prevented from accessing software other than CitectSCADA.

See Also [Running CitectSCADA as a service under NT](#)
[Running CitectSCADA as the shell under NT](#)
[Disabling Windows keyboard commands](#)
[Disabling control menu commands](#)
[Removing the Cancel button](#)

Running CitectSCADA as a service under NT

To prevent access to CitectSCADA or the operating system while CitectSCADA is running, you can run it as a service. Once it is configured this way, CitectSCADA will start when NT starts and it is not 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. You should consult the Citect [knowledge base](#) for the latest information on running CitectSCADA as a service.

See Also [Running CitectSCADA as the shell under NT](#)

Running CitectSCADA as the shell under NT

To prevent users from switching from CitectSCADA back to the Program Manager or Explorer in Windows NT you can run CitectSCADA as the Shell. Normally Windows NT runs either the Program Manager or Explorer as the Shell (depending on your version of NT). You can change this so that another program is run as the Shell when NT starts. To do this you must use the Regedt32.exe to edit the registry. You should consult the Citect Knowledge Base for the latest information on running CitectSCADA as a shell.

See Also [Disabling Windows keyboard commands](#)

Disabling Windows keyboard commands

The Windows 95 and NT 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 may 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.

Using an Alternative INI File

When CitectSCADA starts up, it reads default values from the Citect.ini. (By default, CitectSCADA looks for the ini file in the Citect\Bin directory. If it can't find it there, it will search 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 the Citect Explorer or Runtime.
- 2 Right-click to bring up 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 initialisation file MY.INI, enter the following line:

```
c:\citect\bin\citect.exe /ic:\citect\user\myproj\MY.INI
```

Note: The *Computer Setup Wizard* will use the same INI as specified for the Citect Explorer. You can specify different INIs for both the Runtime and Explorer

programs. However, if you initiate Runtime from the Explorer, it will use the INI 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, which is available as a standard template.

The hardware alarm page is your primary indicator of 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 the SysLog 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](#)

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 may require a bit of practice as the startup procedure may 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
.
.

```

Note: There may be more parameters added later, so always check the COMx information in the Online Help and the Citect Knowledge Base.

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 may 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. Parameters for TCPIP:

[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

- 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 then 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 TCPIP Debug window, 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, 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

- 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, you will never get communications working.

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 may 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

These are drivers such as the Allen Bradley KTX card, Modicon SA85 card, Siemens TIWAY card etc, which have their own low-level driver. Each of these drivers will have some debugging parameters that will make it easier for you to debug problems. Check the Citect Knowledge Base and the Online Help for any possible parameters. The CitectSCADA Knowledge Base will likely have articles describing the methods for debugging each of these Board drivers.

The debugging process is exactly the same as with Serial:

- 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 then up in the manufacturers documentation.

Contacting Citect support

If you can't debug the problems on your own then please contact Citect support. You will need 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 the configuration of the project and examine the results in the CitectSCADA runtime system - with a single button, and without having to shutdown CitectSCADA. You can update your system while CitectSCADA is already running.

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). The changeover is instantaneous.
- If you change any of the databases (e.g. you add a new Alarm Tag, Trend Tag, or Cicode function), CitectSCADA does a **full restart** to run the updated project.

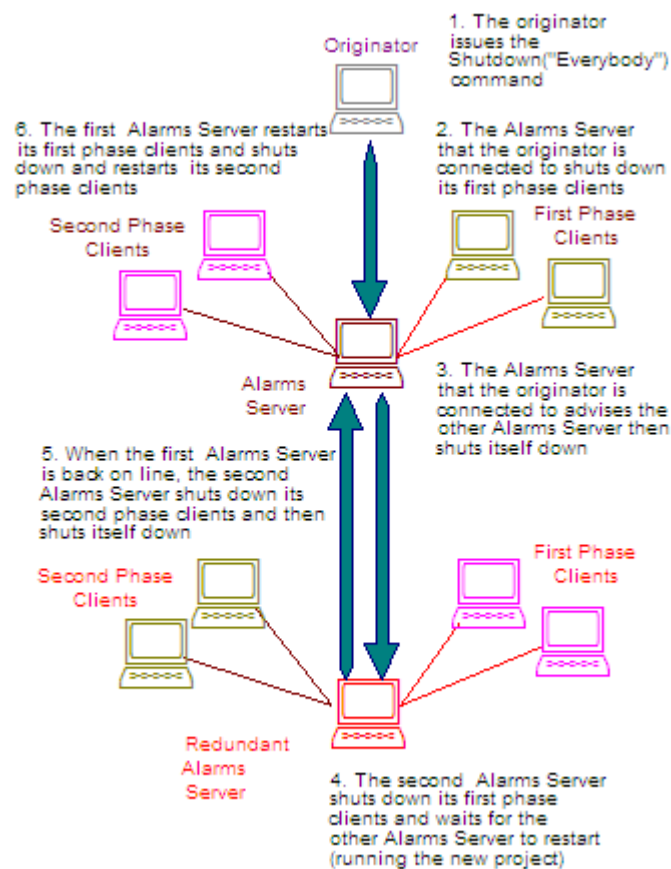
Warning! Do not use this feature if you are making major changes to the project.

See Also [Restarting a networked system online](#)

Restarting a networked system online

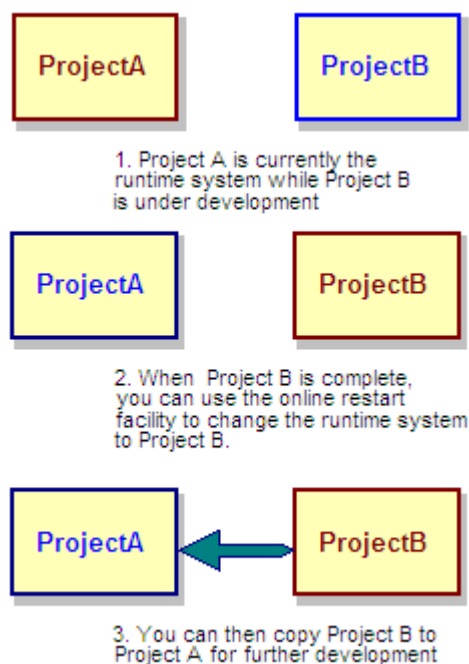
If you are using CitectSCADA on a network, you can use a structured restart procedure that ensures control of the plant is maintained and no data is lost during changeover. You can use any CitectSCADA computer on the network to initiate the online restart.

CitectSCADA automatically manages the online restart in the following sequence:



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:



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 (e.g. 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.

Using a Callback function

You can use a [callback function](#) (with the OnEvent function) to perform any 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 the parallel port of your computer. The hardware key contains details of your user license, such as type and I/O [point limit](#).

See Also [Updating your hardware key](#)
[CiUSAFE dialog properties](#)
[Citect license point count](#)
[Demo mode](#)

Updating your hardware key

When you upgrade to a new version of CitectSCADA, you may need to update your hardware key to enable the system to run. See the CitectSCADA Readme file to confirm whether you need to perform an update.

The hardware key plugs into a parallel printer port on your computer and contains information about your user license including the point limit.

Updating the hardware key involves running the **CitectSCADA Key Update**, which is found in the Help menu of Citect Explorer.

Note: If you have CitectSCADA v5.21 or 5.20, you will need to run CiUSAFE.exe from the Citect BIN directory. You can also download the latest version of the upgrade program from the **Key Upgrade** section of the Citect website.

Each time you launch the CitectSCADA Key Update, the program displays a Key ID. The serial number of the Hardware Key will also be displayed if it has been written to the key. If not, read the number from the printed label on the Hardware Key. To perform the update, visit the Citect web site and enter the serial number. Provided that your Customer Service agreement and license details are valid, an Authorisation Code will display, which you enter in the CiUSAFE dialog.

To update the hardware key:

- 1 In Citect Explorer choose **Help | Citect Key Update**. If you have CitectSCADA 5.21 or 5.20, run CiUSAFE.exe from the Citect BIN directory.
- 2 A Key ID will display. The Hardware Key's serial number may also display. If it does not, read the serial number from the label on the key.
- 3 Visit <http://www.citect.com/> and enter the serial number as prompted. You may also be requested to provide the Key ID and your web login name and password.
- 4 The authorization code appears. Type the code (or copy and paste it from the web site) into the AUTHORISATION CODE field in CiUSAFE. Do not use any spaces when entering the characters.
- 5 Click **Update**.

- 6 The Return Code field will indicate whether the hardware key was updated successfully.

Note: Each time you run the CitectSCADA Key Update, a different Key ID will appear. However, if you obtain an Authorisation Code, but do not immediately update the Hardware Key, you can enter the same Authorisation Code the next time you run the update.

See Also [CiUSAFE dialog properties](#)

CiUSAFE dialog properties

The CiUSAFE dialog box has the following properties which allow you to update your Hardware Key.

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 will need to enter the serial number at the Citect web site to perform the key update.

KeyID

Each time you launch CiUSAFE, a Key ID will display in the KEYID field. You may need to provide the Key ID in addition to the serial number when updating the Hardware Key. This depends on the status of the key in the CitectSCADA license database, and you will be 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 authorisation code into this field. You will be prompted with this code once you have entered the Key ID and serial number, and your license and Customer Service agreement have been verified. Clicking **Update** then updates 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**.

See Also [Citect license point count](#)

Citect license point count

The point limit is the maximum number of I/O device addresses that can be read, and is specified by your CitectSCADA license. CitectSCADA counts static and dynamic points.

Static points are points that are known when the project is configured. This includes all tags used by alarms, trends, reports, events, and pages.

Dynamic points are points that are used dynamically at runtime. Although they exist in the Variables database, dynamic points are not used when the project is configured. They are used when you call a Super Genie, use the TagRead() and TagWrite() Cicode functions, or read or write them using DDE, ODBC, or the CTAPI.

Note the following:

- Dynamic and Static points are counted only once, regardless of how many times they are used.
- At runtime, the static and dynamic point counts are available through the Kernel and the CitectInfo() Cicode function.
- It is important to plan your system and keep aware of your point count so that you do not exceed your point limit. This is particularly important at runtime when you can unexpectedly add to your point count by using tags that have not yet been included in the tally.

When you run CitectSCADA, the static point count is checked against your Hardware Key. If the point count is too large, CitectSCADA will not start up. At runtime, the dynamic point count is added to the static point count. CitectSCADA will not allow you to use a new dynamic point if (at runtime) it pushes the total point count above the point limit - any new references to tags through Super Genies, DDE, ODBC, or the CTAPI will fail.

See Also [Demo mode](#)

Demo mode

CitectSCADA can be run without the hardware key in demonstration (Demo) mode. Demonstration mode lets you use all CitectSCADA features normally, but with restricted run-time and I/O. 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 will start 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 will only be able to write 1 point. If you want 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.

Note: 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 will give 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.

Note the following:

- 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: once you are in the Kernel, you can execute any Cicode function with no privilege restrictions. Anyone using the Kernel has total control of CitectSCADA (and subsequently your plant and equipment).

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

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 tick 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 ticked on the Security Setup - Keyboard page).

Note: You should deselect 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: You should 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	DspKernel(0);
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 at any time by selecting the Close option from the Control menu of the main Kernel window.

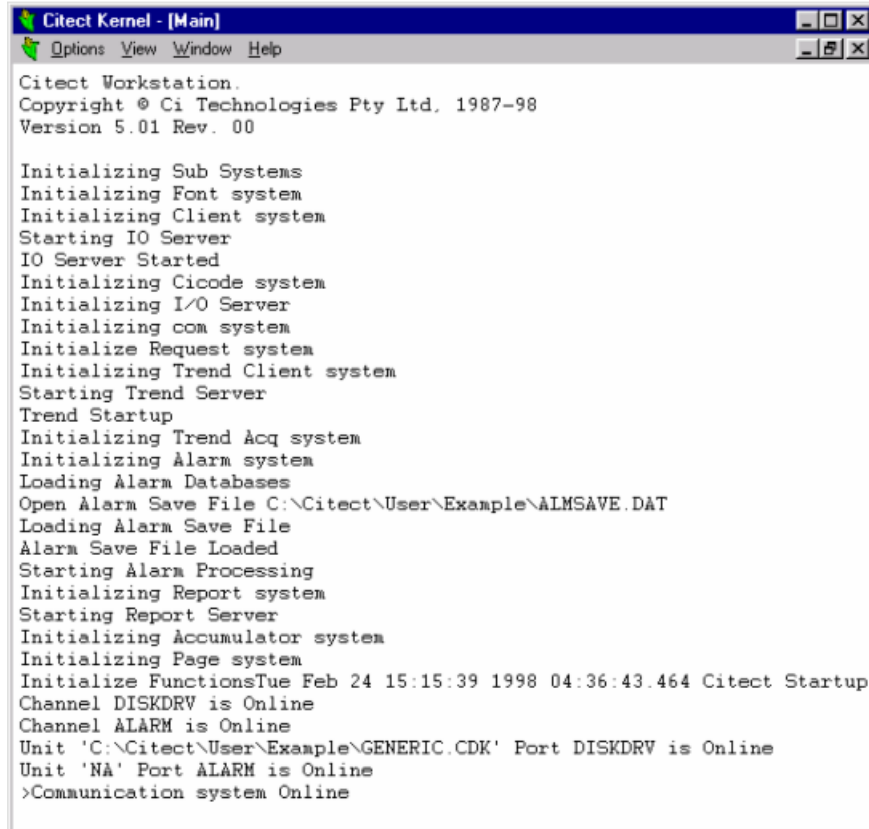
See Also [Inside the kernel](#)

Inside the kernel

When displayed, the CitectSCADA Kernel consists of an application window and one or more child windows. The first time the Kernel is invoked, one child window (called Main) is opened. The Main window contains a command line interface (similar to the Command prompt) where you can type in Kernel commands that perform a Kernel operation or display other child windows.

The Main window displays a line by line description of what CitectSCADA did when it started up. However, CitectSCADA continues to report messages to the

Main Kernel window while it is running, providing a useful history of important events. A typical startup screen will look like this:



```

Citect Kernel - [Main]
Options View Window Help
Citect Workstation.
Copyright © Ci Technologies Pty Ltd. 1987-98
Version 5.01 Rev. 00

