CitectSCADA

v7.20

Technical Reference

October 2010

Legal Notice

DISCLAIMER

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

COPYRIGHT

TRADEMARKS

Schneider Electric (Australia) Pty. Ltd. has made every effort to supply trademark information about company names, products and services mentioned in this manual.

Citect, CitectHMI, and CitectSCADA are registered trademarks of Schneider Electric (Australia) Pty. Ltd.

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

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

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

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

dBASE is a trademark of dataBased Intelligence, Inc.

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

GENERAL NOTICE

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

October 2010 edition for CitectSCADA Version v7.20

Manual Revision Version v7.20.

Contact Schneider Electric (Australia) Pty. Ltd. today at www.Citect.com/citectscada

Contents

Safety Information

Hazard categories and special symbols

The following symbols and special messages may appear in this manual or on the product to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

A lightning bolt or ANSI man symbol in a "Danger" or "Warning" safety label on the product indicates an electrical hazard which, as indicated below, can or will result in personal injury if the instructions are not followed.

The exclamation point symbol in a safety message in a manual indicates potential personal injury hazards. Obey all safety messages introduced by this symbol to avoid possible injury or death.

Symbol	Name
	Lightning Bolt
	ANSI man
	Exclamation Point

⚠ DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, can result in death or serious injury.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

CAUTION

CAUTION used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, can result in property damage.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric (Australia) Pty. Ltd. for any consequences arising out of the use of this material.

Before You Begin

CitectSCADA is a Supervisory Control and Data Acquisition (SCADA) solution. It facilitates the creation of software to manage and monitor industrial systems and processes. Due to CitectSCADA's central role in controlling systems and processes, you must appropriately design, commission, and test your CitectSCADA project before implementing it in an operational setting. Observe the following:

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use CitectSCADA or other SCADA software as a replacement for PLC-based control programs. SCADA software is not designed for direct, high-speed system control.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.*
- Each implementation of a control system created using CitectSCADA must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

* For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control".

Technical Reference

This section contains the following technical reference information and the CitectSCADA Glossary:

Chapter: 1 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 or 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.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Read and understand the applicable material in this manual before changing or removing any citect.ini parameters.
- Never change or remove any undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no necessary or undocumented parameters will be deleted.
- Do not edit your configuration file while your project is running.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Note: Always seek the advice of Technical Support personnel for this product regarding necessary and undocumented features.

You can set operating parameters in:

- The project database.
- From v7.10, CitectSCADA expects the Citect.ini file to exist in the `config` folder of the CitectSCADA User and Data folder selected during installation. If the file is not found in the location, it will not search elsewhere and will instead display an error. If you need to store your INI file elsewhere, specify the path to it on the command line when starting citect32.exe and ctexplor.exe. See Text Editors for more information.
- Both the project database and the `citect.ini` file (see Parameter Precedence).

Rules for using parameters

You need to 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 need to re-compile the project before the parameter settings are used.
- Some `citect.ini` file parameters require a restart of CitectSCADA before they are used, while others are used as soon as the process to which they apply is restarted. For example, an Events parameter for an Alarm Server will be used as soon as the specific Alarm Server is restarted.
- Parameters set in the database are local to the specific CitectSCADA project. Parameters set in the `citect.ini` file apply to every CitectSCADA project (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.
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)

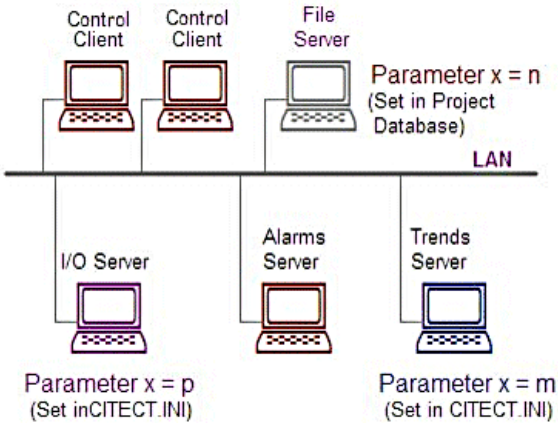
or -

3. Use a text Editor to Edit the `citect.ini` file.
4. Enter the parameter in the following format:

```
[SECTION NAME]
Parameter=<value>
```

Using parameters on a network

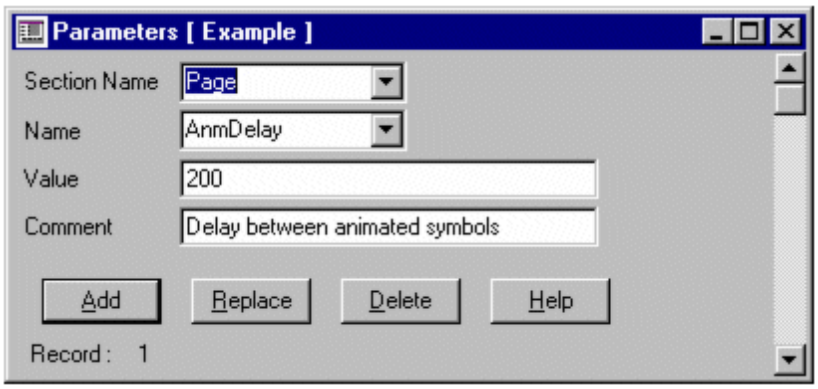
If using CitectSCADA on a network, you can use parameters globally, locally (local to each server and client), or both globally and locally. Any parameter set in the project database applies to every client unless the parameter is also set in the `citect.ini` of a Control Client. The value set in the local `citect.ini file` takes precedence over the project database for that 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 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 48 characters or less.

Name

The name of the parameter for which you want to define a value. Enter a value of 32 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.

Chapter: 2 Configuration Parameters

Parameter Overview

CitectSCADA has a comprehensive set of parameters that are used to configure the operational settings of a project, and, how each computer participates in a CitectSCADA network.

You can set operating parameters in:

- The **project database**
  Parameters set in the project database are local to the specific CitectSCADA project.
- The **Citect.ini file**
  Parameters set in the `Citect.ini` file apply to every CitectSCADA project running on the machine on which the `Citect.ini` file is located.

This chapter covers the following topics:

- Parameter Syntax
  The syntax of the `Citect.ini` file.
- Setting Parameter Values
  The different tools available to set parameter values within CitectSCADA.
- Parameter Precedence
  The rules outlining which parameter value is used when the value is set in both the `Citect.ini` file and the parameter database.
- Hierarchical Parameters
  How to fine tune parameter settings to a specific clusters or server process.
- Comments in Citect.ini
  How to add comments to a `Citect.ini` file.

For a list of the system parameters, refer to the Parameters help file.

Parameter Syntax

Parameters are grouped into Sections according to their purpose.

The syntax used in the `citect.ini` file to define a section is as follows:

```
[Section Name]
<parameter name1> = <parameter value1>
<parameter name2> = <parameter value2>
<parameter nameX> = <parameter valueX>
```

For Example:

```
[Alarm]
SavePeriod = 600
SaveSecondary =
ScanTime = 500
```

The **maximum length** for a parameter is **254** characters.

Sections which relate to server components (Alarms, Trend, Reports, IOServer) also support **hierarchical inheritance** to allow parameters to be fine tuned to the cluster or server component level. The syntax used is as follows:

```
[Section Name.ClusterName.ServerName]
<parameter name1> = <parameter value1>
<parameter name2> = <parameter value2>
<parameter nameX> = <parameter valueX>
```

For Example:

```
[Alarm.Cluster1.Server1]
SavePeriod = 600
ScanTime = 500
```

For more information see Hierarchical Parameters.

Setting Parameter Values

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Read and understand the applicable material in this manual before changing or removing any citect.ini parameters.
- Never change or remove any undocumented citect.ini parameters.
- Before deleting sections of the citect.ini file, confirm that no necessary or undocumented parameters will be deleted.
- Do not edit your configuration file while your project is running.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Note: Always seek the advice of Technical Support personnel for this product regarding undocumented features.

There are a number of methods to create or edit parameter values within CitectSCADA:

- Citect Project Editor
  Used to create or change values in the project database.
- Computer Setup Wizard
  Used to set up both necessary and commonly used `citect.ini` parameters on each machine. This Wizard steps user through a series of pages collecting information used to set parameter values.
- Computer Setup Editor
  Used to create or modify parameters in the `citect.ini` file. This tool provides users with a quick and convenient mechanism to set the value of a specific parameter by combining a graphical interface with a context-sensitive help reference.
- Text Editors
  A text editor can be used to modify the `citect.ini` file, although because of the risk involved where system configuration contains an error, this is not the recommended approach.

If you set (or change) parameters in the `citect.ini` file, you need to **restart** CitectSCADA before the new parameter settings are used. There are a few exceptions to this rule where `citect.ini` parameters are read at regular intervals and can be changed during runtime. Where this is the case, the parameter is documented accordingly.

Citect Project Editor

The only method available to create or change parameters in the project database is to use Citect Project Editor.

To set parameters in the project database:

1. From the **System** menu, select **Parameters** to display the Parameters dialog box.
2. Enter the **Section Name** of the parameter (16 characters or less).
3. Enter the **Name** of the parameter (16 characters or less).
4. Enter a **Value** for the parameter (254 characters or less).
5. Add the record to the database.

> **Note:** To locate an existing parameter use the scroll bar (on the right of the form) to move between each parameter record. The record number is shown in the bottom left hand corner of the form.

If you set (or change) parameters in the project database, you need to re-compile the project before the new parameter settings are used.

Computer Setup Wizard

The Computer Setup Wizard provides the user with simple interface to configure necessary and commonly used system parameters. The Wizard steps the user through a series of pages:

- utilizing the configuration stored in the project database to provide the user with contextual information;
- shielding the user from the need to understand the syntax of the `Citect.ini` file or the parameters; and
- modifying its behavior to reflect any relevant previous values already set within the Wizard.

It is used during the initial setup of each machine running CitectSCADA and can be reused on a machine at a later time to modify parameter settings. Parameter values collected from the user through the Wizard interface are written to the local `Citect.ini` file.

See Using the Computer Setup Wizard for more information.

Computer Setup Editor

The Computer Setup Editor provides the user with a graphical interface to configure `Citect.ini` parameters making it a quick and convenient tool to locate and change values for specific parameters. The Editor includes:

- a graphical interface which represents the parameters within the `Citect.ini` file as an expandable tree with a node for each Section;
- a built-in help reference for `Citect.ini` parameters;
- the ability to generate a comparison report between two separate `Citect.ini` files; and
- the ability to generate an analysis report on a `Citect.ini` file to check validity of parameter values.

For instructions on how to use Computer Setup Editor, see Using Computer Setup Editor Help from the **Help** menu within Computer Setup Editor.

> **Note:** The Computer Setup Editor cannot be used to maintain or set server parameters when being configured at the component or server level using Hierarchical Parameters.

Text Editors

The `citect.ini` file is a text file which stores CitectSCADA's operating parameters. During installation, a default version of this file is copied to the `config` folder of the CitectSCADA User and Data directory, as selected during installation.

If the file is not found in this location, it will not search elsewhere and will display an error. If you need to store your ini file elsewhere, specify the path to it on the command line when starting citect32.exe and ctexplor.exe.

> **Note:** The filename and location of the `citect.ini` file can be changed by using the `-i"file_path.INI"` option when calling the CitectSCADA Explorer or Citect32 runtime.

To set parameters in the local citect.ini file:

1. Use a text editor to open the `citect.ini` file.
2. Enter the parameter in the following format:

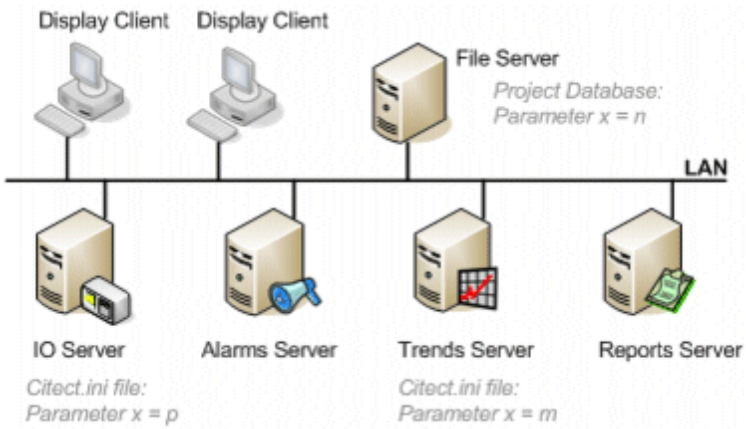
```
[SECTION NAME]
Parameter=<value>
```

3. Save the changes to the `citect.ini` file.

Parameter Precedence

On a machine where a parameter is set in **both** the project database and the `citect.ini` file the value contained in the `citect.ini` will be used by that machine.

For example, in the diagram below the project value for parameterX (which is stored in the project database) is `n`. This is the value used for parameterX on every server and client EXCEPT the I/O Server and Trends Server, both of which use the values set in their local `citect.ini` files (`p` and `m` respectively).



A parameter which is global to a project and applies to the majority of servers running a project is recommended therefore to be defined in the project database where it can be centrally managed and controlled. Any exceptions to this global value can then be managed by modifying the `citect.ini` file on the machine to which the exception applies.

Where a parameter is not set in either the project database or the `citect.ini` file, the **default value** for that parameter will be used by the system.

Hierarchical Parameters

As CitectSCADA supports clustering and the ability to run multiple servers of the same type on one machine, there are circumstances when the server component parameters (Alarm, Report, Trend, IOServer) need to be tuned to a finer level than just machine level. For this reason, these parameters are hierarchical parameters that are capable of being implemented at a number of levels:

- **Component Type Level**
  The widest scope, the parameter value will apply to every instance of the server type.
- **Cluster Level**
  The parameter value will apply to every instance of the server running in the specified cluster.
- **Server Level**
  The parameter value will apply to the instance of the server running in the specified cluster, on the specified machine.

These parameters support **hierarchical inheritance**, that is, a parameter will:

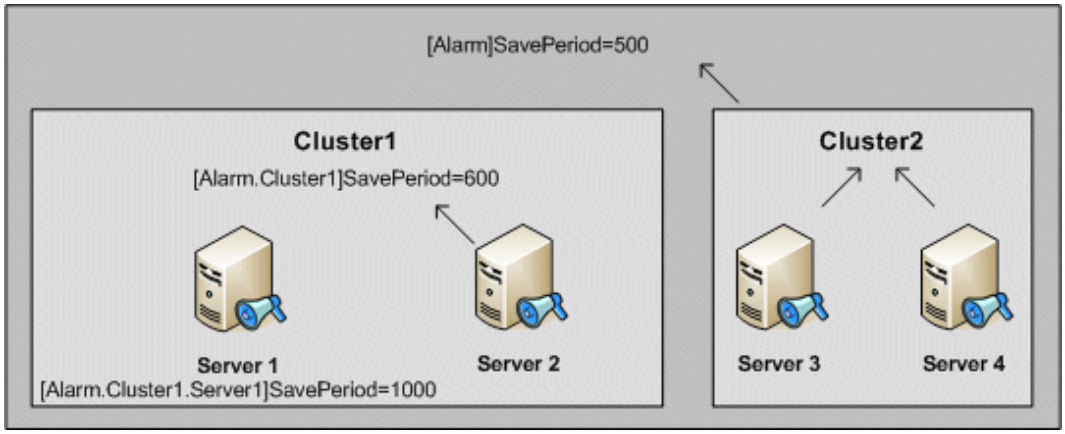
1. Apply a value set for it at a server level;
2. If that is not specified, apply a value set for it at a cluster level;
3. If that is not specified, apply a value set for it at a component level;
4. If that is not specified, apply the default value for that parameter.

Example

The following `Citect.ini file` is applied to Server1 and Server2 both in Cluster1, and to Server3 and Server4 both in Cluster2.

```
[Alarm]SavePeriod = 500
[Alarm.Cluster1]SavePeriod = 600
[Alarm.Cluster1.Server1]SavePeriod = 1000
```

This is illustrated in the diagram below.



The values applied at each server for [Alarm]SavePeriod would be as follows:

Cluster Name	Computer Name	SavePeriod Value
Cluster1	Server1	1000
Cluster1	Server2	600
Cluster2	Server3	500
Cluster2	Server4	500

Note: Hierarchical parameters may be set in the parameters database. In this case normal rules of precedence will apply. For more information see Parameter Precedence.

Comments in Citect.ini

Comments can be placed in the Citect.ini file using the following special characters:

- Use `#' at the start of a new line to add a comment followed by the text of the comment. Relate the comment to the following INI element (section or parameter). The equals character is not acceptable in comments.
- Use `!' at the start of a parameter to show that the parameter is disabled and the default value applies.

Note: Any line containing an '=' will be considered a parameter, regardless of what the line starts with.

Example:

```
[LAN]
#Disable Networking
Disable=1
```

Chapter: 3 Reference Information

This section contains the reference information for CitectSCADA, including:

Specifications

Format Fields

Error Messages

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Always use buffers with 2 extra bytes when reading digital types with CtAPI. This prevents bitwise shift operations from corrupting system memory.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

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 necessary 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 Variable Tags defined in CitectSCADA	4,194,303 (but no more than 500,000* is the recommended) This is a system wide limitation for a running system. Tags in every project included in the project you are currently compiling and running will contribute to this figure, regardless of clustering.

	Be aware that if you are using one or more tag based drivers in your project, the figure depends on the bit length of your variables defined for the driver(s). This is due to the need for these drivers to use OIDs to identify each tag. For example, a tag based driver that uses 32-bit digitals may reduce 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). If no bit size is defined for digitals by 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	32000*
Number of trends displayed on the same chart	16*
Number of alarms	65,535 per alarm type
Number of trends displayed on the same page	16000
Number of user functions	4500*
Number of	700

standard built-in functions provided	
Number of operator commands for the system	3000*
Number of I/O Devices connected to Citect-SCADA	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 impact the Trends Server performance (CPU loading) and the CPU loading of the Client displaying the Process Analyst.

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
Unsiged Integer		
Long Integer	4 bytes	
Long BCG	4 bytes	
Floating Point (Real)	4 bytes	
String	256 bytes (Tags and Cicode Functions)	ASCII (null terminated)
	128 bytes (Global Cicode Variables)	

Reserved ANs

The following table describes 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 alert or error messages. You need to 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.
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) customization 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	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 necessary. 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.
Tab1Menu . . Tab6Menu	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 necessary. 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.
SingleTrend	A template to create trend pages with one trend window. There are several ways to configure the trend pens: 1. Double-click the window. 2. Open the trend page using the PageTrend() function and pass in the trend pens as parameters. 3. Select the pens manually (from the page) at runtime

	SingleTrend pages can be configured with trend tags of type Periodic or Periodic Event.
DoubleTrend	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. DoubleTrend pages can be configured with trend tags of type Periodic or Periodic Event.
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 trend window. There are several ways to configure the trend pens: 1) double-click the window 2) Open the trend page using the PageTrend() function and pass in the trend pens as parameters. 3) 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, open the trend page using the PageTrend() function and pass in the trend pens as parameters, 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 need to 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 need to 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.
Hardware	A template to create a hardware alarm page. You need to 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 need to 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 need to 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 need to 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

This section describes the system keyboard commands that are predefined in the Include Project. (System keyboard commands operate on any graphics page displayed on the computer screen).

- System keyboard commands database
- Predefined keyboard keys
- Keyboard keys database

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 keyboard keys described below 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
DOWN_SHIFT	KEY_DOWN_SHIFT	Shift down key
ENTER	KEY_ENTER	Enter key
LBUTTON_DBL	KEY_LBUTTON_DBL	Left mouse button double click
LBUTTON_DN	KEY_LBUTTON_DN	Left mouse button down
LBUTTON_UP	KEY_LBUTTON_UP	Left mouse button up

Key Name	Key Code	Description
LBUTTON_CMD_DN	KEY_LBTN_CMD_DN	Left mouse button down (command cursor)
LBUTTON_CMD_DNC	KEY_LBTN_CMD_DNC	Ctrl left mouse button down (command cursor)
LBUTTON_CMD_DNS	KEY_LBTN_CMD_DNS	Shift left mouse button down (command cursor)
LBUTTON_CMD_UP	KEY_LBTN_CMD_UP	Left mouse button up (command cursor)
LBUTTON_CMD_UPC	KEY_LBTN_CMD_UPC	Ctrl left mouse button up (command cursor)
LBUTTON_CMD_UPS	KEY_LBTN_CMD_UPS	Shift 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
UP_SHIFT	KEY_UP_SHIFT	Shift up key

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	
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
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

This section describes devices that 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
_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
trninfo.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. 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
Grey	0xBFBFBF

Dark_Grey	0x7F7F7F
Light_Blue	0x0000FF
Light_Green	0x00FF00
Light_Cyan	0x00FFFF
Light_Red	0xFF0000
Light_Magenta	0xFF00FF
Yellow	0xFFFF00
White	0xFFFFFF
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 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 Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
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_LBTN_CMD_UP	0x0601	Left Mouse Button Up (Command Cursor)
KEY_RBTN_CMD_UP	0x0602	Right Mouse Button Up (Command Cursor)
KEY_MBTN_CMD_UP	0x0604	Middle Mouse Button Up (Command Cursor)
KEY_LBTN_CMD_DN	0x0605	Left Mouse Button Down (Command Cursor)
KEY_RBTN_CMD_DN	0x0606	Right Mouse Button Down (Command Cursor)
KEY_MBTN_CMD_DN	0x0608	Middle Mouse Button Down (Command Cursor)
KEY_LBTN_CMD_UPS	0x1601	Shift Left Mouse Button Up (Command Cursor)
KEY_RBTN_CMD_UPS	0x1602	Shift Right Mouse Button Up (Command Cursor)
KEY_MBTN_CMD_UPS	0x1604	Shift Middle Mouse Button Up (Command Cursor)
KEY_LBTN_CMD_DNS	0x1605	Shift Left Mouse Button Down (Command Cursor)
KEY_RBTN_CMD_DNS	0x1606	Shift Right Mouse Button Down (Command Cursor)

Key Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
KEY_MBTN_CMD_DNS	0x1608	Shift Middle Mouse Button Down (Command Cursor)
KEY_LBTN_CMD_UPC	0x2601	Ctrl Left Mouse Button Up (Command Cursor)
KEY_RBTN_CMD_UPC	0x2602	Ctrl Right Mouse Button Up (Command Cursor)
KEY_MBTN_CMD_UPC	0x2604	Ctrl Middle Mouse Button Up (Command Cursor)
KEY_LBTN_CMD_DNC	0x2605	Ctrl Left Mouse Button Down (Command Cursor)
KEY_RBTN_CMD_DNC	0x2606	Ctrl Right Mouse Button Down (Command Cursor)
KEY_MBTN_CMD_DNC	0x2608	Ctrl Middle Mouse Button Down (Command Cursor)
KEY_BACKSPACE	0x0008	Backspace
KEY_TAB	0x0009	Tab
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 Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
KEY_END	0x0123	End
KEY_HOME	0x0124	Home
KEY_LEFT	0x0125	Cursor Left
KEY_UP	0x0126	Cursor Up
KEY_RIGHT	0x0127	Cursor Right
KEY_DOWN	0x0128	Cursor Down
KEY_LEFT_SHIFT	0x1125	Shift Left
KEY_UP_SHIFT	0x1126	Shift Up
KEY_RIGHT_SHIFT	0x1127	Shift Right
KEY_DOWN_SHIFT	0x1128	Shift 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 Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
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 Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
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_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 Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
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 Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
KEY_Y_SHIFT	0x1059	Shift Y
KEY_Z_SHIFT	0x105A	Shift Z
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_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 Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
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_O_ALT	0x404F	Alt O

Key Code (CitectSCADA label)	Key Code (Hex Value)	Key Description
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

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

This section describes the labels that are predefined in the Include Project.

Labels database

The table below defines the names, expressions predefined in the Include Project.

Name	Expr	Comment
__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
__TIME__	$0	Time of compilation
BLANK		NULL Definition
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
Alarm-NextCatRec(hRec,hCat,nType,hArea=-1)	_AlarmQueryNextRec (hRec,hCat,nType,hArea,0)	Get Alarm Cat Rec with Area
Alarm-NextPriRec(hRec,hPri,nType,hArea=-1)	_AlarmQueryNextRec (hRec,hPri,nType,hArea,1)	Get Alarm Pri Rec with Area
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
DELETE_ANM	000	Delete animation
DevFirst(hDev)	DevSeek(hDev,0)	DevSeek with Off-set=0
DspBut-ton(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
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
EVEN_P	2	Even Parity
Exec(sText,mode=1)	_Exec(sText,mode)	Exec program, default to normal
FALSE	0	Boolean False
FlashColourState()	StrToInt(PageInfo(18))	Flashing Color State as a bool-ean
GetBlueValue(PackedRGB)	((PackedRGB / 65536) BITAND 255)	Get the blue com-ponent 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 com-ponent 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
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
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
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 Triag 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
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 priv-ileges
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
$	36	24
%	37	25
&	38	26

'	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
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
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
ƒ	131	83
„	132	84
…	133	85
†	134	86
‡	135	87
ˆ	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
œ	156	9C
	157	9D
	158	9E
Ÿ	159	9F
{NBSP}	160	A0
¡	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
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
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 describes the following:

- Alarm Display Fields
- Alarm Summary Fields
- Command Fields

Alarm display fields

You can use any of the fields listed below, or the Alarm Summary 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 **Note:** If the **Tag** field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.
{TagEx,n}	Alarm Tag with Cluster Name prefix **Note:** If the **TagEx** field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.
{AlarmType,n}	Alarm type (string), not localized. Values are: Digital, Analog, Advanced, Multi-Digital, Argyle Analog, Time Stamped, Time Stamped Digital, Time Stamped Analog.
{TypeNum,n}	Alarm type number (use AlarmType to get string value instead). Values are: -1 Invalid 0 Digital 1 Analog 2 Advanced 3 Multi-Digital 4 ArgAna 5 User Event 6 timestamped 7 hardware 8 timestamped digital 9 timestamped analog
{AlmComment,n}	The text entered into the Comment field of the alarm properties dialog.
{Cluster,n}	Cluster Name

Field Name	Description
{CUSTOM1,n} {CUSTOM2,n} {CUSTOM3,n} {CUSTOM4,n} {CUSTOM5,n} {CUSTOM6,n} {CUSTOM7,n} {CUSTOM8,n}	Alarm custom fields as configured.
{Name,n}	Alarm Name **Note:** If the **Name** field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.
{Native_Name,n}	Alarm Name in the expression **Note:** If the **Native_Name** field is configured to support long names (up to 79 characters), it might cause overlap in an alarm display. Use a smaller display font if long names are expected.
{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
{Priority,n}	Alarm category's priority
{Type,n}	The type of alarm or condition: ACKNOWLEDGED CLEARED DISABLED UNACKNOWLEDGED
{LocalTimeDate,n}	Alarm date and time in the form: "yyyy-mm-dd hh:mm:ss[.ttt]"
{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.)

Field Name	Description
{Date,n}	The date on which the alarm changed state (dd:mm:yyyy). Be aware that you can change the format used via the parameter [ALARM]ExtendedDate.
{DateExt,n}	The date on which the alarm changed state in extended format.
{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
{Millisec,n}	Adds milliseconds to the {Time,n} field
{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
{ErrDesc,n}	Text string associated with a protocol (communication) error. This field is only associated with hardware errors and contains extra information associated with whatever error is detected (for example if the error is associated with a device, the device name is returned; if the error is associated with a Cicode function, the

Field Name	Description
	function name is returned; if the error is associated with an I/O Device, the I/O Device's alert message is returned).
{ErrPage,n}	The page, device, etc. associated with the alarm.
{LogState,n}	The last state that the alarm passed through. (This is useful when logging alarms to a device.)
{State_desc, n}	The configured description (for example healthy or stopped) of a particular state. This description is entered when configuring the Multi-Digital Alarm Properties
{Paging,n}	Indicates whether the alarm has to be paged. When the value is TRUE the alarm will be paged. The default value is FALSE. See Alarm Paging Properties.
{PagingGroup, n}	Indicates the paging group to which the alarm belongs. Maximum length is 80 characters.
{AcqDesc,n}	Textual representation of Alarm Acquisition Error.
{AcqError, n}	Numeric representation of Alarm Acquisition Error.

Where n specifies the display field size.

Notes:
- Any of the above fields can be displayed for any type of alarm. Where not applicable for a particular alarm type, zero or an empty string will be displayed.
- If an alarm value is longer than the field it is to be displayed in (n), it will be truncated or replaced with the #OVR ("overflow of format width") alert message.
- 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.

See Also

Alarm summary fields

Alarm summary fields

You can use any fields listed below (or a combination) to format an alarm summary display and an alarm summary device.

Format the alarm summary for an entire category of alarms by specifying field names in the **Summary Format** field of the Alarm Category Properties dialog box.