Initializing Sub Systems
Initializing Font system
Initializing Client system
Starting IO Server
IO Server Started
Initializing Cicode system
Initializing I/O Server
Initializing com system
Initialize Request system
Initializing Trend Client system
Starting Trend Server
Trend Startup
Initializing Trend Acq system
Initializing Alarm system
Loading Alarm Databases
Open Alarm Save File C:\Citect\User\Example\ALMSAVE.DAT
Loading Alarm Save File
Alarm Save File Loaded
Starting Alarm Processing
Initializing Report system
Starting Report Server
Initializing Accumulator system
Initializing Page system
Initialize FunctionsTue Feb 24 15:15:39 1998 04:36:43.464 Citect Startup
Channel DISKDRV is Online
Channel ALARM is Online
Unit 'C:\Citect\User\Example\GENERIC.CDK' Port DISKDRV is Online
Unit 'NA' Port ALARM is Online
>Communication system Online
  
```

Note: When CitectSCADA starts up, double-click the Main Kernel window title bar, then double-click the Kernel title bar.

- **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.**
- **Adding NetBIOS name**
Adding NetBIOS name - You will only see these on a networked system. This indicates that CitectSCADA has registered NetBIOS names on a protocol stack. This should appear twice, as CitectSCADA has two NetBIOS capable protocol stacks.

- **Starting IO Server** - You will only see these messages if the CitectSCADA computer is an I/O server. If the computer is an I/O 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. **IO Server Started** - The server has started and is functioning correctly. It is unlikely that this will ever fail.
 - **Initializing I/O Server** - Starting to check what is needed to make the I/O server 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 of the 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 - This means that 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 - You will only see these messages 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 start up 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 may cause startup errors are:

- Incorrect computer setup - usually solved by the Computer Setup Wizard.
- Networking faults or bad hardware.
- Communication faults - this is usually just a configuration issue.

The Main window is particularly useful to check that all of your I/O devices come online correctly when starting. First the ports must be initialized OK, then the I/O device itself will come online. If there is a problem, CitectSCADA will display a message; "PLC not responding", "I/O Device Offline" or similar.

Some I/O devices may take two attempts to come online. If this is the case, CitectSCADA will wait (usually 30 seconds) and try again. If your I/O device does not come online after the second attempt, you should check your configuration (at both ends) and cabling.

Note: The Kernel continues to report changes in the status of the I/O devices to the Main window. This information may 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
Cache	Changes the cache timeout for each I/O device.
Cicode	Opens a child window that you can use to call Cicode functions.
Cls	Clears all text from the Main or Cicode windows.
Debug	Enables the debugging of raw data transferred between CitectSCADA and a driver
Exit	Closes a Cicode or Shell window.
Help	Displays a list of some of the commands available in the Kernel.
INI	Displays the local Citect.INI file
Log	Enables or disables the logging of I/O device reads and writes.
NetBIOS	NetBIOS logging.
Page General	Displays general statistics information.
Page Driver	Displays information about each driver in the CitectSCADA system.
Page Memory	Displays the memory debug heap.
Page Netstat	Displays NetBIOS specific statistics and information.
Page Table	Displays information about CitectSCADA's internal data structures.
Page RDB	Displays information about CitectSCADA's Runtime Databases.
Page Unit	Displays information about each I/O device in the CitectSCADA system.
Pause	Pause debug output.
Probe	Displays read and write requests (and responses)
Shell	Opens a new command (shell) window.
Stats	Resets all system statistics.
SysLog	Displays the SysLog.DAT file.

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's 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 may require that you adjust your system to gain 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 may become idle for short periods of time and so waste its bandwidth. Also if the cache is too large, a CitectSCADA client may 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, eg display it on the 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. Use The following method:

Note: This information 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 the 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 may not see any improvement in performance with a single client, as caching will only improve performance with multi clients. You may see improvements if you are also running trends, alarms or reports which are requesting the same data.
- 8 You should then add another CitectSCADA client which 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 27: Using CitectSCADA on a Network

For large applications, you can add a LAN to the CitectSCADA system, or use an existing LAN supported by CitectSCADA.

You can use NetBEUI, IPX/SPX, TCP/IP, and other network protocols with CitectSCADA. CitectSCADA also supports all protocols compatible with NetBIOS, for example, Novell Netware, LAN Manager (with Windows 3.1, Windows for Workgroups, and Windows NT), and various other network operating systems. CitectSCADA can use multiple protocols at the same time.

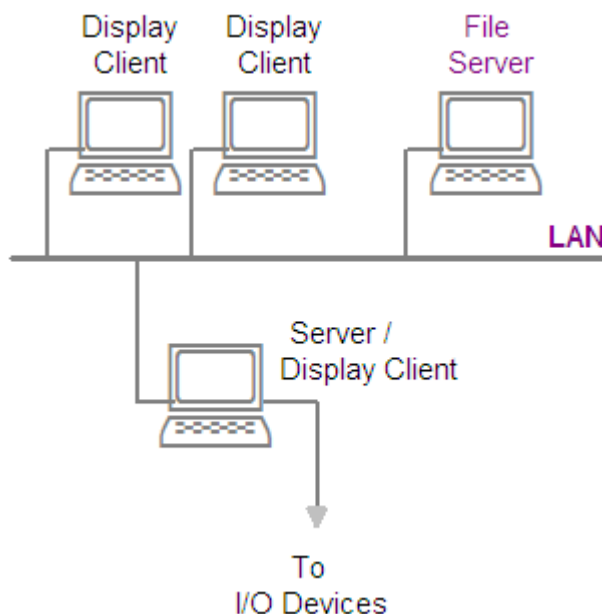
CitectSCADA supports scalable architecture, which lets you initially implement CitectSCADA on a single computer, or over a small network, and then expand the system later without having to change your existing hardware, software, or system configuration.

Using CitectSCADA on a LAN adds more flexibility to the system, and coordination within large plants can be more easily achieved. You can control and monitor autonomous areas within the plant separately, and interrogate the whole plant using any CitectSCADA computer on the network if you want.

A LAN can also be used to:

- Distribute the processing load for large systems. See [Using Distributed Processing](#).
- Provide redundancy. See [Building Redundancy into Your System](#).

The following illustration provides an example of three CitectSCADA Display Clients (with 1 also performing as a CitectSCADA [I/O server](#)) connected to a common existing LAN (which has a non-CitectSCADA [file server](#)).



Each runtime CitectSCADA machine is a [display client](#) of the CitectSCADA system. You can distribute Display Clients throughout your plant, to control and monitor individual regions (or areas), and throughout your offices, to provide high-level information to the appropriate personnel in your organisation. You can connect as many as 256 Display Clients to a CitectSCADA system.

Each CitectSCADA Display Client PC connected to an I/O device must also be set up as a CitectSCADA I/O server. One or more CitectSCADA Display Clients can also be configured to be a [CitectSCADA server](#), used to process alarms, reports, or trends.

See Also [Setting up a Network](#)
[Using Distributed Processing](#)
[Using Distributed Servers](#)
[Building Redundancy into Your System](#)

Setting up a Network

To set-up a [local area network \(LAN\)](#) for CitectSCADA, you must have successfully installed all (non-CitectSCADA) network hardware and software in strict accordance with the instructions provided by the manufacturer as

appropriate, and you should also be quite familiar with the basic operation of the network.

You must install the CitectSCADA software on every PC machine you want to use as a CitectSCADA design-time development machine, runtime CitectSCADA Display Client, CitectSCADA I/O server, and CitectSCADA Alarm, Report, or trend server.

You must also set-up CitectSCADA for your network, using the Computer Setup Wizard on each and every one of the afore-mentioned machines.

You can configure your CitectSCADA system for use with Wide Area Networks (WANs). For details, see [Configuring CitectSCADA to communicate over a WAN](#).

To start the Citect Computer Setup Wizard:

- 1 Open Citect Explorer.
- 2 In the project list area, select **My Projects**.
- 3 Double-click the **Computer Setup Wizard** icon, or choose **Tools | Computer Setup**.

To set up CitectSCADA for your network:

- 1 Run the Computer Setup Wizard.
- 2 Select **Custom Setup**, and click **Next**.
- 3 On the 'Computer Role Setup' page, select the appropriate 'Network Computer' option.
- 4 Follow the prompts given by the wizard. If the computer is to be used as a server, select the appropriate server type (Alarms, Reports, Trends) when the page for that server type displays.
- 5 On the 'Alarms, Reports, and Trends Server Setup' page, enter an appropriate name for this server on the network. This is the name that other machines on the network will be configured to connect with.
- 6 On the 'Network Setup' page, enter the Network Computer Name for this machine. (The Network computer name can be viewed in the 'Network Properties' tab of the Windows System Properties dialog. Right-click the MyComputer Icon and select **Properties**.)
- 7 Continue to the end of the wizard and press the **Finish** button, to save the settings.

To set-up a CitectSCADA display client:

- 1 Run the Computer Setup Wizard.
- 2 Select **Express Setup** (you can run in **Custom** mode if desired), and click **Next**.

- 3 On the 'Computer Role Setup' page, select the appropriate 'Display Client' option.
- 4 Follow the prompts, continue to the end of the wizard, and press the **Finish** button, to save the settings.

To set up an I/O server:

- 1 Run the Computer Setup Wizard.
- 2 Select **Express Setup** (you can run in **Custom** mode if desired), and click **Next**.
- 3 On the 'Computer Role Setup' page, select the appropriate 'Server' option, and click **Next**.
- 4 On the I/O Server Setup page, select **This computer is an I/O Server**.
- 5 Select an appropriate I/O server, and click **Next**.
- 6 Follow the prompts given by the wizard. If the computer is to be used as an Alarm, Report, or Trend server, select the appropriate server type when the page for that server type displays.
- 7 If the server is to be a network server, the 'Alarms, Reports, and Trends Server Setup' page will display. Enter an appropriate name for this server on the network. This is the name that other machines on the network will be configured to connect with.
- 8 Continue to the end of the wizard and press the **Finish** button, to save the settings.

To set up an alarms server:

- 1 Run the Computer Setup Wizard.
- 2 Select **Custom Setup**, and click **Next**.
- 3 On the 'Computer Role Setup' page, select the appropriate 'Server' option, and click **Next**.
- 4 Follow the prompts given by the wizard. If the computer is to be used as an Alarm, Report, or Trend server, select the appropriate server type when the page for that server type displays.
- 5 On the 'Alarms Setup - Advanced' page, enter appropriate values to configure the behavior of the Alarm Server. Click **Help** on the wizard dialog for option details.
- 6 If the server is to be a network server, the 'Alarms, Reports, and Trends Server Setup' page will display. Enter an appropriate name for this server on the network. This is the name that other machines on the network will be configured to connect with.
- 7 Continue to the end of the wizard and press the **Finish** button, to save the settings.

To set up a trends server:

- 1 Run the Computer Setup Wizard.
- 2 Select **Custom Setup**, and click **Next**.
- 3 On the 'Computer Role Setup' page, select the appropriate 'Server' option, and click **Next**.
- 4 Follow the prompts given by the wizard. If the computer is to be used as an Alarm, Report, or Trend server, select the appropriate server type when the page for that server type displays.
- 5 On the 'Trends Setup - Advanced' page, click **Help** on the wizard dialog for option details.
- 6 If the server is to be a network server, the 'Alarms, Reports, and Trends Server Setup' page will display. Enter an appropriate name for this server on the network. This is the name that other machines on the network will be configured to connect with.
- 7 Continue to the end of the wizard and press the **Finish** button, to save the settings.

To set up a reports server:

- 1 Run the Computer Setup Wizard.
- 2 Select **Custom Setup**, and click **Next**.
- 3 On the 'Computer Role Setup' page, select the appropriate 'Server' option, and click **Next**.
- 4 Follow the prompts given by the wizard. If the computer is to be used as an Alarm, Report, or Trend server, select the appropriate server type when the page for that server type displays.
- 5 On the 'Reports Setup - Advanced' page, select a startup report if required. Click **Help** on the wizard dialog for option details.
- 6 If the server is to be a network server, the 'Alarms, Reports, and Trends Server Setup' page will display. Enter an appropriate name for this server on the network. This is the name that other machines on the network will be configured to connect with.
- 7 Continue to the end of the wizard and press the **Finish** button, to save the settings.

To set-up a time server:

Note: A Time Server can ONLY be set-up on a CitectSCADA I/O server machine.

- 1 Run the Computer Setup Wizard.
- 2 Select **Custom Setup**, and click **Next**.

- 3 On the 'Computer Role Setup' page, select the appropriate 'Server' option, and click **Next**.
- 4 On the I/O Server Setup page, select **This computer is an I/O Server**.
- 5 Select an appropriate I/O server, and click **Next**.
- 6 Follow the prompts given by the wizard. On the 'Time Setup' page, select **This computer is the Time Server**, and click **Next**.
- 7 Continue to the end of the wizard and press the **Finish** button, to save the settings.

See Also [Using TCP/IP for network communications](#)

Using TCP/IP for network communications

By default, CitectSCADA uses NetBIOS to facilitate communications across a network. However, you have the option to use TCP/IP instead.

If you plan to implement web-based deployment of your projects using the CitectSCADA Web Client, the use of TCP/IP becomes a requirement.

To switch your system over to TCP/IP-based communications, a number of parameters must be set in the [citect.ini file](#). They are:

- [\[LAN\]TCPIP](#)

Set this parameter to **1** to on all network computers to enable TCP/IP communication. This will make all machines TCP/IP capable.

- [\[LAN\]NetBIOS](#)

Set this parameter to **zero** (0) to disable NetBIOS. This will force the use of TCP/IP.

Note: If both parameters are set to 1, TCP/IP protocol will take precedence over NetBIOS.

You then need to map the domain name for each server to a TCP/IP address. This is done via the [DNS] section of the Citect.ini file using the following parameter.

- [\[DNS\]<Server name>](#)

Simply type in the name for each server, and then identify its IP address.

Note: NetBIOS automatically identifies a server's role by appending the word "Alarm", "Trend" or "Report" to the end of the proposed server name when it is initially created by the Computer Setup Wizard. To identify a server for TCP/IP, you will need to add this appendage yourself.

For example, if you had the following servers in your system:

Citect_IO_one	Citect_IO_two
Citect.PrimaryAlarm	Citect.StandbyAlarm

Citect.PrimaryTrend	Citect.StandbyTrend
Citect.PrimaryReport	Citect.StandbyReport

You would have to determine the IP address for each machine and add them in the Citect.ini file as follows:

```
[DNS]
Citect_IO_one=192.168.0.10
Citect_IO_two=192.168.0.11
Citect.PrimaryAlarm=192.168.0.12
Citect.StandbyAlarm=192.168.0.13
Citect.PrimaryTrend=192.168.0.14
Citect.StandbyTrend=192.168.0.15
Citect.PrimaryReport=192.168.0.16
Citect.StandbyReport=192.168.0.17
```

If the project is set up for redundancy, both servers must appear in the Citect.ini file. If all of your servers (I/O, alarm, trend and report) are consolidated on one machine, and therefore share the same IP address, you can use the following parameter:

```
[DNS]Primary=192.168.0.21
```

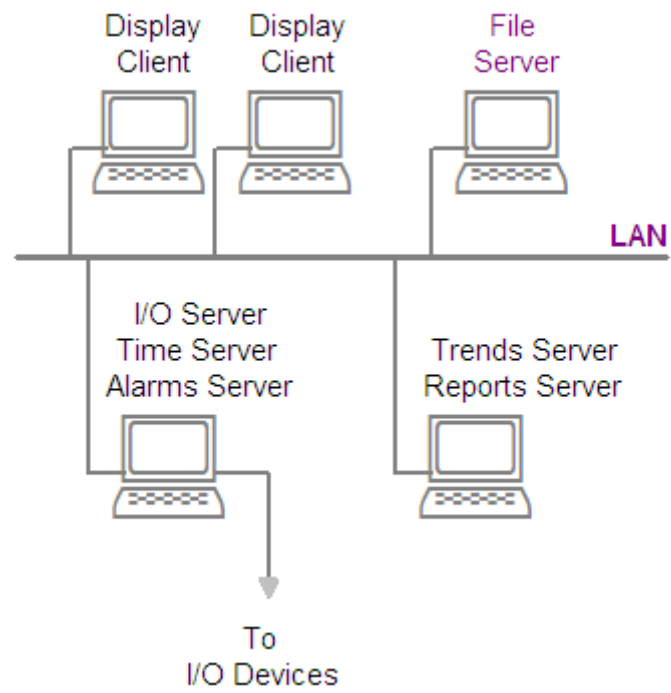
```
[DNS]Standby= 192.168.0.22
```

Using Distributed Processing

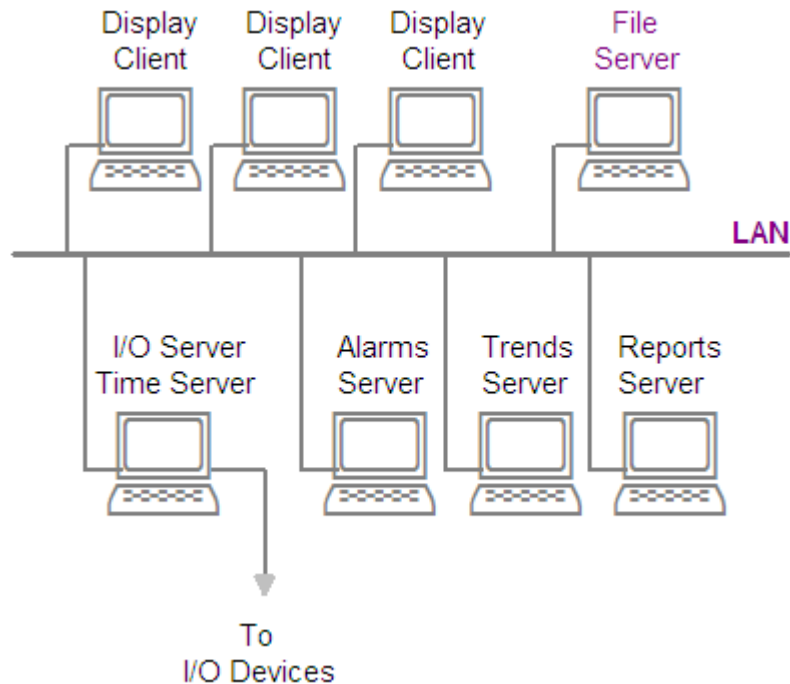
In large applications with large amounts of data, you can reduce the processing load on individual computers by using distributed processing. With CitectSCADA, the function of the [CitectSCADA server](#) can be divided into five individual tasks. These tasks are:

- Communicating with the I/O devices ([I/O server](#))
- Monitoring and processing alarms (alarms server)
- Processing reports (reports server)
- Accumulation and processing of historical data for trending (trends server)
- Synchronisation of system time (Time Server)

These tasks can be performed on a single computer, or can be distributed between two or more computers.



For large applications, you can assign a separate computer for each task.



This is achieved by running the CitectSCADA Computer setup Wizard on the machine you want to become the particular server (I/O, Alarms, Reports, or Trends).

Any CitectSCADA Server machine can be configured to behave as a proxy server by applying appropriate Citect.INI parameters on that machine. For details, see [PROXI parameter settings to make the computer a Proxy server for I/O requests](#).

Any CitectSCADA Server machine can be configured to behave as a File Transfer (FTP) Server for Internet Display Clients if required. See [Internet parameter settings to make the computer an FTP server](#).

Any CitectSCADA Server machine can be configured to use TCP/IP over a Wide Area Network (WAN) if required. See [LAN parameter settings to allow the use of TCPIP over a WAN](#).

The limitations provided by some network configurations can be eased by distributing the processing load across multiple I/O servers. See [Splitting the processing load for multiple I/O servers](#).

See Also [Using Distributed Servers](#)

Splitting the processing load for multiple I/O servers

You can use up to 255 I/O servers on a single CitectSCADA system. The configuration of the I/O servers depends on how each I/O server is connected to the I/O devices. The following guidelines will help you achieve optimum performance in a system with multiple I/O servers.

If all (or most) I/O servers share the same physical link to the I/O devices (e.g. a PLC network) and the PLC network is the performance [bottleneck](#), only one I/O server should communicate with the I/O devices (PLC network).

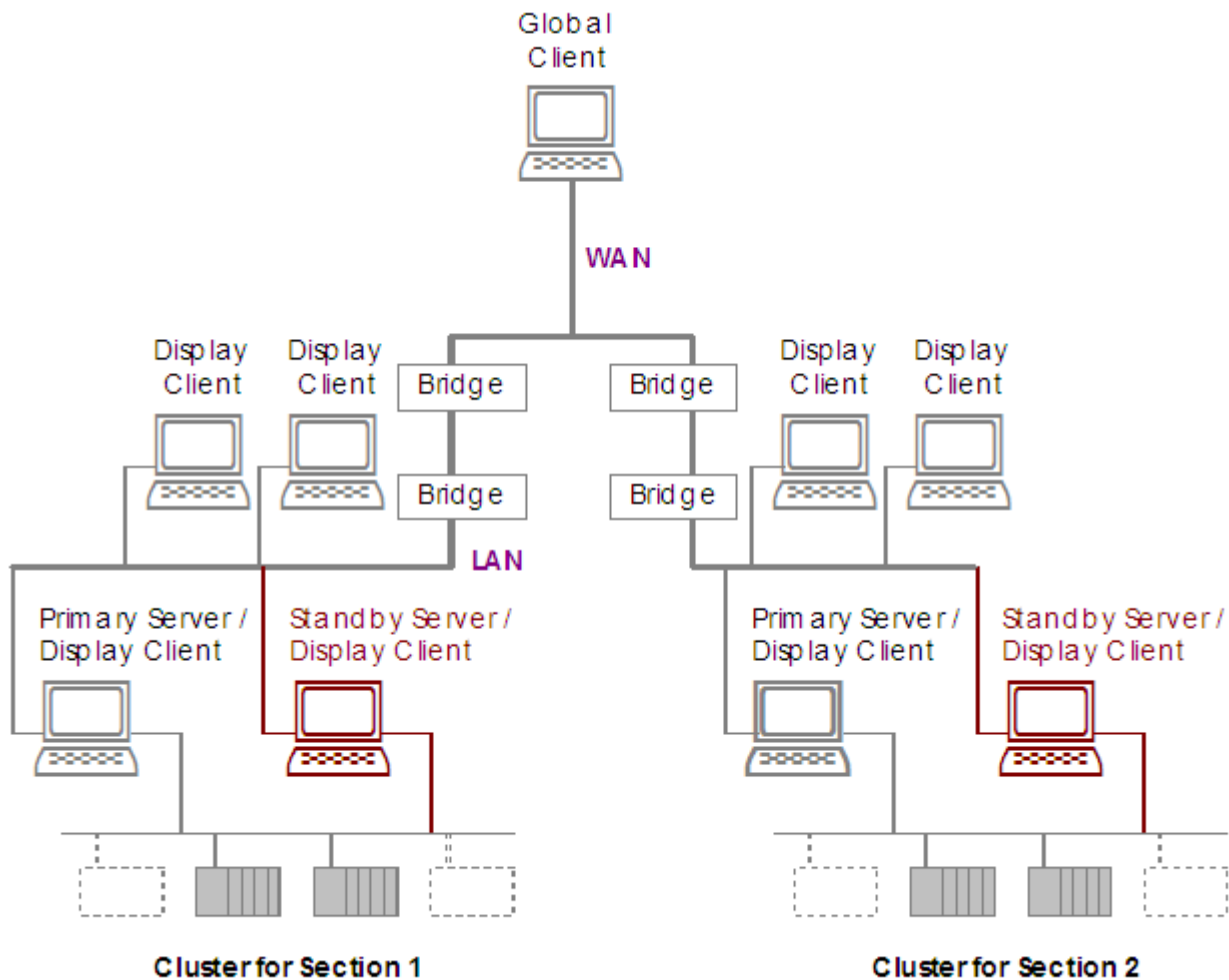
However, with some PLC networks the interface card in the I/O server is the bottleneck. In this situation, you should share the communication load across all I/O servers. You should also share the communication load across all I/O servers when each I/O server has its own physical link to the I/O device (for example, individual serial links).

Note that the configuration depends totally on the type of protocol you are using. Because it is easy to change how the I/O servers are set up (you only need to change the Startup Mode in the I/O Devices form), you should experiment to find the best performance for your plant.

Using Distributed Servers

If your plant consists of several different sections or systems, you can configure a corresponding number of clusters of CitectSCADA Servers, and assign each

[cluster](#) to a different section. All systems can then be monitored using a single Display Client - the [global client](#).



Each cluster runs its own unique project and has unique alarms, trends, reports, and display pages. The Global Client can display information from any cluster projects. For example, at the Global Client, you could display the Trend page from Plant 1, then switch to the Trend page from Plant 2. Ideally, a global system should consist of no more than eight clusters.

Note: Distributed Servers should not be used to split up a single site into discrete areas. A single cluster system with distributed processing would be better suited to this situation, as it would not be hampered by the maintenance overhead of a distributed server system (such as extra project compilations etc.).

See Also [Switching between clusters](#)
[Configuring projects for distributed servers](#)
[Configuring CitectSCADA to communicate over a WAN](#)

Switching between clusters

From the [global client](#), you can display information from any [cluster](#) in your system. To do so, first attach to the relevant cluster server (alarms server, reports server etc.) using the ClusterSetName() function. Then you can display information (such as Trend Tags, Alarm pages etc.) from the cluster's project.

See Also [Configuring projects for distributed servers](#)

Configuring projects for distributed servers

To ensure that the Global Client functions correctly, you must configure your projects correctly. A typical system consists of a:

- Global Include Project
- Cluster Project for each cluster
- Global Display Project

The Global Include Project

The Global Include Project exists purely to be included in each of the Cluster Projects. It would contain Cicode functions, fonts, devices, users, groups, and global keyboard commands.

Cluster Projects

All Cluster Projects must be included in the Global Display Project. Each Cluster Project would contain the following information for its cluster:

Tags	Reports
O Servers	Alarms and Alarm Categories
O Devices	Genies and Super Genies
Templates	Pages
Trends	Symbols

Because all Cluster Projects are included in the Global Display Project, Tag names, I/O device names etc. must be unique for each cluster. For example, you cannot have an I/O device named **IODev_1** in each Cluster Project.

The Cluster Projects will each be compiled and run from a Display Client in the relevant cluster. So **Cluster Project A** will be run from a Display Client in **Cluster A**, and so on.

The Global Display Project

The Global Display Project would be compiled and run from the [global client](#). It would contain a single startup page, and would include each of the Cluster Projects. The startup page could contain several buttons for switching to various pages from each of the clusters (using the ClusterSetName() function).

See Also [Configuring CitectSCADA to communicate over a WAN](#)

Configuring CitectSCADA to communicate over a WAN

A proxy I/O server is used for the optimization of CitectSCADA network traffic for I/O requests. It is therefore particularly suited for use with widely distributed I/O servers over a wide area network. Citect proxy servers are often used with WANs and can also be used as file transfer (FTP) servers for Internet display clients if required.

There are several Citect.INI parameters that work together to achieve the three types of configuration as described below.

- LAN parameter settings to allow the use of TCPIP over the WAN.
- PROXI parameter settings to make the computer a proxy Server for I/O requests.
- INTERNET parameter settings to make the computer an FTP server.

LAN parameter settings to allow the use of TCPIP over a WAN

A typical arrangement of parameters and settings is shown below. The critical setting is 'Tcpip=1' to enable the use of Windows Sockets by CitectSCADA. The Readpool and Sessions parameters have been increased in this example to cater for a large network with many I/O servers connecting to the Proxi Server. TCPIP does not have the maximum sessions limit that NETBIOS has (maximum of 255 sessions), and so permits more CitectSCADA communication sessions than NETBIOS allows.

```
[LAN]
Node=TEST_PC
Disable=0
LanA=-1
Netbios=1
Tcpip=1
Readpool=8096
Sessions=1024
```

You will need to put the 'Tcpip=1' setting into the [LAN] section of the Citect.ini file for all of the I/O servers as well.

The DNS section must define the IP address for the CitectSCADA server and all the I/O servers in the project(s). This is most important for redundancy.