You can also use the `[Alarm]DefSumFmt` parameter to format the alarm summary, particularly if 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 (for example acknowledging the alarm or disabling the alarm, etc.). 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 (for example acknowledging the alarm or disabling the alarm, etc.). 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

Field Name	Description
{SumType,n}	Type of alarm summary (similar to alarm "Type"). Values are ACKNOWLEDGED, CLEARED, DISABLED, UNACKNOWLEDGED
{Native_SumDesc,n}	A description of the alarm summary, in the native language
{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.**

See Also

Changing the Order of the Alarm Summary Display

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 *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 Timothy Lee".

Error Messages

CitectSCADA has two kinds of protocol driver errors:

- generic
- driver-specific

Generic errors are hardware errors 0-31, and are common to every protocol.

Drivers 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 becomes inoperative, there is often both a driver-specific error and a corresponding generic error.

> **Note:** For reference information related to the implementation of device drivers, including driver alert messages, please refer to the **Driver Reference Help**.

See Also

Generic Driver Errors
Driver Specific Errors
Using the Driver Reference Help

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 every protocol.

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

Error number	Error title	Description
		device. Check the Variable Tags database to find the variable in error.
2	Command canceled	The server canceled the command while it was being processed by the driver. The driver may have taken too long to process the command. If a driver does not 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. 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 will not occur during normal operation. Restart the computer to reset every driver and hardware. If the problem persists, contact Technical Support for this product. 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, for example 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 will not occur during normal operation. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Technical Support of this product.
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 loss of communications. 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 increas-

Error number	Error title	Description
		ing 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**: 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.
10	Hardware error	A problem exists with either the communication channel, server, or I/O device hardware. Examine hardware components. The command or data request has not been processed. The server's operation may no longer operate

Error number	Error title	Description
		normally.
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 alert condition active, for example 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 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.)

Error number	Error title	Description
		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 finished.
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 drivers and hardware. If the problem persists, contact Technical Support of this product.
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.
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 the drivers and hardware. If the problem persists, contact Technical Support of this product.
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 drivers and hardware. If the problem persists, contact Technical Support of this product. The command or data request has not finished.

Error number	Error title	Description
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 errors. When this error occurs, 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 finished.
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 drivers and hardware. If the problem persists, contact Technical Support for this product.. The command or data request has not finished.
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 drivers and hardware. If the problem persists, contact Technical support for this product. The command or data request has not finished.
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 finished. Try re-booting the computer to reset drivers and hardware. If the problem persists, contact Contact Technical Support for this product..
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 need to establish complex communication to retrieve data from the I/O device. Ignore this alert.
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 drivers and hardware. If the problem persists, contact Contact Technical Support for this product..
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 error 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.

Error number	Error title	Description
29	Bad user parameters	There is a configuration error, for example invalid special options have been set.
30	Stand-by I/O device error	There is an error in a standby I/O device. Rectify error in the standby I/O device.
31	Request Time-out from I/O Server	One or more requests sent to the I/O Server have not finished in the timeout period. Either the I/O Server is off line or the I/O Server is taking too long to finish the requests. Check the PLC communication link, PLC time-outs, 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 Control Client. If the error persists, increase the [LAN] TimeOut parameter. The default timeout is 8000 milliseconds.
32	Cannot talk to remote unit	The remote I/O device is not connected.
274	Invalid argument passed	An invalid argument has been passed to a Cicode function. This is a general error message and is generated when arguments passed to a function are out of range or are invalid. Check the value of arguments being passed to the function. If arguments are input directly from the operator, check that the correct arguments are being passed to the function.
281	No server could be found	The specified CitectSCADA server cannot be found. Either the server is not running or there is some communication problem with the network. Check that the network is set up correctly, and you are using the same Server Name on both the client and server.
418	No server of type on cluster	There is no server of the necessary type configured on the server.
448	Record size mismatch	An RDB file contains records with the wrong size.
451	Server previous reload busy	Unable to start a reload using the ServerReload Cicode function as a reload is already in progress for the specified server.
452	Invalid RDB version	An RDB file was compiled using an incompatible version of the software.

Error number	Error title	Description
454	Cicode library times-tamp differs	The timestamp of the Cicode library in memory is dif-ferent from the timestamp of the Cicode library on disk. The Cicode libraries are potentially different.
519	Remote Cicode call Interrupted	Cicode call that triggers an RPC remote call is interrupted before the expected result is returned.
520	Alarm category out of range	A category number is out of its valid range (from 0 to 16376 inclusively).
521	Data browse record is deleted	A record was deleted during a reload of ART server.
522	Trend archive prop-erty mis-match	A trend record's archive properties have changed during start-up or reload.
523	Alarm priority out of range	A category priority is out of its valid range (from 0 to 256 inclusively).

Generic driver errors

The following errors are generic to every CitectSCADA driver. A driver error needs to be mapped to a generic error before CitectSCADA can interpret it.

Error	Description
GENERIC_ADDRESS_RANGE_ERROR (0x0001 | SEVERITY_ERROR)	A request was made to a device address that does not exist. For example, an attempt was made to read register number 4000 when there are only 200 registers in the device.
GENERIC_CMD_CANCELED (0x0002 | SEVERITY_ERROR)	The server canceled the command while the driver was processing it. This can happen if the driver takes too long to process the com-mand. Check the timeout and retries for the driver.
GENERIC_INVALID_DATA_TYPE (0x0003 | SEVERITY_ERROR)	A request was made specifying a data type not supported by the protocol. This error will not occur during normal operation.

GENERIC_INVALID_DATA_FORMAT (0x0004 | SEVERITY_ERROR)	A request contains invalid data; for example, writing to a floating-point address with an invalid floating-point number. Check the Citect-SCADA database.
GENERIC_INVALID_COMMAND (0x0005 | SEVERITY_ERROR)	The server sent a command to the driver that it did not recognize. This error will not occur during normal operation.
GENERIC_INVALID_RESPONSE (0x0006 | SEVERITY_ERROR)	The communication channel is not performing normally, and is causing errors in the transmitted data.
GENERIC_UNIT_TIMEOUT (0x0007 | SEVERITY_ERROR)	A device is not responding to read or write requests. The driver sent a command to the device and the device did not respond within the timeout period.
GENERIC_GENERAL_ERROR (0x0008 | SEVERITY_ERROR)	Unmapped driver specific errors are normally reported as a general error. Refer to the protocol-specific errors listed with the protocol you are using.
GENERIC_WRITE_PROTECT (0x0009 | SEVERITY_ERROR)	A write operation was attempted to a location that is protected against unauthorized modification. Change the access rights to this location to permit a write operation.
GENERIC_HARDWARE_ERROR (0x000A | SEVERITY_UNRECOVERABLE)	The communication channel, server, or device hardware is not performing normally. Examine hardware components. The server's operation needs to also be examined for proper operation.
GENERIC_UNIT_WARNING (0x000B | SEVERITY_WARNING)	The communication link between the server and the device is functioning correctly; however, the device is experiencing an error or is in a non-operational state, for example, the device is in program mode.
GENERIC_UNIT_OFFLINE (0x000C | SEVERITY_SEVERE)	The device is in offline mode, preventing any external communication. This error will cause any stand-by units to become active. Citect-SCADA 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_VIOLATION	An attempt has been made by an unauthorized

(0x000E| SEVERITY_ERROR)	user to access information. Check the user's access rights.
GENERIC_NO_MEMORY (0x000F | SEVERITY_UNRE-COVERABLE)	The server or driver has run out of memory and cannot continue execution. Minimize buffer and queue allocation or expand the server computer's memory (physical or virtual memory).
GENERIC_NO_BUFFERS (0x0010 | SEVERITY_ERROR)	There are no communication buffers left to allocate. The performance of the server may be reduced; however, it can still continue to run. Increase the number of communication buffers.
GENERIC_LOW_BUFFERS (0x0011| SEVERITY_WARNING)	This error may occur in periods of high transient loading with no ill effects. If this error occurs frequently, increase the number of communication buffers.
GENERIC_TOO_MANY_COMMANDS (0x0012| SEVERITY_WARNING)	Too many commands have been sent to the driver.
GENERIC_DRIVER_TIMEOUT (0x0013 | SEVERITY_ERROR)	The server is not receiving any response from the driver. This error should not occur during normal operation.
GENERIC_NO_MORE_CHANNELS (0x0014 | SEVERITY_SEVERE)	Each driver can only support a fixed number of communication channels. You have exceeded the limit. The command or data request has not been completed.
GENERIC_CHANNEL_OFFLINE (0x0015 | SEVERITY_SEVERE)	A communication channel is currently offline, disabling communication. The server cannot initialize the communication channel or the channel went offline while running. Every device (units)connected using this channel will be considered to be offline so this will cause any stand-by devices to become active. Citect-SCADA 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 condition persists, contact 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, for example 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 issue. For example, some drivers may use this to alert you to stale data.

Protocol-Specific Errors

Though each protocol may have multiple unique errors, the first 34 protocol-specific errors are standard for every protocol. Every protocol-specific error is 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 Technical Support for this product..
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 disconnected cable 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 disconnected cable 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 disconnected cable to an I/O Device or by excessive noise on the communication link. Contact Technical Support for this product. 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 disconnected cable 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 disconnected cable 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 disconnected cable 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 disconnected cable to an I/O Device or by excessive noise on the communication link.
11 (0x0B)	Cannot reset serial driver	An error has occurred with the serial (COMxI, PCXI, or COMx) driver. Try re-booting the computer to reset drivers and hardware.
12 (0x0C)	Length of request inconsistent	The length of a request is not consistent with the driver's requirements.
15 (0x0F)	Command from server invalid	The command from the server is invalid. Contact Technical Support for this product.

16 (0x10)	Cannot allocate timer resource for driver	Driver timer resources cannot be allocated. Contact Technical Support for this product.
17 (0x11)	Too many channels specified for driver	Too many channels have been specified for the device. Contact Technical Support for this product.
18 (0x12)	Channel number from server not opened	The channel number from the server is not open. Contact Technical Support for this product.
19 (0x13)	Command cannot be cancelled	A driver command cannot be cancelled. Contact Technical Support for this product.
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 disconnected cable 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 disconnected cable 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 Technical Support for this product.
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 Technical Support for this product.
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 Technical Support for this product.

26 (0x1A)	Too many I/O Devices specified for channel	Too many I/O Devices have been specified for the channel. Contact Technical Support for this product.
27 (0x1B)	Too many commands issued to driver	Too many commands have been issued to the driver. Contact Technical Support for this product.
28 (0x1C)	Data read invalid	The data read is not valid. Contact Technical Support for this product.
29 (0x1D)	Command is cancelled	A driver command has been cancelled. Contact Technical Support for this product.
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 Technical Support for this product.
32 (0x20)	Cannot read data from device	CitectSCADA cannot read the data from the I/O Device. Contact Technical Support for this product.
33 (0x21)	Device specified does not exist	The device specified does not exist. Contact Technical Support for this product.
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.

Standard driver errors

The following errors are low-level errors which are generic to evry driver. These errors are mapped to Generic errors so that CitectSCADA can recognize them. Most drivers also have a set of driver specific errors in addition to these errors.

Error	Description
0 (0x00000000) NO_ERROR	No error condition exists.
1 (0x00000001) DRIVER_CHAR_OVERRUN	Transmitted characters could not be received fast enough. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
2 (0x00000002) DRIVER_CHAR_PARITY	Parity error in received characters. This error mapped to Generic Error GENERIC_INVALID_ RESPONSE.
3 (0x00000003) DRIVER_CHAR_BREAK	A break was detected in the receive line. This error mapped to Generic Error GENERIC_INVALID_ RESPONSE.
4 (0x00000004) DRIVER_CHAR_FRAMING	Framing error. Check the baud rate. This error mapped to Generic Error GENERIC_INVALID_ RESPONSE.
5 (0x00000005) DRIVER_MSG_OVERRUN	The message received from the device was too long. This error mapped to Generic Error GENERIC_INVALID_RESPONSE.
6 (0x00000006) DRIVER_BAD_CRC	Checksum in received message does not match the calculated value. Error mapped to Generic Error GENERIC_INVALID_RESPONSE.
7 (0x00000007) DRIVER_NO_STX	Start of text character not present. Error is mapped to Generic Error GENERIC_INVALID_ RESPONSE.
8 (0x00000008) DRIVER_NO_ETX	End of text character not present. Error is mapped to Generic Error GENERIC_INVALID_RESPONSE.
9 (0x00000009) DRIVER_NOT_INIT	Driver has not been initialized. This error is mapped to Generic Error GENERIC_UNIT_OFF-LINE.
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_HARD-WARE_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_CAN-CEL.
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)	The unit type from the server is unknown to the

DRIVER_BAD_UNIT_TYPE	driver. This error is mapped to Generic Error GENERIC_INVALID_DATA_TYPE.
26 (0x0000001A) DRIVER_TOO_MANY_UNITS	Too many units specified for channel. This error is mapped to Generic Error GENERIC_TOO_MANY_ UNITS.
27 (0x0000001B) DRIVER_TOO_MANY_COMMANDS	Too many commands have been issued to the driver. This error code can also occur if you are running a restricted version of a driver (i.e. one that will run for a limited time) for every issued read and write. 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 exists. This error is mapped to Generic Error GENERIC_HARDWARE_ ERROR.
34 (0x00000022) DRIVER_DEVICE_NO_INTERRUPT	Device specified does not support interrupt. This error is mapped to Generic Error GENERIC_HARD-WARE_ERROR.
35 (0x00000023) DRIVER_BAD_SPECIAL	Invalid special options in port database. This error is mapped to Generic Error GENERIC_BAD_PARAM-ETER.
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 con-tinue execution. Minimize buffer and queue allo-cation or expand the computer's memory (physical

	or virtual memory). This error is mapped to Generic Error GENERIC_NO_MEMORY.

Chapter: 4 CtAPI Functions

CitectSCADAAPI

The CTAPI allows access to CitectSCADA I/O variable tags via a DLL interface. This allows 3rd party developers to create applications in C (or other languages) to read and write to the I/O Devices.

The files necessary are ctapi.dll ctapi.lib and ctapi.h, and are located in the [bin] directory.

Using the CTAPI on a remote computer

To use the CTAPI on a remote computer without installing CitectSCADA, you will need to copy the following files from the [bin] directory to your remote computer: ctapi.dll, ct_ipc.dll, cteng32.dll, ctres32.dll, ctutil32.dll, and CiDebugHelp.dll. A project needs to have users defined before you can connect to the CTAPI from a remote computer.

Using CTAPI on Windows 2000

When running an application on Windows 2000 that uses CTAPI, you will need to copy the following files from the [bin] directory to the same directory as your application: ct_ipc.dll, cidebughlp.dll, dbghelp.dll, and ctutil32.dll.

Backward compatibility issues

A non-documented API was provided in the 16 bit version of CitectSCADA. As the 16-bit API is not compatible with the 32 bit environment, this new API has been implemented to replace it. The CTAPI is not compatible with the previous CT_VAR and CTUSER APIs. The CTAPI cannot be made compatible due to changes necessary for 32-bit environment. The changes necessary to port an application from the older API to the new CTAPI are small.

See Also

I/O Point Count
CtAPI Synchronous Operation
Reading Data Using the CTAPI Functions
CTAPI from CitectSCADA or CitectSCADA Driver
Error Codes
Debug Tracing
Function Reference

I/O Point Count

Physical I/O Device tags read, or written to, using the CTAPI are counted as dynamic CitectSCADA points. If the point limit is exceeded by making calls to this interface, then that call will not succeed, and CitectSCADA will not be allocated any more dynamic points.

> **Note:**CitectSCADA's licensing works on the basis of how many points you use. Every tag in your system has the potential to add to your point count. It is important to remember this, and plan your system properly, otherwise you may exceed your point limit.

The point limit is the maximum number of I/O Device addresses (variable tags) that can be read, and is specified by your CitectSCADA license. CitectSCADA counts I/O Device addresses dynamically at runtime. This includes tags used by alarms, trends, reports, events, pages, in Super Genies, use of the TagRead() and TagWrite() Cicode functions, or read or write using DDE, ODBC, or the CTAPI.

It does not count any points statically at compile time.

When your system is running, any new use of tags through Super Genies, DDE, ODBC, or CTAPI can potentially add to your dynamic point count.

See Also

CitectSCADA API Synchronous Operation

CtAPI Synchronous Operation

The CitectSCADA CTAPI supports both synchronous and asynchronous (or overlapped) operations. The **ctCicode()**, **ctListRead()**, and **ctListWrite()** functions can be performed either synchronously or asynchronously. The **ctTagRead()** and **ctTagWrite()** functions can be performed synchronously only.

When a function is executed synchronously, it does not return until the operation has been completed. This means that the execution of the calling thread can be blocked for an indefinite period while it waits for a time-consuming operation to finish. A function called for an overlapped operation can return immediately, even though the operation has not been completed. This enables a time-consuming I/O operation to be executed in the background while the calling thread is free to perform other tasks. For example, a single thread can perform simultaneous operations on different handles, or even simultaneous read and write operations on the same handle. To synchronize its execution with the completion of the overlapped operation, the calling thread uses the

ctGetOverlappedResult() function or one of the wait functions to determine when the overlapped operation has been completed. You can also use the **ctHas-OverlappedIoCompleted()** macro to poll for completion.

To call a function to perform an overlapped operation, the calling thread needs to specify a pointer to a **CTOVERLAPPED** structure. If this pointer is NULL, the function return value may incorrectly indicate that the operation completed. The **CTO-VERLAPPED** structure needs to contain a handle to a manual-reset, not an auto-reset event object. The system sets the state of the event object to non-signaled when a call to the I/O function returns before the operation has been completed. The system sets the state of the event object to signaled when the operation has been completed.

When a function is called to perform an overlapped operation, it is possible that the operation will be completed before the function returns. When this happens, the results are handled as if the operation had been performed synchronously. If the operation was not completed, however, the function's return value is FALSE, and the GetLastError() function returns ERROR_IO_PENDING.

A thread can manage overlapped operations by either of two methods:

- Use the **ctGetOverlappedResult()** function to wait for the overlapped operation to be completed.
- Specify a handle to the **CTOVERLAPPED** structure's manual-reset event object in one of the wait functions and then call **ctGetOverlappedResult()** after the wait function returns. The **ctGetOverlappedResult()** function returns the results of the completed overlapped operation, and for functions in which such information is appropriate, it reports the actual number of bytes that were transferred.

When performing multiple simultaneous overlapped operations, the calling thread needs to specify a **CTOVERLAPPED** structure with a different manual-reset event object for each operation. To wait for any one of the overlapped operations to be completed, the thread specifies the manual-reset event handles as wait criteria in one of the multiple-object wait functions. The return value of the multiple-object wait function indicates which manual-reset event object was signaled, so the thread can determine which overlapped operation caused the wait operation to be completed.

You can cancel a pending asynchronous operation using the ctCancelIO() function. Pending asynchronous operations are canceled when you call ctClose().

Reading Data Using the CTAPI Functions

Reading Data Using the CTAPI Functions

- List functions
- Array support
- Bit shifting when reading digital arrays

See Also
Function Reference

I/O tags interface

The CitectSCADA I/O Server is designed on a client read on demand basis. The Citect-SCADA I/O Server will read I/O tags from the I/O Devices when requested to by a Client. This reduces the load on the I/O Devices and increases the overall system performance.

The client interface to the real time data is more complex as the client needs to wait for a physical I/O cycle to complete before the data can be used. The client needs to request the data it requires from the I/O Server and then wait up to several seconds while the I/O Server reads the requested data. This design is reflected in the operation of the CTAPI interface. Using CTAPI to read a tag can take several seconds to complete. It is up the caller to allow for this in their design in calling this interface.

If you need to use a polling type of service, use the ctList functions.

See Also
The Tag functions

The Tag functions

The simplest way to read data is via the ctTagRead() function. This function reads the value of a single variable, and the result is returned as a formatted engineering string.

List functions

The List functions provide a higher level of performance for reading data than the tag based interface, The List functions also provide support for overlapped operations.

The List functions allow a group of tags to be defined and then read as a single request. They provide a simple tag based interface to data which is provided in formatted engineering data. You may create several lists and control each individually.

Tags can be added to, or deleted from lists dynamically, even if a read operation is pending on the list.

See Also
Array support

Array support

Arrays are supported in the tag functions ctTagWrite(), and ctTagRead(). These functions can take the singular tag name as "PV123", or use the array syntax as "Recipe[10]". When the array syntax is used in the "Recipe[10]" example, the single value can be read or written to, not the entire array.

See Also

Bit shifting when reading digital arrays

Bit shifting when reading digital arrays

When digital types are read, CitectSCADA may adjust the starting position of the first point. This is done to improve the performance of the digital read. For example, if you start reading an array of digital values, CitectSCADA may read several digitals before the start of the array, and the data will be offset. When CitectSCADA shifts the bits, extra data will be read from the I/O Device. CitectSCADA may shift the data up to 15 bits, resulting in an extra 2 bytes of buffer space necessary for reads. Therefore, always use digital buffers which contain 2 bytes extra.

CTAPI from CitectSCADA or CitectSCADA Driver

The CTAPI has been designed to be called from external applications. This API has not been designed to be called from the CitectSCADA Cicode DLL functions or from a Citect-SCADA protocol driver. Calling the CTAPI from Cicode DLL functions or a CitectSCADA protocol driver may cause a deadlock condition to occur. This will result in Citect-SCADA and the protocol driver hanging. If you need to call the CTAPI from a protocol driver, you need to create a new Win32 thread to call the API. You cannot call the CTAPI from the Cicode DLL interface.

See Also

Function Reference

Error Codes

Error Codes

The error codes returns from the CTAPI functions are the Microsoft WIN 32 error codes. These error codes are documented in the Microsoft SDKs. Where the error code is a CitectSCADA special error code, the error code is added to the value -ERROR_USER_DEFINED_BASE.

Note: If a CTAPI function returns the error 233, it typically means the connection to the client is not established. However, it may also mean the client has not logged in correctly. confirm both scenarios.

Example

```
int bRet = ctTagWrite(hCTAPI, "SP123", "12.34");
if (bRet == 0) {
        dwStatus = GetLastError();
        if (dwStatus < ERROR_USER_DEFINED_BASE) {
                // Microsoft error codes see ERROR.H
        } else {
                short    status;
                // status is theCitectSCADA error codes, see CitectSCADA help
                status = dwStatus - ERROR_USER_DEFINE_BASE;
        }
}
```

The following defines have been declared to make this checking easier:

```
IsCitectError(dwStatus)                 // test if CitectSCADA
error
WIN32_TO_CT_ERROR(dwStatus)             // Convert to CitectSCADA
status
```

For example:

```
if (IsCitectError(dwStatus)) {
        short    status;
        // status is the CitectSCADA error codes, see CitectSCADA help
        status = WIN32_TO_CT_ERROR(dwStatus);
}
```

If the connection is lost between your application and CitectSCADA, you need to close the connection and reopen. An inoperative connection will be shown by the returning of a Microsoft error code. If a CitectSCADA status error is returned, the connection has not been lost. The command requested is invalid and the connection does not have to be closed and reopened.

```
int bRet = ctTagWrite(hCTAPI, "SP123", "12.34");
if (bRet == 0) {
        dwStatus = GetLastError();
        if (dwStatus < ERROR_USER_DEFINED_BASE) {
                ctClose(hCTAPI);
                hCTAPI = ctOpen(NULL, NULL, NULL, 0);
                while (hCTAPI == NULL) {
                        Sleep(2000); // wait a while
```

```
                        hCTAPI = ctOpen(NULL, NULL, NULL, 0);
                }
        }
}
```

When the connection between your application and CitectSCADA is lost, any pending overlapped commands will time out and be canceled by CTAPI. You need to destroy handles which are associated with the connection.

In Version 5.10, the CT_OPEN_RECONNECT mode was added to ctOpen(). When this mode is enabled, CTAPI will attempt to re-establish the connection to CitectSCADA if a communication interruption occurs. Handles created with the connection will remain valid when the connection is re-created. While the connection is down, functions will be ineffective and will report errors.

See Also

Debug Tracing

Debug Tracing

Debug tracing of the CTAPI has been added to the kernel. You may enable the debug trace with the command CTAPI 1 in the main kernel window. CTAPI 0 will disable the debug tracing. You may also enable the debug tracking by setting the CITECT.INI parameter:

```
[CTAPI]
Debug=1
```

The debug tracing will display each client CTAPI traffic to CitectSCADA. This tracing may slow down the performance of CitectSCADA and the CTAPI client if a large amount of communication is occurring.

The debug trace is displayed in the main CitectSCADA kernel window and is logged to the syslog.dat file.

Function Reference

The CTAPI functions allow access to CitectSCADA I/O variable tags via a DLL interface. This allows third-party developers to create applications in C or other languages to read and write to the I/O Devices.

Function	Argument(s)	Type	Description

ctCancelIO	hCTAPI, pctO-verlapped	Boolean	Cancels a pending overlapped I/O oper-ation.
ctCiCode	hCTAPI, sCmd, hWin, nMode, sResult, dwLength, pctO-verlapped	DWORD	Executes a Cicode function.
ctClientCreate	*()*	n/a	Initializes the resources for a new CtAPI client instance
ctClientDestroy	hCTAPI	Boolean	The handle to the CTAPI as returned from ctOpen().
ctClose	hCTAPI	Boolean	Closes a connection to the CitectSCADA API.
ctCloseEx	hCTAPI,bDestroy	Handle	The handle to the CTAPI as returned from ctOpen().
ctEngToRaw	pResult, dValue, pScale, dwMode	Boolean	Converts the engi-neering scale variable into raw I/O Device scale.
ctFindClose	hnd	Boolean	Closes a search ini-tiated by ctFindFirst().
ctFindFirst	hCTAPI, szTa-bleName, szFilter, pObjHnd, dwFlags	Handle	Searches for the first object in the specified database which sat-isfies the filter string.
ctFindFirstEx	hCTAPI, szTa-bleName, szFilter, szCluster, pObjHnd, dwFlags	Handle	Searches for the first object in the specified database which sat-isfies the filter string specified by cluster.
ctFindNext	hnd, pObjHnd	Boolean	Retrieves the next object in a search ini-tiated by ctFindFirst().
ctFindPrev	hnd, pObjHnd	Boolean	Retrieves the previous object in a search ini-

			tiated by ctFindFirst().
ctFindScroll	hnd, dwMode, dwOffset, pObjHnd	Handle	Scrolls to the necessary object in a search initiated by ctFindFirst().
ctGetOverlappedResult	hCTAPI, lpctOverlapped, pBytes, bWait	Boolean	Returns the results of an overlapped operation.
ctGetProperty	hnd, szName, pData, dwBufferLength, dwResultLength, dwType	Boolean	Retrieves an object property.
ctHasOverlappedIoCompleted	lpctOverlapped	Boolean	Checks for the completion of an outstanding I/O operation.
ctListAdd	hList, sTag	Handle	Adds a tag to the list.
ctListAddEx	hList, sTag, bRaw, nPollPeriodMS, dDeadband	Handle	Adds a tag to the list with a specified poll period.
ctListData	hTag, pBuffer, dwLength, dwMode	Boolean	Gets the value of a tag on the list.
ctListDelete	hTag	Boolean	Frees a tag created with ctListAdd().
ctListEvent	hCTAPI, dwMode	Handle	Returns the elements in the list which have changed state since they were last read using the ctListRead() function.
ctListFree	hList	Boolean	Frees a list created with ctListNew().
ctListItem	hTag, dwitem, pBuffer, dwLength, dwMode	Boolean	Gets the tag element item data.

ctListNew	hCTAPI, dwMode	Handle	Creates a new list.
ctListRead	hList, pctOverlapped	Boolean	Reads every tag on the list.
ctListWrite	hTag, sValue, pctOverlapped	Boolean	Writes to a single tag on the list.
ctOpen	sComputer, sUser, sPassword, nMode	Handle	Opens a connection to the CitectSCADA API.
ctOpenEx	sComputer, sUser, sPassword, nMode, hCTAPI	Handle	Establishes the connection to the CtAPI server using the given client instance.
ctRawToEng	pResult, dValue, pScale, dwMode	Boolean	Converts the raw I/O Device scale variable into engineering scale.
ctTagGetProperty	hCTAPI, szTagName, szProperty, pData, dwBufferLength, dwType	Boolean	Gets the given property of the given tag.
ctTagRead	hCTAPI, sTag, sValue, dwLength	Boolean	Reads the current value from the given I/O Device variable tag.
ctTagReadEx	hCTAPI, sTag, sValue, dwLength, pctTagvalueItems	Boolean	Performs the same as ctTagRead, but with an additional new argument
ctTagWrite	hCTAPI, sTag, sValue	Boolean	Writes the given value to the I/O Device variable tag.
ctTagWriteEx	hCTAPI, sTag, sValue, pctOverlapped	Boolean	Performs the same as ctTagWrite, but with an additional new argument.

ctCancelIO

Cancels a pending overlapped I/O operation. When the I/O command is canceled, the event will be signaled to show that the command has completed. The status will be set to the CitectSCADA error **CT_ERROR_CANCELED**. If the command completes before you can cancel it, **ctCancelIO()** will return **FALSE**, and **GetLastError()** will return **GENERIC_CANNOT_CANCEL**. The status of the overlapped operation will be the completion status of the command.

The CTAPI interface will automatically cancel any pending I/O commands when you call ctClose().

Syntax

ctCancelIO(*hCTAPI, pctOverlapped*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

pctOverlapped

Type: CTOVERLAPPED*
Input/output: Input
Description: Pointer to the overlapped I/O operation to cancel. If you specify NULL, any pending overlapped I/O operations on the interface will be canceled.

Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call **GetLastError**.

Related Functions

ctOpen, ctClose

Example

```
char                        sVersion[128];
CTOVERLAPPED                        ctOverlapped;
ctOverlapped.hEvent = CreateEvent(NULL, TRUE, TRUE, NULL);
ctCicode(hCTAPI, "Version(0)", 0, 0, sVersion, sizeof(sVersion),
&ctOverlapped);
ctCancelIO(hCTAPI, &ctOverlapped);
```

ctCiCode

Executes a Cicode function on the connected CitectSCADA computer. This allows you to control CitectSCADA or to get information returned from Cicode functions. You may call either built in or user defined Cicode functions. Cancels a pending overlapped I/O operation.

The function name and arguments to that function are passed as a single string. Standard CitectSCADA conversion is applied to convert the data from string type into the type expected by the function. When passing strings put the strings between the CitectSCADA string delimiters.

Functions which expect pointers or arrays are not supported. Functions which expect pointers are functions which update the arguments. This includes functions DspGetMouse(), DspAnGetPos(), StrWord(), and so on. Functions which expect arrays to be passed or returned are not supported, for example TableMath(), TrnSetTable(), TrnGetTable(). You may work around these limitations by calling a Cicode wrapper function which in turn calls the function you require.

If the Cicode function you are calling takes a long time to execute, is pre-empt or blocks, then the result of the function cannot be returned in the sResult argument. The Cicode function will, however, execute correctly.

Syntax

ctCiCode(hCTAPI, sCmd, hWin, nMode, sResult, dwLength, pctOverlapped)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

sCmd

Type: String
Input/output: Input
Description: The command to execute.

vhWin

Type: Dword
Input/output: Input
Description: The CitectSCADA window to execute the function. This is a logical CitectSCADA window (0, 1, 2, 3 etc.) not a Windows Handle.

nMode

Type: Dword
Input/output: Input
Description: The mode of the Cicode call. Set this to 0 (zero).

sResult

Type: LPSTR
Input/output: Output
Description: The buffer to put the result of the function call, which is returned as a string. This may be NULL if you do not need the result of the function.

dwLength

Type: Dword
Input/output: Input
Description: The length of the sResult buffer. If the result of the Cicode function is longer than the this number, then the result is not returned and the function call does not succeed, however the Cicode function is still executed. If the sResult is NULL then this length needs to be 0.

pctOverlapped

Type: CTOVERLAPPED*
Input/output: Input
Description: CTOVERLAPPED structure. This structure is used to control the overlapped notification. Set to NULL if you want a synchronous function call.

Return Value

Type: Dword. TRUE if successful, otherwise FALSE. Use **GetLastError()** to get extended error information.

Related Functions

ctOpen

Example

```
char    sName[32];
ctCicode(hCTAPI, "AlarmAck(0,)", 0, 0, NULL, 0, NULL);
ctCicode(hCTAPI, "PageInfo(0)", 0, 0, sName, sizeof(sName), NULL);
/* to call the Prompt function with the string "Hello Citect", the
C code would be:
*/

ctCicode(hCTAPI, "Prompt(\"Hello Citect\")", 0, 0, NULL, 0, NULL);
/* If the string does not contain any delimiters (for example spaces or commas) you
may omit the string delimiters. For example to display a page called "Menu" the C
code would be:
*/
ctCicode(hCTAPI, "PageDisplay(Menu)", 0, 0, NULL, 0, NULL);
```

ctClientCreate

ctClientCreate initializes the resources for a new CtAPI client instance. Once you have called ctClientCreate, you can pass the handle returned to ctOpenEx to establish communication with the CtAPI server.

Consider a situation where you try to communicate to the CtAPI server and the server takes a long time to respond (or doesn't respond at all). If you just call ctOpen, you haven't been given a handle to the CtAPI instance, so you can't cancel the ctOpen by calling ctCancelIO. But if you use ctClientCreate and then call ctOpenEx, you can use the handle returned by ctClientCreate to cancel the ctOpenEx.

Syntax

ctClientCreate()

Return Value

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. Use GetLastError() to get extended error information.

Related Functions

ctOpen, ctOpenEx, ctClose, ctCloseEx, ctClientDestroy

Example

```
DWORD dwStatus = 0;
HANDLE hCtapi = ctClientCreate();
if (hCtapi == NULL) {
        dwStatus = GetLastError(); // An error has occurred, trap it.
} else {
        if (TRUE == ctOpenEx(NULL, NULL, NULL, 0, hCtapi)) {
                ctTagWrite(hCtapi, "Fred", "1.5");
                if (FALSE == ctCloseEx(hCtapi, FALSE)) {
                        dwStatus = GetLastError(); // An error has occurred, trap it.
                }
        } else {
                dwStatus = GetLastError(); // An error has occurred, trap it.
        }
        if (FALSE == ctClientDestroy(hCtapi)) {
                dwStatus = GetLastError(); // An error has occurred, trap it
        }
}
```

ctClientDestroy

Cleans up the resources of the given CtAPI instance. Unlike ctClose, ctClientDestroy does not close the connection to the CtAPI server.

You need to call ctCloseEx with *bDestroy* equal to FALSE before calling ctClientDestroy.

Syntax

ctClientDestroy(*hCTAPI*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

Related Functions

ctCloseEx, ctClose, ctClientCreate, ctOpen, ctOpenEx

Example

See ctClientCreate for an example.

ctClose

Closes the connection between the application and the CtAPI. When called, any pending commands will be canceled. You need to free any handles allocated before calling ctClose(). These handles are not freed when ctClose() is called. Call this function from an application on shutdown or when a major error occurs on the connection.

Syntax

ctClose(*hCTAPI*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

Related Functions

ctOpen

Example

See the example for ctOpen().

ctCloseEx

Closes the connection to the CtAPI server for the given CtAPI instance. It closes the connection the same way as does the ctClose method, but provides an option for whether or not to destroy the CtAPI instance within the ctCloseEx function call. ctClose always destroys the CtAPI instance within its function call.

For example, consider a situation where when we try to close the connection to the CtAPI server and it takes a long time to respond (or doesn't at all). If you call ctClose, you can't cancel the ctClose by calling ctCancelIO because you can't guarentee that the CtAPI instance is not in the process of being destroyed. But if you call ctCloseEx with the option of not destroying the CtAPI instance, you can call ctCancelIO to cancel the ctCloseEx.

When you call ctCloseEx with *bDestroy* equal to FALSE, you need to then call ctClient-Destroy afterwards to free the CtAPI client instance.

Syntax

ctCloseEx(*hCTAPI,bDestroy*);

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

bDestroy

Type: boolean
Input/output: Input
Description: If TRUE will destroy the CtAPI instance within the ctCloseEx function call. Default is FALSE.

Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

Related Functions

ctClientDestroy, ctClose, ctClientCreate, ctOpen, ctOpenEx

Example

See ctClientCreate for an example.

ctEngToRaw

Converts the engineering scale variable into raw I/O Device scale. This is not necessary for the Tag functions as CitectSCADA will do the scaling. Scaling is not necessary for digitals, strings or if no scaling occurs between the values in the I/O Device and the Engineering values. You need to know the scaling for each variables as specified in the CitectSCADA Variable Tags table.

Syntax

ctEngToRaw(*pResult, dValue, pScale, dwMode*)

pResult

Type: Double
Input/output: Output
Description: The resulting raw scaled variable.

dValue

Type: Double
Input/output: Input
Description: The engineering value to scale.

pScale

Type: CTSCALE*
Input/output: Input
Description: The scaling properties of the variable.

dwMode

Type: Dword
Input/output: Input
Description: The mode of the scaling:

CT_SCALE_RANGE_CHECK: Range check the result. If the variable is out of range then generate an error. The pResult still contains the raw scaled value.

CT_SCALE_CLAMP_LIMIT: Clamp limit to maximum or minimum scales. If the result is out of scale then set result to minimum or maximum scale (which ever is closest). No error is generated if the scale is clamped. Cannot be used with CT_SCALE_RANGE_CHECK or CT_SCALE_NOISE_FACTOR options.

CT_SCALE_NOISE_FACTOR: Allow noise factor for range check on limits. If the variable is our of range by less than 0.1 % then a range error is not generated.

Return Value

TRUE if successful, otherwise FALSE. Use **GetLastError()** to get extended error information.

Related Functions

ctOpen, ctRawToEng, ctTagRead

Example

```
CTSCALE        Scale    = { 0.0, 32000.0, 0.0, 100.0};
double            dSetPoint = 42.23;
double            dRawValue;
ctEngToRaw(&dRawValue, dSetPoint, &Scale, CT_SCALE_RANGE_CHECK);
```

ctFindClose

Closes a search initiated by ctFindFirst.

Syntax

ctFindClose(hnd)

hnd

Type: Handle
Input/output: Input
Description: Handle to the search, as returned by ctFindFirst().

Return Value

If the function succeeds, the return value is non-zero. If the function does not succeed, the return value is zero. To get extended error information, call **GetLastError()**.

Related Functions

ctOpen, ctFindNext, ctFindPrev, ctFindScroll, ctGetProperty

Example

See ctFindFirst

ctFindFirst

Searches for the first object in the specified table, device, trend, or alarm data which satisfies the filter string. A handle to the found object is returned via pObjHnd. The object handle is used to retrieve the object properties. To find the next object, call the ctFindNext function with the returned search handle.

If you experience server performance problems when using ctFindFirst() refer to CPULoadCount and CpuLoadSleepMS.

Syntax

ctFindFirst(*hCTAPI, szTableName, szFilter, pObjHnd,dwFlags*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

szTableName

Type: LPCTSTR
Input/output: Input
Description: The table, device, trend, or alarm data to be searched. The following tables and fields can be searched:

- **Trend** - Trend Tags
  CLUSTER, NAME/TAG, RAW_ZERO, RAW_FULL, ENG_ZERO, ENG_FULL, ENG_UNITS, COMMENT, SAMPLEPER, TYPE
- **DigAlm** - Digital Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT
- **AnaAlm** - Analog Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, ALMCOMMENT
- **AdvAlm** - Advanced Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT
- **HResAlm** - Time-Stamped Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, MILLISEC, DATE, AREA, ALMCOMMENT
- **ArgDigAlm** - Argyle Digital Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, PRIORITY, STATE_DESC, OLD_DESC

- **ArgAnaAlm** - Argyle Analog Alarm Tags
  CLUSTER, TAG, NAME, HELP, ALMCOMMENT, CATEGORY, STATE, TIME, DATE, AREA, VALUE, PRIORITY, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION
- **TsDigAlm** - Time-Stamped Digital Alarm Tags
  CLUSTER, TAG, NAME, DESC, CATEGORY, AREA, ALMCOMMENT
- **TsAnaAlm** - Time-Stamped Analog Alarm Tags
  CLUSTER, TAG, NAME, DESC, CATEGORY, AREA, ALMCOMMENT
- **ArgDigAlmStateDesc** - Argyle Digital Alarm Tag State Descriptions
  CLUSTER, TAG, STATE_DESC0, STATE_DESC1, STATE_DESC2, STATE_DESC3, STATE_DESC4, STATE_DESC5, STATE_DESC6, STATE_DESC7
- **Alarm** - Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, PRIORITY, STATE_DESC, OLD_DESC, ALARMTYPE
- **AlarmSummary** - Alarm Summary
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, PRIORITY, STATE_DESC, OLD_DESC, ALARMTYPE, ONDATE, ONDATEEXT, ONTIME, ONMILLI, OFFDATE, OFFDATEEXT, OFFTIME, OFFMILLI, DELTATIME, ACKDATE, ACKDATEEXT, ACKTIME, ALMCOMMENT, USERNAME, FULLNAME, USERDESC, SUMSTATE, SUMDESC, NATIVE_SUMDESC, COMMENT, NATIVE_COMMENT
- **Accum** - Accumulators
  PRIV, AREA, CLUSTER, NAME, TRIGGER, VALUE, RUNNING, STARTS, TOTALISER
- **Tag** - Variable Tags
- **LocalTag** - Local Tags
- **Cluster** - Clusters

For information on fields, see the Browse Function Field Reference in the Cicode Reference Guide.

> **Note:** The migration tool in CitectSCADAv7.20 converts memory PLC variables to local variable tags which are in a separate table to the variable tags. Calling ctFindFirst with *szTableName* "Tag" will not return the local variable tags. In order to return the local variable tags you need to call ctFindFirst with the *szTableName* of "LocalTag". Local variables do not have clusters and have only one pair of zero/full scales (as opposed to raw and engineering scales for variable tags).

The field names for local variable tags are:

NAME, TYPE, ASIZE (array size), ZERO, FULL, UNITS, COMMENT.

The array size field is available only for local tags.

szFilter

Type: LPCTSTR
Input/output: Input
Description: Filter criteria. This is a string based on the following format:

"PropertyName1=FilterCriteria1;PropertyName2=FilterCriteria2"

The wildcard * may be used as part of the filter criteria to match multiple entries. Use an empty string, or "*" as the filter string to match every entry.

pObjHnd

Type: HANDLE
Input/output: Output
Description: The pointer to the found object handle. This is used to retrieve the properties.

dwFlags

This argument is no longer used, pass in a value of 0 for this argument.

To search a table:

In *szTableName* specify the name of the table.

To search a device:

In *szTableName* specify the name as defined in the CitectSCADA Devices form, for example "RECIPES" for the Example project.

To search trend data:

In *szTableName* specify the trend using the following format (including the quotation marks):

`TRNQUERY,*Endtime,EndtimeMs,Period,NumSamples,Tagname,Displaymode,Datamode*'

See TrnQuery for syntax details.

To search alarm data:

In *szTableName* specifythe alarm data using the following format (including the quotation marks):

`ALMQUERY,*Database,TagName,Starttime,StarttimeMs,Endtime,EndtimeMs,Period*'

See AlmQuery for syntax details.

Return Value

If the function succeeds, the return value is a search handle used in a subsequent call to ctFindNext() or ctFindClose(). If the function does not succeed, the return value is **NULL**. To get extended error information, call **GetLastError()**

Related Functions

ctOpen, ctFindNext, ctFindClose, ctGetProperty, ctFindFirstEx

Example

```
HANDLE     hSearch;
HANDLE     hObject;
HANDLE     hFind;
// Search the Tag table
hSearch = ctFindFirst(hCTAPI, "Tag", NULL, &hObject, 0);
if (hSearch == NULL) {
        // no tags found
} else {
        do {
                char    sName[32];
                // Get the tag name
                ctGetProperty(hObject, "Tag", sName, sizeof(sName), NULL,
               DBTYPE_STR);
        } while (ctFindNext(hSearch, &hObject));
        ctFindClose(hSearch);
        }
// Get Historical Trend data via CTAPI
// Get 100 samples of the CPU trend at 2 second
hFind = ctFindFirst(hCTAPI, "CTAPITrend(\"10:15:00 \", \"11/8/1998\", 2, 100, 0,
 \"CPU\")", &hObject, 0);
while (hFind) {
        char    sTime[32], sDate[32], sValue[32];
        ctGetProperty(hObject, "TIME", sTime, sizeof(sTime), NULL, DBTYPE_STR);
        ctGetProperty(hObject, "DATE", sDate, sizeof(sDate), NULL, DBTYPE_STR);
        ctGetProperty(hObject, "CPU", sValue, sizeof(sValue), NULL, DBTYPE_STR);
        // do something with the trend data.
        if (!ctFindNext(hFind, &hObject)) {
                ctFindClose(hFind);
                hFind = NULL;
                break;
        }
}
```

ctFindFirstEx

Performs the same as ctFindFirst, but with an additional new argument. Searches for the first object in the specified table, device, trend, or alarm data which satisfies the filter string. A handle to the found object is returned via pObjHnd. The object handle is used to retrieve the object properties. To find the next object, call the ctFindNext function with the returned search handle.

If you experience server performance problems when using ctFindFirst() refer to CPU-LoadCount and CpuLoadSleepMS.

If ctFindFirst is called instead of ctFindFirstEx, the szCluster defaults to NULL.

Syntax

ctFindFirstEx*(hCTAPI, szTableName, szFilter, szCluster, pObjHnd, dwFlags)*

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

szTableName

Type: LPCTSTR
Input/output: Input
Description: The table, device, trend, or alarm data to be searched. The following tables and fields can be searched:

- **Trend** - Trend Tags
  CLUSTER, NAME/TAG, RAW_ZERO, RAW_FULL, ENG_ZERO, ENG_FULL, ENG_UNITS, COMMENT, SAMPLEPER, TYPE
- **DigAlm** - Digital Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT
- **AnaAlm** - Analog Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, ALMCOMMENT
- **AdvAlm** - Advanced Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT
- **HResAlm** - Time-Stamped Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, MILLISEC, DATE, AREA, ALMCOMMENT
- **ArgDigAlm** - Argyle Digital Alarm Tags

CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, PRIORITY, STATE_DESC, OLD_DESC

- **ArgAnaAlm** - Argyle Analog Alarm Tags
  CLUSTER, TAG, NAME, HELP, ALMCOMMENT, CATEGORY, STATE, TIME, DATE, AREA, VALUE, PRIORITY, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION
- **TsDigAlm** - Time-Stamped Digital Alarm Tags
  CLUSTER, TAG, NAME, DESC, CATEGORY, AREA, ALMCOMMENT
- **TsAnaAlm** - Time-Stamped Analog Alarm Tags
  CLUSTER, TAG, NAME, DESC, CATEGORY, AREA, ALMCOMMENT
- **ArgDigAlmStateDesc** - Argyle Digital Alarm Tag State Descriptions
  CLUSTER, TAG, STATE_DESC0, STATE_DESC1, STATE_DESC2, STATE_DESC3, STATE_DESC4, STATE_DESC5, STATE_DESC6, STATE_DESC7
- **Alarm** - Alarm Tags
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, STATE, TIME, DATE, AREA, ALMCOMMENT, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, PRIORITY, STATE_DESC, OLD_DESC, ALARMTYPE
- **AlarmSummary** - Alarm Summary
  CLUSTER, TAG, NAME, DESC, HELP, CATEGORY, TIME, DATE, AREA, VALUE, HIGH, LOW, HIGHHIGH, LOWLOW, DEADBAND, RATE, DEVIATION, PRIORITY, STATE_DESC, OLD_DESC, ALARMTYPE, ONDATE, ONDATEEXT, ONTIME, ONMILLI, OFFDATE, OFFDATEEXT, OFFTIME, OFFMILLI, DELTATIME, ACKDATE, ACKDATEEXT, ACKTIME, ALMCOMMENT, USERNAME, FULLNAME, USERDESC, SUMSTATE, SUMDESC, NATIVE_SUMDESC, COMMENT, NATIVE_COMMENT
- **Accum** - Accumulators
  PRIV, AREA, CLUSTER, NAME, TRIGGER, VALUE, RUNNING, STARTS, TOTALISER
- **Tag** - Variable Tags
- **LocalTag** - Local Tags
- **Cluster** - Clusters

For information on fields, see the Browse Function Field Reference in the Cicode Reference Guide.

szFilter

Type: LPCTSTR
Input/output: Input
Description: Filter criteria. This is a string based on the following format:

"PropertyName1=FilterCriteria1;PropertyName2=FilterCriteria2"\.

"*" as the filter to achieve the same result.

szCluster

Type: LPCTSTR
Input/output: Input
Description: Specifies on which cluster the ctFindFirst function will be performed. If left NULL or empty string then the ctFindFirst will be performed on the active cluster if there is only one.

pObjHnd

Type: HANDLE
Input/output: Output
Description: The pointer to the found object handle. This is used to retrieve the properties.

dwFlags

This argument is no longer used, pass in a value of 0 for this argument.

To search a table:

In *szTableName* specify the name of the table.

To search a device:

In *szTableName* specify the name as defined in the CitectSCADA Devices form, for example "RECIPES" for the Example project.

To search trend data:

In *szTableName* specify the trend using the following format (including the quotation marks):

`TRNQUERY,*Endtime,EndtimeMs,Period,NumSamples,Tagname,Displaymode,Datamode*'

See TrnQuery for syntax details.

To search alarm data:

In *szTableName* specifythe alarm data using the following format (including the quotation marks):

`ALMQUERY,*Database,TagName,Starttime,StarttimeMs,Endtime,EndtimeMs,Period*'

See AlmQuery for syntax details.

Return Value

If the function succeeds, the return value is a search handle used in a subsequent call to **ctFindNext()** or **ctFindClose()**. If the function does not succeed, the return value is **NULL**. To get extended error information, call **GetLastError()**

Related Functions

ctOpen, ctFindNext, ctFindClose, ctGetProperty, ctFindFirst

ctFindNext

Retrieves the next object in the search initiated by ctFindFirst.

Syntax

ctFindNext(*hnd, pObjHnd)*

hnd

Type: Handle
Input/output: Input
Description: Handle to the search, as returned by ctFindFirst().

pObjHnd

Type: HANDLE
Input/output: Output
Description: The pointer to the found object handle. This is used to retrieve the properties.

Return Value

If the function succeeds, the return value is TRUE (1). If the function does not succeed, the return value is FALSE (0). To get extended error information, call GetLastError(). If you reach the end of the search, GetLastError() returns CT_ERROR_NOT_FOUND. Once past the end of the search, you cannot scroll the search using ctFindNext() or ctFindPrev() commands. You need to reset the search pointer by creating a new search using ctFindFirst(), or by using the ctFindScroll() function to move the pointer to a valid position.

Related Functions

ctOpen, ctFindFirst, ctFindPrev, ctFindClose, ctGetProperty

Example

See ctFindFirst.

ctFindPrev

Retrieves the previous object in the search initiated by ctFindFirst.

Syntax

ctFindPrev(*hnd, pObjHnd*)

hnd

Type: Handle
Input/output: Input
Description: Handle to the search, as returned by ctFindFirst().

pObjHnd

Type: HANDLE
Input/output: Output
Description: The pointer to the found object handle. This is used to retrieve the properties.

Return Value

If the function succeeds, the return value is TRUE (1). If the function does not succeed, the return value is FALSE (0). To get extended error information, call GetLastError(). If you reach the end of the search, GetLastError() returns CT_ERROR_NOT_FOUND. Once past the end of the search, you cannot scroll the search using ctFindNext() or ctFind-Prev() commands. You need to reset the search pointer by creating a new search using ctFindFirst(), or by using the ctFindScroll() function to move the pointer to a valid position.

Related Functions

ctOpen, ctFindFirst, ctFindNext, ctFindClose, ctGetProperty

Example

See ctFindFirst

ctFindScroll

Scrolls to the necessary object in the search initiated by ctFindFirst.

To find the current scroll pointer, you can scroll relative (dwMode = CT_FIND_SCROLL_RELATIVE) with an offset of 0. To find the number of records returned in a search, scroll to the end of the search.

Syntax

ctFindScroll(*hnd, dwMode, dwOffset, pObjHnd*)

hnd

Type: Handle
Input/output: Input
Description: Handle to the search, as returned by ctFindFirst().

dwMode

Type: DWORD
Input/output:
Description: Mode of the scroll. The following modes are supported:

CT_FIND_SCROLL_NEXT: Scroll to the next record. The dwOffset parameter is ignored.

CT_FIND_SCROLL_PREV: Scroll to the previous record. The dwOffset parameter is ignored.

CT_FIND_SCROLL_FIRST: Scroll to the first record. The dwOffset parameter is ignored.

CT_FIND_SCROLL_LAST: Scroll to the last record. The dwOffset parameter is ignored.

CT_FIND_SCROLL_ABSOLUTE: Scroll to absolute record number. The record number is specified in the dwOffset parameter. The record number is from 1 to the maximum number of records returned in the search.

CT_FIND_SCROLL_RELATIVE: Scroll relative records. The number of records to scroll is specified by the dwOffset parameter. If the offset is positive, this function will scroll to the next record, if negative, it will scroll to the previous record. If 0 (zero), no scrolling occurs.

dwOffset

Type: LONG
Input/output: Input
Description: Offset of the scroll. The meaning of this parameter depends on the dwMode of the scrolling operation.

pObjHnd

Type: HANDLE
Input/output: Output
Description: The pointer to the found object handle. This is used to retrieve the properties.

pObjHnd

Type: HANDLE
Input/output: Output
Description: The pointer to the found object handle. This is used to retrieve the properties.

Return Value

If the function succeeds, the return value is non-zero. If the function does not succeed, the return value is zero. To get extended error information, call **GetLastError()**. If no matching objects can be found, the **GetLastError()** function returns **CT_ERROR_NOT_FOUND**. The return value is the current record number in the search. Record numbers start at 1 (for the first record) and increment until the end of the search has been reached. Remember, 0 (zero) is not a valid record number - it signifies that the function was not successful.

Related Functions

ctOpen, ctFindFirst, ctFindNext, ctFindPrev, ctFindClose, ctGetProperty

Example

```
HANDLE    hSearch;
HANDLE    hObject;
DWORD    dwNoRecords;
// Search the Tag table
hSearch = ctFindFirst(hCTAPI, "Tag", NULL, &hObject, 0);
// Count number of records
dwNoRecords = ctFindScroll(hSearch, CT_FIND_SCROLL_LAST, 0, &hObject);
// scroll back to beginning
ctFindScroll(hSearch, CT_FIND_SCROLL_FIRST, 0, &hObject);
do {
        char    sName[32];
        // Get the tag name
        ctGetProperty(hObject, "Tag", sName, sizeof(sName), NULL, DBTYPE_STR);
} while (ctFindScroll(hSearch, CT_FIND_SCROLL_NEXT, 0, &hObject));
ctFindClose(hSearch);
```

ctGetOverlappedResult

Returns the results of an overlapped operation. The results reported by the **ctGetOverlappedResult()** function are those of the specified handle's last CTOVERLAPPED operation to which the specified **CTOVERLAPPED** structure was provided, and for which the operation's results were pending. A pending operation is indicated when the function that started the operation returns FALSE, and the **GetLastError** function returns ERROR_IO_PENDING. When an I/O operation is pending, the function that started the operation resets the **hEvent** member of the **CTOVERLAPPED** structure to the non-signaled state. Then when the pending operation has been completed, the system sets the event object to the signaled state.

If the *bWait* parameter is TRUE, **ctGetOverlappedResult()** determines whether the pending operation has been completed by waiting for the event object to be in the signaled state.

Specify a manual-reset event object in the CTOVERLAPPED structure. If an auto-reset event object is used, the event handle needs to not be specified in any other wait operation in the interval between starting the CTOVERLAPPED operation and the call to **ctGetOverlappedResult()**. For example, the event object is sometimes specified in one of the wait functions to wait for the operation's completion. When the wait function returns, the system sets an auto-reset event's state to non-signaled, and a subsequent call to **ctGetOverlappedResult()** with the bWait parameter set to TRUE causes the function to be blocked indefinitely.

Syntax

ctGetOverlappedResult(*hCTAPI, lpctOverlapped, pBytes, bWait)*

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

lpctOverlapped

Type: CTOVERLAPPED*
Input/output: Input
Description: Address of the CTOVERLAPPED structure which was used when an overlapped operation was started.

pBytes

Type: DWORD*
Input/output: Input
Description: Address of actual bytes transferred. For the CTAPI this value is undefined.

bWait

Type: BOOL
Input/output: Input
Description: Specifies whether the function waits for the pending overlapped operation to be completed. If TRUE, the function does not return until the operation has been completed. If FALSE and the operation is still pending, the function returns FALSE and the GetLastError function returns ERROR_IO_INCOMPLETE.

Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. Use **GetLastError()** to get extended error information.

Related Functions

ctOpen, ctHasOverlappedIoCompleted

Example

```
DWORD                                Bytes;
char                                sVersion[128];
CTOVERLAPPED                                ctOverlapped;
ctOverlapped.hEvent = CreateEvent(NULL, TRUE, TRUE, NULL);
ctCicode(hCTAPI, "Version(0)", 0, 0, sVersion, sizeof(sVersion), &ctOverlapped);
//..
// do something else.
```

```
//..
// wait for the ctCicode to complete
ctGetOverlappedResult(hCTAPI, &ctOverlapped, &Bytes, TRUE);
```

ctGetProperty

Retrieves an object property or meta data for an object. Use this function in conjunction with the ctFindFirst() and ctFindNext() functions. i.e. First, you find an object, then you retrieve its properties.