```
[DNS]
Primary=192.168.10.33 (The Citect Primary R.A.T. server)
Secondary= (any Stand by R.A.T. server)
IOServerA=192.168.10.11 (identify every single I/O server here)
```

```
IOServerN=192.168.10.99
```

PROXI parameter settings to make the computer a Proxy server for I/O requests

A CitectSCADA machine can be set to perform as a proxy I/O server through the use of the PROXI parameter settings of the project Citect.INI file on the proxy server machine. For example:

```
[PROXI]
```

```
IOServerA=MyProxy (any name you want to give the proxy server)
```

```
IOServerN=MyProxy
```

OR if a single proxy I/O server is to be used, the following setting can be used; however,, the above makes a lot more sense to other people maintaining the system.

```
[PROXI]
```

```
ALL=MyProxy
```

The other settings required are as follows:

```
[IOSERVER]
```

```
Server=1
```

```
Name=MyProxy
```

Where "MyProxy" is any Proxy I/O server name you want to give it. This machine will actually run up as an I/O server, and get its actual I/O data from the list of I/O servers.

Internet parameter settings to make the computer an FTP server

Typical settings to do this are shown in the following example:

```
[INTERNET]
```

```
Server=1
```

```
display=patrick (any text password for a display license)
```

```
manager=jimmeh (any text password for a manager license)
```

```
RunFTP=1
```

```
ZipFiles=0
```

```
LogFile=D:\
```

Note: Do not make the manager and display passwords the same.

To use this configuration with a large network containing many I/O servers and network Clients, some default resource settings of CitectSCADA may need to be increased. In particular, the networking resources may need to be increased e.g. [LAN]Readpool and [LAN]Sessions parameters. The [KERNEL]Queue parameter may also need to be increased.

Building Redundancy into Your System

While reliability is a key feature of most current computer hardware, breakdowns can still occur. If some (or all) of the processes in your plant are critical, or if the potential down time through failure could be excessive, you should design a level of redundancy into your CitectSCADA system. A system with in-built redundancy minimizes interruptions due to equipment failure. You can choose a level of redundancy to suit the application.

Redundancy is designed into CitectSCADA and can be implemented without changing the project configuration. (CitectSCADA was designed for total redundancy support. Almost everything in CitectSCADA can be made redundant: system display, alarms, trends, reports, I/O servers, external I/O devices, disk I/O devices, network cables, network file servers, FTP servers, and so on.)

The CitectSCADA Computer Setup Wizard allows you to set up redundancy when you define the function of each computer on the network.

See Also

[I/O server redundancy](#)

[Redundancy and persistence](#)

[Data path redundancy](#)

[Alarms, reports, and trends server redundancy](#)

[How CitectSCADA handles file server redundancy](#)

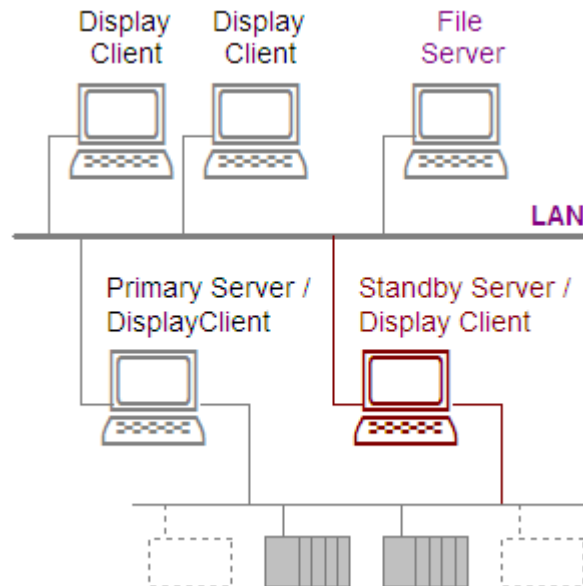
[How CitectSCADA handles FTP server redundancy](#)

[LAN redundancy](#)

I/O server redundancy

Systems with a single [I/O server](#) have a single point of failure. If the server fails, control and monitoring of the system is lost. The single point of failure can be

eliminated with a redundant I/O server that is connected to the same I/O devices. These CitectSCADA servers are called *primary* and *standby* servers.



When the system is in operation, CitectSCADA maintains both servers identically. If the primary server fails, the standby server assumes total control without any interruption to the system. When the primary server is returned to service, CitectSCADA automatically returns control to the primary server. CitectSCADA also ensures that no data is lost.

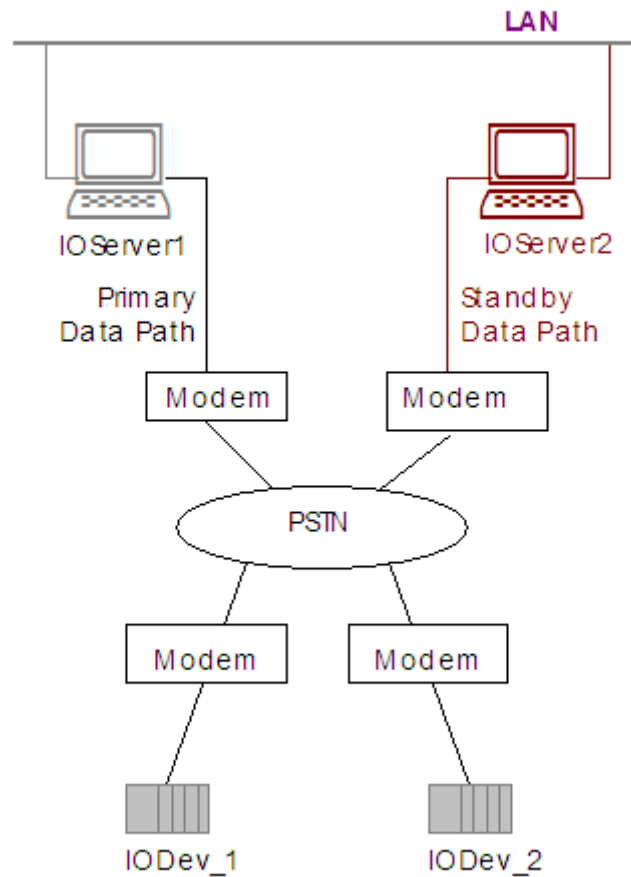
I/O server redundancy stabilizes the system by removing the single point of failure (the CitectSCADA I/O server). However, in the event of failure by the LAN, control and monitoring by the Display Clients is lost (although control and monitoring by the servers is maintained).

When the system is running, you can use redundant I/O servers to split the processing load. Redundant I/O servers result in higher performance, because all I/O servers can be running in parallel when servicing the I/O devices.

See Also [Redundancy and persistence](#)

Redundancy and persistence

If you are using Server redundancy, Persistence Caches keep Standby Servers updated with the most recently read device data. A Persistence Cache is created for each cached I/O device. Consider the following setup:



Every [IOServer]SavePeriod, IO Server1 saves its in-memory cache to disk. The cache is saved in Persistence Caches, one for each cached device. IO Server1 broadcasts to all other I/O servers the UNC path of the Persistence Caches (set with [IOServer]SaveNetwork). From these Persistence Caches, IO Server2 updates its in-memory cache for its I/O devices.

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 Citect'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. You can, therefore, set up several I/O servers to update their in-memory caches with the most recently read data.

For this example, set the [IOServer]SaveFile and [IOServer]SaveNetwork parameters like this:

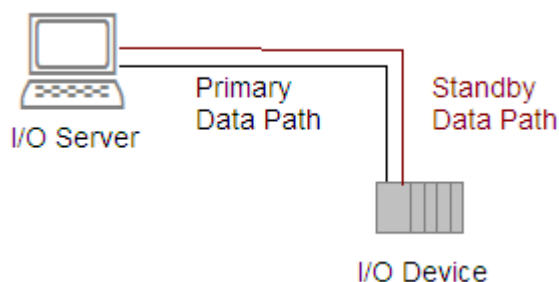
On IOServer1	On IOServer2
[IOServer]	[IOServer]
SaveFile=C:\Data\IOServer1.dat	SaveFile=C:\Data\IOServer2.dat
SaveNetwork=\\IOServer1\Data\IOServer1.dat	SaveNetwork=\\IOServer2\Data\IOServer2.dat

IOServer1 would broadcast the path '\\IOServer1\Data\IOServer1.dat' to the other I/O servers. 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](#)

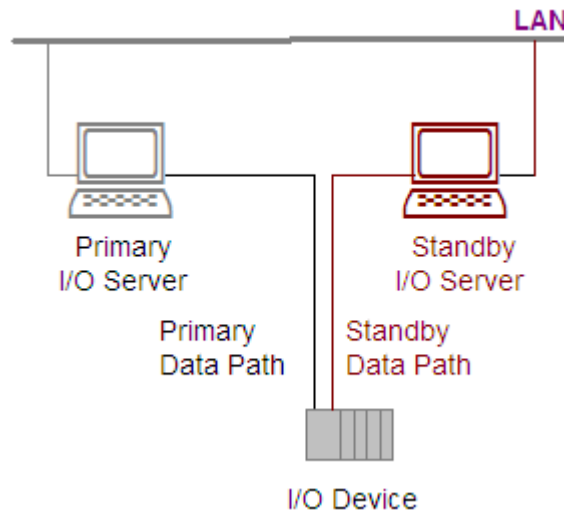
Data path redundancy

With most brands of PLCs, you can install a parallel data path from the I/O server to the I/O device. A parallel data path ensures that if one data path fails, your system can continue without interruption.



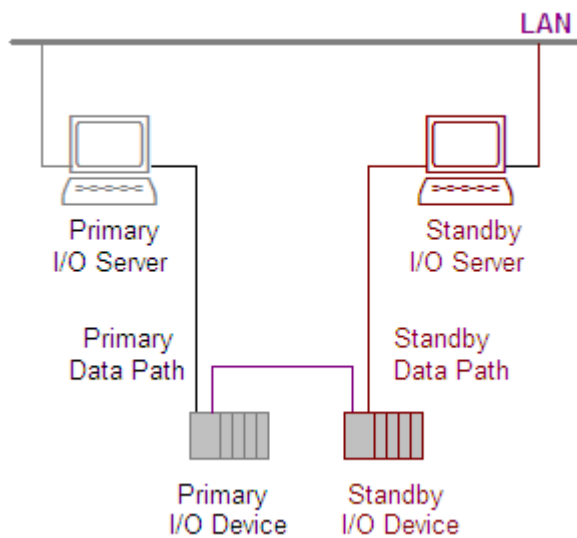
When you start your runtime system, CitectSCADA connects to the I/O device using the primary data path. If communication with the I/O device fails at any time (e.g. if the communications cable is cut), CitectSCADA switches to the standby data path. CitectSCADA reconnects through the primary data path when it is returned to service.

You can also use data path redundancy on a network, as shown here:



In this example, I/O device communication is maintained if either one of the I/O servers or its communications cable fails.

If your I/O devices support peer-to-peer communication, you can provide total redundancy to your system by duplicating I/O devices, as shown here:



One of these I/O devices is the primary I/O device, and the other is the standby I/O device. (You can also have more than one standby I/O device.) When both I/O devices are running, CitectSCADA processes the I/O on the Primary I/O device.

This reduces the I/O load on the I/O device (and PLC network), which is critical for the best performance. You do not have to synchronize data between the primary and standby I/O devices.

Note: Although I/O servers do not adopt a Primary or Standby role, they are generally labelled "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.

To use this arrangement, the I/O devices must support hot-standby redundancy. While CitectSCADA can send write requests to both the primary and standby I/O devices (with the **Startup mode StandbyWrite** option), CitectSCADA cannot synchronize the I/O devices or plant-floor equipment.

CitectSCADA clients communicate with all configured I/O servers at the same time. (On startup, the clients try to connect to all configured I/O servers. If they cannot find an I/O server, a hardware error is generated.) The CitectSCADA client routes the particular I/O request to the active I/O device. For example, if you have three I/O servers configured as follows:

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)

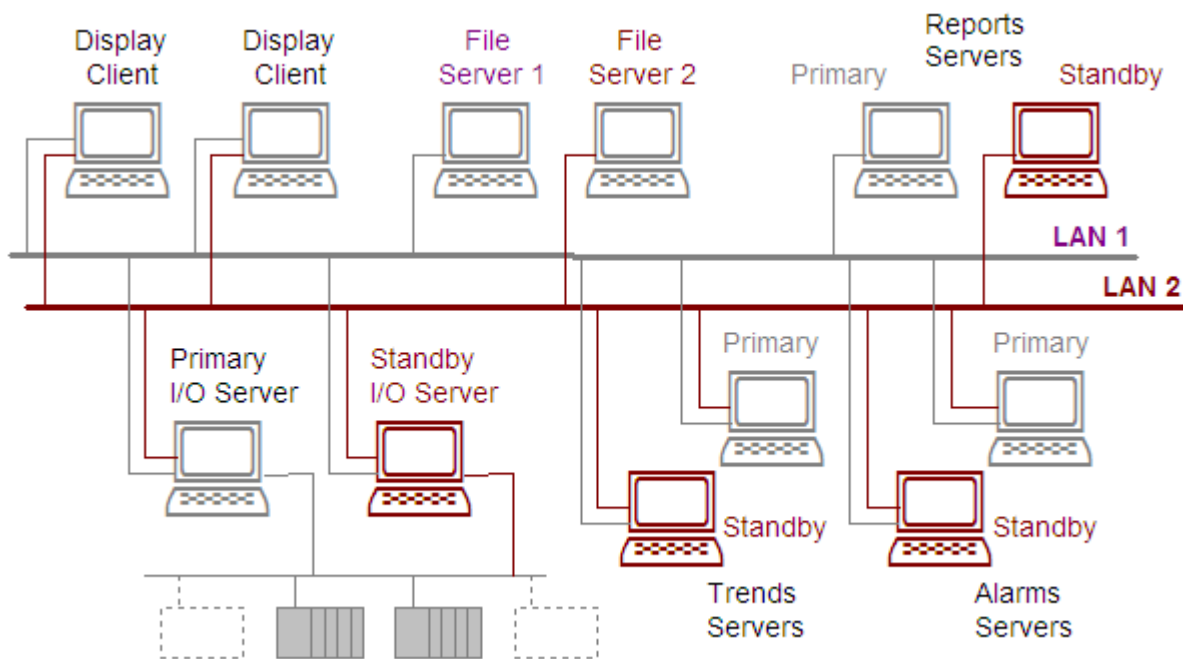
If all I/O devices are communicating correctly, a CitectSCADA client creates network sessions to all three I/O servers. The client then sends requests for I/O device1 and I/O device3 to IOServer1, and requests for I/O device2 to IOServer2. If I/O device1 fails on IOServer1, the client sends requests for this I/O device to IOServer2, while it still sends requests for I/O device3 to IOServer1. If I/O device1 also fails on IOServer2, the client sends requests to IOServer3. When I/O device1 on IOServer1 comes back on line, the clients begin sending their requests to IOServer1.

Because you 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. (This might not apply for all protocols, because the loading could be dependent on the PLC network, 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

On large systems with multiple servers, you can parallel the Alarms, Reports, and Trends Servers. To achieve this level of redundancy, you configure three other computers (or Display Clients) as standby servers. Then if a primary server fails, its operation is immediately transferred to its standby server.



When the system is in operation, CitectSCADA mirrors the primary and standby servers. If the primary Reports, Alarms, or trend server fails, all clients access the appropriate standby server for data. When the primary server restarts, the clients stay on the standby server unless the standby server fails, or the client is shutdown and restarted. (Because CitectSCADA maintains identical data on both 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 and some with the standby server. This also saves the extra overhead of checking if a primary server has come back online.)

See Also [How CitectSCADA handles alarms server redundancy](#)
[How CitectSCADA handles reports server redundancy](#)
[How CitectSCADA handles trends server redundancy](#)

How CitectSCADA handles alarms server redundancy

You can configure two Alarms Servers in a CitectSCADA project: 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 running, 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 starts up, it tries to connect to the other alarms server. If it can connect, it transfers the dynamic alarm data from the running alarms server. (This data includes summary data and the current alarm states.) If another alarms server cannot be found, 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 Server and one from the Standby Server, CitectSCADA uses the save file with the later date. If no save file is configured, the alarms server cannot get the initial state of the alarms, and no summary information is available. In this case, the alarms server starts processing the alarms, and then acknowledges all the new alarms.

While both Alarms Servers are active, they 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 connect to the Primary alarms server. If the Primary alarms server is not running, they try to connect to the Standby alarms server. If the Primary alarms server comes back on line, any clients connected to the Standby alarms server remain connected to the Standby Server. (It does not matter which alarms server the clients talk to, because they both contain the same (mirrored) data.)

See Also [How CitectSCADA handles reports server redundancy](#)

How CitectSCADA handles reports server redundancy

You can configure two Reports Servers in a CitectSCADA project - a Primary reports server and a Standby reports server. When both Reports Servers are running, 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.) No report data is transferred between the Primary and Standby 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 connect to the Primary reports server.

If the Primary reports server is not running, they try to connect to the Standby reports server. If the Primary reports server comes back on line, any clients connected to the Standby reports server remain connected to the Standby server.

See Also [How CitectSCADA handles trends server redundancy](#)

How CitectSCADA handles trends server redundancy

You can configure two Trends Servers in a CitectSCADA project - a Primary trends server and a Standby trends server. When both Trends Servers are running, 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 connect to the other trends server. If it can connect, it transfers 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 connect to the Primary trends server. If the Primary trends server is not running, they try to connect to the Standby trends server. If the Primary trends server comes back on line, any clients connected to the Standby trends server remain connected to the Standby trends server. (It does not matter which trends server the clients talk to, 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.

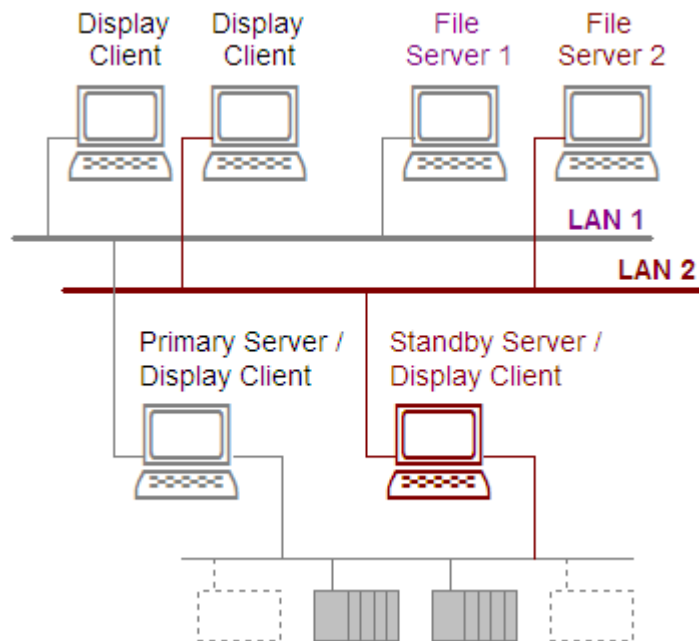
FTP Server Redundancy is configured by setting parameters in the [CLIENT] and [DNS] sections of the Primary FTP Server's Citect.ini file. These parameters are downloaded by the Internet Display Client (IDC) to its own Citect.ini file if the Primary FTP Server fails, provided the [INTERNET]Redundancy 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.

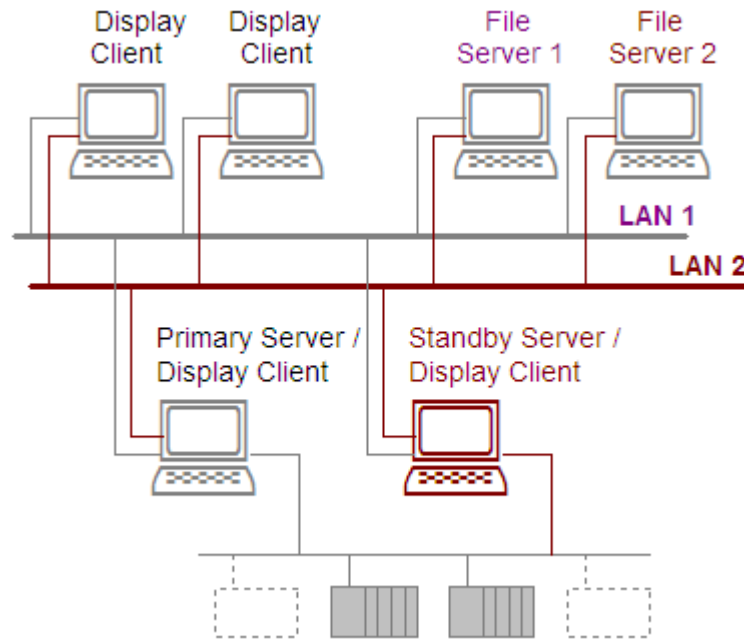
See Also [LAN redundancy](#)

LAN redundancy

A second LAN and [file server](#) would ensure system stability even in the event of network failure.



Here, half of the computers remain operable if one of the LANs or a server fails. With two network cards in each computer, full operation of all computers can be maintained in the event of a failure of one of the LANs (or a server).



NetBIOS Errors

This section describes the NetBIOS errors.

1024 No NetBIOS error

This error should not occur in normal operation. Contact Citect Support.

1025 Invalid buffer length

This error should not occur in normal operation. Contact Citect Support.

1027 Invalid command

This error should not occur in normal operation. Contact Citect Support.

1029 Command timed out

CitectSCADA is timing out when sending data on the network. If this error occurs frequently, increase the timeout period in the [LAN] SendTimeout parameter. This error is likely to occur if you are running CitectSCADA on a slow network or a Wide Area Network.

1030 Incomplete receive message

This error should not occur in normal operation. Contact Citect Support.

1032 Invalid session number

This error should not occur in normal operation. Contact Citect Support.

1033 No resource available

Increase network resources or memory. Increase the Windows parameter NetHeapSize in the SYSTEM.INI file (or other network parameters). See [Setting up a Network](#).

1034 Session has been closed

This error should not occur in normal operation. Contact Citect Support.

1035 Command cancelled

This error should not occur in normal operation. Contact Citect Support.

1037 Duplicate name in local table

This error should not occur in normal operation. Contact Citect Support.

1038 NetBIOS name table full

Increase the number of names in the local name table setup in the network NetBIOS configuration. See [Setting up a Network](#).

1041 NetBIOS session table full

CitectSCADA has run out of NetBIOS sessions. Increase the number of NetBIOS sessions in the network setup. See [Setting up a Network](#).

1044 Server name not found

The specified server cannot be found on the network. Either the server has not started or a network problem is preventing communication.

1046 Name in use on remote adaptor

Two CitectSCADA servers on the network are trying to use the same name. Configure each CitectSCADA server with a unique name.

1049 Name conflict

Two CitectSCADA servers on the network are trying to use the same name. Configure each CitectSCADA server with a unique name.

1058 Too many commands outstanding

CitectSCADA has run out of NetBIOS control blocks (NCBs). Increase the number of NCBs in the network NetBIOS configuration or reduce CitectSCADA's use of NCBs in the CITECT.INI file. See [Setting up a Network](#).

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.

Appendix A: Parameters

Using Parameters

Parameters determine how each CitectSCADA computer operates in the CitectSCADA configuration and runtime environments. For example, there is a parameter which allows you to show/hide the toolbar in the Citect Project Editor, and there is a parameter which determines whether the Primary and Redundant Reports Servers send out heartbeat signals to each other at runtime.

You can set operating parameters in:

- The project database.
- The CITECT.INI file (By default, CitectSCADA looks for the ini file in the Citect\Bin directory. If it can't find it there, it will search the WINDOWS directory of each CitectSCADA computer.)
- Both the project database and the CITECT.INI file.

You must observe the following rules when using parameters:

- Parameters set in the CITECT.INI file take precedence over parameters set in the project database.
- If you set (or change) parameters in the project database, you must re-compile the project before the parameter settings are used.
- If you set (or change) parameters in the CITECT.INI file, you must restart CitectSCADA before the parameter settings are used.
- Parameters set in the database are local to the specific Citect project. Parameters set in the CITECT.INI file apply to all CitectSCADA projects (if you are using multiple CitectSCADA systems).

To set parameters in the project database:

- 1 Choose **System | Parameters**.
- 2 Enter the Section Name of the Parameter.
- 3 Enter the Name of the Parameter.
- 4 Enter a value for the Parameter.
- 5 Add the record to the database.

To set parameters in the local CITECT.INI file:

- 1 Locate the parameter in Help

Rules for using parameters

- 2 Use the button (below the default value) to edit the value.

Note: The current value of the parameter is displayed in the dialog field. (If the dialog field is blank, the parameter is set to its default value)

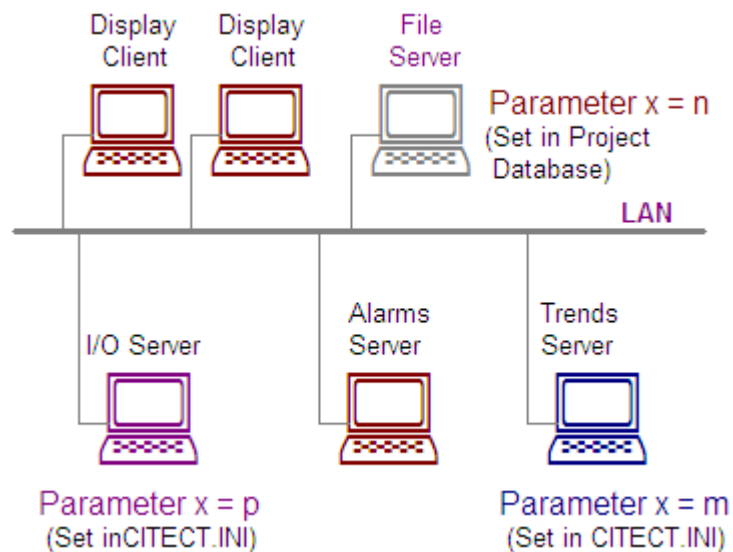
or -

- 3 Use a text Editor to Edit the CITECT.INI file (in the Windows directory)
- 4 Enter the parameter in the following format:

```
[SECTION NAME]
Parameter=<value>
```

Using parameters on a network

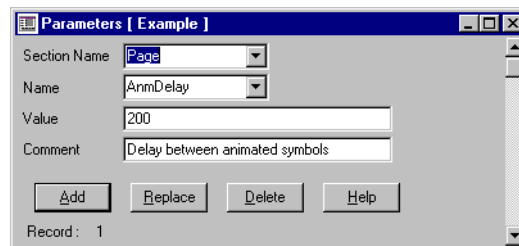
If using CitectSCADA on a network, you can use parameters globally, locally (local to each server and ["display client"](#)), or both globally and locally. Any parameter set in the project database applies to all Display Clients unless the parameter is also set in the CITECT.INI of a Display Client. The value set in the local CITECT.INI file takes precedence over the project database for that Display Client only. For example:



Here, a parameter (Parameter x) is set to a value **n** in the project database (on the ["file server"](#)). When the system is running, this value (**n**) applies to the alarms server and both Display Clients. The same parameter is set to different values for both the ["I/O server"](#) and the trends server (set locally in the respective CITECT.INI files). When the system is running, the I/O server uses the value **p** for the parameter, and the trends server uses the value **m**.

Parameters Dialog

You use the Parameters dialog box to assign properties to your parameters.



Parameter Properties

Parameters have the following properties:

Section Name

The parameter section. Enter a value of 16 characters or less.

Name

The name of the parameter for which you want to define a value. Enter a value of 16 characters or less.

Value

The value of the parameter. Enter a value of 254 characters or less.

Comment

Any useful comment. Enter a value of 48 characters or less.

For example, to define the computer as an I/O server, enter **IOSERVER** as the Section Name, **Server** as the Name, and **1** as the Value. In the CITECT.INI file, the parameter would appear as follows:

```
[IOSERVER]
Server=1
```


Appendix B: CitectSCADA Reference Information

Specifications

This section defines the reference information for CitectSCADA specifications.

- [Graphics](#)
- [Projects](#)
- [I/O device data types](#)
- [Reserved ANs](#)
- [Predefined Templates](#)
- [Predefined Commands](#)
- [Predefined Character Sets](#)
- [Predefined Fonts](#)
- [Predefined Devices](#)
- [Predefined Cicode Files](#)
- [Predefined Color Names and Codes](#)
- [Predefined Keyboard Key Codes](#)
- [Predefined Labels](#)
- [ASCII/ANSI Character Code Listings](#)

Graphics

The table below defines CitectSCADA graphics specifications.

High-resolution color presentation	VGA, SVGA, XGA, SXGA (Any resolution)
Colors in Palette	255
User definable colors can be selected from a palette of	16.8 Million
Free-form graphics/display pages	65000*
Object animation points (ANs) per page	32000
Screen update time	500 milliseconds (see note)

Note: Screen update time depends on the I/O device protocols used and your system design. The minimum update rate is 1 millisecond, which can be achieved only if the PLC can provide data fast enough. CitectSCADA maintains

the fastest possible screen update rate with the use of read on demand and dynamic optimization. These technologies allow CitectSCADA to read only what is required from the PLC, making the most of the communication channel to the I/O devices. Performance test from real installed systems with 100,000 points can maintain screen update rates of 400 milliseconds.

Projects

The table below defines CitectSCADA projects specifications:

Configuration Projects	1022*
Number of Variables defined in CitectSCADA	4,194,303 (but 500,000* is the max recommended) IMPORTANT: The max depends on the bit length of your variables. For example, a 32-bit digital reduces the number of possible variables by 32 (i.e. if your project contains only 32-bit digitals, you can have a maximum of 131,072). The bit size of the digitals depends on the driver being used. If no bit size is given for the driver, it defaults to 16.
Number of Included projects	240 (including the Include project)
Simultaneously logged-in users	250*
Number of reports	1000*
Number of I/O device addresses monitored by alarms	150000*
Number of historical trends	8000*
Number of trends displayed on the same chart	8
Number of trends displayed on the same page	16000
Number of user functions	4500*
Number of standard in-built functions provided	700
Number of operator commands for the system	3000*
Number of I/O devices connected to CitectSCADA	16383
Number of simultaneous multiple protocols	4095
Number of areas	255
Number of alarm categories	16376
Maximum simultaneous multi-tasking threads	512

*There is no actual limit to these values; they are maximum recommended values only. However, exceeding this number can reduce system performance.

I/O device data types

The table below displays the data types, size, and allowed values.

Data Type	Size	Allowed Values
BCD	2 bytes	0 to 9,999
Byte	1 byte	0 to 255
Digital	1 bit or 1 byte	0 or 1
Integer	2 bytes	-32,768 to 32,767

Integer 2 bytes -32,768 to 32,767

Reserved ANs

The following table describes all reserved ANs:

AN	Description	Comments
1	Keyboard Entry Line	Where keyboard input from the operator is echoed (displayed).
2	Prompt Line	The Prompt Line is used to convey important information to your operators. You can use the Prompt() function to display prompts to help an operator with a process, or DspError() to display warning messages. You must use a Cicode function to display a prompt message.
The following ANs are reserved for Version 2.xx style templates only (or for pages that are upgraded from Version 2.xx):		
3	Reserved	
4	Reserved	
5	Unacknowledged Alarms	If an alarm has not been acknowledged, the message "UNACKNOWLEDGED ALARMS" is displayed. You could then select an alarm page to display details of the alarm.
6	Hardware Alarms	If a hardware alarm is detected by CitectSCADA, the message "HARDWARE ALARMS" is displayed. You could then select an alarm page to display details of the alarm.
7	Disabled Alarms	If an alarm is disabled, the message "DISABLED ALARM" is displayed. You could then select an alarm page to display details of the alarm.
8	Reserved	
9	Time	The current system time is displayed. To set the format for the time display, use the Windows Control Panel in the Main Windows program group, or set a CitectSCADA Parameter.
10	Date	The current system date is displayed. To set the format for the date display, use the Windows Control Panel in the Main Windows program group, or set a CitectSCADA Parameter.
11	Last Alarm	Where the last activated alarm is displayed.
12	Page Title	Where the title of the graphics page is displayed.
13	Page Name	Where the name of the graphics page is displayed.
14	Command Help	Where help text associated with a button or animation object is displayed.
15	Buttons	Page-dependent buttons.
16	Buttons	Page-dependent buttons.
17	Buttons	Page-dependent buttons.

AN	Description	Comments
18	Last Page Button	A button to select the graphics page that was displayed before the current page.
19	Page Up Button	A button to select the next graphics page in the page sequence.
20	Page Down Button	A button to select the previous graphics page in the page sequence.

Predefined Templates

The following templates are provided in various styles. Most of these templates are completely configured; you can create pages with little (or no) customisation of the templates.

Template Name	Description
Normal	A template for basic graphics display pages. This template contains buttons for basic page control (such as displaying alarm and menu pages) and a large blank area for drawing plant layouts, control buttons, etc.
Blank	A completely blank template. Use this template to configure an entire page.
PageMenu	A template to create a simple menu page. CitectSCADA automatically generates a menu page (based on this template) as you develop your project. You can modify the menu page to suit your specific requirements.
Book1Menu . . Book5Menu	<p>Templates to create alternative menu pages (in open book format). An operator can move through the menu pages by clicking on the appropriate tab. To use these templates, add buttons to each menu page to display other graphics pages as required.</p> <p>You should create your pages with the same names as the templates to avoid extra configuration. If your pages are not linked , you can modify the menus so that any page name will be accepted.</p>
Tab1Menu . . Tab6Menu	<p>Templates to create alternative menu pages (in tab format). An operator can move through the menu pages by clicking on the appropriate tab. To use these templates, add buttons to each menu page to display other graphics pages as required.</p> <p>You should create your pages with the same names as the templates to avoid extra configuration. If your pages are not linked , you can modify the menus so that any page name will be accepted.</p>
SingleTrend	<p>A template to create trend pages with one trend window. There are several ways to configure the trend pens:</p> <ol style="list-style-type: none"> 1. Double-click the window. 2. Pass the trend pens with PageTrend() function on page entry at runtime 3. Select the pens manually (from the page) at runtime <p>SingleTrend pages can be configured with trend tags of type Periodic or Periodic Event.</p>
DoubleTrend	<p>A template to create trend pages with two trend windows. To configure the trend pens for each window, double-click either window, or select the pens manually (from the page) at runtime. Add a button to the menu page to display each trend page.</p> <p>DoubleTrend pages can be configured with trend tags of type Periodic or Periodic Event.</p>

Template Name	Description
CompareTrend	A template to create trend pages with two trends - one overlaid on the other. To configure the trend pens for each trend (maximum of 4 each), double-click the trend window, or select the pens manually (from the page) at runtime. Add a button to the menu page to display each trend page. CompareTrend pages can be configured with trend tags of type Periodic or Periodic Event.
EventTrend	A template to create trend pages with one event trendwindow. There are several ways to configure the trend pens: i) double-click the window ii) pass the trend pens using the PageTrend() function on page entry at runtime iii) select the pens manually (from the page) at runtime EventTrend pages can only be configured with trend tags of type Event.
ZoomTrend	A template to create trend pages with one trend window and a zoom window. To configure the trend pens, double-click in the window, or select the pens manually (from the page) at runtime. ZoomTrend pages can be configured with trend tags of type Periodic or Periodic Event.
PopTrend	A template to create a small trend page to display as a pop-up trend. To configure the trend pens, pass the PageTrend() function on page entry at runtime, or select the pens manually (from the page) at runtime. PopTrend pages can be configured with trend tags of type Periodic or Periodic Event.
MeanMeanChart	A template to create SPC pages with two mean windows.
RangeChart	A template to create SPC pages with mean and range windows.
StandardChart	A template to create SPC pages with mean and standard deviation windows.
SPCCPK	A template to create SPC capability charts. To configure your pen, double-click the window, or select the pen manually (from the page) at runtime. SPCCPK pages can be configured with SPC tags of type Periodic or Event.
SPCPareto	A template to create SPC Pareto charts. To configure your variable tags (NOT trend tags), double-click the window.
SPCXRSChart	A template to create SPC control chart with mean, range, and standard deviation windows. To configure your pen, double-click the window, or select the pen manually (from the page) at runtime. SPCXRSChart pages can be configured with SPC tags of type Periodic or Event.
EventSPCXRS	A template to create trend pages with one event SPCXRS window. To configure your pen, double-click the window, or select the pen manually (from the page) at runtime. EventSPCXRS pages can only be configured with SPC tags of type Event.
Alarm	A template to create an alarm display page. You must create a page called "Alarm" based on this template, so that the alarm display button (on other pages such as the menu page) operates correctly. (The alarm display button calls the PageAlarm() function.) You can create the "Alarm" page directly from this template (without modification), or modify the page to suit your requirements.
Summary	A template to create an alarm summary page. You must create a page called "Summary" based on this template, so that the alarm summary button (on other pages such as the menu page) operate correctly. (The alarm summary button calls the PageSummary() function.) You can create the "Summary" page directly from this template (without modification), or modify the page to suit your requirements.

Template Name	Description
Hardware	A template to create a hardware alarm page. You must create a page called "Hardware" based on this template, so that the hardware alarm button (on other pages such as the menu page) operate correctly. (The hardware alarm button calls the PageHardware() function.) You can create the "Hardware" page directly from this template (without modification), or modify the page to suit your requirements.
Disabled	A template to create a disabled alarm page. You must create a page called "Disabled" based on this template, if you use the PageDisabled() function. You can create the "Disabled" page directly from this template (without modification), or modify the page to suit your requirements.
File	A template to create a file-to-screen display page. You can use this page to display any ASCII files (such as reports or other information). You must create a page called "File" based on this template, if you use the PageFile() function. You can create the "File" page directly from this template (without modification), or modify the page to suit your requirements.
GroupStatus	A template to create a status table page for groups of plant floor devices.
TrnPopStat	A template to create a page displaying trend statistics. You must create a page called "TrendStats" based on this template. When called from a trend window, it will display the statistics of the trend pens used in that window (such as Min, Max, Average etc.). With the TrnPopStat window displayed, you can also rubber-band an area of the trend, and the statistics for that area will display.

Predefined Commands

The following System Keyboard commands are predefined in the Include Project(System Keyboard commands operate on any graphics page displayed on the computer screen):

System keyboard commands database

The table below gives the key sequences associated with commands and their function.

Key Sequence	Command	Description
*BS	KeyBS()	Backspace over the current key
DOWN	KeyDown()	Move the cursor down
PGDN	PagePrev()	Display the previous page
PGUP	PageNext()	Display the next page
RIGHT	KeyRight()	Move the cursor right
UP	KeyUp()	Move the cursor up

Usually you can override a predefined command by configuring a new command in your project with the same key sequence. The only command that you cannot override is the *BS command, as this sequence is a hotkey used to remove the last key from the key command line.

Note: Do not modify the Include Project. Your changes to the Include project will be lost when you reinstall CitectSCADA or upgrade to a new version.

Predefined Keyboard Keys

The following keyboard keys are predefined in the Include Project. You can use these keys in any key sequence field; for example, to define the keyboard commands for an object:

Keyboard keys database

The table below defines the key names, codes, and description.

Key Name	Key Code	Description
BS	KEY_BACKSPACE	BackSpace key
DOWN	KEY_DOWN	Cursor down
ENTER	KEY_ENTER	Enter key
LBUTTON_DN	KEY_LBUTTON_DN	Left mouse button down
LBUTTON_UP	KEY_LBUTTON_UP	Left mouse button up
LBUTTON_CMD_DN	KEY_LBTN_CMD_DN	Left mouse button down (command cursor)
LBUTTON_CMD_UP	KEY_LBTN_CMD_UP	Left mouse button up (command cursor)
LEFT	KEY_LEFT	Cursor left
MBUTTON_DN	KEY_MBUTTON_DN	Middle mouse button down
MBUTTON_UP	KEY_MBUTTON_UP	Middle mouse button up
PGDN	KEY_PGDN	Page down key
PGUP	KEY_PGUP	Page up key
RBUTTON_DN	KEY_RBUTTON_DN	Right mouse button down
RBUTTON_UP	KEY_RBUTTON_UP	Right mouse button up
RBUTTON_CMD_DN	KEY_RBTN_CMD_DN	Right mouse button down (command cursor)
RBUTTON_CMD_UP	KEY_RBTN_CMD_UP	Right mouse button up (command cursor)
RIGHT	KEY_RIGHT	Cursor right
UP	KEY_UP	Cursor up

Note: Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade CitectSCADA.

Predefined Character Sets

The following character sets are predefined as labels in the Include Project:

Label	Value	Description
DEFAULT_CHARSET	1	Use the default Windows character set
SHIFTJIS_CHARSET	128	Japanese character set
HANGEUL_CHARSET	129	Korean character set
GB2312_CHARSET	134	Chinese character set
CHINESEBIG5_CHARSET	136	Chinese character set
JOHAB_CHARSET	130	
HEBREW_CHARSET	177	
ARABIC_CHARSET	178	
GREEK_CHARSET	161	
TURKISH_CHARSET	162	

Label	Value	Description
VIETNAMESE_CHARSET	163	
THAI_CHARSET	222	
EASTEUROPE_CHARSET	238	
RUSSIAN_CHARSET	204	
BALTIC_CHARSET	186	

Note: Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade CitectSCADA.

Predefined Fonts

The following fonts are predefined in the Include Project:

Font Name	Font Type	Size	Color
AlmAccOffFont	Arial	10	White
AlmAccOnFont	Arial	10	Cyan
AlmDisabledFont	Arial	10	White
AlmUnAccOffFont	Arial	10	Brown
AlmUnAccOnFont	Arial	10	Yellow
ButtonFont	Arial	10	Black
Casanova	Arial	-10	Black
ControlLimits	Times New Roman	14	Black
DefaultFont	Courier New	14	White
DisabledFont	Arial	10	White
FontOP	Courier New	14	Light Cyan
FontPV	Courier New	14	Light Green
FontSP	Courier New	14	Light Red
FontTune	Courier New	14	Yellow
GraphBigFont	Arial	60	Black
GraphColour	Arial	32	Blue
GraphColourBig	Arial	60	Red
GraphColourSmall	Courier New	20	Black
GraphFont	Arial	32	Black
GraphSmallFont	Courier New	20	Black
HardwareFont	Arial	10	Light_Red
Pen1SpcFont	Courier	10	White
Pen1TrendFont	Courier New	14	Light_Green
Pen2SpcFont	Courier New	14	Light_Green
Pen2TrendFont	Courier New	14	Yellow
Pen3SpcFont	Courier New	14	Light_Cyan
Pen3TrendFont	Courier New	14	Light_Red
Pen4SpcFont	Courier New	14	Light_Blue
Pen4TrendFont	Courier New	14	Light_Cyan

Font Name	Font Type	Size	Color
Pen5TrendFont	Courier New	14	Light_Magenta
Pen6TrendFont	Courier New	14	White
Pen7TrendFont	Courier New	14	Light_Blue
Pen8TrendFont	Courier New	14	Gray
PromptFont	Arial	10	White
SpcFont	Courier New	14	White
TextFont	Arial	10	White
TimeFont	Arial	10	Black
TrendFont	Courier New	14	White
TrendHistFont	Courier New	14	Yellow
TrendSHistFont	Arial	-10	Magenta
TrendSFont	Arial	-10	Black
UnacceptedFont	Arial	10	Yellow
Vanuatu	Arial	-9	Black
System	Arial	10	Black
TrendSCentreFont	Arial	-10	Yellow
PopFont	Arial	9	Black

You can override a predefined font by adding a new font with the same name to your project.

Note: Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade CitectSCADA.

Predefined Devices

The following devices are predefined in the Include Project.

Devices database

The table below shows the devices supported by CitectSCADA.

Device Name	Type	Description
ASCII_DEV	0	Ascii Device number
PRINTER_DEV	1	Printer Device number
dBASE_DEV	2	dBASE device number
SQL_DEV	4	SQL device number
AlarmDisk	0 (ASCII File)	Default alarm log file
AlarmPrint	0 (ASCII File)	Default alarm print device
KeyDisk	0 (ASCII File)	Default keyboard log file
KeyPrint	1 (Printer)	Default keyboard printer log
Printer1	1 (Printer) LPT1:	Printer 1 device
Printer2	1 (Printer) LPT2:	Printer 2 device
SummaryPrint	0 (ASCII File)	Default alarm summary printer device
SummaryDisk	0 (ASCII File)	Default alarm summary log file

Device Name	Type	Description
_Trend	3 (dBase)	Trend RDB device
Scratch	0 (ASCII File)	Device for DevModify function

Note: Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade CitectSCADA.

Predefined Cicode Files

The following Cicode files are part of the Include Project:

File Name	Description
citect.ci	General utility functions
debug.ci	User Cicode debugging functions
export.ci	Information functions
graph.ci	Trend data export functions
info.ci	Information functions
numpad.ci	Number entry keypad functions
page.ci	Graphics page utility functions
pareto.ci	Functions for the Pareto charts
spc.ci	Default SPC functions
spcplus.ci	SPC functions - extension
statpop.ci	Trend statistic functions
tag.ci	Functions for Tag assignment and manipulation
trend.ci	Default trend functions
trinfo.ci	Functions to gather trend information
zoom.ci	Trend zoom functions

Note: Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade CitectSCADA.

Predefined Color Names and Codes

Sixteen standard colors are available for use with your CitectSCADA system. They have been predefined in the Include Project. You should refer to these colors by name which make then more readily understandable, wherever you would use the code value:

Color Label	Code
Black	0x000000
Blue	0x000080
Green	0x008000
Cyan	0x008080
Red	0x800000
Magenta	0x800080
Brown	0x808000
Gray	0xBF8080
Dark_Gray	0x7F7F7F

Color Label	Code
Light_Blue	0x0000FF
Light_Green	0x00FF00
Light_Cyan	0x00FFFF
Light_Red	0xFF0000
Light_Magenta	0xFF00FF
Yellow	0xFFFF00
White	0FFFFFFF
TRANSPARENT	0xFF000000

Note: Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade CitectSCADA.

Predefined Keyboard Key Codes

The following meaningful key code labels are predefined in the CitectSCADA Include Project. They can be entered as Key Codes when you define your keyboard keys, so you don't need to remember the Hex value that is associated with each key.

Key Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
KEY_LBUTTON	0x0001	Left Mouse Button
KEY_RBUTTON	0x0002	Right Mouse Button
KEY_MBUTTON	0x0004	Middle Mouse Button
KEY_LBUTTON_UP	0x0201	Left Mouse Button Up
KEY_RBUTTON_UP	0x0202	Right Mouse Button Up
KEY_MBUTTON_UP	0x0204	Middle Mouse Button Up
KEY_LBUTTON_DBL	0x0401	Left Mouse Button Double Click
KEY_RBUTTON_DBL	0x0402	Right Mouse Button Double Click
KEY_MBUTTON_DBL	0x0403	Middle Mouse Button Double Click
KEY_LBUTTON_DN	0x0801	Left Mouse Button Down
KEY_RBUTTON_DN	0x0802	Right Mouse Button Down
KEY_MBUTTON_DN	0x0804	Middle Mouse Button Down
KEY_LF	0x000A	Line Feed
KEY_VT	0x000B	Vertical Tab
KEY_FF	0x000C	Form Feed
KEY_RETURN	0x000D	Return
KEY_ENTER	0x000D	Enter (same key as above)
KEY_ESCAPE	0x001B	Escape
KEY_ESC	0x001B	Escape (same key as above)
KEY_DELETE	0x012E	Delete
KEY_PGUP	0x0121	PageUp
KEY_PGDN	0x0122	PageDown
KEY_END	0x0123	End
KEY_HOME	0x0124	Home

Key Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
KEY_LEFT	0x0125	Cursor Left
KEY_UP	0x0126	Cursor Up
KEY_RIGHT	0x0127	Cursor Right
KEY_DOWN	0x0128	Cursor Down
KEY_INSERT	0x012D	Insert
KEY_HELP	0x012F	Help
KEY_F1	0x0170	F1
KEY_F2	0x0171	F2
KEY_F3	0x0172	F3
KEY_F4	0x0173	F4
KEY_F5	0x0174	F5
KEY_F6	0x0175	F6
KEY_F7	0x0176	F7
KEY_F8	0x0177	F8
KEY_F9	0x0178	F9
KEY_F10	0x0179	F10
KEY_F11	0x017A	F11
KEY_F12	0x017B	F12
KEY_F13	0x017C	F13
KEY_F14	0x017D	F14
KEY_F15	0x017E	F15
KEY_F16	0x017F	F16
KEY_F1_SHIFT	0x1170	Shift F1
KEY_F2_SHIFT	0x1171	Shift F2
KEY_F3_SHIFT	0x1172	Shift F3
KEY_F4_SHIFT	0x1173	Shift F4
KEY_F5_SHIFT	0x1174	Shift F5
KEY_F6_SHIFT	0x1175	Shift F6
KEY_F7_SHIFT	0x1176	Shift F7
KEY_F8_SHIFT	0x1177	Shift F8
KEY_F9_SHIFT	0x1178	Shift F9
KEY_F10_SHIFT	0x1179	Shift 10
KEY_F11_SHIFT	0x117A	Shift F11
KEY_F12_SHIFT	0x117B	Shift F12
KEY_F13_SHIFT	0x117C	Shift F13
KEY_F14_SHIFT	0x117D	Shift F14
KEY_F15_SHIFT	0x117E	Shift F15
KEY_F16_SHIFT	0x117F	Shift F16
KEY_F1_CTRL	0x2170	Ctrl F1
KEY_F2_CTRL	0x2171	Ctrl F2

Key Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
KEY_F3_CTRL	0x2172	Ctrl F3
KEY_F4_CTRL	0x2173	Ctrl F4
KEY_F5_CTRL	0x2174	Ctrl F5
KEY_F6_CTRL	0x2175	Ctrl F6
KEY_F7_CTRL	0x2176	Ctrl F7
KEY_F8_CTRL	0x2177	Ctrl F8
KEY_F9_CTRL	0x2178	Ctrl F9
KEY_F10_CTRL	0x2179	Ctrl F10
KEY_F11_CTRL	0x217A	Ctrl F11
KEY_F12_CTRL	0x217B	Ctrl F12
KEY_F13_CTRL	0x217C	Ctrl F13
KEY_F14_CTRL	0x217D	Ctrl F14
KEY_F15_CTRL	0x217E	Ctrl F15
KEY_F16_CTRL	0x217F	Ctrl F16
KEY_A_SHIFT	0x1041	Shift A
KEY_B_SHIFT	0x1042	Shift B
KEY_C_SHIFT	0x1043	Shift C
KEY_D_SHIFT	0x1044	Shift D
KEY_E_SHIFT	0x1045	Shift E
KEY_F_SHIFT	0x1046	Shift F
KEY_G_SHIFT	0x1047	Shift G
KEY_H_SHIFT	0x1048	Shift H
KEY_I_SHIFT	0x1049	Shift I
KEY_J_SHIFT	0x104A	Shift J
KEY_K_SHIFT	0x104B	Shift K
KEY_L_SHIFT	0x104C	Shift L
KEY_M_SHIFT	0x104D	Shift M
KEY_N_SHIFT	0x104E	Shift N
KEY_O_SHIFT	0x104F	Shift O
KEY_P_SHIFT	0x1050	Shift P
KEY_Q_SHIFT	0x1051	Shift Q
KEY_R_SHIFT	0x1052	Shift R
KEY_S_SHIFT	0x1053	Shift S
KEY_T_SHIFT	0x1054	Shift T
KEY_U_SHIFT	0x1055	Shift U
KEY_V_SHIFT	0x1056	Shift V
KEY_W_SHIFT	0x1057	Shift W
KEY_X_SHIFT	0x1058	Shift X
KEY_Y_SHIFT	0x1059	Shift Y
KEY_Z_SHIFT	0x105A	Shift Z

Key Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
KEY_A_CTRL	0x2041	Ctrl A
KEY_B_CTRL	0x2042	Ctrl B
KEY_C_CTRL	0x2043	Ctrl C
KEY_D_CTRL	0x2044	Ctrl D
KEY_E_CTRL	0x2045	Ctrl E
KEY_F_CTRL	0x2046	Ctrl F
KEY_G_CTRL	0x2047	Ctrl G
KEY_H_CTRL	0x2048	Ctrl H
KEY_I_CTRL	0x2049	Ctrl I
KEY_J_CTRL	0x204A	Ctrl J
KEY_K_CTRL	0x204B	Ctrl K
KEY_L_CTRL	0x204C	Ctrl L
KEY_M_CTRL	0x204D	Ctrl M
KEY_N_CTRL	0x204E	Ctrl N
KEY_O_CTRL	0x204F	Ctrl O
KEY_P_CTRL	0x2050	Ctrl P
KEY_Q_CTRL	0x2051	Ctrl Q
KEY_R_CTRL	0x2052	Ctrl R
KEY_S_CTRL	0x2053	Ctrl S
KEY_T_CTRL	0x2054	Ctrl T
KEY_U_CTRL	0x2055	Ctrl U
KEY_V_CTRL	0x2056	Ctrl V
KEY_W_CTRL	0x2057	Ctrl W
KEY_X_CTRL	0x2058	Ctrl X
KEY_Y_CTRL	0x2059	Ctrl Y
KEY_Z_CTRL	0x205A	Ctrl Z
KEY_A_ALT	0x4041	Alt A
KEY_B_ALT	0x4042	Alt B
KEY_C_ALT	0x4043	Alt C
KEY_D_ALT	0x4044	Alt D
KEY_E_ALT	0x4045	Alt E
KEY_F_ALT	0x4046	Alt F
KEY_G_ALT	0x4047	Alt G
KEY_H_ALT	0x4048	Alt H