To retrieve property meta data such as type, size and so on, use the following syntax for the szName argument:

- `object.fields.count` - the number of fields in the record
- `object.fields(n).name` - the name of the nth field of the record
- `object.fields(n).type` - the type of the nth field of the record
- `object.fields(n).actualsize` - the actual size of the nth field of the record

Syntax

ctGetProperty(*hnd, szName, pData, dwBufferLength, dwResultLength, dwType*)

hnd

Type: Handle
Input/output: Input
Description: Handle to the search, as returned by ctFindFirst().

szName

Type: LPCTSTR*
Input/output: Input
Description: The name of the property to be retrieved. The following properties are supported:

Name - The name of the tag.

FullName - The full name of the tag in the form *cluster.tagname*.

Network - The unique I/O Device Number.

BitWidth - Width of the data type in bits. for example digital will be 1, integer 16, long 32, etc.

UnitType - The protocol specific unit type.

UnitAddress - The protocol specific unit address.

UnitCount - The protocol specific unit count.

RawType - The raw data type of the point. The following types are returned: 0 (Digital), 1 (Integer), 2 (Real), 3 (BCD), 4 (Long), 5 (Long BCD), 6 (Long Real), 7 (String), 8 (Byte), 9 (Void), 10 (Unsigned integer).

Raw_Zero - Raw zero scale.

Raw_Full - Raw full scale.

Eng_Zero - Engineering zero scale.

Eng_Full - Engineering full scale.

pData

Type: VOID*
Input/output: Output
Description: The result buffer to store the read data. The data is raw binary data, no data conversion or scaling is performed. If this buffer is not large enough to receive the data, the data will be truncated, and the function will return false.

dwBufferLength

Type: DWORD
Input/output: Input
Description: Length of result buffer. If the result buffer is not large enough to receive the data, the data will be truncated, and the function will return false.

dwResultLength

Type: DWORD*
Input/output: Output
Description: Length of returned result. You can pass NULL if you want to ignore this parameter

dwType

Type: DWORD
Input/output: Input
Description: The desired return type as follows:

Value	Meaning
DBTYPE_UI1	UCHAR
DBTYPE _I1	1 byte INT
DBTYPE _I2	2 byte INT
DBTYPE _I4	4 byte INT
DBTYPE _R4	4 byte REAL
DBTYPE _R8	8 byte REAL
DBTYPE _BOOL	BOOLEAN
DBTYPE_BYTES	Byte stream

DBTYPE _STR	NULL Terminated STRING

Return Value

If the function succeeds, the return value is non-zero. If the function does not succeed, the return value is zero. To get extended error information, call GetLastError().

Related Functions

ctOpen, ctFindFirst, ctFindNext, ctFindPrev, ctFindClose

Example

Also see ctFindFirst().

```
// get the property of the TAG field
ctGetProperty(hObject, "TAG", sName, sizeof(sName), NULL, DBTYPE_STR);
// Use the meta property fields to enumerate the entire row of data
// first get number of fields in the row
ctGetProperty(hObject, "object.fields.count", &dwFields, sizeof(dwFields),
 NULL, DBTYPE_I4);
for (i = 0; i < dwFields; i++) {
        sprintf(sObject, "object.fields(%d).name", i + 1);
        // get name of field
        if (ctGetProperty(hObject, sObject, sName, sizeof(sName), NULL, DBTYPE_STR)) {
                // get value of field
                if (ctGetProperty(hObject, sName, sData, sizeof(sData),
                NULL, DBTYPE_STR)) {
                        printf("%8.8s ", sData);
                }
        }
}
```

ctHasOverlappedIoCompleted

Provides a high performance test operation that can be used to poll for the completion of an outstanding I/O operation.

Syntax

ctHasOverlappedIoCompleted(*lpctOverlapped*)

lpctOverlapped

Type: CTOVERLAPPED*
Input/output: Input
Description: Address of the CTOVERLAPPED structure which was used when an overlapped operation was started.

Return Value

TRUE if the I/O operation has completed, and FALSE otherwise.

Return Value

ctOpen, ctGetOverlappedResult

ctListAdd

Adds a tag or tag element to the list. Once the tag has been added to the list, it may be read using ctListRead() and written to using ctListWrite(). If a read is already pending, the tag will not be read until the next time ctListRead() is called. ctListWrite() may be called immediately after the ctListAdd() function has completed.

Syntax

ctListAdd(*hList, sTag*)

hList

Type: HANDLE
Input/output: Input
Description: The handle to the list, as returned from ctListNew().

sTag

Type: LPCSTR
Input/output: Input
Description: The tag or tag name and element name, separated by a dot to be added to the list. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference.

Return Value

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. To get extended error information, call GetLastError()

If a tag not currently defined in your system is specified using this function then the return value will specify a valid handle. Calling ctListRead will allow identification of the true state of the tag. Passing an empty tag to this function will result in the function exiting immediately and returning NULL.

Related Functions

ctOpen, ctListNew, ctListFree, ctListRead, ctListWrite, ctListData, ctListAddEx

Example

```
HANDLE  hCTAPI;
HANDLE  hList;
HANDLE  hTagOne;
HANDLE  hTagOneField;
HANDLE  hTagOneControlMode;
HANDLE  hTagOneStatus;
char   sProcessValue[20];
char   sProcessValueField[20];
char   sProcessValueControlMode[20];
char   sProcessValueStatus[20];
hCTAPI                        = ctOpen(NULL, NULL, NULL, 0);
hList                  = ctListNew(hCTAPI, 0);
hTagOne                       = ctListAdd(hList, "TagOne");
hTagOneField           = ctListAdd(hList, "TagOne.Field");
hTagOneControlMode     = ctListAdd(hList, "TagOne.ControlMode");
hTagOneStatus          = ctListAdd(hList, "TagOne.Status");
ctListRead(hList, NULL);

ctListData(hTagOne, sProcessValue, sizeof(sProcessValue), 0);
ctListData(hTagOneField, sProcessValueField, sizeof(sProcessValueField) , 0);
ctListData(hTagOneControlMode, sProcessValueControlMode, size-
of(sProcessValueControlMode) , 0);
ctListData(hTagOneStatus, sProcessValueStatus, sizeof(sProcessValueStatus) , 0);
ctListFree(hList);
```

ctListAddEx

Performs the same as ctListAdd, but with 2 additional new arguments. Adds a tag, or tag element, to the list. Once the tag has been added to the list, it may be read using ctListRead() and written to using ctListWrite(). If a read is already pending, the tag will not be read until the next time ctListRead() is called. ctListWrite() may be called immediately after the ctListAdd() function has completed.

If ctListAdd is called instead of ctListAddEx, The poll period of the subscription for the tag defaults to 500 milliseconds, and the bRaw flag defaults to the engineering value of FALSE.

Syntax

ctListAddEx(*hList, sTag, bRaw, nPollPeriodMS, dDeadband*)

hList

Type: HANDLE
Input/output: Input
Description: The handle to the list, as returned from ctListNew().

sTag

Type: LPCSTR
Input/output: Input
Description: The tag or tag name and element name, separated by a dot to be added to the list. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference.

bRaw

Type: BOOL
Input/output: Input
Description: Specifies whether to subscribe to the given tag in the list using raw mode if TRUE or engineering mode if FALSE.

nPollPeriodMS

Type: INTEGER
Input/output: Input
Description: Dictates the poll period used in the subscription made for the tag (in milliseconds).

dDeadband

Type: DOUBLE
Input/output: Input
Description: Percentage of the variable tag's engineering range that a tag needs to change by in order for an update to be sent through the system. A value of -1.0 indicates that the default deadband specified by the tag definition is to be used.

Return Value

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. To get extended error information, call GetLastError()

If a tag not currently defined in your system is specified using this function then the return value will specify a valid handle. Calling ctListRead will allow identification of the true state of the tag. Passing an empty tag to this function will result in the function exiting immediately and returning NULL.

Related Functions

ctOpen, ctListNew, ctListFree, ctListRead, ctListWrite, ctListData, ctListItem

Example

See ctListNew

ctListDelete

Frees a tag created with ctListAdd. Your program is permitted to call ctListDelete() while a read or write is pending on another thread. The ctListWrite() and ctListRead() will return once the tag has been deleted.

Syntax

ctListDelete(*hTag*)

hTag

Type: HANDLE
Input/output: Input
Description: The handle to the tag, as returned from ctListAdd().

Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call GetLastError().

Related Functions

ctOpen, ctListNew, ctListFree, ctListAdd, ctListRead, ctListWrite, ctListData, ctListItem

Example

```
HANDLE         hList;
HANDLE         hTagOne;
HANDLE         hTagTwo;
hList     = ctListNew(hCTAPI, 0);
hTagOne    = ctListAdd(hList, "TagOne");
hTagTwo    = ctListAdd(hList, "TagTwo");
ctListRead(hList, NULL);        // read TagOne and TagTwo
ctListData(hList, hTagOne, sBufOne, sizeof(sBufOne), 0);
ctListData(hList, hTagTwo, sBufTwo, sizeof(sBufTwo) , 0);
ctListDelete(hTagOne);        // delete TagOne;
ctListRead(hList, NULL);        // read TagTwo only
ctListData(hList, hTagTwo, sBufTwo, sizeof(sBufTwo) , 0);
```

ctListEvent

Returns the elements in the list which have changed state since they were last read using the ctListRead() function. You need to have created the list with CT_LIST_EVENT mode in the ctListNew() function.

Syntax

ctListEvent(*hCTAPI, dwMode*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctListNew().

dwMode

Type: Dword
Input/output: Input
Description: The mode of the list event. You need to use the same mode for each call to ctListEvent() until NULL is returned before changing mode. The following modes are supported:

CT_LIST_EVENT_NEW - Gets notifications when tags are added to the list. When this mode is used, you will get an event message when new tags added to the list.

CT_LIST_EVENT_STATUS - Gets notifications for status changes. Tags will change status when the I/O Device goes offline. When this mode is used, you will get a notification when the tag goes into #COM and another one when it goes out of #COM. You can verify that the tag is in #COM when an error is returned from ctListData() for that tag.

Return Value

If the function succeeds, the return value specifies a handle to a tag which has changed state since the last time ctListRead was called. If the function does not succeed or there are no changes, the return value is NULL. To get extended error information, call GetLastError().

Related Functions

ctListAdd, ctListDelete, ctListRead, ctListWrite, ctListData, ctListItem

Example

```
HANDLE     hList; HANDLE     hTag[100];
hList = ctListNew(hCTAPI, CT_LIST_EVENT);
hTagArray[0] = ctListAdd(hList, "TagOne");
hTagArry[1] = ctListAdd(hList, "TagTwo");
and so on...
while (TRUE) {
        ctListRead(hList, NULL);
        hTag = ctListEvent(hList, 0);
        while (hTag != NULL) {
                // hTag has changed state, do whatever you need
                hTag = ctListEvent(hList, 0);
        }
}
```

ctListFree

Frees a list created with ctListNew. Every tag added to the list is freed, you do not have to call ctListDelete() for each tag. not call ctListFree() while a read operation is pending. Wait for the read to complete before freeing the list.

Syntax

ctListFree(*hList*)

hList

Type: HANDLE
Input/output: Input
Description: The handle to the list, as returned from ctListNew().

Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call GetLastError().

Related Functions

ctOpen, ctListNew, ctListAdd, ctListDelete, ctListRead, ctListWrite, ctListData ,

Example

See ctListNew

ctListRead

Reads the tags on the list. This function will read tags which are attached to the list. Once the data has been read from the I/O Devices, you may call ctListData()to get the values of the tags. If the read does not succeed, ctListData() will return an error for the tags that cannot be read.

While ctListRead() is pending you are allowed to add and delete tags from the list. If you delete a tag from the list while ctListRead() is pending, it may still be read one more time. The next time ctListRead() is called, the tag will not be read. If you add a tag to the list while ctListRead() is pending, the tag will not be read until the next time ctListRead() is called. You may call ctListData() for this tag as soon as you have added it. In this case ctListData() will not succeed, and GetLastError() will return GENERIC_INVALID_DATA.

You can only have 1 pending read command on each list. If you call ctListRead() again for the same list, the function will not succeed.

Before freeing the list, check that there are no reads still pending. wait for the any current ctListRead() to return and then delete the list.

Syntax

ctListRead(*hList, pctOverlapped*)

hList

Type: HANDLE
Input/output: Input
Description: The handle to the list, as returned from ctListNew().

pctOverlapped

Type: CTOVERLAPPED*
Input/output: Input
Description: CTOVERLAPPED structure. This structure is used to control the overlapped notification. Set to NULL if you want a synchronous function call.

Return Value

If the function succeeds, the return value is TRUE. If the function does not succeed, the return value is FALSE. To get extended error information, call GetLastError().

If an error occurred when reading any of the list data from the I/O Device, the return value will be FALSE and GetLastError() will return the associated CitectSCADA error code. As a list can contain tags from many data sources, some tags may be read correctly while other tags may not. If any tag read does not succeed, ctListRead() will return FALSE, however, the other tags will contain valid data. You can call ctListData() to retrieve the value of each tag and the individual error status for each tag on the list.

Related Functions

ctOpen, ctListNew, ctListFree, ctListAdd, ctListWrite, ctListData, ctListItem

Example

See ctListNew

To read the Paging Alarm property using ctListRead:

```
HANDLE     hList;
HANDLE     hAlarmOne;
HANDLE     hAlarmTwo;
hList = ctListNew(hCTAPI, 0);
hTagOne = ctListAdd(hList, "AlarmOne.Paging");
hTagTwo = ctListAdd(hList, "AlarmTwo.Paging");
while (you want the data) {
        ctListRead(hList, NULL);
        ctListData(hAlarmOne, sBufOne, sizeof(sBufOne), 0);
        ctListData(hAlarmTwo, sBufTwo, sizeof(sBufTwo) , 0);
}
```

```
ctListFree(hList);
```

ctOpen

Opens a connection to the CitectSCADA API. The CTAPI.DLL is initialized and a connection is made to CitectSCADA. If CitectSCADA is not running when this function is called, the function will exit and report an error. This function needs to be called before any other CTAPI function to initialize the connection to CitectSCADA.

If you use the CT_OPEN_RECONNECT mode, and the connection is lost, the CTAPI will attempt to reconnect to CitectSCADA. When the connection has been re-established, you can continue to use the CTAPI. However, while the connection is down, every function will return errors. If a connection cannot be created the first time ctOpen() is called, a valid handle is still returned; however GetLastError() will indicate an error.

If you do not use the CT_OPEN_RECONNECT mode, and the connection to CitectSCADA is lost, you need to free handles returned from the CTAPI and call ctClose() to free the connection. You need to then call ctOpen() to re-establish the connection and re-create any handles.

> **Note:** To use the CTAPI on a remote computer without installing CitectSCADA, you will need to copy the following files from the `[bin]` directory to your remote computer: CTAPI.DLL, CT_IPC.DLL, CTENG32.DLL, CTRES32.DLL, CTUTIL32.DLL, and CIDEBUGHELP.DLL.

If calling this function from a remote computer, a valid username and a non-blank password needs to be used.

Syntax

ctOpen(*sComputer, sUser, sPassword, nMode)*

sComputer

Type: LPCSTR
Input/output: Input
Description: The computer you want to communicate with via CTAPI. For a local connection, specify NULL as the computer name. The Windows Computer Name is the name as specified in the Identification tab, under the Network section of the Windows Control Panel.

sUser

Type: LPCSTR
Input/output: Input
Description: Your username as defined in the CitectSCADA project running on the computer you want to connect to. This argument is only necessary if you are calling this function from a remote computer. On a local computer, it is optional.

sPassword

Type: LPCSTR
Input/output: Input
Description: Your password as defined in the CitectSCADA project running on the computer you want to connect to. This argument is only necessary if you are calling this function from a remote computer. You need to use a non-blank password. On a local computer, it is optional.

nMode

Type: DWORD
Input/output: Input
Description: The mode of the Cicode call. Set this to 0 (zero). The following modes are supported:

CT_OPEN_RECONNECT - Reopen connection on error or communication interruption. If the connection to CitectSCADA is lost CTAPI will continue to retry to connect to CitectSCADA.

CT_OPEN_READ_ONLY - Open the CTAPI in read only mode. This allows read only access to data - you cannot write to any variable in CitectSCADA or call any Cicode function.

CT_OPEN_BATCH - Disables the display of message boxes when an error occurs.

Return Value

If the function succeeds, the return value specifies a handle. If the function does not succeed, the return value is NULL. Use GetLastError() to get extended error information.

Related Functions

ctCiCode, ctClose, ctEngToRaw, ctGetOverlappedResult, ctHasOverlappedIoCompleted, ctRawToEng, ctTagRead, ctTagWrite, ctTagWrite

Example

```
HANDLE     hCTAPI;
hCTAPI = ctOpen(NULL, NULL, NULL, 0);
if (hCTAPI == NULL) {
        dwStatus = GetLastError();    // get error
} else {
        ctTagWrite(hCTAPI, "SP123", "1.23");
        ctClose(hCTAPI);
```

```
}
// example of open for remote TCP/IP connection.
hCTAPI = ctOpen("203.19.130.2", "ENGINEER", "CITECT", 0);
```

ctOpenEx

Establishes the connection to the CtAPI server using the given client instance. Create the client instance prior to calling ctOpenEx, using the function ctClientCreate.

ctOpenEx provides exactly the same connection functionality as ctOpen, the only difference being that ctOpen also creates the CtAPI client instance. See ctOpen for details on the connection mechanism and the parameters involved.

Syntax

ctOpenEx(*sComputer, sUser, sPassword, nMode, hCTAPI*);

sComputer

Type: LPCSTR
Input/output: Input
Description: The computer you want to communicate with via CTAPI. For a local connection, specify NULL as the computer name. The Windows Computer Name is the name as specified in the Identification tab, under the Network section of the Windows Control Panel.

sUser

Type: LPCSTR
Input/output: Input
Description: Your username as defined in the CitectSCADA project running on the computer you want to connect to. This argument is only necessary if you are calling this function from a remote computer. On a local computer, it is optional.

sPassword

Type: LPCSTR
Input/output: Input
Description: Your password as defined in the CitectSCADA project running on the computer you want to connect to. This argument is only necessary if you are calling this function from a remote computer. You need to use a non-blank password. On a local computer, it is optional.

nMode

Type: Dword
Input/output: Input
Description:The mode of the Cicode call. Set this to 0 (zero).

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

Related Functions

ctClientCreate, ctOpen, ctClose, ctCloseEx, ctClientDestroy

Example

See ctClientCreate

ctRawToEng

Converts the raw I/O Device scale variable into Engineering scale. This is not necessary for the Tag functions as CitectSCADA will do the scaling. Scaling is not necessary for digitals, strings or if no scaling occurs between the values in the I/O Device and the Engineering values. You need to know the scaling for each variables as specified in the CitectSCADA Variable Tags table.

Syntax

ctRawToEng(*pResult, dValue, pScale, dwMode*)

pResult

Type: Double
Input/output: Output
Description: The resulting raw scaled variable.

dValue

Type: Double
Input/output: Input
Description: The engineering value to scale.

pScale

Type: CTSCALE*
Input/output: Input
Description: The scaling properties of the variable.

dwMode

Type: Dword
Input/output: Input
Description: The mode of the scaling. The following modes are supported:

CT_SCALE_RANGE_CHECK - Range check the result. If the variable is out of range then generate an error. The pResult still contains the raw scaled value.

CT_SCALE_CLAMP_LIMIT - Clamp limit to max or minimum scales. If the result is out of scale then set result to minimum or maximum scale (which ever is closest). No error is generated if the scale is clamped. Cannot be used with CT_SCALE_RANGE_CHECK or CT_SCALE_NOISE_FACTOR options.

CT_SCALE_NOISE_FACTOR - Allow noise factor for range check on limits. If the variable is our of range by less than 0.1 % then a range error is not generated.

Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

Related Functions

ctOpen, ctEngToRaw

Example

```
// SP123 is type INTEGER and has raw scale 0 to 32000 and Eng scale
0 to 100
HANDLE  hList         = ctListNew(s_hCTAPI, 0);
HANDLE  hTag          = ctListAddEx(hList, "SP123", TRUE, 500, -1);
CTSCALE Scale         = { 0.0, 32000.0, 0.0, 100.0};
CHAR    valueBuf[256] = {0};
double  dRawValue     = 0.0;
double  dSetPoint     = 0.0;
ctListRead(hList, NULL);
ctListData(hTag, valueBuf, sizeof(valueBuf), 0);
dRawValue = strtod(valueBuf, NULL);
ctEngToRaw(&dSetPoint, dRawValue, &Scale, CT_SCALE_RANGE_CHECK);
// dSetPoint now contains the Engineering scaled setpoint.
```

ctTagGetProperty

Gets the given property of the given tag.

Syntax

ctTagGetProperty(*hCTAPI, szTagName, szProperty, pData, dwBufferLength, dwType*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

szTagName

Type: LPCSTR
Input/output: Input
Description: The name of the tag. To specify cluster add "ClusterName." in front of the tag. For example Cluster1.Tag1 (note the period at the end of the cluster name).

szProperty

Type: LPCSTR
Input/output: Input
Description: The property to read. Property names are case sensitive. Supported properties are:

ArraySize: Array size of the associated tag. Returns 1 for non-array types.

DataBitWidth: Number of bits used to store the value.

Description: Tag description.

EngUnitsHigh: Maximum scaled value.

EngUnitsLow: Minimum scaled value.

Format: Format bit string. The format information is stored in the integer as follows:

- Bits 0-7 - format width
- Bits 8-15 - number of decimal places
- Bits 16 - zero-padded
- Bit 17- left-justified
- Bit 18 - display engineering units
- Bit 20 - exponential (scientific) notation

FormatDecPlaces: Number of decimal places for default format.

FormatWidth: Number of characters used in default format.

RangeHigh: Maximum unscaled value.

RangeLow: Minimum unscaled value.

Type: Type of tag as a number:

- 0 = Digital
- 1 = Byte
- 2 = Integer16
- 3 = UInteger16
- 4 = Long

- 5 = Real
- 6 = String
- 7 = ULong
- 8 = Undefined

Units: Engineering Units for example %, mm, Volts.

pData

Type: VOID*
Input/output: Output
Description: The output data buffer for the property value retrieved.

dwBufferLength

Type: DWORD
Input/output: Input
Description: The length of the output data buffer in bytes.

dwType

Type: DWORD
Input/output: Input
Description: The type of data to return.

Value	Meaning
DBTYPE_U11	UCHAR
DBTYPE_I1	1 byte INT
DBTYPE_I2	2 byte INT
DBTYPE_I4	4 byte INT
DBTYPE_R4	4 byte REAL
DBTYPE_R8	8 byte REAL
DBTYPE_BOOL	BOOLEAN
DBTYPE_BYTES	Byte Stream
DBTYPE_STR	NULL terminated STRING

Return Value

If the function succeeds, the return value is non-zero. If the function does not succeed, the return value is zero. To get extended error information, call GetLastError().

ctTagRead

Reads the value, quality and timestamp, not only a value. The data will be returned in string format and scaled using the CitectSCADA scales.

The function will request the given tag from the CitectSCADA I/O Server. If the tag is in the I/O Servers device cache the data will be returned from the cache. If the tag is not in the device cache then the tag will be read from the I/O Device. The time taken to complete this function will be dependent on the performance of the I/O Device. The calling thread is blocked until the read is completed.

Syntax

ctTagRead(*hCTAPI, sTag, sValue, dwLength*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

sTag

Type: LPCSTR
Input/output: Input
Description: The tag name or tag name and element name, separated by a dot. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference. You may use the array syntax [] to select an element of an array.

sValue

Type: LPCSTR
Input/output: Output
Description: The buffer to store the read data. The data is returned in string format.

dwLength

Type: Dword
Input/output: Input
Description: The length of the read buffer. If the data is bigger than the dwLength, the function will not succeed.

Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

Related Functions

ctOpen, ctTagWrite, ctTagWriteEx

Example

```
HANDLE   hCTAPI = ctOpen(NULL, NULL, NULL, 0);

char   sProcessValue[20];
char   sProcessValueField[20];
char   sProcessValueControlMode[20];
char   sProcessValueStatus[20];
ctTagRead(hCTAPI,"PV123", sProcessValue, sizeof(sProcessValue));
ctTagRead(hCTAPI,"PV123.Field", sProcessValueField, sizeof(sProcessValueField));
ctTagRead(hCTAPI,"PV123.Field.V", sProcessValueField, sizeof(sProcessValueField));
ctTagRead(hCTAPI,"PV123.ControlMode",sProcessValueControlMode, size-
of(sProcessValueControlMode));
ctTagRead(hCTAPI, "PV123.Status", sProcessValueStatus, sizeof(sProcessValueStatus));
```

ctTagWrite

Writes to the given CitectSCADA I/O Device variable tag. The value, quality and timestamp, not only a value, is converted into the correct data type, then scaled and then written to the tag. If writing to an array element only a single element of the array is written to. This function will generate a write request to the I/O Server. The time taken to complete this function will be dependent on the performance of the I/O Device. The calling thread is blocked until the write is completed. Writing operation will succeed only for those tag elements which have read/write access.

Syntax

ctTagWrite(*hCTAPI, sTag, sValue*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

sTag

Type: LPCSTR
Input/output: Input
Description: The tag name or tag name and element name, separated by a dot. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference. You may use the array syntax [] to select an element of an array.

sValue

Type: LPCSTR
Input/output: Input
Description: The value to write to the tag as a string.

Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

Related Functions

ctOpen, ctTagWrite, ctTagRead

Example

```
HANDLE   hCTAPI = ctOpen(NULL, NULL, NULL, 0);

ctTagWrite (hCTAPI,"PV123", "123.12");
ctTagWrite (hCTAPI,"PV123.Field", "123.12");
ctTagWrite (hCTAPI,"PV123.Field.V", "123.12");
ctTagWrite (hCTAPI,"PV123.ControlMode", "1");
ctTagWrite (hCTAPI,"PV123.Status", "0");
```

ctTagWriteEx

Performs the same as ctTagWrite, but with an additional new argument. Writes to the given CitectSCADA I/O Device variable tag. The value, quality and timestamp, not only a value, is converted into the correct data type, then scaled and then written to the tag. If writing to an array element only a single element of the array is written to. This function will generate a write request to the I/O Server. The time taken to complete this function will be dependent on the performance of the I/O Device.

If the value of pctOverlapped is NULL, the function behaves the same as ctTagWrite, and the calling thread is blocked until the write is completed. If the value of pctOverlapped is not NULL, the write is completed asynchronously and the calling thread is not blocked.

Syntax

ctTagWriteEx(*hCTAPI, sTag, sValue, pctOverlapped*)

hCTAPI

Type: Handle
Input/output: Input
Description: The handle to the CTAPI as returned from ctOpen().

sTag

Type: LPCSTR
Input/output: Input
Description: The tag name or tag name and element name, separated by a dot to write to. If the element name is not specified, it will be resolved at runtime as for an unqualified tag reference. You may use the array syntax [] to select an element of an array.

sValue

Type: LPSTR
Input/output: Input
Description: The value to write to the tag as a string.

pctOverlapped

Type: CTOVERLAPPED*
Input/output: Input
Description: Passes in an overlapped structure so ctTagWriteEx can complete asynchronously. If the pctOverlapped structure is NULL, the function will block, completing synchronously.

Return Value

TRUE if successful, otherwise FALSE. Use GetLastError() to get extended error information.

Related Functions

ctOpen, ctTagRead

AlmQuery

Provides an interface into the alarm summary archive from external applications, replacing the old CtAPIAlarm query. AlmQuery performs significantly better than CtAPIAlarm.

AlmQuery is performed through the same mechanism as CtAPIAlarm. To establish the query and return the first record, you call ctFindFirst. Then, to browse the remaining records, you call ctFindNext. To access the data of the current record, ctGetProperty is called for each field of the record.

ctFindFirst is called with the following parameters:

- *hCtapi*: Handle to a valid CtAPI client instance.
- *szTableName*: Command string for the almquery, see below.
- *szFilter*: Not used for Almquery. Just pass in NULL.
- *hObject*: Handle to the first record retrieved for the query.
- *dwFlags*: Not used for Almquery. Just pass in 0.

The *szTableName* is the command string for the query and contains the parameters for the query.

Syntax

`**ALMQUERY**,*Database*,*TagName*,*Starttime*,*StarttimeMs*,*Endtime*,*EndtimeMs*,*Period*'

> **Note:** Arguments need to be comma-separated. Spaces between arguments are supported but not necessary. We recommend no spaces between arguments as they require more processing and take up more space in the query string.