KEY_I_ALT	0x4049	Alt I
KEY_J_ALT	0x404A	Alt J
KEY_K_ALT	0x404B	Alt K
KEY_L_ALT	0x404C	Alt L
KEY_M_ALT	0x404D	Alt M
KEY_N_ALT	0x404E	Alt N

Key Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
KEY_O_ALT	0x404F	Alt O
KEY_P_ALT	0x4050	Alt P
KEY_Q_ALT	0x4051	Alt Q
KEY_R_ALT	0x4052	Alt R
KEY_S_ALT	0x4053	Alt S
KEY_T_ALT	0x4054	Alt T
KEY_U_ALT	0x4055	Alt U
KEY_V_ALT	0x4056	Alt V
KEY_W_ALT	0x4057	Alt W
KEY_X_ALT	0x4058	Alt X
KEY_Y_ALT	0x4059	Alt Y
KEY_Z_ALT	0x405A	Alt Z

Note: To define a key with:

- The Shift key, add 0x1000 to the value of the key.
- The Ctrl key, add 0x2000 to the value of the key.
- The Alt key, add 0x4000 to the value of the key.

The above key definitions are standard IBM-compatible keys.

Note: Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade CitectSCADA.

Predefined Labels

The following labels are predefined in the Include Project.

Labels database

The table below defines the names, expressions predefined in the Include Project.

Name	Expr	Comment
@<edtsp.cii>	_ExecuteDTSPkg(sFile,s1,s2,s3 , s4,s5,s6,s7,s8,s9,s10,s11)	Execute DTS package Default value macro
@<obsvinv.cii>	_ObjectServerInvoke(sp,sf,s1,s2 , s3,s4,s5,s6,s7,s8,s9,s10)	Invoke Object Server Function
__DATE__	\$1	Date of compilation
__DB__	\$4	Compiler database name
__FIELD__	\$6	Compiler field name
__FILE__	\$2	Compiler file name
__LINE__	\$3	Compiler line number
__RECORD__	\$5	Compiler record number

Name	Expr	Comment
__TIME__	\$0	Time of compilation
BLANK		NULL Definition
AlarmDsp(hAn,count,type=1,scope=0)	_AlarmDsp(hAn,count,type,scope)	Display alarm list
AlarmDspLast(hAn,count=1,type=1)	_AlarmDsp(hAn,count,type,1)	Display last alarm.
AlarmFirstCatRec(hCat,nType,hArea=-1)	_AlarmQueryFirstRec(hCat,nType,hArea,0)	Get Alarm Cat Rec with Area
AlarmFirstPriRec(hPri,nType,hArea=-1)	_AlarmQueryFirstRec(hPri,nType,hArea,1)	Get Alarm Pri Rec with Area
AlarmGetFieldRec(nRID,sField,nVER=0)	_AlarmGetFieldRec(nRID,sField,nVER)	Alarm Get Field Record macro
AlarmNextCatRec(hRec,hCat,nType,hArea=-1)	_AlarmQueryNextRec(hRec,hCat,nType,hArea,0)	Get Alarm Cat Rec with Area
AlarmNextPriRec(hRec,hPri,nType,hArea=-1)	_AlarmQueryNextRec(hRec,hPri,nType,hArea,1)	Get Alarm Pri Rec with Area
AlarmSetQuery(hAN,sQF,sArg="")	_AlarmSetQuery(hAN,sQF,sArg)	Set user defined alarm filter function
ANIMATE	2	Display mode 2
ANM_ARRAY	16	Animated symbols in array mode
ANSI_CHARSET	0	ANSI character set
Arg1	GetGlbStr(0)	keyboard argument 1
Arg2	GetGlbStr(1)	keyboard argument 2
Arg3	GetGlbStr(2)	keyboard argument 3
Arg4	GetGlbStr(3)	keyboard argument 4
Arg5	GetGlbStr(4)	keyboard argument 5
Arg6	GetGlbStr(5)	keyboard argument 6
Arg7	GetGlbStr(6)	keyboard argument 7
Arg8	GetGlbStr(7)	keyboard argument 8
ArgValue1	StrToValue(Arg1)	Get the value of argument 1
Assert(arg)	IF NOT (arg) THEN _Assert(#arg, __FILE__, __LINE__); END	Process an assertion
BAD_HANDLE	-1	Bad Handle
BORDER	2	Border Only
BORDER_3D	1	3D Transparent Button
CreateControlObject(sCls,sName,x1,y1,x2,y2,sEventCls="")	_CreateControlObject(sCls,sName,x1,y1,x2,y2,sEventCls)	CreateControlObject default event class
DateDay(time)	_TimeSub(time,3)	Get days from time
DateDayMonth(time)	_TimeSub(time,10)	Get the last day of the month
DateMonth(time)	_TimeSub(time,5)	Get month from time
DateWeekDay(time)	_TimeSub(time,4)	Get weekday from time

Name	Expr	Comment
DDERead(sApp,sTopic,sItem,bAdvise=1)	_DDERead(sApp,sTopic,sItem,bAdvise)	DDE read with optional advise
DELETE_ANM	000	Delete animation
DevClose(hDev,iMode=0)	_DevClose(hDev,iMode)	Device Close, User Mode
DevFirst(hDev)	DevSeek(hDev,0)	DevSeek with Offset=0
DevOpen(sName,iMode=0)	_DevOpen(sName,iMode)	Device Open, Share Mode
DspAnCreateControlObject(hAn,sCls,w,h,sEventCls="")	_DspAnCreateControlObject(hAn,sCls,w,h,sEventCls)	DspAnCreateControlObject default event class
DspButton(hAn,UK=0,sText,hFont=0,nW=0,nH=0,DK=0,RK=0,nM=0)	_DspButton(hAn,UK,sText,hFont,nW,nH,DK,RK,nM)	Display button
DspButtonFn(hAn,UF=0,sText,hFont=0,nW=0,nH=0,DF=0,RF=0,nM=0)	_DspButtonFn(hAn,UF,sText,hFont,nW,nH,DF,RF,nM)	Display a button
DspChart(hAn,sTrn,v1,v2=0,v3=0,v4=0,v5=0,v6=0,v7=0,v8=0)	_DspChart(hAn,sTrn,v1,v2,v3,v4,v5,v6,v7,v8)	Trend display macro
DspGetAnFromPoint(x,y,hPrevAn=0)	_DspGetAnFromPoint(x,y,hPrevAn)	Allow getting all ANs at a point
DspSetTooltipFont(sName,nSize=12,sAttribs="")	_DspSetTooltipFont(sName,nSize,sAttribs)	Tool tip font macro
DspSym(hAn,sSym,mode=0)	_DspSym(hAn,sSym,mode)	Display symbol
DspSymAnm(hAn,s1,s2=0,s3=0,s4=0,s5=0,s6=0,s7=0,s8=0)	_DspSymAnm(hAn,s1,s2,s3,s4,s5,s6,s7,s8,0,"")	Display multi symbols
DspSymAnmEx(hAn,mode,s1,s2=0,s3=0,s4=0,s5=0,s6=0,s7=0,s8=0,s9=0)	_DspSymAnm(hAn,s1,s2,s3,s4,s5,s6,s7,s8,mode,s9)	DspSymAnm with mode
DspTrend(hAn,sTrn,v1=0,v2=0,v3=0,v4=0,v5=0,v6=0,v7=0,v8=0)	_DspTrend(hAn,sTrn,v1,v2,v3,v4,v5,v6,v7,v8)	Trend display macro
DspTrendInfo(sName,nType,hAn=-1)	_DspTrendInfo(sName,nType,hAn)	Get Trend Info, with AN
ErrGetHw(nDevice,nType=-1)	_ErrGetHw(nDevice,nType)	Get Hardware Error macro
ErrSetHw(nDevice,nError,nType=-1)	_ErrSetHw(nDevice,nError,nType)	Set Hardware Error macro
EVEN_P	2	Even Parity
Exec(sText,mode=1)	_Exec(sText,mode)	Exec program, default to normal
FALSE	0	Boolean False
FlashColourState()	StrToInt(PagelInfo(18))	Flashing Color State as a boolean
FormComboBox(Col,Row,Width,Height,sBuf,mode=0)	_FormComboBox(Col,Row,Width,Height,sBuf,mode)	Form Combo box with mode

Name	Expr	Comment
FormGroupBox(Col,Row,Width,Height,sText="")	_FormGroupBox (Col,Row,Width,Height,sText)	Group box with text
FormListBox(Col,Row,Width,Height,sBuf,mode=0)	_FormListBox (Col,Row,Width,Height,sBuf,mode)	Form List box with mode
FormSaveAsFile(sTitle,sDefault,sFilters,sDefExt="")	_FormSaveAsFile (sTitle,sDefault,sFilters,sDefExt)	
GetBlueValue(PackedRGB)	((PackedRGB / 65536) BITAND 255)	Get the blue component of a packed RGB color
GetGreenValue(PackedRGB)	((PackedRGB / 256) BITAND 255)	Get the green component of a packed RGB color
GetRedValue(PackedRGB)	(PackedRGB BITAND 255)	Get the red component of a packed RGB color
GetVar(sTag,sField)	\$7	Get variable field data
GetVarDef(sTag,sField,sDefault)	\$10	Get variable field data if defined
GetVarStr(sTag,sField)	\$8	Get variable field data as str
GetVarStrDef(sTag,sField,sDefault)	\$11	Get variable field data as a str if defined
GRAY_ALL	3	Gray the entire button
GRAY_HIDE	4	Hide object when grayed
GRAY_PART	2	Sink and gray the text / symbol
GRAY_SUNK	1	Sink the text / symbol
IFDEF(sTag,sTrue,sFalse)	\$9	Inline IF defined macro
InAnimationCycle()	StrToInt(PageInfo(19))	In Animation Cycle as a boolean
InCommunicationsCycle()	StrToInt(PageInfo(20))	In Communications Cycle as a boolean
KeyDown()	KeyMove(4)	Move Cursor down
KeyLeft()	KeyMove(1)	Move Cursor left
KeyReplay()	_KeyReplay(1)	Key Replay - last key
KeyReplayAll()	_KeyReplay(0)	Key Replay All
KeyRight()	KeyMove(2)	Move Cursor right
KeyUp()	KeyMove(3)	Move Cursor up
NONE	0	No Parity
NORMAL	0	Normal Button
ObjectAssociatePropertyWithTag (Obj,sPName,sTName,sEvName="")	_ObjectAssociatePropertyWithTag (Obj,sPName,sTName,sEvName)	ObjectAssociatePropertyWithTag default event
ODD_P	1	Odd Parity
OVERLAP	1	Display mode 1
PackedRGB(Red,Green,Blue)	(Red + Green * 256 + Blue * 65536)	Make a packed RGB color from its components

Name	Expr	Comment
PlotInfo(hPlot,nType,sInput="")	_PlotInfo(hPlot,nType,sInput)	Get information about a plot system
Print(sText,nMode=0)	DevPrint(DevCurr(),sText,nMode)	Print output to device
PrintLn(sText)	DevPrint(DevCurr(),sText,1)	Print output to device, newline
Pulse(arg)	arg = TRUE; Sleep(2); arg = FALSE;	Pulse the variable
RAboveUCL	8192	
RBelowLCL	16384	
ROutsideCL	4096	
Shutdown(sDest="",sProject="",nMode=1)	_Shutdown(sDest,sProject,nMode)	Shutdown macro
SOFT	0	Display mode 0
TableMath(Table, Size, Command, mode=0)	_TableMath(Table, Size, Command,mode)	mathematical operations on a tab
TARGET	3	Screen Target
TaskHnd(sName="")	_TaskHnd(sName)	Get Task handle
TestRandomWave(p=60,lo=0,hi=100,off=0)	_Wave(4,p,lo,hi,off)	Test random wave
TestSawWave(p=60,lo=0,hi=100,off=0)	_Wave(3,p,lo,hi,off)	Test Saw wave
TestSinWave(p=60,lo=0,hi=100,off=0)	_Wave(0,p,lo,hi,off)	Test sin wave
TestSquareWave(p=60,lo=0,hi=100,off=0)	_Wave(1,p,lo,hi,off)	Test square wave
TestTriangWave(p=60,lo=0,hi=100,off=0)	_Wave(2,p,lo,hi,off)	Test Triang wave
TimeHour(time)	_TimeSub(time,0)	Get hours from time
TimeMidNight(time)	_TimeSub(time, 7)	Extract time at midnight
TimeMin(time)	_TimeSub(time,1)	Get minutes from time
TimeSec(time)	_TimeSub(time,2)	Get seconds from time
TimeSecond(time)	_TimeSub(time, 2)	Get seconds from time
TimeYearDay(time)	_TimeSub(time, 8)	
Toggle(arg)	arg = NOT arg;	Toggle the variable
TRN_EVENT	2	Event trend
TRN_PERIODIC	1	Periodic trend
TRN_PERIODIC_EVENT	3	Periodic Event trend
TrnGetTable(Name,time,period,len,Buf, mode,msec=0)	_TrnGetTable (Name,time,period,len, Buf,mode,msec)	
TrnNew(hAn,sTrn,s1="",s2="",s3="",s4="",s5="",s6="",s7="",s8="")	_TrnNew(hAn,sTrn,s1, s2,s3,s4,s5,s6,s7,s8)	Trend new macro

Name	Expr	Comment
TrnScroll(hAn,hPen,nScroll,nMode=1)	_TrnScroll (hAn,hPen, nScroll,nMode)	
TrnSetTable(Name,time,period,len,Buf, msec=0)	_TrnSetTable (Name,time,period,len,Buf,msec)	
TRUE	1	Boolean True
UnitControl(IODev,Type,Data)	IODeviceControl(IODev,Type,Data)	
UnitInfo(IODev,Type)	IODeviceInfo(IODev,Type)	
UnitStats()	IODeviceStats()	
UserCreate(s1,s2,s3,s4,s5="",pG="", p1="",p2="",p3="",p4="",p5="",p6="",p7 ="",p8="")	_UserCreate(s1,s2,s3,s4,s5, pG,p1,p2,p3,p4,p5,p6,p7,p8)	Create a new user with privileges
UserPassword(sUser,sNewPassword,s OldPassword="")	_UserPassword(sUser, sNewPassword,sOldPassword)	Set user password
UserPasswordExpiryDays(sUser,sPass word="")	_UserPasswordExpiryDays (sUser,sPassword)	Get user password expiry days
WinCopy(cx=1,cy=1,sPal="")	_WinCopy(cx,cy,sPal)	Print macro
WinFile(sFile,cx=1,cy=1,sPal="")	_WinFile(sFile,cx,cy,sPal)	Print file macro
WinPrint(sPort,cx=0,cy=0,sPal="[run]:pr inter.pal")	_WinPrint(sPort,cx,cy,sPal)	Print macro
WinPrintFile(sFile,sPort,cx=0,cy=0,sPal ="[run]:printer.pal")	_WinPrintFile(sFile,sPort, cx,cy,sPal)	Print file macro
WRITE_ON_DRAG	1	Write mode for slider
WRITE_ON_DROP	0	Write mode for slider
XAboveUCL	4	
XBelowLCL	8	
XDownTrend	64	
XErratic	512	
XFreak	1	
XGradualDown	256	
XGradualUp	128	
XMixture	2048	
XOutsideCL	2	
XOutsideWL	16	
XStratification	1024	
XUpTrend	32	

Note: Do not modify the Include Project. Changes to the Include project are lost when you reinstall or upgrade CitectSCADA.

ASCII/ANSI Character Code Listings

The code table shows the Latin 1 ANSI character set. Codes 0–31 are control codes. The standard ASCII codes are from 32–127 (decimal) and are common regardless of the ANSI set used. The remaining codes from 160–255 (decimal) vary between languages depending upon the ANSI set used.

Symbol	Decimal	Hex
{NUL}	0	00
{SOH}	1	01
{STX}	2	02
{ETX}	3	03
{EOT}	4	04
{ENQ}	5	05
{ACK}	6	06
{BEL}	7	07
{BS}	8	08
{HT}	9	09
{LF}	10	0A
{VT}	11	0B
{FF}	12	0C
{CR}	13	0D
{SO}	14	0E
{SI}	15	0F
{DLE}	16	10
{DC1}	17	11
{DC2}	18	12
{DC3}	19	13
{DC4}	20	14
{NAK}	21	15
{SYN}	22	16
{ETB}	23	17
{CAN}	24	18
{EM}	25	19
{SUB}	26	1A
{ESC}	27	1B
{FS}	28	1C
{GS}	29	1D
{RS}	30	1E
{US}	31	1F
{SPC}	32	20
!	33	21
"	34	22
#	35	23

Symbol	Decimal	Hex
\$	36	24
%	37	25
&	38	26
'39	39	27
(40	28
)	41	29
*	42	2A
+	43	2B
,	44	2C
-	45	2D
.	46	2E
/	47	2F
0	48	30
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35
6	54	36
7	55	37
8	56	38
9	57	39
:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F
@	64	40
A	65	41
B	66	42
C	67	43
D	68	44
E	69	45
F	70	46
G	71	47
H	72	48
I	73	49
J	74	4A
K	75	4B

Symbol	Decimal	Hex
L	76	4C
M	77	4D
N	78	4E
O	79	4F
P	80	50
Q	81	51
R	82	52
S	83	53
T	84	54
U	85	55
V	86	56
W	87	57
X	88	58
Y	89	59
Z	90	5A
[91	5B
\	92	5C
]	93	5D
^	94	5E
_	95	5F
	96	60
a	97	61
b	98	62
c	99	63
d	100	64
e	101	65
f	102	66
g	103	67
h	104	68
i	105	69
j	106	6A
k	107	6B
l	108	6C
m	109	6D
n	110	6E
o	111	6F
p	112	70
q	113	71
r	114	72
s	115	73

Symbol	Decimal	Hex
t	116	74
u	117	75
v	118	76
w	119	77
x	120	78
y	121	79
z	122	7A
{	123	7B
	124	7C
}	125	7D
~	126	7E
{Delete}	127	7F
	128	80
	129	81
,	130	82
f	131	83
”	132	84
...	133	85
†	134	86
‡	135	87
^	136	88
‰	137	89
Š	138	8A
‹	139	8B
Œ	140	8C
	141	8D
	142	8E
	143	8F
	144	90
‘	145	91
’	146	92
	"147	93
	"148	94
•	149	95
—	150	96
—	151	97
~	152	98
™	153	99
š	154	9A
›	155	9B

Symbol	Decimal	Hex
œ	156	9C
	157	9D
	158	9E
ÿ	159	9F
{NBSP}	160	A0
ı	161	A1
¢	162	A2
£	163	A3
¤	164	A4
¥	165	A5
¦	166	A6
§	167	A7
¨	168	A8
©	169	A9
ª	170	AA
«	171	AB
172	AC	
-	173	AD
®	174	AE
¯	175	AF
°	176	B0
±	177	B1
²	178	B2
³	179	B3
´	180	B4
µ	181	B5
182	B6	
•	183	B7
¸	184	B8
¹	185	B9
º	186	BA
»	187	BB
¼	188	BC
½	189	BD
¾	190	BE
¿	191	BF
À	192	C0
Á	193	C1
Â	194	C2
Ã	195	C3

Symbol	Decimal	Hex
Ä	196	C4
Å	197	C5
Æ	198	C6
Ç	199	C7
È	200	C8
É	201	C9
Ê	202	CA
Ë	203	CB
Ì	204	CC
Í	205	CD
Î	206	CE
Ï	207	CF
Ð	208	D0
Ñ	209	D1
Ò	210	D2
Ó	211	D3
Ô	212	D4
Õ	213	D5
Ö	214	D6
×	215	D7
Ø	216	D8
Ù	217	D9
Ú	218	DA
Û	219	DB
Ü	220	DC
Ý	221	DD
Þ	222	DE
ß	223	DF
à	224	E0
á	225	E1
â	226	E2
ã	227	E3
ä	228	E4
å	229	E5
æ	230	E6
ç	231	E7
è	232	E8
é	233	E9
ê	234	EA
ë	235	EB

Symbol	Decimal	Hex
ì	236	EC
í	237	ED
î	238	EE
ï	239	EF
ð	240	F0
ñ	241	F1
ò	242	F2
ó	243	F3
ô	244	F4
õ	245	F5
ö	246	F6
÷	247	F7
ø	248	F8
ù	249	F9
ú	250	FA
û	251	FB
ü	252	FC
ý	253	FD
þ	254	FE
ÿ	255	FF

Format Fields

This section contains information about:

- [Alarm Display Fields](#)
- [Alarm Summary Fields](#)
- [Command Fields](#).

Alarm Display Fields

You can use any of the following fields (or combination of fields) to format an Alarm Display (see [Alarm Categories](#)) and an Alarm Log Device (see Formatting a Device):

Field Name	Description
{Tag,n}	Alarm Tag
{Name,n}	Alarm Name NOTE: If the Tag and Name fields are configured to support long names (up to 79 characters), it may cause overlap in an alarm display. It is recommended you use a smaller display font if long names are expected.
{Native_Name,n}	Alarm Name in the native language
{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: DISABLED ACKNOWLEDGED 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.
You can use the following fields for Digital Alarms only :	
{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**:

{Millisec,n}	Adds milliseconds to the {Time,n} field
--------------	---

You can use any of the following fields for **Analog Alarms only**:

{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

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. DEVIATION RATE LOW LOWLOW HIGH HIGHHIGHCLEARED
You can use any of the following fields for Hardware Alarms (Category 255) only :	
{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 :	
{LogState,n}	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 :	
{State_desc, n}	The configured description (e.g. healthy or stopped) of a particular state. This description is entered when configuring the Multi-Digital Alarm
where n specifies the display field size.	Properties.

Note the following:

- 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. See [General]FormatCheck parameter for details.
- You should 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.
- You should use only the fields above to format an Alarm Display or Alarm Log Device. Using alarm summary fields may produce unreliable results.

See Also [Alarm Summary Fields](#)

Alarm Summary Fields

You can use any of the fields listed below (or a combination of fields) 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.

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,n}	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,n}	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,n}	The text related to the user event
{OnDate,n}	The date when alarm was activated
{OnDateExt,n}	The date (in extended format) when the alarm was activated (dd/mm/yyyy)
{OffDate,n}	The date when the alarm returned to its normal state
{OffDateExt,n}	The date (in extended format) when the alarm returned to its normal state (dd/mm/yyyy)
{OnTime,n}	The time when the alarm was activated
{OffTime,n}	The time when the alarm returned to its normal state
{DeltaTime,n}	The time difference between OnDate/OnTime and OffDate/OffTime, in seconds
{OnMilli,n}	Adds milliseconds to the time the alarm was activated.
{OffMilli,n}	Adds milliseconds to the time the alarm returned to its normal state.
{AckTime,n}	The time when the alarm was acknowledged
{AckDate,n}	The date when the alarm was acknowledged
{AckDateExt,n}	The date (in extended format) when the alarm was acknowledged (dd/mm/yyyy)
{SumState,n}	Describes the state of the alarm when it occurred
{SumDesc,n}	A description of the alarm summary
{Native_SumDesc,n}	A description of the alarm summary, in the native language
{AlmComment,n}	The text entered into the Comment field of the alarm properties dialog.
{Comment,n}	A comment the operator adds to an Alarm Summary entry during runtime. The comment is specified using the AlarmComment() function.
{Native_Comment,n} where n specifies the display field size.	Native language comments the operator adds to an Alarm Summary entry during runtime.

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 may produce unreliable results.

See Also [Alarm Display Fields](#)

Command Fields

You can use any of the following fields (or combination of fields) 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 Message Log 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 native language version of the message sent as the Message Log 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 Henry Chang".

Error Messages

This section contains information about CitectSCADA-related errors.

- [Protocol Generic Errors](#)
- [Protocol-Specific Errors](#)

See Also [Cicode Errors](#)

Protocol Generic Errors

CitectSCADA has two kinds of protocol driver errors: generic and [Protocol-Specific Errors](#). Generic errors are hardware errors 0-31, and are common to all protocols.

Protocol drivers also have their own specific errors, which can be unique and therefore cannot be recognized by the hardware alarm system. The drivers convert their specific errors into generic errors that can be identified by the I/O Server. For example, when a driver has a fault, there is often both a protocol-specific error and a corresponding generic error.

Generic Errors

The table below describes the generic protocol errors.

Error number	Error title	Description
1	Address is out of range	A request was made to a device address that does not exist. For example, you tried to read register number 4000 when there are only 200 registers in the I/O device. Check the Variable Tags database to find the variable in error.
2	Command cancelled	The server cancelled the command while it was being processed by the driver. The driver may have taken too long to process the command. If a driver fails to respond during the specified time limit, CitectSCADA cancels the command. The time limit is the product of the timeout period and the number of times to retry a command after each timeout. You can increase these values in the Timeout and Retry parameters for the protocol. You should also check the WatchTime parameter for the frequency with which the driver checks the link to the I/O device. Check also for communication errors.
3	Unknown data type	A request was made that specified a data type not supported by the protocol. This error should not occur during normal operation. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support. If you have written your own protocol driver, this error is caused by a mismatch in the compiler specification and the driver's database.
4	Unknown data format	A write request contains invalid data, e.g. you tried to write to a floating-point address with an invalid floating-point number. Check the CitectSCADA database.
5	Command is unknown	The server sent a command that the driver did not recognize. This error should not occur during normal operation. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support.

Error number	Error title	Description
6	Response bad or garbled	A problem exists with the communication channel, causing errors in the transmitted data. Inspect the setup for the communication channel hardware. For example, there may be a mismatch in parity, baud rate, stop bits, or data bits between the transmitter and receiver. Check that the setup of the I/O device and the field data in the CitectSCADA Ports and Boards forms are the same.
7	I/O device not responding	An I/O device is not responding to read or write requests. The driver sent a command to the I/O device and the I/O device did not respond within the timeout period. This is usually the first indication of a communication failure. Check that the I/O device is correctly connected to the server and is switched on. This error can also occur if the timeout period is too short. Try increasing the timeout period in the Timeout parameter for the protocol. You could also increase the delay time between receiving a response and sending the next command, by increasing the Delay parameter.
8	General error	CitectSCADA has established communications with the I/O device; however, the I/O device has detected an error in the protocol. This error could be caused by a fault in the communications link , or an error in the ladder logic (in the I/O device).
		Solution: <ol style="list-style-type: none"> 1. Check that the I/O device is operating correctly. 2. Check the communication cable is connected correctly (at both ends). 3. Use the Communications Express Wizard to check that the configuration of the I/O device (in particular, the Address and Special Options fields) matches the recommended settings and the settings on the I/O device. 4. If you are using serial communications, use the Communications Express Wizard to check that the configuration of the Port (in particular the Baud Rate, Data Bits, Stop Bits, and Parity) matches the recommended settings and the settings on the I/O device. 5. Display the hardware alarm page, and note the protocol error that is displayed. 6. Use the documentation that was supplied with your I/O device, network, and communication board to locate the error. 7. Check the ladder logic in the I/O device for errors. 8. Run the Computer Setup Wizard. 9. Re-compile the project and start the CitectSCADA runtime.
9	Write location is protected	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.

Error number	Error title	Description
10	Hardware error	A problem exists with either the communication channel, server, or I/O device hardware. Examine all hardware components. The command or data request has not been processed. The server's operation may no longer be reliable.
11	I/O device warning	The communication link between the server and the I/O device is functioning correctly, however the I/O device has some warning condition active, e.g. the I/O device is in program mode. Check that the I/O device is in the correct mode.
12	I/O device off-line, cannot talk	The I/O device is in off-line mode, preventing any external communication.

Solution:

1. Check that the I/O device is operating correctly.
2. Check the communication cable for breakage.
3. Check the communication cable is connected correctly (at both ends).
4. If you are using serial communications, check that the communication cable matches the diagram in the help system.
5. Use the Communications Express Wizard to check that the configuration of the I/O device (in particular, the Address and Special Options fields) matches the recommended settings and the settings on the I/O device.
6. If you are using serial communications, use the Communications Express Wizard to check that the configuration of the port (in particular the baud rate, data bits, stop bits, and parity) matches the recommended settings and the settings on the I/O device.
7. Run the Computer Setup Wizard.
8. Check the citect.ini file (in the Windows directory) for the following:

```
[IOSERVER]
Server=1
Name=<name>
```

where:

<name> is the name of the server configured in the CitectSCADA project. (Use Custom Setup to check the server name.)

9. Re-compile the project and start the CitectSCADA runtime system.

NOTE: If you have standby I/O devices configured, this error will cause any standby I/O devices to become active. The command or data request current when the I/O device went off-line has not been completed.

Error number	Error title	Description
13	Driver software error	An internal software error has occurred in the driver. This error should not occur during normal operation. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support.
14	User access violation	An attempt has been made by an unauthorized user to access information. Check the user's access rights.
15	Out of memory - FATAL	The server is out of memory and cannot continue execution. Minimize buffer and queue allocation or expand memory in the server computer. The command or data request has not been processed.
16	No buffers, cannot continue	There are no communication buffers available to be allocated, or the computer is out of memory. The performance of the server may be reduced, however it can continue to run. Increase the memory.
17	Low buffer warning	This error may occasionally occur in periods of high transient loading, with no ill effects. If this error occurs frequently, increase the number of communication buffers.
18	Too many commands to driver	Too many commands have been sent to the driver. If you are using a NetBIOS driver, increase the number of NetBIOS control blocks (NCBs).
19	Driver is not responding	The server is not receiving any response from the driver. This error should not occur during normal operation. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support.
20	Too many channels opened	Each driver can only support several communication channels. You have exceeded the limit. This error may occur if you abnormally terminate from the server and then restart it. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support. The command or data request has not been completed.
21	Channel off-line, cannot talk	A communication channel is currently off-line, disabling communication. Either the server cannot initialize the communication channel or the channel went off-line while running. Check the channel hardware for failure. When this error occurs, all I/O devices connected to this channel are considered off-line, and standby I/O devices become active. The command or data request has not been completed.
22	Channel not yet opened	The server has attempted to communicate with a channel that is not open. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support. The command or data request has not been completed.

Error number	Error title	Description
23	Channel not yet initialized	The server is attempting to communicate with a channel that has not been initialized. This error should not occur during normal operation. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support. The command or data request has not been completed.
24	Too many I/O devices per channel	A channel has too many I/O devices attached to it. This error should not occur during normal operation. The command or data request has not been completed. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support.
25	Data not yet valid	The data requested is still being processed and will be returned in due course. This error only occurs with drivers that must establish complex communication to retrieve data from the I/O device. Ignore this warning.
26	Could not cancel command	The server tried to cancel a command, but the driver could not find the command. This error should not occur during normal operation. Try re-booting the computer to reset all drivers and hardware. If the problem persists, contact Citect Support.
27	Stand-by I/O device activated	Communication has been switched from the primary to the standby I/O device(s). The server returns this message when a "hot" changeover has occurred. Rectify the fault in the primary I/O device(s).
28	Message overrun	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.
29	Bad user parameters	There is a configuration error, e.g. invalid special options have been set.
30	Stand-by I/O device error	There is an error in a standby I/O device. Rectify the fault in the standby I/O device.
31	Request Timeout from I/O Server	One or more requests sent to the I/O server have not been completed in the timeout period. Either the I/O Server is off line or the I/O Server is taking too long to complete the requests. Check the PLC communication link, PLC timeouts, PLC retries, and network communication. This error can occur even if you have no network, i.e. if the I/O Server is the same computer as the display client . If the error persists, increase the [LAN] TimeOut parameter. The default timeout is 8000 milliseconds.

Protocol-Specific Errors

Though each protocol may have multiple unique errors, the first 34 protocol-specific errors are standard for all protocols. All protocol-specific errors are also reported under error numbers 1 to 31 above. Although these errors have their own error number (also given in hexadecimal), it is only used as a notation.

Note: Errors that are protocol-specific are listed in the Protocol-Specific Errors help topic for each protocol. Refer to the documentation that was supplied with your I/O device if you cannot locate an error description.

Error number	Error title	Description
1 (0x01)	Cannot process received characters fast enough	Cannot process received characters fast enough. Lower the baud rate or use a faster computer. If the error persists, contact Citect Support.
2 (0x02)	Parity error	The received message has a parity error. Check that the correct baud rate, parity, stop bits, and data bits are specified in the Citect Ports form. This error may be caused by a faulty cable connection to the I/O device or by excessive noise on the communication link.
3 (0x03)	Break detected in receive line	A break has been detected in the receive line. This error may be caused by a faulty cable connection to an I/O device or by excessive noise on the communication link.
4 (0x04)	Framing error	The wrong baud rate may have been specified. Check that the correct baud rate is specified in the Citect Ports form.
5 (0x05)	Message too long	The message received from the I/O device is too long. This error may be caused by a faulty cable connection to an I/O device or by excessive noise on the communication link. Contact Citect Support if the error continues.
6 (0x06)	Invalid checksum	The checksum in the received message does not match the calculated value. Check that the correct baud rate, parity, stop bits, and data bits are specified in the Citect Ports form. This error may be caused by a faulty cable connection to the I/O device or by excessive noise on the communication link. You can also try increasing the number of retries in the Retry parameter for the protocol.
7 (0x07)	Start of text missing	A start of text (STX) character is not present in the received message. Check that the correct baud rate, parity, stop bits, and data bits are specified in the Citect Ports form. This error may be caused by a faulty cable connection to the I/O device or by excessive noise on the communication link.
8 (0x08)	End of text missing	An end of text (ETX) character is not present in the received message. Check that the correct baud rate, parity, stop bits, and data bits are specified in the Citect Ports form. This error may be caused by a faulty cable connection to the I/O device or by excessive noise on the communication link.
10 (0x0A)	Cannot transmit message	CitectSCADA cannot transmit the message. This error may be caused by a faulty cable connection to an I/O device or by excessive noise on the communication link.
11 (0x0B)	Cannot reset serial driver	A fault has occurred with the serial (COMX1, PCXI, or COMX) driver. Try re-booting the computer to reset all drivers and hardware.
15 (0x0F)	Command from server invalid	The command from the server is invalid. Contact Citect Support.

Error number	Error title	Description
16 (0x10)	Cannot allocate timer resource for driver	Driver timer resources cannot be allocated. Contact Citect Support.
17 (0x11)	Too many channels specified for driver	Too many channels have been specified for the device. Contact Citect Support.
18 (0x12)	Channel number from server not opened	The channel number from the server is not open. Contact Citect Support.
19 (0x13)	Command cannot be cancelled	A driver command cannot be cancelled. Contact Citect Support.
20 (0x14)	Channel not on-line	The channel is not on-line. This error can occur if timeouts are occurring, and may be caused by a faulty cable connection to an I/O device or by excessive noise on the communication link.
21 (0x15)	Timeout error	No response was received from the I/O device within the specified timeout period. This error may be caused by a faulty cable connection to the I/O device or by excessive noise on the communication link. You can try increasing the number of retries in the Retry parameter for the protocol.
22 (0x16)	I/O device number from server not active or out of range	The I/O device number from the server is not active or is out of range. Contact Citect Support.
23 (0x17)	I/O device not on-line	Check that the I/O device Address specified in the Citect I/O Devices form is the same as that configured on the I/O device.
24 (0x18)	Data type from server unknown to driver	The data type from the server is unknown to the driver. Contact Citect Support.
25 (0x19)	I/O device type from server unknown to driver	The I/O device type from the server is unknown to the driver. Contact Citect Support.
26 (0x1A)	Too many I/O devices specified for channel	Too many I/O devices have been specified for the channel. Contact Citect Support.
27 (0x1B)	Too many commands issued to driver	Too many commands have been issued to the driver. Contact Citect Support.
28 (0x1C)	Data read invalid	The data read is not valid. Contact Citect Support.
29 (0x1D)	Command is cancelled	A driver command has been cancelled. Contact Citect Support.
30 (0x1E)	Address invalid or out of range	The address you tried to access has an invalid data type or is out of range. Check that you are using data types and ranges of addresses that are valid for the I/O device.
31 (0x1F)	Data length from server incorrect	The data length from the server is wrong. Contact Citect Support.
32 (0x20)	Cannot read data from device	CitectSCADA cannot read the data from the I/O device. Contact Citect Support.
33 (0x21)	Device specified does not exist	The device specified does not exist. Contact Citect Support.

Error number	Error title	Description
34 (0x22)	Device specified does not support interrupt	The I/O device specified does not support interrupt handling. You have specified an interrupt, either on the Boards form or by setting the PollTime parameter to 0, for a hardware device that does not support interrupts. Check the interrupt set for the board and set the PollTime parameter for the protocol.

NetBIOS Errors

The table below describes the NetBIOS errors.

Error number	Error title	Description
1024	No NetBIOS error	This error should not occur in normal operation. Contact Citect Support.
1025	Invalid buffer length	This error should not occur in normal operation. Contact Citect Support.
1027	Invalid command	This error should not occur in normal operation. Contact Citect Support.
1029	Command timed out	CitectSCADA is timing out when sending data on the network. If this error occurs frequently, increase the timeout period in the [LAN] SendTimeout parameter. This error is likely to occur if you are running CitectSCADA on a slow network or a Wide Area Network.
1030	Incomplete receive message	This error should not occur in normal operation. Contact Citect Support.
1032	Invalid session number	This error should not occur in normal operation. Contact Citect Support.
1033	No resource available	Increase network resources or memory. Increase the Windows parameter NetHeapSize in the system.ini file (or other network parameters).
1034	Session has been closed	This error should not occur in normal operation. Contact Citect Support.
1035	Command cancelled	This error should not occur in normal operation. Contact Citect Support.
1037	Duplicate name in local table	This error should not occur in normal operation. Contact Citect Support.
1038	NetBIOS name table full	Increase the number of names in the local name table setup in the network NetBIOS configuration.
1041	NetBIOS session table full	CitectSCADA has run out of NetBIOS sessions. Increase the number of NetBIOS sessions in the network setup.
1044	Server name not found	The specified server cannot be found on the network. Either the server has not started or a network problem is preventing communication.

Error number	Error title	Description
1046	Name in use on remote adaptor	Two CitectSCADA servers on the network are trying to use the same name. Configure each CitectSCADA server with a unique name.
1049	Name conflict	Two CitectSCADA servers on the network are trying to use the same name. Configure each CitectSCADA server with a unique name.
1058	Too many commands outstanding	CitectSCADA has run out of NetBIOS control blocks (NCBs). Increase the number of NCBs in the network NetBIOS configuration or reduce CitectSCADA's use of NCBs in the citect.ini file.

Glossary

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.

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.

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.

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.

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.

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.

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.

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.

file server A computer with a large data storage capacity (magnetic disk, laser disk, magnetic tapes, and so on) that is totally dedicated to a local area network (LAN). The file server stores common data and distributes it over the network as required.

full duplex Simultaneous two-way (in both directions) independent transmission (4 Wires).

generic protocol A pseudo-protocol supported by memory I/O devices and disk I/O devices that provides a convenient way to represent memory and 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.

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.

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 digital alarms Alarms that 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).

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.

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 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.

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.

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: $2048/16 = 128$.

memory I/O device A utility that resides in the memory of a computer and emulates a real I/O device. The value of each variable in the memory I/O device is stored in the computer's memory; however the memory I/O device is not connected to any field equipment in the plant.

millisecond trending Allows you to use a trends sample period of less than one second.

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.

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.

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).

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.

proxy/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.

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.

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.

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.

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 such as NT, 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 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.

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.

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.

wizard A CitectSCADA facility that simplifies an otherwise complex procedure by presenting the procedure as a series of simple steps.

Index

Numerics

- 10base2, defined, 673
- 10base5, defined, 673
- 10baseT, defined, 673
- 9-to-25-pin converter, 512

A

- access
 - disabling, 191
 - object, 188
- access properties, 189
- access property, object, 288
- access table, reading data from, 441
- accredited - level 1, defined, 673
- accredited - level 2, defined, 673
- accumulator, defined, 673
- accumulators, 259
 - configuring, 259
- action queries, calling, 443
- active alarm, defined, 673
- ActiveX objects, 137
 - identifying, 141
 - managing associated data sources, 137
- adding, user records, 374
- advanced alarms, 229, 673
- alarm categories, defined, 673
- alarm display fields, 241
- alarm display page, defined, 673
- alarm filters, 204, 676
- alarm summary fields, 243
- alarm summary page, defined, 673
- alarms, 203
 - advanced, 229, 673
 - advanced alarm properties, 229
 - alarm category properties, 208
 - analog, 224, 673
 - analog alarm properties, 224
 - categories of, 207
 - configured, 203
 - creating pages for, 249
 - custom filters, 204
 - debugging hardware, 575
 - delay, 204
 - digital, 212
 - digital alarm properties, 212
 - display order, 245
 - displaying variable data in, 240
 - filters, 204
 - formatting display, 239
 - multi-digital, 215
 - multi-digital alarm properties, 216
 - pages, displaying, 249
 - properties supported as tags, 245
 - redundancy, 621
 - runtime, 248
 - setting up, 248
 - time-stamped, 220
 - time-stamped alarm properties, 221
 - time-stamped analog, 234
 - time-stamped analog alarm properties, 235
 - time-stamped digital, 231
 - time-stamped digital alarm properties, 231
 - using properties as tags, 245
 - writing to properties of, 247
- alarms server redundancy, 622
- alarms server, defined, 673
- Align dialog box, 296
- aligning objects, 296, 342, 343
- alternative .ini file, 574
- analog alarms, 224, 673
- animation number files, defined, 674
- Animation Point (AN), 299, 346, 635, 674
- ANSI character codes, 653
- ANSI character sets, 471
- appearance
 - graphics page, 336
 - of objects, 282
- appending data with ODBC, 442
- archiving
 - data, 408
 - projects, 25
- area prefix in tag names, 51
- areas, 288, 674
 - and security, 380
 - defining, 378
 - groups, 381, 382

- labels for, 380
- privileges, 383, 384
- Super Genies and, 462
- viewing, 384
- arguments
 - defined, 674
 - in labels, 389
 - values for, 389
- arrangements, pipe, 110
- array size, 49
- arrays, 47
 - Super Genies and, 459
- ASCII character codes, 653
- ASCII character sets, 471
- ASCII devices, 393
 - format, 401
- assigning variable tags, 39
- Attach Super Genie dialog box, 461
- attachment unit interface (AUI), defined, 674
- attribute section in tag names, 52
- attributes for tag names, 52
- automation component, 674

B

- Backup Project dialog box, 26
- backup utility, 29
- Backup/Restore Password Encryption dialog box, 27
- backups, project, 25
- baud rate, defined, 674
- BCD variable, 674
- bitmaps, 281, 358
- boards
 - proprietary, 500
 - serial, 498
- bottleneck, defined, 674
- breaking link to data source, 55
- Bring Forwards command, 295
- Bring to Front command, 295
- browse sequence
 - defined, 674
 - pages, 330
- buttons (objects), 120
- byte variable, defined, 674

C

- cables, wiring, 512
- cache
 - data, 488
 - defined, 674
 - tuning, 598
- calculating
 - array size, 49
 - disk storage (trends), 275
- callback functions, 586, 675
- calling action queries, 443
- capability charts, 305
- capability, process, 303
- categories, alarm, 207
- center limit, 304
- changing
 - alarm summary display, 245
 - languages, 467
- character codes, 653
- character sets, 471
 - predefined, 639
- charts
 - capability, 305
 - control, 304
- Cicode blocking function, defined, 675
- Cicode blocks in reports, 369
- Cicode Editor, 7
- Cicode files, predefined, 642
- Cicode objects, 132
- Cicode, defined, 675
- CiNet, 628
- circles, drawing, 102
- Citect driver accreditation, 518
- Citect Explorer, 4
- Citect Internet Client Setup dialog box, 572
- Citect support, 584
- citect.ini file, defined, 675
- CitectSCADA Bitmap Editor, 358
- CitectSCADA computer, defined, 675
- CitectSCADA project, defined, 675
- CitectSCADA server, defined, 675
- CiUSAFE dialog box, 589
- client syntax and DDE, 412
- client, defined, 675
- clusters

- defined, 675
 - switching between, 612
- color codes, predefined, 642
- color names, predefined, 642
- color, fill, 163, 164, 166, 169
- command fields, 409
- commands
 - defined, 675
 - kernel, 596
 - keyboard, 180
 - predefined, 638
 - SQL, 431
 - touch, 177
- comments in reports, 369
- communications
 - configuring, 503
 - report errors, 371
 - scheduling, 526
 - serial, 495
 - troubleshooting, 541
 - types, 483
 - with I/O devices, 483
- communications link, defined, 675
- communications port, defined, 675
- communications test project, 576
- compilation
 - debugging, 560
 - incremental, 560
 - project, 559
- Computer Setup Wizard, 473, 603
 - advanced alarms, 476
 - advanced reports, 478
 - advanced trends, 478
 - alarm, 476
 - computer role, 475
 - events, 479
 - flow diagram, 475
 - general options setup, 481
 - I/O server, 476
 - Internet server, 476
 - keyboard security, 481
 - menu security, 480
 - miscellaneous security, 481
 - network, 479
 - project, 476
 - reports, 477
 - server, 478
 - time, 480
 - trends, 478
- COMx driver, 495
 - debugging, 578
- configuration environment, 2
- configuration, startup/runtime, 568, 570
- configured alarm types, 203
- configuring, 605
 - accumulators, 259
 - alarms, 203, 248
 - communications, 503
 - DDE conversations, 413
 - devices, 396
 - events, 255
 - history files, 276
 - network DDE shares, 422
 - networks, 602
 - ODBC drivers, 435
 - parameters, 629
 - projects, 10
 - reports, 363
 - reports server, 605
 - time server, 605
 - trend tags, 264
 - trends server, 605
 - variable tags, 39
- connecting
 - database, 416
 - network DDE shared application, 426
 - SQL database, 430
- constants and Super Genies, 459
- control charts, 304
- control, statistical, 302
- conversations, DDE, 412
- converting
 - date and time, 445
 - values to strings, 389
- Copy Project dialog properties, 32
- copying
 - objects, 294
 - projects, 31
- cp index, 305
- cpk index, 305

- creating
 - alarm page, 249
 - browse sequence, 330
 - Genies, 449
 - graphics pages, 323
 - projects, 16
 - templates, 329, 330
- CtBackup utility, 30
- CtBackup32 utility, 30
- custom alarm filters, 204, 676
- D**
- data
 - appending, 442
 - archiving, 408
 - caching, 488
 - displaying in alarms, 240
 - editing, with ODBC, 442
 - exchanging, 411, 415
 - exporting trend, 272
 - formatting device, 400
 - logging in different languages, 471
 - printing trend, 272
 - reading data from access table, 441
 - trending, 263
 - using devices to read, 396
- data acquisition board, defined, 676
- data bits, defined, 676
- data communications equipment (DCE), 513, 676
- data path redundancy, 618
- data sources, external, 80
- data terminal equipment (DTE), 513, 678
- data transfer, defined, 676
- data type (I/O device), defined, 676
- data types
 - DDE, 415
 - variable tag, 41
- database device, 403
- database editors, 12
- database field format, 13
- database files, 13
- database structure, 12
- database transactions, 432