Database:

The Alarm database that the alarm is in (alarm type). The following databases are supported: DigAlm (Digital), AnaAlm (Analog), AdvAlm (Advanced), HResAlm (Time Stamped), ArgDigAlm (Multi-Digital), ArgAnaAlm (Argyle Analog), TsDigAlm (Time Stamped Digital), TsAnaAlm (Timestamped Analog).

TagName:

The Alarm tag as a string. This query only supports the retrieval of alarm data for one alarm at a time. Although it is supported by CitectSCADA, do not declare two different alarms with the same tag and of the same type. You will not be able to retrieve the alarm data for both as this query expects the combination of alarm type (database) and tag to be unique.

Starttime:

The start time of the alarm query in seconds since 1970 as an integer in UTC time.

StarttimeMs:

The millisecond portion of the start time as an integer. It is expected to be a number between 0 and 999.

Endtime:

The end time of the alarm query in seconds since 1970 as an integer in UTC time.

EndtimeMs:

Millisecond portion of the end time as an integer. It is expected to be a number between 0 and 999.

Period:

Time period in seconds between the samples returned as a floating point value. The only decimal separator supported is the `.'.

Return Value

The maximum number of samples returned is the time range divided by the period, plus 3 (one for the sample exactly on the end time, and two for the previous and next samples).

Note: Divide the period evenly into the time range, otherwise one extra sample may be returned.

The AlmQuery does not return interpolated samples in periods where there were no alarm samples. However, to stay within the allowable number of samples, the raw alarm samples will be compressed when more than one sample occurs in one period.

When this compression occurs, the returned sample is flagged as a multiple sample. The timestamp is then an average of the samples within the period. The value and comment returned reflects that of the last sample in the period.

The following properties are returned for each data record of the query.

- *DateTime*: The time of the alarm sample in seconds since 1970 as an integer. This Time is in UTC (Universal Time Coordinates).
- *MSeconds*: The millisecond component of the time of the trend sample as an integer. This value is in between 0 and 999.
- *Comment*: The comment associated with the alarm sample as a string.
- *Value*: The alarm value of the sample as an unsigned integer. See below for a detailed description of the alarm value. The alarm value contains information describing the state of the alarm at the time of the sample:

  bGood (Bit 0)- Future use only, intended to show when the quality of the alarm data goes bad. At the moment every sample has this bit set to 1 to say the sample is good.

  bDisabled (Bit 1)- 1 if the alarm is disabled at the sample's time, 0 otherwise.

  bMultiple (Bit 2)- 1 if the alarm sample is based on multiple raw samples, 0 if it is based on only 1 raw sample.

  bOn (Bit 3)- 1 if the alarm is on at the sample's time, 0 otherwise.

  bAck (Bit 4)- 1 if the alarm is acknowledged at the sample's time, 0 otherwise.

  state (Bits 5 - 7)- Contains the state information of the alarm at the sample's time.

  The alarm state represents the different states of the different alarm types. The state only contains relevant information if the alarm is on.

  For analog, Argyle analog, and time-stamped analog alarms the state can be as follows:

  - **Expired (0)**- The alarm state information has expired. We no longer know what state the alarm was, we just know the alarm was on at this time. This occurs if you set the Citect.ini parameter [Alarm]SumStateFix = 0.
  - **Deviation High (1)**- The alarm has deviated above the Setpoint by more than the specified threshold.

- **Deviation Low (2)**- The alarm has deviated below the Setpoint by more than the specified threshold.
- **Rate of Change (3)**- The alarm has changed at a faster rate than expected.
- **Low (4)**- The alarm has entered the low alarm range of values.
- **High (5)**- The alarm has entered the high alarm range of values.
- **Low Low (6)**- The alarm has entered the low low alarm range of values.
- **High High (7)**- The alarm has entered the high high alarm range of values.

For Multi-Digital Alarms the state can be as follows:

- **000 (0)**- Digital tags for the alarm are off.
- **00A (1)**- Tag A is on, B and C are off.
- **0B0 (2)**- Tag B is on, A and C are off.
- **0BA (3)**- Tags B and A are on, C is off.
- **C00 (4)**- Tag C is on, B and C are off.
- **C0A (5)**- Tag C and A are on, B is off.
- **CB0 (6)**- Tag C and B are on, A is off.
- **CBA (7)**-Digital tags for the alarm are on.

For the rest of the alarm types ignore the state information.

TrnQuery

Provides a powerful interface into the trend achive from external applications, replacing the old CtAPITrend query. TrnQuery performs significantly better than CtAPITrend.

TrnQuery is performed through the same mechanism as CtAPITrend. To establish the query and return the first record, you call ctFindFirst. Then, to browse the remaining records, you call ctFindNext. To access the data of the current record, ctGetProperty is called for each field of the record.

ctFindFirst is called with the following parameters:

- *hCtapi*: handle to a valid Ctapi client instance.
- *szTableName*: command string for the Trnquery, see below.
- *szFilter*: Not used for Trnquery. Just pass in NULL.
- *hObject*: handle to the first record retrieved for the query.
- *dwFlags*: Not used for Trnquery. Just pass in 0.

The *szTableName* is the command string for the query. It contains the parameters for the query.

Syntax

TRNQUERY,*Endtime*,*EndtimeMs*,*Period*,*NumSamples*,*Tagname*,*Displaymode*,*Datamode*,*Instant-Trend*,*SamplePeriod*'

> **Note:** Arguments needs to be comma-separated. Spaces between arguments are supported but not necessary. We recommend no spaces between arguments as they require more processing and take up more space in the query string.

Endtime:

End time of the trend query in seconds since 1970 as an integer. This time is expected to be a UTC time (Universal Time Coordinates).

EndtimeMs:

Millisecond portion of the end time as an integer, expected to be a number between 0 and 999.

Period:

Time period in seconds between the samples returned as a floating point value. The only decimal separator supported is the `.'.

NumSamples:

Number of samples requested as an integer. The start time of the request is calculated by multiplying the Period by NumSamples - 1, then subtracting this from the EndTime.

The actual maximum amount of samples returned is actually NumSamples + 2. This is because we return the previous and next samples before and after the requested range. This is useful as it tells you where the next data is before and after where you requested it.

TagName:

The name of the trend tag as a string. This query only supports the retrieval of trend data for one trend at a time.

DisplayMode:

Specifies the different options for formatting and calculating the samples of the query as an unsigned integer. See Display Mode for information.

DataMode:

Mode of this request as an integer. 1 if you want the timestamps to be returned with their full precision and accuracy. Mode 1 does not interpolate samples where there were no values. 0 if you want the timestamps to be calculated, one per period. Mode 0 does interpolate samples, where there was no values.

InstantTrend:

An integer specifying whether the query is for an instant trend. 1 if for an instant trend. 0 if not.

SamplePeriod:

An integer specifying the requested sample period in milliseconds for the instant trend's tag value.

Return Value

See Returned Data for return values.

Display Mode

The data returned can vary drastically depending on the display mode of the TrnQuery. The display mode is split into the following mutually exclusive options:

Ordering Trend sample options

- 0 - Order returned samples from oldest to newest
- 1 - Order returned samples from newest to oldest. This mode is not supported when the Raw data option has been specified.

Condense method options

- 0 - Set the condense method to use the mean of the samples.
- 4 - Set the condense method to use the minimum of the samples.
- 8 - Set the condense method to use the maximum of the samples.
- 12 - Set the condense method to use the newest of the samples.

Stretch method options

- 0 - Set the stretch method to step.
- 128 - Set the stretch method to use a ratio.
- 256 - Set the stretch method to use raw samples (no interpolation).

Gap Fill Constant option

- n - the number of missed samples that the user wants to gap fill) x 4096.

Last valid value option

- 0 - If we are leaving the value given with a bad quality sample as 0.
- 2097152 - If we are to set the value of a bad quality sample to the last valid value (zero if there is no last valid value).

Raw data option

- 0 - If we are not returning raw data, that is we are using the condense and stretch modes to compress and interpolate the data.
- 4194304 - If we are to return totally raw data, that is no compression or interpolation. This mode is only supported if we have specified the DataMode of the query = 1.

When using this mode, more samples than the maximum specified above will be returned if there are more raw samples than the maximum in the time range.

Returned Data

The following properties are returned for each data record of the query.

- *DateTime*: Time of the trend sample in seconds since 1970 as an integer in UTC (Universal Time Coordinates).
- *MSeconds*: Millisecond component of the time of the trend sample as an integer. This value is inbetween 0 and 999.
- *Value*: Trend value of the sample as a double.
- *Quality*: The quality information associated with the trend sample as an unsigned integer. The Quality property contains different information in different bits of the unsigned integer as follows:
  Value Type (Bits 0 - 3)
  - ValueType_None (0): There is no value in the given sample. Ignore the sample value, time and quality.
  - ValueType_Interpolated (1): The value has been interpolated from data around it.
  - ValueType_SingleRaw (2): The value is based on one raw sample.
  - ValueType_MultipleRaw (3): The value has been calculated from multiple raw samples.

  Value Quality (Bits 4 - 7)
  - ValueQuality_Bad (0): Ignore the value of the sample as there was no raw data to base it on.
  - ValueQuality_Good (1): The value of the sample is valid, and is based on some raw data.

  Last Value Quality (Bits 8 - 11)
  - LastValueQuality_Bad (0: The value of the sample should be ignored as there was no raw data to base it on.
  - LastValueQuality_Good (1): The value quality of the last raw sample in the period was good.
  - LastValueQuality_NotAvailable (2): The value quality of the last raw sample in the period was Not Available.
  - LastValueQuality_Gated (3): The value quality of the last raw sample in the period was Gated.

  Partial Flag (Bit 12)
  When the Partial Flag is set to 1 it indicates that the sample may change the next time it is read. This occurs when you get samples right at the current time, and a sample returned is not necessarily complete because more samples may be acquired in this period.

CtAPIAlarm

Provides an interface into the alarm summary archive from external applications. For performance improvements, use the AlmQuery function instead.

To establish the query and return the first record, you call ctFindFirst. Then, to browse the remaining records, you call ctFindNext. To access the data of the current record, ctGetProperty is called for each field of the record.

ctFindFirst is called with the following parameters:

- *hCtapi*: Handle to a valid CtAPI client instance.
- *szTableName*: Command string for the almquery, see below.
- *szFilter*: Not used for Almquery. Just pass in NULL.
- *hObject*: Handle to the first record retrieved for the query.
- *dwFlags*: Not used for Almquery. Just pass in 0.

The *szTableName* is the command string for the query and contains the parameters for the query.

Syntax

CTAPIAlarm(*Category,Type,Area*)

> **Note:**Arguments needs to be comma-separated. Spaces between arguments are supported but not necessary. We recommend no spaces between arguments as they require more processing and take up more space in the query string.

Category:

The alarm category or group number to match. Set Category to 0 (zero) to match every alarm categorie.

Type:

The type of alarms to find:

Non-hardware alarms

- 1. Unacknowledged alarms, ON and OFF.
- 2. Acknowledged ON alarms.
- 3. Disabled alarms.
- 4. Configured alarms, i.e. Types 0 to 3, plus acknowledged OFF alarms.

If you do not specify a Type, the default is 0.

Area:

The area in which to search for alarms. If you do not specify an area, or if you set Area to -1, only the current area will be searched.

To simplify the passing of this argument, you could first pass the CTAPIAlarm() function as a string, then use the string as the *szTableName* argument (without quotation marks).

CtAPITrend

Provides an interface into the trend archive from external applications, replacing the old CtAPITrend query. For performance improvements, use the TrnQuery function instead.

To establish the query and return the first record, you call ctFindFirst. Then, to browse the remaining records, you call ctFindNext. To access the data of the current record, ctGetProperty is called for each field of the record.

ctFindFirst is called with the following parameters:

- *hCtapi*: handle to a valid Ctapi client instance.
- *szTableName*: command string for the Trnquery, see below.
- *szFilter*: Not used for Trnquery. Just pass in NULL.
- *hObject*: handle to the first record retrieved for the query.
- *dwFlags*: Not used for Trnquery. Just pass in 0.

The *szTableName* is the command string for the query. It contains the parameters for the query.

Syntax

CTAPITrend(*sTime,sDate,Period,Length,Mode,Tag*)

> **Note:** Arguments needs to be comma-separated. Spaces between arguments are supported but not necessary. We recommend no spaces between arguments as they require more processing and take up more space in the query string.

sTime:

The starting time for the trend. Set the time to an empty string to search the latest trend samples.

sDate:

The date of the trend.

Period:

The period (in seconds) that you want to search (this period can differ from the actual trend period).

The Period argument used in the CTAPITrend() function needs to be 0 (zero) when this function is used as an argument to ctFindFirst() for an EVENT trend query.

Length:

The length of the data table, i.e. the number of rows of samples to be searched.

Mode:

The format mode to be used:

Periodic trends

- 1 - Search the Date and Time, followed by the tags.
- 2 - Search the Time only, followed by the tags.
- 3 - Ignore any invalid or gated values. (This mode is only supported for periodic trends.)

Event trends

- 1 - Search the Time, Date, and Event Number, followed by the tags.
- 2 - Search the Time and Event Number, followed by the tags.

Tag:

The trend tag name for the data to be searched.

To simplify the passing of this argument, you could first pass the CTAPITrend() function as a string, then use the string as the *szTableName* argument (without quotation marks).

Chapter: 5 CSV_Include Reference

This section provides information on:

CSV_Include Cicode functions

CSV_Include Parameters

There are a number of Citect.ini files that can be used specifically for the CSV_Include project. For information on these parameters refer to the Parameters topic of the Citect-SCADA on line help in the section "CSV_Include Parameters".

CSV_Include Functions

The table below contains the CSV_Include categories of functions:

Function Category	Functions
CSV_Include Alarms	CSV_Alarms_Ack CSV_Alarms_AckHardware CSV_Alarms_AckPage CSV_Alarms_AckRec CSV_Alarms_AdvFilter CSV_Alarms_AdvFilterConfig CSV_Alarms_AdvFilterQuery CSV_Alarms_AdvFilterSetDateTime CSV_Alarms_CheckSound CSV_Alarms_ClearGroupFilter CSV_Alarms_Disable CSV_Alarms_DisableRec CSV_Alarms_DspGroupFilter CSV_Alarms_DspGroupList CSV_Alarms_DspInfo CSV_Alarms_DspInfoRec CSV_Alarms_DspLast CSV_Alarms_Enable CSV_Alarms_EnableRec CSV_Alarms_GetAckPrivilege() CSV_Alarms_GetDisablePrivilege() CSV_Alarms_GetGroupFilter CSV_Alarms_GetGroupFilterID CSV_Alarms_GetUniqueGroupName CSV_Alarms_GroupAdd CSV_Alarms_GroupConfig() CSV_Alarms_GroupRemove

Function Category	Functions
	CSV_Alarms_GroupEdit CSV_Alarms_GroupFilter CSV_Alarms_GroupSelect CSV_Alarms_GroupsInit() CSV_Alarms_Help CSV_Alarms_HelpRec CSV_Alarms_ListHeading CSV_Alarms_ListHeadingFont() CSV_Alarms_PopupMenu CSV_Alarms_Sound() CSV_Alarms_SoundActive() CSV_Alarms_Silence()
CSV_Include Database	CSV_DB_BOF CSV_DB_Close CSV_DB_EOF() CSV_DB_Execute CSV_DB_GetExecuteError CSV_DB_GetFieldCount CSV_DB_GetFieldIndex CSV_DB_GetFieldName CSV_DB_GetFieldText CSV_DB_GetRowCount CSV_DB_GetRowCurrent CSV_DB_GetRowFieldText CSV_DB_MoveFirst CSV_DB_MoveLast CSV_DB_MoveNext CSV_DB_MoveOffset CSV_DB_MovePrev CSV_DB_StandbyConnectionActive CSV_DB_StrToSQL
CSV_Include Display	CSV_Display_Display_Logo CSV_Display_Display_ServicePack() CSV_Display_Title() CSV_Display_Version()
CSV_Include File	CSV_File_Display CSV_File_Print CSV_File_Save
CSV_Include Form	CSV_Form_Centre CSV_Form_Login() CSV_Form_NumPad CSV_Form_Position CSV_Form_Shutdown() CSV_Form_UserCreate() CSV_Form_UserEdit() CSV_Form_UserPassword()
CSV_Include ListBox	CSV_ListBox_AddItem CSV_ListBox_Clear CSV_ListBox_Create() CSV_ListBox_Destroy CSV_ListBox_GetCategory CSV_ListBox_GetItem CSV_ListBox_GetItemID CSV_ListBox_GetSelectedItem

Function Category	Functions
	CSV_ListBox_GetSelectedItemCategory CSV_ListBox_GetSelectedItemID CSV_ListBox_GetTagComment CSV_ListBox_GetTagDescFromTag CSV_ListBox_GetTagName CSV_ListBox_GetTrendDescFromTag() CSV_ListBox_Hide CSV_ListBox_RemoveItem CSV_ListBox_SelectCategories CSV_ListBox_SelectTags() CSV_ListBox_SelectTrends() CSV_ListBox_SetText CSV_ListBox_Show CSV_ListBox_TagFormat CSV_ListBox_Visible
CSV_Include Math	CSV_Math_RoundDown CSV_Math_Truncate
CSV_Include MenuConfig	CSV_MenuConfig_Close() CSV_MenuConfig_Display() CSV_MenuConfig_LoadDflt() CSV_MenuConfig_UserPages()
CSV_Include MessageBox	CSV_MessageBox
CSV_Include Misc	CSV_Misc_CheckNumPadValue CSV_Misc_IntRange CSV_Misc_MouseOver
CSV_Include MultiMonitors	CSV_MM_BackEmpty() CSV_MM_ConfigInit() CSV_MM_FwdEmpty() CSV_MM_GetMonitor() CSV_MM_GetMonitors() CSV_MM_GetMouseX CSV_MM_GetMouseY CSV_MM_GetOffset CSV_MM_GetScreenWidth() CSV_MM_ListLastPages CSV_MM_MonitorFromPoint CSV_MM_MonitorFromWindow CSV_MM_MonitorGoto CSV_MM_NextEmpty() CSV_MM_PageDisplay CSV_MM_PageLast CSV_MM_PageNext() CSV_MM_PagePrev() CSV_MM_PagesInit() CSV_MM_PreviousEmpty() CSV_MM_StoreLastPage CSV_MM_WinDrag() CSV_MM_WinDragEnd() CSV_MM_WinFree() CSV_MM_WinNewAt CSV_MM_WinPopup CSV_MM_WinTitle

Function Category	Functions
CSV_Include Navigation	CSV_Nav_Alarms() CSV_Nav_AlarmsDisabled() CSV_Nav_AlarmsHardware() CSV_Nav_AlarmsSummary() CSV_Nav_CloseWindow() CSV_Nav_DisableMenuItem CSV_Nav_DisplayMenuBar CSV_Nav_DisplayPopupMenu CSV_Nav_File CSV_Nav_GetEngToolsPrivilege() CSV_Nav_Home() CSV_Nav_Login() CSV_Nav_LoginMenu() CSV_Nav_MenuBar_MenuClick CSV_Nav_Network() CSV_Nav_NetworkBtnEnabled() CSV_Nav_PageExists CSV_Nav_PagePrint() CSV_Nav_Parent() CSV_Nav_ParentBtnEnabled() CSV_Nav_Report() CSV_Nav_ReportBtnEnabled() CSV_Nav_ReportMenu CSV_Nav_Tools() CSV_Nav_ToolsBtnEnabled() CSV_Nav_ToolsMenu() CSV_Nav_Trend() CSV_Nav_TrendBtnEnabled() CSV_Nav_TrendMenu() CSV_Nav_TrendX() CSV_Nav_TickMenuItem
CSV_Include Security	CSV_Sec_ShowLoginMenu
CSV_Include Strings	CSV_String_GetField CSV_String_GetLines CSV_String_Replace
CSV_Include Tags	CSV_Tag_Debug
CSV_Include Tag	CSV_Trend_AutoScale CSV_Trend_DspGroup CSV_Trend_DspGroupList CSV_Trend_DspPopupMenu CSV_Trend_DspScaleRange CSV_Trend_DspTrendText CSV_Trend_GetCursorPos CSV_Trend_GetCursorTypeStr CSV_Trend_GetCursorValueStr CSV_Trend_GetDate CSV_Trend_GetMode CSV_Trend_GetPen CSV_Trend_GetPenFocus CSV_Trend_GetSettings CSV_Trend_GetSettings CSV_Trend_GetSpan

Function Category	Functions
	CSV_Trend_GetTime CSV_Trend_GroupConfig() CSV_Trend_Page CSV_Trend_Popup CSV_Trend_ScaleDigital CSV_Trend_SelectGroup CSV_Trend_SelectPen CSV_Trend_SetCursor CSV_Trend_SetDate CSV_Trend_SetDateTime CSV_Trend_SetPens CSV_Trend_SetRange CSV_Trend_SetScale CSV_Trend_SetSpan CSV_Trend_SetTime CSV_Trend_SetTimebase CSV_Trend_UpdatePens CSV_Trend_Win
CSV_Include TrendX	CSV_TrendX_AddVariable CSV_TrendX_AgeTrends() CSV_TrendX_ClearTrend CSV_TrendX_Close CSV_TrendX_DeletePen() CSV_TrendX_Display() CSV_TrendX_DspPopupMenu CSV_TrendX_GenericToTag CSV_TrendX_GenericToTagStr CSV_TrendX_GetComment CSV_TrendX_GetCursor CSV_TrendX_GetDuration() CSV_TrendX_GetSamplePeriod CSV_TrendX_GetScale CSV_TrendX_GetTrendName CSV_TrendX_GetTrigger CSV_TrendX_GetVal CSV_TrendX_InitClient() CSV_TrendX_InitSrvr() CSV_TrendX_MapTrendTags() CSV_TrendX_RefreshTrendPage CSV_TrendX_SetDuration CSV_TrendX_SetDuration CSV_TrendX_SetPen() CSV_TrendX_SetSamplePeriod CSV_TrendX_SetScale CSV_TrendX_TagSelect CSV_TrendX_TagSelectFrmCursor() CSV_TrendX_TagToGeneric CSV_TrendX_TrendTimeout
CSV_Include WinUtilities	CSV_WinUtl_DestroyCursor() CSV_WinUtl_GetColourRes() CSV_WinUtl_GetCpuUsage CSV_WinUtl_GetSystemDir() CSV_WinUtl_GetTotalCpuUsage() CSV_WinUtl_GetWindowsDir() CSV_WinUtl_GetWinMode()

Function Category	Functions
	CSV_WinUtl_LoadCursor CSV_WinUtl_LockWindowUpdate CSV_WinUtl_NormalCursor CSV_WinUtl_ShellExec CSV_WinUtl_UpdateTotalCpuUsage() CSV_WinUtl_WaitCursor

CSV_Alarms_Ack

Acknowledges an alarm at a specified animation point in an alarm list.

Syntax

CSV_Alarms_Ack(*iAN*)

iAN:

Animation point number of alarm to acknowledge.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_AckHardware

Acknowledges a hardware alarm at a specified animation point in an alarm list.

Syntax

CSV_Alarms_AckHardware(*iAN*)

iAN:

Animation point number of alarm to acknowledge.

> **Note:** Hardware alarms are not stored in the same way as standard alarms. Therefore, AlarmGetDsp() does not return any information for a hardware alarm. Thus, CSV_Alarms_Ack will not function correctly for hardware alarms.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_AckPage

Acknowledges a page of alarms, starting at a specified animation point. Silences the alarm sound.

Syntax

CSV_Alarms_AckPage(*iAN*)

iAN:

Starting animation point number of page of alarms to acknowledge.

CSV_Alarms_AckRec

Acknowledges an alarm by record number, and silences the alarm sound.

Syntax

CSV_Alarms_AckRec(*iRecNo*)

iRecNo:

Record number of alarm to acknowledge.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_AdvFilter

Applies an advanced filter to the alarm list displayed at a specified AN. The advanced filter allows alarms to be filtered based on Date, Time, Tag, Name, Description, Area, Category, Priority, State and Type (or any combination of these).

Syntax

CSV_Alarms_AdvFilter(*iAN*,*iAlarmType*,*iMonitor*)

iAN:

Animation point where the alarm list is displayed.

iAlarmType:

Type of alarm list associated with filter:

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iMonitor:

The number of the monitor to associate the filter with (each monitor can display and store a different filter).

Return Value

0

CSV_Alarms_AdvFilterConfig

Displays a popup window allowing the user to configure advance alarm filtering.

Syntax

CSV_Alarms_AdvFilterConfig(*iAlarmType,iMonitor*)

iAlarmType:

Type of alarm list associated with filter:

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iMonitor:

The number of the monitor to associate the filter with (each monitor can display and store a different filter).

Return Value

0 if Advanced filter applied, otherwise -1.

CSV_Alarms_AdvFilterQuery

Called for each alarm to determine which alarm is displayed when a user defined advanced filter has been applied.

Syntax

CSV_Alarms_AdvFilterQuery(*iRecNo,nVer,sFromDate,sFromTime,sToDate,sToTime,sTag,sName,sArea,sCategory,sPriority,sState,sType*)

iRecNo:

Record number of the alarm.

nVer:

Version (not used).

sFromDate:

Alarms prior to this date won't be displayed ("" sets FromDate to earliest possible).

sFromTime:

Alarms prior to this time won't be displayed ("" sets FromTime to 12:00 Midnight).

sToDate:

Alarms subsequent to this date won't be displayed ("" sets ToDate to current date).

sToTime:

Alarms subsequent to this time won't be displayed ("" sets ToDate to current time).

sTag:

Alarm Tag needs to be 'Like' sTag i.e. = *sTag*.

sName:

Alarm Name needs to be 'Like' sName i.e. = *sName*.

sArea:

Area of alarm (or group of areas).

sCategory:

Alarm category (or group of categories).

sPriority:

Alarm priority (or group of priorities).

sState:

Alarm state.

sType):

Alarm type.

> **Note:** Setting any filter argument to "" will result in that filter criteria being ignored.

Return Value

1 if alarm is to be displayed (i.e., matches criteria), otherwise 0.

CSV_Alarms_AdvFilterSetDateTime

Writes the date and time entered via a keypad form to specified Text boxes. (Used in the Advanced Alarm Filter form).

Syntax

CSV_Alarms_AdvFilterSetDateTime(*iDateAN, iTime*)

iDateAN:

AN number of Date Text Box.

iTime:

AN number of Time Text Box.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_CheckSound

Checks alarm summary records between a specified index and the current index until an unacknowledged alarm is found (for given area/s) with a priority higher than a specified priority.

> **Note:** Only call this function on an Alarm Server.

Syntax

CSV_Alarms_CheckSound(*iAlarmIndexPrevious,iPriorityPrevious,sArea*)

iAlarmIndexPrevious:

Index in alarm summary to begin checking from (i.e., the index of the last alarm checked).

iPriorityPrevious:

Priority to compare with.

sArea:

Current logged in areas (i.e., only check alarms within these areas).

Return Value

This function returns a string containing three values separated by a single space:

- **Alarm Priority**: Priority of higher priority alarm if one is found, otherwise iPriorityPrevious as originally passed to function.
- **Alarm Index**: Index of most recent alarm checked.

- **Alarm Acknowledged**: 1 if an alarm has been acknowledged and no further alarms have since been triggered, otherwise 0.

CSV_Alarms_ClearGroupFilter

Clears the filter applied to the specified alarm list.

Syntax

CSV_Alarms_ClearGroupFilter(*iAN*,*iAlarmType*, *iMonitor*)

iAN:

Animation point number of start of alarm list.

iAlarmType:

Type of alarm list associated with filter:

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iMonitor:

Number of monitor displaying alarm list (-1 = active monitor).

CSV_Alarms_Disable

Disables an alarm at a specified animation point in an alarm list.

Syntax

CSV_Alarms_Disable(*iAN*)

iAN:

Animation point number of alarm to disable.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_DisableRec

Disables an alarm by record number.

Syntax

CSV_Alarms_DisableRec(*iRecNo*)

iRecNo:

Record number of alarm to disable.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_DspGroupFilter

Displays the Alarm Group listbox, and stores the selected filter for the specified alarm page and the specified monitor.

Syntax

CSV_Alarms_DspGroupFilter(*iAlarmType,iMonitor*)

iAlarmType:

Type of alarm list associated with filter:

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iMonitor:

Number of monitor displaying alarm list (-1 = active monitor).

Return Value

Name of Alarm Group selected, or "" if selection canceled.

CSV_Alarms_DspGroupList

Displays the Alarm Group listbox.

Syntax

CSV_Alarms_DspGroupList

sSelectedGroup:

Name of group to preselect in the list.

sAreas:

Areas to enable in the list; i.e., only alarm groups belonging to these areas are displayed.

Return Value

Alarm group (description) selected from the list, or "" if cancel is pressed.

CSV_Alarms_DspInfo

Displays information popup for alarm at specified animation point in alarm list.

Syntax

CSV_Alarms_DspInfo(*iAN*)

iAN:

Animation point number of alarm to display information for.

CSV_Alarms_DspInfoRec

Displays information popup for alarm at the specified record number.

Syntax

CSV_Alarms_DspInfoRec(*iRecNo*)

iRecNo:

Record number of alarm to display information for.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_DspLast

Displays specified number of most recent alarms, starting at a specified animation point.

Syntax

CSV_Alarms_DspLast(*iAN*,*iAlarmCount*,*iType*)

iAN:

Animation point number of start of alarm list.

iAlarmCount:

Number of alarms to display.

iType:

The type of alarms to display.

Non-hardware alarms

- -1 = Alarms specified by [Alarm]LastAlarmType parameter
- 0 = Active alarms, i.e. Types 1 and 2.
- 1 = Unacknowledged alarms, ON and OFF.
- 2 = Acknowledged ON alarms.
- 3 = Disabled alarms.
- 4 = Configured (non-hardware) alarms, i.e. Types 0 to 3, plus acknowledged OFF alarms.

Hardware alarms

- 5 = Active alarms; i.e., types 6 and 7.
- 6 = Unacknowledged alarms, ON and OFF.
- 7 = Acknowledged ON alarms.
- 8 = Disabled alarms.
- 9 = Configured alarms; i.e., types 5 to 8.

Alarm Summary

- 10 = Summary alarms.

Alarm General

- 11 = ON alarms.
- 12 = OFF alarms.
- 13 = ON hardware alarms.
- 14 = OFF hardware alarms.

CSV_Alarms_Enable

Enables an alarm at a specified animation point in an alarm list.

Syntax

CSV_Alarms_Enable(*iAN*)

iAN:

Animation point number of alarm to enable.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_EnableRec

Enables an alarm by record number.

Syntax

CSV_Alarms_EnableRec(*iRecNo*)

iRecNo:

Record number of alarm to enable.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_GetAckPrivilege()

Checks that the user has privilege level necessary for acknowledging alarms.

Return Value

1 if user has necessary privilege level, otherwise 0.

CSV_Alarms_GetDisablePrivilege()

Checks that the user has privilege level necessary for disabling alarms.

Return Value

1 if user has necessary privilege level, otherwise 0.

CSV_Alarms_GetGroupFilter

Returns the description of the filter currently applied to the alarm list.

Syntax

CSV_Alarms_GetGroupFilter(*iAlarmType,iMonitor,iChars*)

iAlarmType:

Type of alarm list associated with filter:

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iMonitor:

Number of monitor displaying alarm list (-1 = active monitor).

iChars:

Number of characters per line (-1 = single line).

Return Value

Description of the filter currently applied to a specified alarm list, returned as lines if a maximum number of characters per line is specified.

CSV_Alarms_GetGroupFilterID

Returns the name of the group associated with the filter currently applied to a specified alarm list.

Syntax

CSV_Alarms_GetGroupFilterID(*iAlarmType,iMonitor*)

iAlarmType:

Type of alarm list associated with filter:

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iMonitor:

Number of monitor displaying alarm list (-1 = active monitor).

Return Value

Name of alarm group, or "_AdvFilter_" if advanced filter applied, "" if no filter applied.

CSV_Alarms_GetUniqueGroupName

Checks if a group of a specified name exists. If a group already exists with the specified name then a new name is found by appending a number to the original name.

Syntax

CSV_Alarms_GetUniqueGroupName(*sGroupName*)

sGroupName:

Name of a group to check.

> **Note:** Call this function to verify a new group can be created with a specified name, before attempting to create the group.

Return Value

Name of a group not yet assigned (= sGroupName, or modified version of sGroupName).

CSV_Alarms_GroupAdd

Adds an alarm group to the Alarm Group Listbox, and creates a group to store the associated alarm categories. The alarm group is also added to AlarmGrp.dbf. The name of the group is stored in the second field of the listbox (non-visible field), as well as in the "Name" field of the AlarmGrp.dbf.

> **Note:** Alarm groups are used to filter alarms on an alarm page. When a group is selected from the list only alarms having the associated categories are displayed on the alarm page.

Syntax

CSV_Alarms_GroupAdd(*sGroupName*, *sDesc*, *sCategories*, *sArea*)

sGroupName:

Name/ID of alarm group (needs to be unique).

sDesc:

Text describing alarm group that will appear in listbox.

sCategories:

String listing categories represented by alarm group. To have the same format as a standard Citect-SCADA group; for example, "1,5,7..9" = categories 1,5,7,8,9.

sArea:

Area the group applies to. Empty string = every area.

Return Value

Name of the group created, or "" if unsuccessful.

CSV_Alarms_GroupConfig()

Displays a popup window allowing the user to browse/edit/add/delete records in the AlarmGrp.dbf at runtime.

> **Note:** Modifications can be made to alarm groups at run-time, that will be reflected in the list box displaying available alarm groups for filtering.

CSV_Alarms_GroupRemove

Removes an alarm group from the Alarm Group Listbox, and deletes the CitectSCADA group of the same name. The alarm group is also removed from the AlarmGrp.dbf.

> **Note:** Alarm groups are used to filter alarms on an alarm page. When a group is selected from the list, only alarms having the associated categories are displayed on the alarm page.

Syntax

CSV_Alarms_GroupRemove(*sGroupName*)

sGroupName:

Unique Name/ID of alarm group (= second field (non-visible) of Alarm Group listbox, which can be retrieved by calling CSV_ListBox_GetSelectedItemID.)

Return Value

0 if successful, otherwise -1.

CSV_Alarms_GroupEdit

Edits an existing alarm group in the Alarm Group Listbox, and updates the AlarmGrp.dbf.

> **Note:** Alarm groups are used to filter alarms on an alarm page. When a group is selected from the list, only alarms having the associated categories are displayed on the alarm page.

Syntax

CSV_Alarms_GroupEdit(*sGroupName,sDesc,sCategories,sArea*)

sGroupName:

Name/ID of alarm group (needs to be unique).

sDesc:

Text describing alarm group that will appear in listbox.

sCategories:

String listing categories represented by alarm group. To have the same format as a standard Citect-SCADA group; for example, "1,5,7..9" = categories 1,5,7,8,9.

sArea:

Area the group applies to. Empty string = every area.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_GroupFilter

Filters the alarm list starting at a specified animation point for a group of categories.

> **Note:** If the group name = "_AllAlarms_", the "all alarms' is displayed; i.e., the filter is cleared. If the group name = "_AdvFilter_", the selected advanced filter is applied to the alarm list.

Syntax

CSV_Alarms_GroupFilter(*iAN,sGroupName,iAlarmType,iMonitor*)

iAN:

Animation point number of start of alarm list.

sGroupName:

Name/ID of alarm group to filter for.

iAlarmType:

Type of alarm list associated with filter.

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iMonitor:

Number of monitor displaying alarm list (-1 = active monitor).

Return Value

Handle to group, otherwise -1.

CSV_Alarms_GroupSelect

Filters the alarm list starting at a specified animation point for a group of categories. The alarm group may be specified by either the group name or the group description (as found in the AlarmGrp.dbf). Stores the applied filter for a specified monitor and specified alarm page type.

Syntax

CSV_Alarms_GroupSelect(*iAN,sGroupID,sGroupIDType,iAlarmType, iMonitor*)

iAN:

Animation point number of start of alarm list.

sGroupID:

Name/Desc of alarm group to filter for. If sGroupID = "", the filter is cleared.

sGroupIDType:

- 0 = `sGroupID` specifies the alarm group Name.
- 1 = `sGroupID` specifies the alarm group description.

iAlarmType:

Type of alarm list associated with filter:

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iMonitor:

Number of monitor displaying alarm list (-1 = active monitor).

CSV_Alarms_GroupsInit()

Initializes Alarm Group Listbox with groups specified in AlarmGrp.dbf. For each alarm group listed in AlarmGrp.dbf, a group is created to store the alarm categories assigned to the alarm group. Groups are used to filter alarm list. When a group is selected from the list, only alarms having those categories are displayed on the alarm page.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_Help

For alarm at specified animation point in alarm list: Displays page specified in help field of alarm dbf, or if help field begins with "?", calls the function named in the field (i.e., the text following "?").

Return Value

CSV_Alarms_Help(*iAN*)

iAN:

Animation point number of alarm to display help for.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_HelpRec

For alarm at specified record number: displays the page specified in the help field of alarm dbf, or if the help field begins with "?", calls the function named in the field (i.e., the text following "?").

Syntax

CSV_Alarms_HelpRec(*iAN*)

iAN:

Animation point number of alarm to display help for.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_ListHeading

Returns a formatted heading for the specified alarm list type. The heading format is specified by the ini parameters `[Alarm]ActiveHeading`, `[Alarm]SummaryHeading`, `[Alarm]DisabledHeading`, and `[Alarm]HardwareHeading`.

Syntax

CSV_Alarms_ListHeading(*iAlarmType*)

iAlarmType:

Type of alarm list associated with filter.

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

Return Value

Alarm list heading. Returns "" if no heading has been specified.

Example

```
[Alarm]
ActiveHeading = {DATE,12}^t{TIME,14}^t{NAME,15}^t{DESC,40}^t{STATE,10}
```

CSV_Alarms_ListHeadingFont()

Returns the font to use for alarm list headings. The font is specified by the ini parameter `[Alarm]HeadingFont`. If no font is specified the default, (Tahoma, bold, 9 blue) is used.

Return Value

Alarm list heading font.

CSV_Alarms_PopupMenu

Displays popup menu for alarm at specified animation point in alarm list. Available menu items:

- Alarm information
- Acknowledge
- Disable
- Enable
- Help

> **Note:** The Disable/Enable options are available only to user with privilege level specified by `[Privilege] DisableAlarms` parameter. Acknowledge option available only to user with privilege level specified by `[Privilege] AckAlarms` parameter.

Syntax

CSV_Alarms_PopupMenu(*iAN,iAlarmType,iPreserveWinNo*)

iAN:

Animation point number of alarm to display menu for.

iAlarmType:

Type of alarm list:

- 0 = Last alarms list.
- 1 = Active alarms list.
- 2 = Alarm summary list.
- 3 = Hardware alarms list.
- 4 = Disabled alarms list.

iPreserveWinNo:

An optional argument which restores the original window selected upon exiting the function if set to 1.

Allowable Values:

0 = Preserves the original behavior (default).

1 = Restores the original window selected.

Return Value

0 if successful, otherwise -1.

CSV_Alarms_Sound()

Checks if there are unacknowledged alarms in the system, and if there are it sounds the relevant alarm.

CSV_Alarms_SoundActive()

Checks if an alarm is being sounded. This function is used to animate siren in templates, and so on.

Return Value

1 if sound is active, otherwise 0.

CSV_Alarms_Silence()

Silences alarm by setting `miResetAlarmSound`.

CSV_DB_BOF

Checks for the beginning of file flag for a recordset.

Syntax

CSV_DB_BOF(*hRecordSet*)

hRecordSet:

Handle to recordset (as returned from CSV_DB_Execute())

Return Value

0 if not at beginning of file.

CSV_DB_Close

Closes a specified recordset.

Syntax

CSV_DB_Close(*hRecordSet*)

hRecordSet:

Handle to recordset (as returned from CSV_DB_Execute())

Return Value

0 if successful, otherwise -1.

CSV_DB_EOF()

Checks for the end of file flag for a recordset.

Return Value

0 if not at end of file.

CSV_DB_Execute

Executes a command on a specified database. A connection string is used to specify how to connect to the database. If a standby connection string is specified then the standby path is used if the primary path is offline. Make the command an SQL type command, for example:

```
"SELECT * FROM MyTable WHERE TimeValue(Time) > #10:00:00#" etc.
```

Example connection strings:

SQL Server:

```
"Provider=sqloledb;Data Source=MySQLServerName;Initial
Catalog=MyDatabase;User Id=MyUserID;Password=MyPassword;"
```

Access:

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=\somepath\mydb.mdb;User Id=MyUserID;Password=MyPassword;"
```

Oracle:

```
"Provider=OraOLEDB.Oracle;Data Source=MyOracleDB;User
Id=MyUserID;Password=MyPassword;"
```

Excel:

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\somepath\MyExcel.xls; Extended Properties=Excel
8.0;HDR=Yes;IMEX=1"
```

where:

`HDR=Yes`; indicates that the first row contains columnnames, not data

`IMEX=1`; tells the driver to always read "intermixed" data columns as text

Text:

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=c:\somepath\MyTxtFilesFolder\;Extended
Properties=text;HDR=Yes;FMT=Delimited"
```

where:

"`HDR=Yes`;" indicates that the first row contains column names, not data

DBF:

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=c:\somepath\MyDbfFolder;Extended Properties=dBASE IV;User
ID=Admin;Password="
```

DSN:

```
"DSN=MyDsn;Uid=MyUserID;Pwd=MyPassword;"
```

UDL:

```
"File Name=c:\somepath\myDataLink.udl;"
```

Syntax

CSV_DB_Execute(*sCommand,sPrimaryConnection,sStandbyConnection*)

#sCommand:

Command to execute

#sPrimaryConnection:

Connection string for primary connection path.

#sStandbyConnection:

Connection string for standby connection path.

Return Value

Handle to the resulting recordset if successful, otherwise -1.

CSV_DB_GetExecuteError

Returns a description of the error that occurred for the last CSV_DB_Execute command call.

Syntax

CSV_DB_GetExecuteError(*nMode*)

#nMode

Return Value

Error description.

CSV_DB_GetFieldCount

Returns the number of fields contained in a specified recordset.

Syntax

CSV_DB_GetFieldCount(*hRecordSet*)

Return Value

Number of fields if successful, otherwise -1.

CSV_DB_GetFieldIndex

Returns the index of a specified field in a specified recordset.

Syntax

CSV_DB_GetFieldIndex(*hRecordSet,sField*)

#sField:

Name of field.

Return Value

Index of fields if successful, otherwise -1.

CSV_DB_GetFieldName

Returns the name of a field contained in a specified recordset. The field is identified by a field index.

Syntax

CSV_DB_GetFieldName(*hRecordSet,nFieldIndex*)

#nFieldIndex:

Index of field (first field has nFieldIndex = 0).

Return Value

Name of fields if successful, otherwise "".

CSV_DB_GetFieldText

Returns the value of a field (as a string) contained in a specified recordset. The field is identified by a field index.

Syntax

CSV_DB_GetFieldText(*hRecordSet,sField,nFieldIndex,sNullValue*)

#sField:

Name of field. (Leave blank "" if nFieldIndex is to be used instead.)

#nFieldIndex:

Index of field.The first field has nFieldIndex = 0. (Used only if sField = "")

#sNullValue:

Value to return if the field value is Null

> **Note:** If the value of the field is Null then this function will return the string specified by the argument sNullValue.

Return Value

Value of fields if successful, otherwise sNullValue.

CSV_DB_GetRowCount

Returns the number of rows contained in a specified recordset.

Syntax

CSV_DB_GetRowCount(*hRecordSet*)

Return Value

Number of rows if successful, otherwise -1.

CSV_DB_GetRowCurrent

Returns the row number of the current record in a specified recordset.

Syntax

CSV_DB_GetRowCurrent(*hRecordSet*)

Return Value

Current row number if successful, otherwise -1.

CSV_DB_GetRowFieldText

Get the value of a specified field in a specified row of a specified recordset.

Syntax

CSV_DB_GetRowFieldText(*hRecordSet*,*nRowOffset*,*sField*,*nFieldIndex*, *sNullValue*)

#nRowOffset:

Offset of row (from current position)

CSV_DB_MoveFirst

Finds the first record in a specified recordset.

Syntax

CSV_DB_MoveFirst(*hRecordSet*)

Return Value

First record if successful, otherwise -1.

CSV_DB_MoveLast

Finds the last record in a specified recordset.

Syntax

CSV_DB_MoveLast(*hRecordSet*)

Return Value

Last record if successful, otherwise -1.

CSV_DB_MoveNext

Finds the next record in a specified recordset.

Syntax

CSV_DB_MoveNext(*hRecordSet*)

Return Value

Next record if successful, otherwise -1.

CSV_DB_MoveOffset

Finds the record at a specified offset from the current record in a specified recordset.

Syntax

CSV_DB_MoveOffset(*hRecordSet*)

Return Value

Record at specified offset if successful, otherwise -1.

CSV_DB_MovePrev

Finds the previous record in a specified recordset.

Syntax

CSV_DB_MovePrev(*hRecordSet*)

Return Value

Previous record if successful, otherwise -1.

CSV_DB_StandbyConnectionActive

Check the last connection state of the specified primary connection and the specified standby connection.

Syntax

CSV_DB_StandbyConnectionActive(*sPrimaryConnection*, *sStandbyConnection*)

Return Value

If the primary connection state is offline and the standby connection state is online then return 1 otherwise return 0.

CSV_DB_StrToSQL

Replaces single quote with two single quotes to verify SQL interprets single quote as text only.

Syntax

CSV_DB_StrToSQL(*sText*)

#sText:

The text to convert to SQL format

Return Value

Converted text.

CSV_Display_Logo

Displays company logo at specified x and y coordinates. The logo needs to be a 256-color (or less) bitmap file. The default file is "logo.bmp" located in the [RUN] directory. Alternatively, a file name and path may be specified.

> **Note:** The logo will only be displayed on the first scanupdate of the page.

Syntax

CSV_Display_Logo(*iX,iY,sLogoFile*)

#iX:

X coordinate to display top-left corner of logo.

#iY:

Y coordinate to display top-left corner of logo.

#sLogoFile:

File name to display (including path).

CSV_Display_ServicePack()

Gets CitectSCADA Service Pack in the form 'Service Pack A' . The function will only return a value for officially released service packs. Any hotfix being applied will result in the function returning 'Unknown'.

Return Value

CitectSCADA Service Pack as string.

CSV_Display_Title()

Gets window title to display in title bar.

Return Value

Window title.

CSV_Display_Version()

Gets CitectSCADA Version number in the form 5.41.128.

Return Value

CitectSCADA Version number as string.

CSV_File_Display

Displays textRich text file at a text box object AN.

Syntax

CSV_File_Display(*sFile,iAN,iMode,sFontName,sFontSize, iFontColour,iBackColour,iWord-Wrap,iScrollbars*)

#sFile:

Name of file to display.

#iAN:

Animation point number of text box object.

#iMode:

- 1 = Locked (don't allow editing).
- 2 = Allow save (enables save option in popup context menu).
- 4 = Allow create (creates the file if it doesn't already exist).
- 8 = Allow open (enable open option in popup context menu, which allows user to browse for another file to open).

#sFontName:

Name of font.

#sFontSize:

Size of font.

#iFontColour:

Color of font.

#iBackColour:

Color of text box.

#iWordWrap:

Wrap text (for text files only; i.e., not rtf files).

#iScrollbars:

Display scrollbars:

- 0 = None
- 1 = Horizontal
- 2 = Vertical
- 3 = Both

CSV_File_Print

Prints text/Rich text file.

Syntax

CSV_File_Print(*Name*)

#Name:

Name of file to print.

Return Value

0 if successful, otherwise -1.

CSV_File_Save

Saves text/Rich text file.

Syntax

CSV_File_Save(*sFile*)

#sFile:

Name of File to save.

Return Value

0 if successful, otherwise -1.

CSV_Form_Centre

Displays a form in the center of the current monitor screen.

Syntax

CSV_Form_Centre(*iFormX*, *iFormY*)

#iFormX:

Width of form.

#iFormY:

Height of form

CSV_Form_Login()

Displays the login form, gets the user name and password, and then tries to log the user in. If the login does not succeed, it will retry until login is OK or user presses the Cancel button.

Return Value

0 if login successful, otherwise an error (298).

CSV_Form_NumPad

Generates a form that allows the user to enter values through a number pad. The form is displayed on the current ('active') monitor, at the cursor position.

Syntax

CSV_Form_NumPad(*sTitle,sInput,iMode*)

#sTitle:

Title of numeric pad form.

#sInput:

Initial default value.

#iMode:

Indicates the input mode:

- 0 = Normal keypad.
- 1 = With a Password style edit field.
- 2 = Mode not yet implemented.
- 4 = With "+-" button.
- 8 = With "" button.
- 16 = With "." button.
- 32 = With ":" button, not compatible with mode "+-".
- 64 = With "AM" and "PM" buttons, not comparable with mode "" or "."
- 128 = With "Now" button.
- 512 = With "1hr", "2hr", "8hr", "24hr" buttons, not compatible with mode "Now".

Return Value

Returns the string of value entered through the keypad if closed with the accept button, or a null string if closed any other way.

CSV_Form_Position

Displays a form at the specified x,y coordinates, and commands that the entire form be displayed within the boundaries of the current monitor. (i.e., that the x,y coordinates are automatically adjusted if necessary).

Syntax

CSV_Form_Position(*iX*, *iY*, *iFormX*, *iFormY*)

#iX:

Desired X position of top-left of form.

#iY:

Desired Y position of top-left of form.

#iFormX:

Width of form.

#iFormY:

Height of form.

CSV_Form_Shutdown()

Displays a dialog box to verify that the user really wants to shut down the CitectSCADA system. If the user selects [Yes], CitectSCADA will be shut down.

Return Value

0 if shutdown confirmed, otherwise an error (298).

CSV_Form_UserCreate()

Displays a form to create a record for a new user. A new user of the specified type is created. The name of the user needs to be unique.

Return Value

0 if new user is created successfully, otherwise an error.

CSV_Form_UserEdit()

Displays a form to allow the user to create or delete any user record in the database. Give this function restricted access. Changes are written to both the Users database and the runtime database in memory.

Return Value

0 if successful, otherwise an error.

CSV_Form_UserPassword()

Displays a form to allow users to change their own passwords. Changes are written to both the Users database and the runtime database in memory.

Return Value

0 if successful, otherwise an error.

CSV_ListBox_AddItem

Adds item to combo box in ActiveX tag list object.

Syntax

CSV_ListBox_AddItem(*hList,sItem,sCategory,sItemID*)

#hList:

Handle to list object.

#sItem:

Item text to add to list.

#sCategory:

Category of item (list can be filtered by category).

#sItemID:

ID of item (optional, but if used make it unique for each item).

Return Value

0 if successful, otherwise -1.

CSV_ListBox_Clear

Clears ActiveX list object.

Syntax

CSV_ListBox_Clear(*hList*)

#hList:

Handle to list object.

Return Value

0 if successful, otherwise -1.

CSV_ListBox_Create()

Creates ActiveX list object.

> **Note:** This object displays a form that contains a combobox. The form may be displayed or hidden at any time. Items may be added to or removed from the combobox at any time (whether or not the combobox is currently being displayed). The combobox remains in memory until the `CSV_List_Destroy()` function is called for that

combobox.

Return Value

0 if list box was created successfully, otherwise an error.

CSV_ListBox_Create()

Creates new list.

Return Value

Handle to the created list if list box was created successfully; otherwise -1.

CSV_ListBox_Destroy

Destroys ActiveX list object.

Note: Call this function if the listbox is no longer necessary to free memory.

Syntax

CSV_ListBox_Destroy(*hList*)

#hList:

Handle to list object.

Return Value

0 if successful, otherwise -1.

CSV_ListBox_GetCategory

Returns the item category associated with a given ItemID.

Syntax

CSV_ListBox_GetCategory(*hList,sItemID*)

#hList:

Handle to list object.

#sItemID:

ItemID of item.

Return Value

Category of item having ItemID = sItemID.

CSV_ListBox_GetItem

Returns the item text associated with a given ItemID.

Syntax

CSV_ListBox_GetItem(*hList,sItemID*)

#hList:

Handle to list object.

#sItemID:

ItemID of item.

Return Value

Item having ItemID = sItemID.

CSV_ListBox_GetItemID

Returns the item ID associated with a given Item text.

Syntax

CSV_ListBox_GetItemID(*hList,sItem*)

#hList:

Handle to list object.

#sItem:

Item text (as it appears in the listbox).

Return Value

ItemID.

CSV_ListBox_GetSelectedItem

Called when list is displayed. Returns the selected item.

Syntax

CSV_ListBox_GetSelectedItem(*hList*)

#hList:

Handle to list object.

Return Value

Item selected from list.

CSV_ListBox_GetSelectedItemCategory

Call after selection has been made from list. Returns the category of the selected item.

Syntax

CSV_ListBox_GetSelectedItemCategory (*hList*)

#hList:

Handle to list object.

Return Value

Category of item selected from list.

CSV_ListBox_GetSelectedItemID

Call after item has been selected from list to retrieve its Item ID.

Syntax

CSV_ListBox_GetSelectedItemID(*hList*)

#hList:

Handle to list object.

Return Value

ItemID of item selected from list.

CSV_ListBox_GetTagComment

Extracts the tag comment from a string containing the name followed by, in brackets, the tag comment.

Syntax

CSV_ListBox_GetTagComment(*sItem*)

#sItem:

String containing tag name and comment.

Return Value

Comment of tag contained in string sItem

CSV_ListBox_GetTagDescFromTag

Extracts the tag name and comment from the tag name.

Syntax

CSV_ListBox_GetTagDescFromTag(*sTrendTag*)

#sTrendTag:

Name of tag.

Return Value

String containing formatted tag name and comment.

CSV_ListBox_GetTagName

Extracts the tag name from a string containing the name followed by, in brackets, the tag comment.

Syntax

CSV_ListBox_GetTagName(*sItem*)

#sItem:

String containing tag name and comment.

Return Value

Name of tag contained in string sItem.

CSV_ListBox_GetTrendDescFromTag()

Remove item from combo box in ActiveX list object.

Syntax

CSV_ListBox_GetTrendDescFromTag(*#sTrendTag*)

#sTrendTag:

Name of trend tag.

Return Value

String containing formatted trend tag name and comment.

CSV_ListBox_RemoveItem

Removes item from combo box in ActiveX list object.

> **Note:** Two options: 1) Specify both sItem AND sCategory; or 2) Set sItem = "", sCategory = "", and specify only sItemID.

Syntax

CSV_ListBox_RemoveItem(*hList,sItem,sCategory,sItemID*)

#hList:

Handle to list object.

#sItem:

Item to remove from list.

#sCategory:

Category of item.

#sItemID:

ID of item.

Return Value

0 if successful, otherwise -1.

CSV_ListBox_Hide

Hides list.

Syntax

CSV_ListBox_Hide(*hList*)

#hList:

Handle to list object.

Return Value

0 if successful, otherwise -1.

CSV_ListBox_SelectCategories

Select categories of items to be displayed in list (filters list to display only items having specified category. More than one category can be displayed by separating each category with a comma (and no spaces between categories).

Note: Categories = "" -> removes category filter.

Syntax

CSV_ListBox_SelectCategories(*hList*, *sCategories*)

#hList:

Handle to list object.

#sCategories:

Categories to filter list for.

Return Value

0 if successful, otherwise -1.

CSV_ListBox_SelectTags()

Creates an ActiveX object which provides a combo box to allow a tag to be selected from a list. If a tag list object already exists a new instance of it is created.

Return Value

Handle to tag list object.

CSV_ListBox_SelectTrends()

Creates an ActiveX object which provides a combo box to allow a trend tag to be selected from a list. If a trend tag list object already exists a new instance of it is created.

Return Value

Handle to trend list object.

CSV_ListBox_SetText

Set title, description, **OK** button, and **Cancel** button text on ActiveX list object.

Syntax

CSV_ListBox_SetText(*hList*,*sTitle*,*sDesc*,*sOK*,*sCancel*)

#hList:

Handle to list object.

#sTitle:

Title appearing on form.

#sDesc:

Description appearing on form.

#sOK:

Text displayed on OK button.

#sCancel:

Text displayed on Cancel button.

Return Value

0 if successful, otherwise -1.

CSV_ListBox_Show

Displays list of tags.

Syntax

CSV_ListBox_Show(*hList,sTitle,sDesc,sOK,sCancel,iX,iY*)

#hList:

Handle to list object.

#sTitle:

Title appearing on form.

#sDesc:

Description appearing on form.

#sOK:

Text displayed on OK button.

#sCancel:

Text displayed on Cancel button.

#iX:

X coordinate of left corner, or -9999 to center horizontally on active monitor.

#iY:

Y coordinate of top corner, or -9999 to center vertically.

Return Value

Item selected from list (returns empty string if no item selected)

CSV_ListBox_TagFormat

Formats a string to contain the name of the specified variable followed by, in brackets, the comment associated with the variable. Called before adding a variable to a drop down list of variables available for trending. Formats each item in the drop down list.

Syntax

CSV_ListBox_TagFormat(*sVariable*)

#sVariable:

Name of variable to be formatted.

CSV_ListBox_Visible

Checks if a ListBox is currently visible.

Syntax

CSV_ListBox_Visible(*hObject*)

#hObject:

Handle to list object.

Return Value

1 if list is currently visible, otherwise 0.