- databases
 - external, 429
 - language, 468
- date conversions, 445
- dates in SQL, 432
- DB-15, defined, 676
- DB-25, defined, 676
- DB-9, defined, 676
- dBASE database devices, 393
- dBASE databases, 429
- dBASE devices
 - format, 402
- dBase III format specifications, 13
- DDE conversations, 412
 - configuring, 413
- DDE services, 420
- DDE shares, 424
- DDE trusted shares, 425
- debugging
 - compilation, 560
 - COMx driver, 578
 - I/O devices, 576
 - proprietary board drivers, 584
 - protocol drivers, 582
 - protocols, 576
 - runtime system, 575
 - TCP/IP drivers, 581
- debugging trends, 277
- decimal notation, 46
- defaults, graphics page, 341
- defining
 - access to objects, 188
 - alarm categories, 207
 - areas, 378
 - colors on graphics pages, 348
 - fonts, 250
 - labels, 390
 - page properties, 333
 - paths, 276
 - substitutions for Genies, 450
 - substitutions for Super Genies, 457
 - trend pages, 270
 - user privileges, 376
- delaying alarms, 204
- deleting
 - objects, 292
 - projects, 18

- rows from access table, 443
 - demo mode, 590
 - design, project, 10, 11
 - DevAppend() function, 403
 - DevDelete() function, 405
 - DevFind() function, 404
 - DevGetField() function, 404
 - deviation alarm, defined, 677
 - device history files, 406
 - devices, 393
 - ASCII devices format, 401
 - configuring, 396
 - dBASE devices format, 402
 - formatting data in, 400
 - groups of, 395
 - predefined, 641
 - printer devices format, 401
 - properties, 397
 - reading data, 396
 - SQL devices format, 402
 - using a database device, 403
 - DevOpen() function, 403
 - DevSetField() function, 403
 - DevWrite() function, 404
 - dial back, defined, 677
 - dial-in modem, defined, 677
 - dial-out modem, defined, 677
 - Digiboard, defined, 677
 - digital alarms, 212, 677
 - digital variable (I/O Device), defined, 677
 - disabling object access, 191
 - disk I/O devices, 545, 547
 - defined, 677
 - redundant, 549
 - setup, 547
 - disk storage, calculating, 275
 - display
 - formatting alarm, 239
 - display client, 603
 - display client, defined, 677
 - display period, defined, 677
 - display, formatting alarm, 239
 - displaying
 - alarm pages, 249
 - lists in alarms, 240
 - tables in alarms, 240
 - distributed processing, 607, 677
 - distributed servers, 610, 612, 677
 - dither (imported bitmaps), defined, 677
 - domain name server (DNS), defined, 677
 - dot notation, defined, 678
 - drawing
 - buttons, 120
 - circles, 102
 - ellipses, 101
 - lines, 95, 97
 - pipes, 109, 110
 - polygons, 106
 - polylines, 106
 - rectangles, 99
 - squares, 99
 - drawing environment, 342
 - options, 345
 - driver accreditation, 518
 - driver errors, standard, 523
 - driver information, 519
 - drivers
 - defined, 678
 - generic errors, 520
 - ODBC, 435
 - ODBS, 433
 - drwFlip_properties, 298
 - DspGetEnv() function, 340
 - duplex, defined, 678
 - Dynamic Data Exchange (DDE), 411
 - and Office applications, 419
 - connecting to tag database, 416
 - data types, 415
 - posting data, 416
 - reading values from, 418
 - writing values to, 417
 - dynamic data exchange (DDE) server, defined, 676
 - dynamic data exchange (DDE), defined, 676
- ## E
- Edit Favorite Colors dialog box, 349
 - editing
 - databases, 12, 13
 - project properties, 19
 - using ODBC, 442

- editors, database, 12
 - effects, applying three-dimensional, 143
 - elements, array, 48
 - ellipse objects, 101
 - empty value, defined, 678
 - engineering units, 46
 - environment
 - configuration, 2
 - drawing, 342
 - graphics pages, 340
 - environment variables, 340
 - Super Genie, 462
 - errors, 664
 - generic driver, 520
 - netBIOS, 626
 - ethernet, defined, 678
 - event trend/SPC, defined, 678
 - events, 255
 - configuring, 255
 - page, 339
 - running, 257
 - times and periods, 257
 - triggers for, 258
 - Excel macros, 427
 - exchanging data, 411, 415
 - exponential notation, 46
 - Export Variable Tags dialog box, 80
 - exporting
 - tags, 79
 - trend data, 272
 - Express Communications Wizard, 551
 - caller ID, 554
 - device selection, 552
 - I/O device address, 553
 - I/O device communications selection, 552
 - I/O device connection schedule, 553
 - I/O device type, 552
 - link to external database, 555
 - serial device, 556
 - server selection, 552
 - summary, 557
 - TCP/IP address, 552
 - expression, defined, 678
 - expressions (Cicode) in reports, 368
 - external data sources, 80
 - external databases, using, 429
- ## F
- FastLinx for Mitsubishi
 - defining variable tag names, 60
 - reserved variable tag names, 60
 - tag browser properties, 59
 - field conversion, 87
 - fields
 - alarm display, 241
 - alarm summary, 243
 - command, 409
 - file names in DDE, 419
 - file server redundancy, 623
 - file server, defined, 678
 - files
 - alternative .ini, 574
 - Cicode, predefined, 642
 - device history, 406
 - include, 24
 - reconfiguring history, 276
 - report format, 368
 - syslog.dat, 575
 - trend history, 273
 - updating server-client, 570
 - fill color, 163, 164, 166, 169
 - fill level, 174
 - fill property (object), 285
 - filters, custom alarm, 204, 676
 - Find dialog box, 328
 - finding objects, 298
 - fixed text
 - alarm displays, 240
 - reports, 368
 - fonts
 - defining, 250
 - predefined, 640
 - properties, 251
 - report, 368
 - form feed in reports, 369
 - format file (import, export, linking), 82
 - format files, 83, 92
 - format, database, 13
 - formatting
 - alarm display, 239

- device data, 400
- numeric variables, 44
- Free Hand Line tool, 95
- freehand line properties, 96
- FTP server redundancy, 624
- full duplex, defined, 678

G

- generic driver errors, 520
- generic errors, 664
- generic protocol, defined, 678
- Genie controller, defined, 679
- Genies, 447, 448
 - and substitutions, 454
 - and Super Genies, 458
 - creating, 449
 - defined, 678
 - opening, 449
 - properties, 453
 - saving, 450
 - substitutions for, 450
 - tag names, 463
 - using, 451
- global Cicode variable, defined, 679
- global client, defined, 679
- Goto Object dialog box, 299
- graphics bounding box, defined, 679
- Graphics Builder, 7
- graphics objects, hiding, 465
- graphics pages
 - appearance, 336
 - bitmaps, 358
 - color definition, 348
 - creating, 323
 - defaults, 341
 - defined, 679
 - defining colors for, 348
 - environment, 340
 - events, 339
 - keyboard commands for, 337
 - libraries, 356
 - locating, 325
 - opening, 323
 - properties, 333
 - saving, 324
 - screen resolution, 332
 - sizing, 332
 - sliders, 183
 - symbols, 357
 - zooming, 355
- graphics specifications, 633
- graphics, importing, 281
- graphs, trend, 269
- Grid Setup dialog box, 343
- grid, for alignment, 342
- group (of objects), defined, 679
- grouping
 - objects, 293
 - registers, 489
- groups (areas), 381, 382
- groups (devices), 395
- groups, object, 280
- guidelines (alignment), 343
- Guidelines Setup dialog box, 345

H

- half duplex, defined, 679
- hardware alarms, debugging, 575
- hardware key, updating, 588
- hiding
 - graphics objects, 465
 - objects, 147
- hierarchy, privilege, 378
- histogram, defined, 679
- history
 - device, 406
 - trend, 273
- history files, 276
- horizontal movement, 149

I

- I/O device address, defined, 680
- I/O device constraints, 535
- I/O device data type specifications, 634
- I/O device data, errors in, 371
- I/O device properties, 507
- I/O device variable, defined, 680
- I/O device, defined, 679
- I/O devices

- communication with, 483
- debugging, 576
- disk devices, 547
- memory devices, 545
- properties, 507
- reading from, 527
- remote multidrop, 536
- transparent, 517
- writing to, 527
- I/O server redundancy, 540, 615
- I/O server, defined, 680
- IFDEF format, 465
- Import dialog box, 360
- Import Variable Tags dialog box, 58
- importing
 - graphics, 281
 - variable tags, 56, 57
- include files, 24, 679
- Include project, 25
- included projects, 24
- Included Projects dialog box, 24
- including projects, 23
- incremental compilation, 560
- input (keyboard commands), 181
- input property (objects), 286
- Insert Function dialog box, 37
- Insert Tag dialog box, 37
- inserting variable tags, 37
- integer variable (Cicode), defined, 680
- integer variable (I/O device), defined, 680
- Internet client setup properties, 572
- Internet display client, 569, 680
- Internet server, 570
- interpolation, trend, 271
- interrupt, defined, 680
- IODeviceControl () function, 526
- IP address, defined, 680

K

- kernel, 591
- kernel commands, 596, 597
- kernel window, 591, 592
- keyboard commands, 180, 337, 680
- keyboard key codes, predefined, 643
- keyboard keys, predefined, 639

- knowledge base, defined, 681
- kurtosis, defined, 681

L

- labels, 39, 387
 - arguments in, 389
 - defining, 390
 - for areas, 380
 - predefined, 647
- LAN redundancy, 625
- language database, defined, 681
- language databases, 468
- languages
 - changing, 467
 - changing at runtime, 470
 - logging in different, 471
 - multiple, 469
- layout, format file, 83
- level, fill, 174
- libraries, 356
- license point count, 590
- line objects, 280
- link, defined, 681
- linked symbols, 356
- linked tags, refreshing, 55
- linking
 - projects, 19
 - tags, 54
 - templates, 329
 - to projects, 22
- links, breaking, 55
- lists, displaying in an alarm, 240
- local area network (LAN), defined, 681
- local Cicode variable, defined, 681
- local language, 467, 681
- locating objects, 298
- locking objects, 293
- login records, 373
- long BCD variable (I/O device), defined, 681
- long file names (with DDE), 419
- long variable (I/O Device), defined, 681
- loop-back test, 501
- low and low low alarms, defined, 681
- lower control limit, 304
- lower specification limit, 305

M

- macros, 14, 427
- management
 - print, 409
- manager client, defined, 682
- manipulating objects, 289
- manual configuration, 503
- maximum request length, defined, 682
- memory I/O devices, 545, 682
 - setup, 546
- methods, storage, 274
- Microsoft Excel, 13
- millisecond trending, defined, 682
- mirroring objects, 298
- mode, demo, 590
- modems, 530, 531
- module Cicode variable, defined, 682
- movement, 148
 - horizontal, 149
 - of objects, 291
 - properties, 283
 - vertical, 151
- multi-digital alarms, 215, 682
- multidrop remote I/O devices, 536
- multi-dropping, 535
- multi-language projects, 467
- multiple
 - languages, 469
 - projects, 470

N

- native language, 467, 682
- nesting Super Genies, 462
- netBIOS errors, 626
- network
 - configuring, 602
 - defined, 682
 - using, 601
 - using parameters on, 630
- Network communications
 - using TCP/IP, 606
- network computer, defined, 682
- network DDE, 420, 426, 682
- network DDE shared application, 426
- network DDE shares, 422

- networked system, restarting, 585
- New Project dialog box, 17
- New Style dialog box, 330
- nodes, defined, 682
- normal distribution, defined, 683
- null value, defined, 683
- numbers (objects), 111
- numeric variables, 44

O

- object alignment, 296, 342, 343
- object ID (OID), defined, 683
- object properties, 281
 - disable access, 191
- object types, 95
 - ActiveX, 137
 - buttons, 120
 - Cicode, 132
 - ellipse, 101
 - free hand line, 95
 - numbers, 111
 - pasted symbol, 134
 - polygons, 106
 - rectangles, 99
 - straight line, 97
 - symbol sets, 122
 - text, 111
 - trends, 129
- object variable (Cicode), defined, 683
- objects, 279
 - 3D effects, 144
 - access, 188, 189, 191
 - access property, 288
 - aligning, 296
 - appearance, 282
 - applying effects, 143
 - copying/pasting, 294
 - defined, 683
 - deleting, 292
 - fill color, 163, 164, 166, 169
 - fill level, 174
 - fill property, 285
 - grouping, 293
 - groups, 280
 - horizontal movement property, 149

- input property, 286
- keyboard commands, 181
- line, 280
- locating, 298
- locking/unlocking, 293
- manipulating, 289
- mirroring, 298
- movement properties, 283
- moving, 148, 291
- pipes, 110
- rectangles, 100
- reshaping, 280
- resizing, 291
- rotating, 297
- rotational property, 186
- scaling, 156, 160, 284
- sliders, 184, 185, 287
- touch commands for, 177
- touch properties, 178
- vertical movement, 151
- visibility, 147, 148
- objects types
 - pipes, 109
- objects, library, 356
- occurrence section in tag names, 52
- ODBC drivers, 433, 435
- ODBC server, using CitectSCADA as, 439
- OEM character sets, 472
- OID validation, 520
- OLE objects in reports, 368
- OPC Data Access Server Tag Browser, 78
- open database connectivity (ODBC), defined, 683
- Open/Save As dialog box, 327
- opening
 - Genies, 449
 - projects, 18
- options, 345

P

- pack, defined, 684
- page environment variable, defined, 684
- page events, 339
- Page Properties dialog box, 334
- Page recompiled against different Variable.DBF, 520
- page style, graphics, 329

- page templates, 328
- pages
 - alarm, 249
 - displaying alarm, 249
 - startup, 331
- parameter queries, 444
- parameters, 629
 - on networks, 630
- Parameters dialog box, 631
- Pareto charts, 305
- parity, defined, 684
- pasted symbol objects, 134
- pasting objects, 294
- path definitions, 276
- path substitution, 276
- performance, 488, 598
- periodic trend, defined, 683
- periods, specifying for events, 257
- persistance and redundancy, 617
- persistence cache, defined, 684
- pipe arrangements, 110
- pipe objects, 109
- PLC interface board, defined, 684
- plots, in reports, 369
- point limit, defined, 683
- polygon objects, 106
- polygon properties, 107
- polylines, drawing, 106
- port(s), defined, 684
- posting data using DDE, 416
- predefined
 - character sets, 639
 - Cicode files, 642
 - color codes and names, 642
 - commands, 638
 - devices, 641
 - fonts, 640
 - keyboard key codes, 643
 - keyboard keys, 639
 - labels, 647
- predefined templates, 636
- prefixes in tag names, 51
- primary alarms server, defined, 684
- primary reports server, defined, 684
- primary trends server, defined, 684

- print management, 409
 - printer devices, 393
 - format, 401
 - printing
 - project details, 33
 - trend data, 272
 - privilege hierarchy, 378
 - privilege/area combinations, 384
 - privileges, using areas with, 383
 - process capability, 303
 - process variation, 301
 - processing, distributed, 607
 - project configuration, 10
 - project design, 10, 11
 - Project Editor, 5
 - Project Properties dialog box, 20
 - project restoration and security, 373
 - Project tag mismatch detected, 520
 - projects
 - backing up, 25
 - compiling, 559
 - configuring, 10
 - copying, 31
 - creating, 16
 - deleting, 18
 - editing properties, 19
 - including, 23
 - linking, 19
 - linking to, 22
 - multi-language, 467
 - multiple, 470
 - opening, 18
 - printing details, 33
 - restoring, 28
 - projects specifications, 634
 - properties
 - 3D effects, 144
 - access, 189
 - ellipses, 103
 - fill level, 174
 - font, 251
 - freehand line, 96
 - Genies, 453
 - movement, 148
 - object, 281
 - object visibility, 148
 - page, 333
 - pipe, 110
 - polygon, 107
 - rectangles, 100
 - rotational, 186
 - scaling, 156, 160
 - slider, 184
 - straight lines, 98
 - text, 112, 113, 114, 116, 118
 - touch, 178
 - variable tags, 41
 - vertical movement, 151
 - writing to alarm, 247
 - properties, object, defined, 683
 - proprietary board drivers, debugging, 584
 - proprietary boards, 500
 - protection, software, 588
 - protocol drivers
 - debugging, 582
 - protocol generic errors, 664
 - protocol, defined, 684
 - protocols, debugging, 576
 - proxi/proxy server, defined, 684
- ## Q
- queries, parameter, 444
- ## R
- rate of change alarms, defined, 684
 - reading from I/O devices, 527
 - real variable (Cicode), defined, 684
 - real variable (I/O device), defined, 685
 - record name, defined, 685
 - rectangle objects, 99, 100
 - redundancy, 615, 621, 685
 - and persistence, 617
 - disk I/O devices, 549
 - refreshing linked tags, 55
 - registers, grouping, 489
 - Remapping
 - properties, 493
 - remapping
 - variables, 491

- remote communications, defined, 685
 - remote terminal, defined, 685
 - report
 - defined, 685
 - report format file, 368
 - report format file, defined, 685
 - reports, 363
 - Cicode, 369
 - comments in, 369
 - communication errors, 371
 - configuring, 363
 - expressions and variables, 368
 - fixed text, 368
 - fonts, 368
 - form feed, 369
 - I/O device data errors, 371
 - OLE objects in, 368
 - plots, 369
 - redundancy, 621
 - report format file, 368
 - running, 366
 - suppressing, 372
 - trend data, 369
 - trend graphs, 369
 - triggers for, 367
 - reports server redundancy, 622
 - reports server, configuring, 605
 - reports server, defined, 685
 - reserved ANs, 635
 - reserved words, defined, 685
 - reshaping objects, 280
 - resizing objects, 291
 - restarting the system, 584
 - Restore Project dialog box, 28
 - restoring projects, 28
 - RJ11, defined, 685
 - RJ12, defined, 685
 - RJ45, defined, 685
 - Rotate dialog box, 298
 - rotating objects, 297
 - rotational property, 186
 - rows, deleting, 443
 - RS232/485 converter, 513
 - RS-232C standard, defined, 685
 - RS-422 (or EIA-422), 515
 - RS-422, defined, 686
 - RS-485, 516, 686
 - running
 - a system, 568
 - events, 257
 - reports, 366
 - runtime
 - changing languages at, 470
 - displaying alarm pages at, 249
 - handling alarms at, 248
 - runtime configuration, 568, 570
 - runtime information, 597
 - runtime security, 573
 - runtime system, 2, 3
 - debugging, 575
 - runtime system, defined, 685
- ## S
- sample period, 265
 - Save_DBF macro, 14
 - saving Genies, 450
 - scalable architecture, defined, 686
 - scaling objects, 156, 160, 284
 - scan time, page, 335
 - schedule period, defined, 686
 - schedules, specifying, 526
 - scheduling
 - communications, 526
 - screen resolution and graphics pages, 332
 - security, 373
 - MS Office, 419
 - runtime, 573
 - specifying requirements for, 383
 - using areas for, 380
 - Send Backwards command, 294
 - Send to Back command, 294
 - serial boards, 498
 - serial communications, 495, 686
 - serial communication standards, 514
 - serial port loop-back cable, 502
 - serial port loop-back test, 501
 - server, defined, 686
 - server-client file updates, 570
 - servers, distributed, 610, 612
 - services, starting network DDE, 420

- SetLanguage() function, 467, 470
- shares, DDE, 424
- shortform notation, 47
- simplex transmission, defined, 686
- size, calculating array, 49
- sizing
 - graphics pages, 332
 - objects by scaling, 156
- skewness, defined, 686
- slider control, defined, 686
- slider property, 287
- sliders, 183, 186
- Snap to Grid command, 342
- soft PLC, defined, 686
- software protection, 588, 686
- specifications
 - graphics, 633
 - I/O device data types, 634
 - projects, 634
- specifying
 - schedule, 526
- SQL database devices, 393
 - format, 402
- SQL databases, 429
- SQLExec() function, 431
- squares, drawing, 99
- standard driver errors, 523
- standards, common serial communication, 514
- standby alarms server, defined, 686
- standby reports server, defined, 687
- standby trends server, defined, 687
- starting network DDE services, 420
- startup configuration, 568, 570
- startup page, 331
- statistical control, 302
- statistical process control (SPC), 301
- stop bits, defined, 687
- storage method, trend data, 274
- storage, calculating disk, 275
- straight line objects, 97
- straight line properties, 98
- string arrays, 49
- string substitution, 390
- structure, database, 12
- structured query language (SQL), 430

- commands, 431
- times and dates, 432
- structured tag names, 51, 463
- substitutions
 - defining for Super Genies, 457
 - path, 276
 - string, 390
- summary display, alarms, 245
- Super Genies, 447
 - and Genies, 458
 - areas, 462
 - arrays and, 459
 - constants and, 459
 - environment variables, 462
 - nesting, 462
 - substitutions for, 457
 - tag names, 463
 - using, 454
- suppressing reports, 372
- Swap Color dialog box, 352
- switching between clusters, 612
- symbol set objects, 122
- symbols, 357, 687
- syntax, ODBC, 436
- syslog.dat file, 575
- system
 - running a, 568
 - runtime, 2, 3
- system commands (predefined), 638
- system redundancy, 615
- system tuning, 598

T

- tables, displaying, in alarms, 240
- tag browser
 - FastLink for Mitsubishi, 59
 - OPC Data Access Server, 78
- tag database, connecting to, 416
- tag names, 52
- tag-based driver data validation, 520
- tags
 - exporting, 79
 - importing, 56
 - linking, 54
 - names for, 51, 52

- supported alarm properties, 245
 - using alarm properties as, 245
 - task, defined, 687
 - TCP/IP
 - special options for, 497
 - using for network communications, 606
 - TCP/IP drivers, debugging, 581
 - templates, 18, 687
 - creating, 329, 330
 - for graphics pages, 326, 328
 - Genie substitutions in, 454
 - linking, 329
 - opening, 330
 - predefined, 636
 - text
 - in alarm displays, 240
 - marking for language change, 467
 - text box, defined, 687
 - text objects, 111
 - text properties, 112, 113, 114, 116, 118
 - thread, defined, 687
 - three-dimensional (3D) effects, 143, 144
 - time conversions, 445
 - time server, configuring, 605
 - time server, defined, 687
 - time stamped analog alarms, 687
 - time stamped digital alarms, 687
 - timeout, defined, 687
 - times
 - in SQL, 432
 - specifying for events, 257
 - time-stamped alarms, 220, 679
 - time-stamped analog alarms, 234
 - time-stamped digital alarms, 231, 679
 - tool tip, defined, 688
 - touch (object at runtime), defined, 688
 - touch commands, 177, 688
 - touch properties, 178
 - transactions, database, 432
 - transparent I/O devices, 517
 - trend data, report, 369
 - trend graphs, 369
 - trend history files, 273
 - trend line, defined, 688
 - trend objects, 129
 - trend plot, defined, 688
 - trend tags
 - configuring, 264
 - properties, 264
 - trend, defined, 688
 - TrendDebug, 277
 - trending data, 263
 - trends
 - creating pages for, 270
 - debugging, 277
 - exporting data for, 272
 - graphs of, 269
 - interpolation, 271
 - printing data, 272
 - storage methods, 274
 - trends server, 605, 688
 - trends server redundancy, 621, 623
 - triggers
 - and reports, 367
 - event, 258
 - TrnGetTable() function, 369
 - TrnPlot() function, 369
 - troubleshooting communications, 541
 - tuning
 - cache, 598
 - system, 598
- ## U
- unlinked symbols, 356
 - unlocking objects, 293
 - unsigned integer variable (I/O device), defined, 688
 - updating
 - hardware key, 588
 - server-client files, 570
 - upper control limit, 304
 - upper specification limit, 305
 - Use Template dialog box, 326
 - user privileges, 375, 376
 - user records, adding, 373, 374
 - Users dialog box, 374
 - using
 - distributed processing, 607
 - distributed servers, 610
 - external databases, 429
 - Genies, 451

- networks, 601
- serial board, 498
- Super Genies, 454
- symbols, 357

V

- validating tag-based driver data, 520

- values

- argument, 389
 - reading from DDE application, 418
 - string conversion, 389
 - writing to DDE application, 417

- variable data, displaying in alarms, 240

- variable tags

- assigning, 39
 - configuring, 39
 - data types, 41
 - importing, 57
 - inserting into field, 37
 - properties, 41

- variable type (Cicode), defined, 688

- variables

- environment, 340
 - in reports, 368

- remapping, 491
- variation, process, 301
- vertical movement, 151
- vertical sliders properties, 185
- viewing
 - Include project, 25
 - plant areas, 384
- virtual, defined, 688
- visibility, 147, 148

W

- wide area network (WAN), 613

- wiring cables, 512

- wizard, defined, 688

- writing

- to alarm properties, 247
 - to I/O devices, 527

X

- XRS control charts, 303

Z

- zooming, 355