CSV_Math_RoundDown

Rounds a real value down (toward 0) to a specified number of decimal places.

Syntax

CSV_Math_RoundDown(*rValue*, *iDecPlaces*)

#rValue:

The value to be rounded down.

#iDecPlaces:

The number of decimal places the value is rounded down to.

Example

```
CSV_Math_RoundDown(4.328, 2) = 4.32
CSV_Math_RoundDown(4.321, 2) = 4.32
CSV_Math_RoundDown(-4.321, 2) = -4.32
CSV_Math_RoundDown(512.3, -2) = 500
```

Return Value

Rounded value.

CSV_Math_Truncate

Truncates a real value down to an integer value.

Syntax

CSV_Math_Truncate(*rValue*)

Example

```
CSV_Math_Truncate(4.328) = 4
CSV_Math_Truncate(5.867) = 5
```

Return Value

Truncated value (as integer).

CSV_MenuConfig_Close()

Closes the Menu Configuration popup. If changes have not been saved, a prompt to save the configuration will appear.

Return Value

0 if successful, otherwise -1.

CSV_MenuConfig_Display()

Displays Menu configuration popup, which gives the user the ability to configure menus at runtime.

Return Value

0 if successful, otherwise -1.

CSV_MenuConfig_LoadDflt()

Loads a default menu configuration from the [Bin] directory.

Return Value

0 if successful, otherwise -1.

CSV_MenuConfig_UserPages()

Updates the menu configuration to allow the user to select from the "Pages" menu every non-system page (maximum number of pages = 25).

Return Value

0 if successful, otherwise -1.

CSV_MessageBox

Displays a message box centered on the active monitor screen and waits for the user to select a button. Can display up to three buttons, as well as a checkbox. Can disappear after specified timeout. The maximum timeout is 30s if this is used. If 0 is passed in then no timeout applies.

When using 1,2 or 3 custom buttons: due to the way the underlying widget works, 1 or 2 button custom popups do not have the Cancel or Timeout Feature. The 3 button version does have Timeout. It is recommended when using 2 buttons to use this syntax :

"button1", "button2", "Cancel"

This will allow your 2 button selection to have a timeout feature. In this example your Cicode needs to use the '2' button reply as meaning cancel (299).

Error 359 is returned when a 2nd popup is attempts to display with the same title. The location of the message box is the same so multiple popup boxes // can be problematic.

Syntax

CSV_MessageBox(*sTitle,sPrompt,iMode,iTimeout,sButton1Text,sButton2Text,sBut-ton3Text,sCheckboxText*)

#sTitle:

Message box title

#sPrompt:

Message box prompt

#iMode :

- 0 - OK button only (default)
- 1 - OK and Cancel buttons
- 2 - Abort, Retry, and Ignore buttons
- 3 - Yes, No, and Cancel buttons
- 4 - Yes and No buttons
- 5 - Retry and Cancel buttons
- 16 - Critical message
- 32 - Warning query
- 48 - Warning message
- 64 - Information message
- 0 - First button is default (default)
- 256 - Second button is default
- 512 - Third button is default
- 768 - Fourth button is default
- 0 - Application modal message box (default)
- 4096 - System modal message box
- 16384 - Adds Help button to the message box
- 65536 - Specifies the message box window as the foreground window
- 524288 - Text is right aligned
- 1048576 - Specifies text to appear as right-to-left reading on Hebrew and Arabic systems

#iTimeout:

The number of seconds before the message box disappears.

#sButton1Text:

Text for first button

#sButton2Text:

Text for second button

#sButton3Text:

Text for third button

#sCheckBoxText:

Text for the checkbox

Return Value

If *sButtonText1*="" OR the 3 TextBoxes are in use then:

Return Value	Description
0	OK button pressed
299	Cancel button pressed
359	A Popup with the same title is already displayed
512	A timeout has occurred
3	Abort button pressed
4	Retry button pressed
5	Ignore button pressed
6	Yes button pressed
7	No button pressed

Else:

Return Value	Description
0	First button pressed
1	Second button pressed
2	Third button pressed
359	A Popup with the same title is already displayed

If *sCheckBoxText* <> "" then 1024 is added to the above return values.

CSV_Misc_CheckNumPadValue

Uses the MultiMonitor Numpad to get a value, then checks the value's range and returns the new value, or the old if range is incorrect.

Syntax

CSV_Misc_CheckNumPadValue(*sDESC*, *rValue*, *rUpLimit*, *rLowLimit*)

#sDESC:

The description to appear in the form numpad title (as a string)

#rValue:

The original value to be changed (as a real or int)

#rUpLimit:

The Upper limit that the original value can be changed to (as a real or int)

#rLowLimit:

The Lower limit that the original value can be changed to (as a real or int)

Return Value

The new value or the original value if out of range.

Example

```
Tag = CSV_Misc_CheckNumPadValue("change Value", Tag, 190, 10)
! This will means that Tag can only have values of 10 - 190 written to it via the form-
NumPad.
```

CSV_Misc_IntRange

Checks the range is valid for Integers; if not, a message box appears informing the user of the correct range.

Syntax

CSV_Misc_IntRange(*LowerRange*,*UpperRange*,*OriginalValue*,*NewValue*)

#LowerRange:

The Lower range of the necessary Range

#UpperRange:

The Upper range of the necessary Range

#Original Value:

The value to be change back too; this is used if the value is invalid or out of range.

#New Value:

The new value to change to.

Return Value

The new value, or the original value if out of range.

CSV_Misc_MouseOver

Returns TRUE if the mouse is inside the region defined by the extents of the object at 'hAN'.

Syntax

CSV_Misc_MouseOver(*hAN*)

#hAN:

The animation number of the display object

Return Value

- TRUE (1) if the mouse cursor is inside the region bounded by the extents of the specified display object
- FALSE (0) otherwise.

CSV_MM_BackEmpty()

Checks if backward navigation is possible.

Note: If CSV_MM_BackEmpty() = 1, disable backward navigation button (there are no pages on the last-page stack that may be navigated in a backward direction from the current position).

Return Value

1 if backward navigation is not possible, otherwise 0.

CSV_MM_ConfigInit()

Initializes parameters needed for multi-monitor functionality. Initializes queues for storing last pages displayed (last page stack) Parameter values are read from .ini file `[MultiMonitors]` section:

- **Number of monitors** (parameter = "Monitors", default = 1).
- **Screen width of monitor** (parameter = "ScreenWidth", default = 1024).
- **Startup pages for each monitor** (parameter = "Startup1","Startup2",... etc. depending on number of monitors, default = "Startup").

Size of last page stack (parameter = "LastPageStackSize", default = 10).

CSV_MM_FwdEmpty()

Checks if forward navigation is possible (only possible if backward navigation has been used).

Return Value

1 if forward navigation is not possible, otherwise 0.

> **Note:** If CSV_MM_FwdEmpty() = 1 then disable forward navigation button (there are no pages on the last-page stack that may be navigated in a forward direction from the current position).

CSV_MM_GetMonitor()

Gets the number of the currently active monitor. The 'active' monitor is the monitor that contains the largest part of the area of the currently active window.

Return Value

Number of currently active monitor.

CSV_MM_GetMonitors()

Gets number of monitors, as set by `Monitors` parameter in [`MultiMonitors`] section of the .ini file.

Return Value

Number of monitors.

CSV_MM_GetMouseX

Gets X coordinate of mouse position with respect to desktop, monitor, or window.

Syntax

CSV_MM_GetMouseX(*iMode*)

#iMode :

- 0 = Gets mouse position with respect to top-left corner of desktop.
- 1 = Gets mouse position with respect to top-left corner of active monitor.
- 2 = Gets mouse position with respect to top-left corner of active window.

CSV_MM_GetMouseY

Gets Y coordinate of mouse position with respect to desktop, monitor, or window.

Syntax

CSV_MM_GetMouseY(*iMode*)

#iMode :

- 0 = Gets mouse position with respect to top-left corner of desktop.
- 1 = Gets mouse position with respect to top-left corner of active monitor.

- 2 = Gets mouse position with respect to top-left corner of active window.

CSV_MM_GetOffset

Gets X-offset of selected monitor. To display a page on the selected monitor the page needs to have its X coordinate set to this value.

Syntax

CSV_MM_GetOffset(*iMonitorNo*)

#iMonitorNo:

Number of monitor to get X-offset for.

Return Value

Offset of monitor.

CSV_MM_GetScreenWidth()

Gets width of screen, as set by `ScreenWidth` parameter in `[MultiMonitors]` section of .ini file.

Return Value

Width of screen.

CSV_MM_ListLastPages

Displays on the active monitor a menu listing pages that may be navigated backwards or forwards from the current page. A stack stores recently displayed pages in the order in which they were displayed. This function can be used to allow these pages to be selected for display.

Syntax

CSV_MM_ListLastPages(*Mode*)

#Mode :

- 0 = Lists pages which may be navigated backwards.
- 1 = Lists pages which may be navigated forwards.

Return Value

0 if successful, otherwise -1.

CSV_MM_MonitorFromPoint

Gets number of monitor containing point specified.

Syntax

CSV_MM_MonitorFromPoint(*iX, iY*)

#iX:

X-coordinate of point.

#iY:

Y-coordinate of point.

Return Value

Number of monitor containing specified point.

CSV_MM_MonitorFromWindow

Gets number of monitor intersecting the largest area of the specified window.

Syntax

CSV_MM_MonitorFromWindow(*iWindowNo*)

#iWindowNo:

Window number to get monitor number for.

Return Value

Number of monitor associated with window.

CSV_MM_MonitorGoto

Goes to main window of specified monitor.

Syntax

CSV_MM_MonitorGoto(*iMonitor*)

#iMonitor:

Number of monitor to go to.

Return Value

Number of main window associated with monitor if successful, otherwise -1.

CSV_MM_NextEmpty()

Checks if a 'Next' page has been defined for the current page.

> **Note:** If `CSV_MM_NextEmpty() = 1` then disable 'Next Page' navigation button.

Return Value

1 if 'Next Page' has not been defined, otherwise 0.

CSV_MM_PageDisplay

Displays selected page on the 'active' monitor, or a pre-selected monitor.

Syntax

CSV_MM_PageDisplay(*sPage,iMonitor,bStoreLastPage,sStoreFunction*)

#sPage:

Name of page to display.

#iMonitor:

Number of monitor to display page on. First monitor = '0', Second = '1' etc. If iMonitor = -1 then page is displayed on the 'active' (that is. currently selected) monitor.

#bStoreLastPage:

Add page to last page stack. If bStoreLastPage = 0 then the page is not written to the queue that stores the previous pages displayed.

#sStoreFunction:

Name of function to store on last page stack.

If a function has been specified then that function will be called when navigating through the last pages, rather than displaying the page.

To include arguments append a space and then a comma-separated list of the arguments (string constants) to the function name.

Return Value

`0' if successful, otherwise an error number.

CSV_MM_PageLast

Navigates last page stack. Allows moving backward and (subsequently) forward through a predefined number of previously displayed pages, in the order in which they were displayed. The stack is unique to the currently active monitor. that is. only the last pages displayed on the active monitor are navigated.

Syntax

CSV_MM_PageLast(*iMode*)

#iMode:

Direction of navigation:

- 0 = backwards (Default).
- 1 = forwards.

Return Value

0 if successful, otherwise -1

CSV_MM_PageNext()

Displays 'Next page' of currently active page. Page is displayed on same monitor (that is. currently active monitor).

Return Value

0 if successful, otherwise -1.

CSV_MM_PagePrev()

Displays 'Previous page' of currently active page. Page is displayed on same monitor (i.e., currently active monitor).

Return Value

0 if successful, otherwise -1.

CSV_MM_PagesInit()

Displays startup pages. Parameter values are read from .ini file [MultiMonitors] section.

> **Note:** This function is to be called on startup for clients requiring multiple-monitor support. To implement this without requiring a call to this function from within the startup Cicode function, it has been configured as a periodic event (listed as a 'CSV_MultiMonitor' event). The first time the event is processed the multi-monitor functionality is initialized. Subsequent calls return immediately without effect.

CSV_MM_PreviousEmpty()

Checks if a 'Previous' page has been defined for the current page.

> **Note:** If CSV_MM_PreviousEmpty() = 1 then disable 'Previous Page' navigation button.

Return Value

1 if 'Previous Page' has not been defined, otherwise 0.

CSV_MM_StoreLastPage

Adds page to last page stack for selected monitor. Page Title is written to queue that stores pages in order of access. (Each monitor has its own queue.) The action to perform when navigating through the last page stack is also stored.

Syntax

CSV_MM_StoreLastPage(*iMonitorNo,sPageAction,sPageTitle*)

#iMonitorNo:

Number of monitor page was displayed on.

#sPageAction:

Name of action to store on last page stack.

To specify a function, prefix the function name with "?" If a function has been specified then that function will be called when navigating through the last pages, rather than displaying the page.

To include arguments, append a space and then a comma-separated list of the arguments (string constants) to the function name.

#sPageTitle:

Name of page displayed.

Return Value

1 if backward navigation is not possible, otherwise 0.

CSV_MM_WinDrag()

Moves active window with mouse; i.e., window position will track mouse movements.

> **Note:** Call `CSV_MM_WinDragEnd` to end dragging of window.

CSV_MM_WinDragEnd()

Ends window dragging initiated by `CSV_MM_WinDrag()`.

CSV_MM_WinFree()

Closes active window, if active window is not main window for a monitor.

Calling `CSV_MM_WinFree` rather than `WinFree` verifies that assigned monitors maintain at least one open window. That window will be the one opened by the `CSV_MM_PageDisplay` function.

Always call `CSV_MM_WinFree` to close a window if multi-monitor functionality has been implemented.

Return Value

0 if successful, otherwise an error is returned. -1 indicates that you attempted to close the main window of a monitor.

CSV_MM_WinNewAt

Displays a new window at the X and Y coordinates relative to the top-left corner of active monitor.

Syntax

CSV_MM_WinNewAt(*sPage,iX,iY,iMode*)

#sPage:

Name of pagewindow to display.

#iX:

X-offset to display window at relative to left of monitor.

#iY:

Y-offset to display window at relative to top of monitor.

#iMode:

Display mode (same settings as for 'WinNewAt' function, except that the window by default will be 'always on top', regardless of whether or not you add 64 to the mode.This verifies that the popup window does not disappear behind the main window. To de-select this option add 2048 to the mode). Dynamic resizing will be disabled unless 4096 is added to the mode. To center the window within the page add 8192 to the mode.

Return Value

The window number of the window, or -1 if the window cannot be opened.

CSV_MM_WinPopup

Display popup window at x and y coordinates relative to top left corner of active monitor.

Syntax

CSV_MM_WinPopup(*sWindow*, *iX*, *iY*, *iHideTitleBar*)

#sWindow:

Name of page window to display.

#iX:

X offset to display window at relative to left of monitor.

#iY:

Y offset to display window at relative to top of monitor.

#iHideTitleBar :

- 0 = display window standard title bar.
- 1 = don't display title bar (for XP style window).

Return Value

The window number of the window, or -1 if the window cannot be opened.

> **Note:** The entire window is displayed within the borders of a single screen. If iX = -1 and iY = -1, the window is centered on screen.

CSV_MM_WinTitle

Sets the window title. Call this function rather than `WinTitle` to set window title. Changes the title of the page on the last page stack if the window is a main page. Shows the correct page title in the forward/back navigation drop down list.

Syntax

CSV_MM_WinTitle(*sTitle*)

#sTitle:

Title of window.

Return Value

0 if successful, otherwise an error.

CSV_Nav_Alarms()

Displays Alarm page, or calls function defined for alarm page.

> **Note:** The Network page is defined by the parameter [Navigation] AlarmPage. To specify a function prefix the function name with "?"

Return Value

0 if successful, otherwise -1.

CSV_Nav_AlarmsDisabled()

Displays Disabled Alarm page, or calls function defined for disabled alarm page.

> **Note:** The Network page is defined by the parameter `[Navigation]DisabledPage`. To specify a function prefix the function name with "?".

Return Value

0 if successful, otherwise -1.

CSV_Nav_AlarmsHardware()

Displays Hardware Alarm page, or calls function defined for hardware alarm page.

> **Note:** The Network page is defined by the parameter [Navigation] HardwarePage. To specify a function prefix the function name with "?"

Return Value

0 if successful, otherwise -1.

CSV_Nav_AlarmsSummary()

Displays Alarm page, or calls function defined for alarm page.

> **Note:** The Network page is defined by the parameter `[Navigation]SummaryPage`. To specify a function prefix the function name with "?"

Return Value

0 if successful, otherwise -1.

CSV_Nav_CloseWindow()

Displays form to enable user to shutdown CitectSCADA.

CSV_Nav_DisableMenuItem

Disables/enables a specified item in a specified popup menu. A disabled menu item appears embossed in the popup menu and cannot be selected.

Syntax

CSV_Nav_DisableMenuItem(*iMode,sMenuItem,sSubMenuItem,sMenuName,sPageName*)

#iMode :

- 1 = disable menu item.
- 0 = enable menu item.

#sMenuItem:

Menu item to enable/disable.

#sSubMenuItem:

Submenu item to enable/disable(if applicable).

#sMenuName:

Name of menu (that is. button associated with popup menu).

#sPageName:

Name of page associated with menu.

Return Value

0 if successful, otherwise -1.

CSV_Nav_DisplayMenuBar

Creates menu bar for specified page. The PageMenu.dbf (previously named Menu.dbf) is accessed to determine what buttons appear in the menu bar. A new menu bar (ActiveX object) is created with the specified buttons.

Syntax

CSV_Nav_DisplayMenuBar(*sPageName,iX,iY,nBackColour,nForeColour*)

#sPageName:

Name of page.

#iX:

X-coordinate of top left corner of menu bar.

#iY:

Y-coordinate of top left corner of menu bar.

#nBackColour:

Background color of menu bar (CitectSCADA palette number).

#nForeColour:

Foreground (font) color of menu bar (CitectSCADA palette number).

Return Value

0 if successful, otherwise -1.

CSV_Nav_DisplayPopupMenu

Displays popup menu for specified page and specified menu. Top left corner of menu is displayed at nominated x,y coordinates.

Syntax

CSV_Nav_DisplayPopupMenu(*sPageName,sMenuName,iX,iY*)

#sPageName:

Name of page.

#sMenuName:

Name of menu.

#iX:

X-coordinate of top left corner of popup menu.

#iY:

Y-coordinate of top left corner of popup menu.

Return Value

0 if successful, otherwise -1.

CSV_Nav_File

Displays text/Rich text file.

Syntax

CSV_Nav_File(*sTitle,sFile,iMode,sFontName,iFontSize,iFontColour, iBackColour,iWordWrap*)

#sTitle:

Title to appear on file page.

#sFile:

File name including path (for example, "[Run]:\file.txt").

#iMode :

- 1 = Locked (don't allow editing).
- 2 = Allow save (enables save option in popup context menu).
- 4 = Allow create (creates the file if it doesn't already exist).
- 8 = Allow open (enable open option in popup context menu - allows user to browse for another file to open).

#sFontName:

Name of font to display file in (if not an rtf file).

#iFontSize:

Size of font (if not an rtf file).

#iFontColour:

Color of font (if not an rtf file).

#iBackColour:

Color of background.

#iWordWrap:

Enable word wrap.

Return Value

0 if successful, otherwise -1.

CSV_Nav_GetEngToolsPrivilege()

Checks that the user has the privilege level necessary for engineering tools.

Return Value

1 if user has necessary privilege level, otherwise 0.

CSV_Nav_Home()

Displays Home page, or calls function defined for home page.

Note: The Home page is defined by the parameter `[Navigation]HomePage`. To specify a

function prefix the function name with "?"

Return Value

0 if successful, otherwise -1.

CSV_Nav_Login()

Displays popup form allowing user to login.

CSV_Nav_LoginMenu()

Displays popup menu for Screen Login.

Note: Login popup menu is defined by the "Template" page and "Login" menu in the PageMenu.dbf (previously named Menu.dbf). If no "Login" menu has been defined in this section of the PageMenu.dbf then a default menu is displayed.

CSV_Nav_MenuBar_MenuClick

Event triggered by clicking a button in the ActiveX menu bar.

Syntax

CSV_Nav_MenuBar_MenuClick(*sPageName,sButtonName,iX,iY*)

#sPageName:

Name of page containing menu bar.

#sButtonName:

Name of button clicked.

#iX:

X-coordinate of top-left corner of menu bar.

#iY:

Y-coordinate of top-left corner of menu bar.

CSV_Nav_Network()

Displays Network page, or calls function defined for network page.

Note: The Network page is defined by the parameter [Navigation] NetworkPage. To

specify a function prefix the function name with "?"

Return Value

0 if successful, otherwise -1.

CSV_Nav_NetworkBtnEnabled()

Checks if network page exists.

Return Value

1 if network page exists, or function has been specified for network page.

CSV_Nav_PageExists

Checks if a page exists by attempting to locate its associated runtime file.

Syntax

CSV_Nav_PageExists(*sPage*)

#sPage:

Name of page to check.

Return Value

1 if page exists, otherwise 0.

CSV_Nav_PagePrint()

Creates a screen print of the active page, or calls the function defined for page print.

Note: The print function is defined by the page environment variable "PrintPage" if it exists, otherwise by the parameter `[Navigation] PrintPage`. To specify a function prefix the function name with "?". If no function has been defined, a screen print will be performed.

CSV_Nav_Parent()

Displays page configured as ParentPage environment variable for current page, or calls function specified by ParentPage.

Return Value

0 if successful, otherwise -1.

> **Note:** To specify a function prefix the function name with "?".

CSV_Nav_ParentBtnEnabled()

Checks if a page has been defined for the current page.

Return Value

1 if parent page has been defined.

CSV_Nav_Report()

Displays Report page, or calls function defined for report page.

> **Note:** The Network page is defined by the parameter `[Navigation]ReportPage`. To specify a function prefix the function name with "?".

Return Value

0 if successful, otherwise -1.

CSV_Nav_ReportBtnEnabled()

Checks if Report page exists.

Return Value

1 if Report page exists, or function has been specified for Report page.

CSV_Nav_ReportMenu

Displays popup menu for Reports.

> **Note:** Report popup menu is defined by the "Template" page and "Reports" menu in the PageMenu.dbf (previously named Menu.dbf).

Syntax

CSV_Nav_ReportMenu(*iX*,*iY*)

#iX:

X-coordinate of popup menu position.

#iY:

Y-coordinate of popup menu position.

CSV_Nav_Tools()

Displays Tools page, or calls function defined for tools page.

> **Note:** The Tools page is defined by the parameter `[Navigation]ToolsPage`.

To specify a function, prefix the function name with "?".

Return Value

0 if successful, otherwise -1.

CSV_Nav_ToolsBtnEnabled()

Checks if Tools page exists.

Return Value

1 if Tools page exists, or function has been specified for Tools page.

CSV_Nav_ToolsMenu()

Displays popup menu for Screen Tools.

> **Note:** Tools popup menu is defined by the "Template" page and "Tools" menu in the PageMenu.dbf (previously named Menu.dbf). If no Tools menu has been defined in this section of the PageMenu.dbf, a default menu is displayed.

CSV_Nav_Trend()

Displays Trend page, or calls function defined for trend page.

> **Note:** The Trend page is defined by the parameter [Navigation] TrendPage. To specify a function prefix the function name with "?".

Return Value

0 if successful, otherwise -1.

CSV_Nav_TrendBtnEnabled()

Checks if Trend page exists.

Return Value

1 if Trend page exists, or function has been specified for Trend page.

CSV_Nav_TrendMenu()

Displays popup menu for Trends.

> **Note:** Trend popup menu is defined by the "Template" page and "Trends" menu in the PageMenu.dbf (previously named Menu.dbf).

CSV_Nav_TrendX()

Displays Instant Trend page.

> **Note:** To implement this function, you need to add the CSV_InstantTrend project as an Included project. (See "Including a project in the current project" in the Citect-SCADA User Guide.)

Return Value

0 if successful, otherwise -1.

CSV_Nav_TickMenuItem

Checks/unchecks a specified item in a specified popup menu. A checked menu item appears with a tick beside it in the popup menu.

Syntax

CSV_Nav_TickMenuItem(*iMode,sMenuItem,sSubMenuItem,sMenuName,sPageName*)

#iMode :

- 1 = Check menu item.
- 0 = Uncheck menu item.

#sMenuItem:

Menu item to check/uncheck.

#sSubMenuItem:

Submenu item to check/uncheck (if applicable).

#sMenuName:

Name of menu (i.e., button associated with popup menu).

#sPageName:

Name of page associated with menu.

Return Value

0 if successful, otherwise -1.

CSV_Sec_ShowLoginMenu

Displays a popup menu allowing user to login, logout, change the password, and, if the user has the necessary privilege, to edit a user or add a user.

Syntax

CSV_Sec_ShowLoginMenu(*iXpos,iYpos,iUserEditPrivilege*)

#iXpos:

X position of top-left corner of popup menu.

#iYpos:

Y position of top-left corner of popup menu.

#iUserEditPrivilege:

Privilege necessary to edit or add a user.

CSV_String_GetField

Gets a field value (text) from a string, where the string consists of a number of fields separated by a field separation character.

Syntax

CSV_String_GetField(*sText,iField,sFieldSeparator*)

#sText:

String containing fields.

#iField:

Index of field value to return (starting at 1).

#SFieldSeparator:

Field separation character.

Return Value

Field value as string.

Example

```
sText = "ab?cde?fghi?j";
sField = CSV_String_GetField(sText,3,"?");
In this case sField = "fghi"
```

CSV_String_GetLines

Returns the number of lines in a string, given a maximum number of characters per line.

Syntax

CSV_String_GetLines(*sText, iChars*)

#sText:

Text to convert to lines.

#iChars:

Maximum number of characters per line.

Return Value

Number of lines that text would be converted to.

CSV_String_Replace

Returns a string in which a specified substring has been replaced with another substring a specified number of times.

Syntax

CSV_String_Replace(*sTextString,sFind,sReplace,iStart,iCount*)

#sTextString:

Expression containing substring to replace.

#sFind:

Substring being searched for.

#sReplace:

Replacement substring.

#iStart:

Optional. Position within expression where substring search is to begin. If omitted, 0 is assumed.

#iCount:

Optional. Number of substring substitutions to perform. If omitted, the default value is -1, which means make every possible substitution.

CSV_Tag_Debug

Builds a form to provide simple user access to every Variable Tag during runtime. Reading and writing are supported. The Form is always on top, and only one instance is allowed.

Syntax

CSV_Tag_Debug()

Return Value

Name of selected tag.

> **Note:** Uses a listbox object to display every tag in system. List may be filtered.

CSV_Trend_AutoScale

Auto scales trend pens, such that the 100% scale is 10% of the full tag range above the maximum tag value in the viewable trend window, and the 0% scale is 10% of the tag range below the minimum tag value in the viewable trend window.

Syntax

CSV_Trend_AutoScale(*hTrendAN*)

#hTrendAN:

Animation point number of the trend.

CSV_Trend_DspGroup

Displays a specified group of trend pens on a specified trend page. The group of trend pens need to have been defined in the TrendGrp.dbf file in the [RUN] directory. The group may be specified by either the group name or the group description.

Syntax

CSV_Trend_DspGroup(*sTitle,sTrendPage,hTrendAN,sTrendID,iTrendIDType, iTrendDataSet*)

#sTitle:

Title to appear on trend page.

#sTrendPage:

Name of trend page to display.

#hTrendAN:

Animation point number of trend.

#sTrendID:

Name or Desc of trend group (found in TrendGrp.dbf).

#iTrendIDType:

The type of the trend. Two possible values:

- 0 = `sTrendID` specifies the Name of the trend group.
- 1 = `sTrendID` specifies the description of the trend group.

#iTrendDataSet:

Identifies the data set to be used for the group.

Normal trend page uses data set 0, double trend page uses data sets 1 and 2.

CSV_Trend_DspGroupList

Displays available groups of trend tags in a listbox. Returns the description of the item selected from the list. Groups are configured in the TrendGrp.dbf file found in the [RUN] directory.

Syntax

CSV_Trend_DspGroupList(*sSelectedGroup,sAreas*)

#sSelectedGroup:

Name of group to preselect in the list.

#sAreas:

Areas to enable in the list; i.e., only trend groups belonging to these areas are displayed.

Return Value

Trend group (description) selected from the list, or "" if cancel is pressed.

CSV_Trend_DspPopupMenu

Displays a popup menu to allow the user to add or clear the selected pen.

Syntax

CSV_Trend_DspPopupMenu(*hTrendAN*,*iPen*)

#hTrendAN:

Animation point number of the trend.

#iPen:

Number of selected pen.

Return Value

Description of trend group.

CSV_Trend_DspScaleRange

Returns the current displayed scale range for a specified trend pen, in the format: "Lo - HiEU" where Lo = RangeMin, Hi = RangeMax, and EU = engineering units.

Syntax

CSV_Trend_DspScaleRange(*hTrendAN*,*iPen*)

#hTrendAN:

Animation point number of the trend.

#iPen:

Number of the trend pen.

Return Value

Formatted range value as a string.

CSV_Trend_DspTrendText

Returns the comment for the trend tag plotted by the specified pen if a comment exists, otherwise returns the name of the trend tag.

Syntax

CSV_Trend_DspTrendText(*hTrendAN*,*iPen*)

#hTrendAN:

Animation point number of the trend.

#iPen:

Number of the trend pen.

Return Value

Trend tag comment if it exists, otherwise the trend tag name (all capitalized).

CSV_Trend_GetCursorPos

Gets the offset of a trend cursor from its origin, in samples.

Syntax

CSV_Trend_GetCursorPos(*hTrendAN*)

#hTrendAN:

Animation point number of the trend.

Return Value

The offset of a trend cursor from its origin, in samples, or -1 if the trend cursor is disabled.

CSV_Trend_GetCursorTypeStr

Returns text indicating whether the cursor is displayed. Used in conjunction with `CSV_Trend_GetCursorValueStr()` to notify the user whether the displayed trend tag value corresponds to the value at the cursor, or the current value.

Syntax

CSV_Trend_GetCursorTypeStr(*hTrendAN*)

#hTrendAN:

Animation point number of the trend.

Return Value

Returns "Current Value" if the cursor is not displayed, or "Cursor Value" if the cursor is displayed.

CSV_Trend_GetCursorValueStr

Gets the value of a trend pen at the cursor position, or the current value of the trend pen if the cursor is disabled. The value is returned as a string, optionally followed by the engineering units of the tag.

Syntax

CSV_Trend_GetCursorValueStr(*hTrendAN*, *iPen*, *iEngUnits*)

#hTrendAN:

Animation point number of the trend.

#iPen:

Number of the trend pen.

#iEngUnits:

Append the engineering units to the cursor value returned.

Return Value

Value of the trend pen at the cursor position, or its current value if the cursor is not displayed.

CSV_Trend_GetGroup

Gets the description of the group of trends (as defined in TrendGrp.dbf) currently displayed (or last displayed) on a specified monitor.

Syntax

CSV_Trend_GetGroup(*iMonitor*, *iTrendDataSet*)

#iMonitor:

Number of monitor the trend is/was displayed on.

#iTrendDataSet:

Identifies the data set to be used for the group of trend tags. Normal trend page uses data set 0; a double trend page uses data sets 1 and 2.

Return Value

Description of trend group.

CSV_Trend_GetMode

Gets the mode (real-time or historical trending) of the trend pen.

Syntax

```
CSV_Trend_GetMode(hTrendAN)
```

#hTrendAN:

Animation point number of the trend.

Return Value

The current mode: 0 for real-time or 1 for historical.

CSV_Trend_GetPen

Gets the trend tag being plotted by a specified pen.

Syntax

CSV_Trend_GetPen(*hTrendAN, iPen*)

#hTrendAN:

Animation point number of the trend.

#iPen:

Number of pen.

Return Value

Trend tag of specified pen.

CSV_Trend_GetPenFocus

Gets the trend pen currently in focus.

Syntax

CSV_Trend_GetPenFocus(*hTrendAN*)

#hTrendAN:

Animation point number of the trend.

Return Value

Number of pen in focus.

CSV_Trend_GetSettings

Reads an .ini file to recall (Get) the settings (Tags displayed and scales) for the current page. This function will allocate a separate section in the .ini file for each page.

Syntax

CSV_Trend_GetSettings(*sPage, hTrendAN*)

#sPage:

The reference for the settings to recall.

#hTrendAN:

Animation point number of the trend.

Example

```
[TrendPage1]
Tag_1=TrendTag1
Zero_1=0.
Full_1=1000.
Tag_2=TrendTag2
Zero_2=0.
Full_2=1000.
Tag_3=TrendTag3
Zero_3=0.
Full_3=1000.
Tag_4=TrendTag4
Zero_4=0.
Full_4=1000.
Tag_5=
Tag_6=
Tag_7=
Tag_8=
```

Note: Call this function on entry to the Trend Page.

CSV_Trend_GetSettings

Writes an .ini file to recall (Get) the settings (tags displayed and scales) for the current page. This function allocates a separate section in the .ini file for each page.

Syntax

CSV_Trend_GetSettings(*sPage,hTrendAN*)

Example

```
[TrendPage1]
Tag_1=TrendTag1
Zero_1=0.
Full_1=1000.
Tag_2=TrendTag2
Zero_2=0.
Full_2=1000.
Tag_3=TrendTag3
Zero_3=0.
Full_3=1000.
Tag_4=TrendTag4
Zero_4=0.
Full_4=1000.
Tag_5=
Tag_6=
Tag_7=
Tag_8=
```

Note: Call this function on exiting the Trend Page.

CSV_Trend_GetSpan

Gets the time span as a time formatted string "HH:MM:SS" for a specified trend.

Syntax

CSV_Trend_GetSpan(*hTrendAN*)

#hTrendAN:

Animation point number of the trend.

Return Value

The formatted time string.

CSV_Trend_GetTime

Gets the time of the trend at a percentage along the trend, using the time of the right-most sample displayed. The time associated with the right-most sample displayed is known as the end time. The start time is the time of the left-most sample displayed. Percent 0 (zero) will correspond to the end time, and Percent 100 will correspond to the start time.

Syntax

CSV_Trend_GetTime(*hTrendAN, iPercent*)

#hTrendAN:

Animation point number of the trend.

#iPercent:

The percentage of the trend from the time of the right-most sample displayed.

Return Value

The time of the trend in the format hh:mm:ss.

CSV_Trend_GetDate

Gets the date of the trend at a percentage along the trend, using the date of the right-most sample displayed. The date associated with the right-most sample displayed is known as the end date.

The start date is the date of the left-most sample displayed. Percent 0 (zero) will correspond to the end date, and Percent 100 will correspond to the start date.

Syntax

GetDate(*hTrendAN,iPercent*)

#hTrendAN:

Animation point number of the trend.

#iPercent:

The percentage of the trend from the date of the right-most sample displayed.

Return Value

The date of the trend in the format month day year.

CSV_Trend_GroupConfig()

Displays a popup window allowing the user to browse/edit/add/delete records in the TrendGrp.dbf at runtime.

CSV_Trend_Page

Builds a trend page with the specified pens.

> **Note:** Because you cannot mix templates in a project, CSV_Trend_Page only works on trend pages based on XP-style templates. When using CSV_Trend_Page to go to a page based on a standard template, the page displays, but no trend tag is added. This also applies for the PageTrend Cicode function.

Syntax

CSV_Trend_Page(*sPage,sPen1,sPen2,sPen3,sPen4,sPen5,sPen6,sPen7,sPen8*)

#sPage:

Name of trend page to display.

#sPen1:

Trend tag to be trended by pen 1.

#sPen2:

Trend tag to be trended by pen 2.

#sPen3:

Trend tag to be trended by pen 3.

#sPen4:

Trend tag to be trended by pen 4.

#sPen5:

Trend tag to be trended by pen 5.

#sPen6:

Trend tag to be trended by pen 6.

#sPen7:

Trend tag to be trended by pen 7.

#sPen8:

Trend tag to be trended by pen 8.

Return Value

0 if successful, otherwise an error number.

CSV_Trend_Popup

Builds a Pop-up trend page in a new window with the specified pens. The window is centered on the active monitor.

Syntax

CSV_Trend_Popup(*sPage,sPen1,sPen2,sPen3,sPen4*)

#sPage:

Name of trend page to display.

#sPen1:

Trend tag to be trended by pen 1.

#sPen2:

Trend tag to be trended by pen 2.

#sPen3:

Trend tag to be trended by pen 3.

#sPen4:

Trend tag to be trended by pen 4.

Return Value

Window number of popup trend window; otherwise -1 if the window couldn't be created.

CSV_Trend_ScaleDigital

Rescales digital pens between -2 and 2.

> **Note:** To be rescaled trend tags need to have same name as digital variable tag.

Syntax

CSV_Trend_ScaleDigital(*hTrendAN,iPen*)

#hTrendAN:

Animation point number of the trend.

#iPen:

Number of pen to scale, or -1 for every pen.

CSV_Trend_SelectGroup

Allows the user to select a group of trend tags from a listbox. Each group has an associated name, description and list of up to 8 tags. This function stores the selected group data and returns the name of the group selected from the list.

> **Note:** Groups are configured in the TrendGrp.dbf file found in the [RUN] directory.

Syntax

```
CSV_Trend_SelectGroup(iMonitor,iTrendDataSet)
```

#iMonitor:

Number of monitor the trend is/was displayed on.

#iTrendDataSet:

Identifies the data set to be used for the group of trend tags. A normal trend page uses data set 0, double trend page uses data sets 1 and 2.

Return Value

Trend group (description) selected from the list, or "" if Cancel is pressed.

CSV_Trend_SelectPen

Displays a listbox to allow the user to select a tag to trend with the selected pen.

Syntax

CSV_Trend_SelectPen(*sSelectedPen*)

#sSelectedPen:

Name of trend tag to pre-select.

Return Value

Name of trend tag selected from list, or "" if action is canceled.

CSV_Trend_SetCursor

If no trend pen has the focus, this function returns, otherwise it moves the trend cursor by a specified number of samples. If the trend cursor is disabled, this function enables it. If the cursor is enabled and the number of samples is 0 (zero), the cursor is disabled. If the cursor is moved off the current trend frame, the trend scrolls.

Syntax

CSV_Trend_SetCursor(*hTrendAN*)

#hTrendAN:

Animation point number of the trend.

CSV_Trend_SetDate

Sets the 0% date of the trend via a keypad form. This allows the user to view trend information up to the date entered.

Syntax

CSV_Trend_SetDate(*hTrendAN*,*sValue*)

#hTrendAN:

Animation point number of the trend.

#sValue:

The date to set the 0% trend date to. If sValue = "", a form is displayed for the user to select a date.

Return Value

New date (as string).

CSV_Trend_SetDateTime

Sets the 0% date and time of the trend via a keypad form. This allows the user to view trend information up to the time and date entered.

Syntax

CSV_Trend_SetDateTime(*hTrendAN*)

#hTrendAN:

Animation point number of the trend.

Return Value

New time and date, separated by a space.

CSV_Trend_SetPens

Allocates trend tags to trend pens. The names of the trend tags are extracted from a string that stores the last group of trend tags displayed on a particular monitor.

Syntax

CSV_Trend_SetPens(*hTrendAN*, *iMonitor*, *iTrendDataSet*)

#hTrendAN:

Animation point number of the trend.

#iMonitor:

Number of monitor the trend is displayed on (-1 for active monitor).

#iTrendDataSet:

Identifies the data set to be used for the group of trend tags. Normal trend page uses data set 0; double trend page uses data sets 1 and 2.

CSV_Trend_SetRange

Gets the default range for trend pens and sets page strings 10-17 to the values of the ranges.

Syntax

CSV_Trend_SetRange(*hTrendAN*)

#hTrendAN:

Animation point number of trend.

CSV_Trend_SetScale

Allows the user to set the zero and full scale values of the trend. The scale may be changed for every trend or only the current trend.

Syntax

CSV_Trend_SetScale(*hTrendAN,iPercentage,sValue*)

#hTrendAN:

Animation point number of trend for which the timebase is to be set.

#iPercentage:

Scale percentage to set (0 or 100).

#sValue:

Value to set scale percentage to. If sValue = "", a form will be displayed allowing the user to select a new scale.

Return Value

New scale value as string.

CSV_Trend_SetSpan

Sets the span (total amount of time visible) on the trend.

Syntax

CSV_Trend_SetSpan(*hTrendAN,sSpan*)

#hTrendAN:

Animation point number of trend.

#sSpan:

Value to set the span to. If sSpan = "", a form will be displayed allowing the user to select the trend span.

Return Value

New span as string.

CSV_Trend_SetTime

Sets the 0% time of the trend via a keypad form. This allows the user to view trend information up to the time entered.

Syntax

CSV_Trend_SetTime(*hTrendAN,sValue*)

#hTrendAN:

Animation point number of the trend.

#sValue:

The time to set the 0% trend time to. If sValue = "", a form is displayed for the user to select a time.

Return Value

New time (as string).

CSV_Trend_SetTimebase

Allows the operator to set the time interval between each sample.

Syntax

CSV_Trend_SetTimebase(*hTrendAN,sValue*)

#hTrendAN:

Animation point number of trend for which the timebase is to be set.

#sValue:

Value to set timebase to. If sValue = "", a form will be displayed allowing the user to select a new timebase.

Return Value

New timebase as string.

CSV_Trend_UpdatePens

Stores the names of tags currently trended at a specified AN to a string as a comma separated list. A separate string is assigned to each monitor. The string is used to restore the last tags trended when the trend page is redisplayed.

Syntax

CSV_Trend_UpdatePens(*hTrendAN,iMonitor,iTrendDataSet*)

#hTrendAN:

Animation point number of the trend.

#iMonitor:

Number of monitor the trend is displayed on (-1 for active monitor).

#iTrendDataSet:

Identifies the data set to be used for the group of trend tags. Normal trend page uses data set 0, double trend page uses data sets 1 and 2.

CSV_Trend_Win

Builds a trend page in a new window with the specified pens.

Syntax

CSV_Trend_Win(*sPage,iX,iY,iMode,sPen1,sPen2,sPen3,sPen4,sPen5,sPen6,sPen7, sPen8*)

#sPage:

Name of trend page to display.

#iX:

X coordinate of top left corner of window.

#iY:

Y coordinate of top left corner of window.

#iMode:

Mode of the window (= mode used by WinNewAt).

#sPen1:

Trend tag to be trended by pen 1.

#sPen2:

Trend tag to be trended by pen 2.

#sPen3:

Trend tag to be trended by pen 3.

#sPen4:

Trend tag to be trended by pen 4.

#sPen5:

Trend tag to be trended by pen 5.

#sPen6:

Trend tag to be trended by pen 6.

#sPen7:

Trend tag to be trended by pen 7.

#sPen8:

Trend tag to be trended by pen 8.

Return Value

Window number of the window; otherwise -1 if window can't be opened.

CSV_TrendX_AddVariable

Assigns a variable to the first available instant trend tag. An instant trend tag is available if no variable is currently being trended by it; that is, `msTrendXVariable[iTrendNo] = ""`, where `iTrendNo` is the number of the instant trend.

> **Note:**This function is to be called only on a Trends Server. To maintain redundancy the function is also called with the same arguments on the second/redundant Trends Server.

The variable is assigned a trend duration. The variable name is also added to the end of a queue storing currently assigned variables in the order in which they were assigned.

If there are no available trend tags then the variable is not assigned to be trended.

Syntax

CSV_TrendX_AddVariable(*sVariable, iDuration, IupdateRedundantSrvr*)

#sVariable:

Name of variable to be trended.

#iDuration:

Value to preset trend tag timer to. This determines the number of seconds that the variable will be trended for.

#iUpdateRedundantSrvr:

- 1 = update second Trends Server with same info, i.e. RPC same function on second Trends Server. Set to 0 only in RPC call from within function itself.
- 0 = don't RPC second Trends Server.

Note: Number of instant trend tag assigned to trending sVariable if successful, otherwise -1.

CSV_TrendX_AgeTrends()

Decrements trend countdown timers.

CSV_TrendX_ClearTrend

Clears trend cache and delete trend file associated with specified trend.

This function needs to be called before a new variable can be assigned to a Instant Trend tag. This needs to be done as the trend tag may have been previously assigned to a different variable, in which case scrolling back through the trends history could display data not associated with the current variable.

Note: This function is to be called only on a Trends Server. To maintain redundancy the function is also called with the same arguments on the second/redundant Trends Server.

Syntax

CSV_TrendX_ClearTrend(*iTrendNo, IUpdateRedundantSrvr*)

#iTrendNo:

Number of Instant Trend to be cleared.

#iUpdateRedundantSrvr :

- 1 = update second Trends Server with same info; i.e., RPC same function on second Trends Server.

- 0 = don't RPC second Trends Server. Set to 0 only in RPC call from within function itself.

CSV_TrendX_Close

Frees instant trend tags associated with trend pens. Close the instant trend popup.

Syntax

CSV_TrendX_Close(*hAN*)

#hAN:

AN number of instant trend.

CSV_TrendX_DeletePen()

Deletes trend pen on instant trend page. Stop trending variable assigned to instant trend Tag.

Syntax

CSV_TrendX_DeletePen(*hAN*, *iPenNo*)

#hAN:

AN number of Instant Trend

#iPenNo:

Number of trend pen to delete.

CSV_TrendX_Display()

Displays the Instant Trend popup. Set trend duration to default value.

CSV_TrendX_DspPopupMenu

Creates a popup at the location of the mouse on an Instant Trend page, giving the user a choice of selecting a trend pen (i.e., selecting a tag to be trended by the selected pen), or clearing a trend pen.

If the user chooses 'select trend pen' then a form is displayed allowing the user to select a variable tag to be trended by the pen from a menu of available variable tags. If the user chooses 'clear trend pen', the selected trend pen is deleted. Called when the user right-clicks a trend pen marker.

Syntax

CSV_TrendX_DspPopupMenu(*hTrendAN*, *iPenNo*)

#hTrendAn:

AN number of Instant Trend.

#iPenNo:

Number of trend pen to select/clear.

CSV_TrendX_GenericToTag

Converts raw integer value (0-32000) to real value scaled between specified tag's engineering zero and engineering full scale.

Syntax

CSV_TrendX_GenericToTag(*iValue,sTagName*)

#iValue:

Raw value scaled between 0 - 32000.

#sTagname:

Name of tag whose eng zero and eng full scale values are to be used to scale iValue.

Return Value

Value scaled between tag's eng zero scale and eng full scale.

CSV_TrendX_GenericToTagStr

Converts raw integer value (0-32000) to real value scaled between specified tag's engineering zero and engineering full scale, then returns that value as a string.

> **Note:** Instant trend data is stored in generic format. i.e., as a raw integer with range 0-32000. Call this function to convert raw trend value into scaled value to be displayed on the trend popup.

Syntax

CSV_TrendX_GenericToTagStr(*iValue,sTagName*)

#iValue:

Raw value scaled between 0 - 32000.

#sTagname:

Name of tag whose eng zero and eng full scale values are to be used to scale iValue.

Return Value

Value (as string) scaled between tag's eng zero scale and eng full scale.

CSV_TrendX_GetComment

Gets comment associated with variable tag.

Syntax

CSV_TrendX_GetComment(*sVariable*)

#sVariable:

Name of tag to retrieve comment for.

Return Value

Comment associated with variable tag sVariable.

CSV_TrendX_GetCursor

Gets value of instant trend pen at cursor.

Syntax

CSV_TrendX_GetCursor(*hAN*, *iPenNo*)

#hAN:

AN number of Instant Trend.

#iPenNo:

Pen to get cursor value for.

Return Value

Value of trend pen at cursor (returned as string). Value is scaled between eng zero and eng full for variable being trended, as specified by in variable tag configuration.

CSV_TrendX_GetDuration()

Gets duration associated with instant trend popup.

Return Value

Trend duration of instant trend popup, in long time period format (hh:mm:ss).

CSV_TrendX_GetSamplePeriod

Gets period at which trend tag is being sampled.

Syntax

CSV_TrendX_GetSamplePeriod(*iTrendNo*)

#iTrendNo:

Number of trend tag to get sample period for.

Return Value

Sample period of specified Instant Trend (in seconds).

> **Note:** This is not the same as the sample period specified in the trend tag configuration form (which is set to 1 sec). The sample period for a Instant Trend can be set dynamically at run time.

CSV_TrendX_GetScale

Gets value representing a percentage of the displayed range for trend pen in focus. Used for determining/displaying 0, 50, 100% etc, scale on Instant Trend page.

Syntax

CSV_TrendX_GetScale(*hAN*, *iPercent*)

#hAN:

AN number of Instant Trend.

#iPercent:

Percentage of full scale.

Return Value

Scale value.

CSV_TrendX_GetTrendName

Gets name of instant trend from number of instant trend.

Syntax

CSV_TrendX_GetTrendName(*iTrendNo*)

#iTrendNo:

Number of instant trend tag.

Return Value

Name of trend tag.

CSV_TrendX_GetTrigger

Description This function is called in the Trigger field of the Trend Tag configuration form for Instant Trend tags.

Syntax

CSV_TrendX_GetTrigger(*iTrendNo*)

#iTrendNo:

Number of the instant trend tag.

Return Value

Return Value Trigger setting for each Instant Trend tag.

CSV_TrendX_GetVal

This function is called in the Expression field of the Trend Tag configuration form for instant trend tags. Makes the element of the array that stores the value assigned to a trend tag available to the trend system.

Syntax

CSV_TrendX_GetVal(*iTrendNo*)

#iTrendNo:

Number of the instant trend tag.

Return Value

Last stored value of the variable associated with the instant trend tag, as an integer between -1 and 32000.

CSV_TrendX_InitClient()

Initializes trend client for instant trending.

> **Note:** This function is to be called on startup for each trend client if instant trend functionality is necessary. To implement this without requiring a call to this function from within the startup Cicode function, it has been configured as a periodic event

(listed as a `CSV_TrendXClient` event). The first time the event is processed the instant trend client functionality is initialized. Any subsequent calls return immediately without effect.

CSV_TrendX_InitSrvr()

Initializes Trends Server for instant trending. Set up table used for clearing data in trend cache. Set instant trend triggers to 1. Initializes queue for storing names of variables being trended by instant trend system.

Note: This function is to be called on startup for Trends Servers if instant trend functionality is necessary. To implement this without requiring a call to this function from within the startup Cicode function, it has been configured as a periodic event (listed as a `CSV_TrendXServer` event). The first time the event is processed the instant Trends Server functionality is initialized. Subsequent calls return immediately without effect.

CSV_TrendX_MapTrendTags()

Wrapper function for `_CSV_TrendX_MapTrendTags`. Called as an event on Trends Server every 1 second, to update trend tag values (if `CSV_TrendXServer` event has been enabled).

CSV_TrendX_RefreshTrendPage

Refreshes trend page. Called after a variable has been added to instant trend system. Scrolls to current time.

Syntax

CSV_TrendX_RefreshTrendPage(*hAN*)

#hAN:

AN number of instant trend.

Note: Calling `TrendSetNow` results in old/invalid data being cleared from the screen. This is necessary when the variable being trended by a pen changes.

CSV_TrendX_SetDuration

Sets duration of Instant Trend popup.

Syntax

CSV_TrendX_SetDuration(*iDuration*, *iDspNumPad*)

#iDuration:

Duration of popup (in seconds).

#iDspNumPad:

Display number pad for data entry.

CSV_TrendX_SetDuration

Sets duration of Instant Trend on Trends Server.

> **Note:** This function is to be called only on a Trends Server. To maintain redundancy, the function is also called with the same arguments on the second/redundant Trends Server.

Syntax

CSV_TrendX_SetDuration(*iTrendNo*,*iDuration*,*iUpdateRedundantSrvr*)

#iTrendNo:

Number of trend to set duration for.

#iDuration:

Duration of popup (in seconds).

#iUpdateRedundantSrvr :

- 1 = Update second Trends Server with same info; i.e., RPC same function on second Trends Server.
- 0 = Don't RPC second Trends Server. Set to 0 only in RPC call from within function itself.

Return Value

0 if successful, otherwise -1.

CSV_TrendX_SetPen()

Displays form allowing user to select variable to assign to trend pen.

CSV_TrendX_SetSamplePeriod

Sets the sample period for a specified instant trend pen. For display purposes only, the sample period is stored as a page-based integer. This is updated when this function is called. The sample period is updated on the Trends Server.

Syntax

CSV_TrendX_SetSamplePeriod(*hAN, iPenNo, iPeriod*)

#hAN:

Number of Instant Trend AN.

#iPenNo:

Number of pen to update sample period.

#iPeriod:

Time (in seconds) to set new sample period to.

CSV_TrendX_SetScale

Sets scale for instant trend. Scale may be set for every pen or current pen only.

Syntax

CSV_TrendX_SetScale(*hAN,iPercent,iScaleVal,iDspNumPad*)

#hAN:

AN number of Instant Trend.

#iPercent:

Percent of displayed range that scale setting represents.

#iScaleVal:

New scale value.

#iDspNumPad:

Display number pad for setting scale.

CSV_TrendX_TagSelect

Assigns a variable to a pen on the Instant Trend page. The variable will be assigned to the first available Instant Trend tag. The local page based variables accessed by the trend page are updated.

Return Value

Number of instant trend tag assigned to trending sVariable if successful, otherwise -1.

Syntax

CSV_TrendX_TagSelect(*hAN,iPenNo,sVariable*)

#hAN:

AN number of Instant Trend.

#iPenNo:

Number of pen to assign to variable.

#sVariable:

Name of variable to assign to pen.

CSV_TrendX_TagSelectFrmCursor()

Assigns a variable to a pen on the Instant Trend page by positioning the mouse pointer over an animation point. The variable associated with the AN point will be selected.

CSV_TrendX_TagToGeneric

Converts real value scaled between specified tag's engineering zero and engineering full scale, to a raw integer value (0 - 32000).

Instant Trend data is stored in generic format. i.e. as a raw integer with range 0 - 32000.

Syntax

CSV_TrendX_TagToGeneric(*rValue, sTagName*)

#rValue:

Scaled value to convert to raw integer 0-32000.

#sTagname:

Name of tag whose eng zero and eng full scale values rValue is scaled between.

Return Value

Value scaled between 0-32000.

CSV_TrendX_TrendTimeout

Monitors time remaining for trends associated with instant trend popup.

Syntax

CSV_TrendX_TrendTimeout(*hAN*)

#hAN:

Number of Instant Trend AN.

Return Value

1 if trend has timed out, 0 otherwise.

CSV_WinUtl_DestroyCursor()

Deletes the specified cursor and sets the cursor to the normal cursor.

CSV_WinUtl_GetColourRes()

Gets the screen color resolution.

Return Value

Screen color resolution: 0 = 256 colors, 1 = High color (16 bit), 2 = True color (24 bit/32 bit), -1 = Error.

CSV_WinUtl_GetCpuUsage

Gets the percent CPU usage of a specified process, or the total CPU usage.

Note: This function has been deprecated on Windows Vista, and will return 0 when called on this operating system.

Syntax

CSV_WinUtl_GetCpuUsage(*sProcessName*)

#sProcessName:

Name of process, or "" to get total CPU usage.

Return Value

Percentage CPU usage.

CSV_WinUtl_GetSystemDir()

Gets the windows system directory.

Return Value

Windows system directory path.

CSV_WinUtl_GetTotalCpuUsage()

Gets the total percent CPU usage.

> **Note:** Call `CSV_WinUtl_UpdateTotalCpuUsage` to refresh the data (`CSV_WinUtl_UpdateTotalCpuUsage` prevents a 'Foreground Cicode run too long' error).

Return Value

Total CPU Usage.

CSV_WinUtl_GetWindowsDir()

Gets the windows directory.

Return Value

Windows directory path.

CSV_WinUtl_GetWinMode()

Returns 1 if CitectSCADA is in FullScreen mode.

Return Value

1 if fullscreen mode([Animator]FullScreen = 1), otherwise 0.

CSV_WinUtl_LoadCursor

Loads the cursor for a specified window from a file (.ani or .cur).

Syntax

CSV_WinUtl_LoadCursor(*sCursor,hWnd*)

#sCursor:

File (including path) containing cursor.

#hWnd:

Handle of window to change cursor for.

Return Value

Handle to new cursor.

CSV_WinUtl_LockWindowUpdate

Freezes the specified window (prevents CitectSCADA repainting it).

Syntax

CSV_WinUtl_LockWindowUpdate(*hWnd*)

#hWnd:

Handle of window to freeze, or -1 to unfreeze any frozen window.

Return Value

0 if successful, otherwise -1.

CSV_WinUtl_NormalCursor

Loads the normal cursor for a specified window.

Syntax

CSV_WinUtl_NormalCursor(*hWnd*)

#hWnd:

Handle of window to change cursor for.

Return Value

Handle to normal cursor.

CSV_WinUtl_ShellExec

Opens or prints a specified file.

Syntax

CSV_WinUtl_ShellExec(*sFile,sArgs,sDir,sOperation,iShowCmd*)

#sFile:

Specifies the file to open or print or the folder to open or explore. The function can open an executable file or a document file. The function can print a document file.

#sArgs:

If sFile specifies an executable file, sArgs specifies the parameters to be passed to the application. If sFile specifies a document file, make sArgs as "".

#sDir:

Specifies the default directory.

#sOperation:

Specifies the operation to perform. The following operation strings are valid:

- open - Opens the file specified by the lpFile parameter. The file can be an executable file or a document file. It can also be a folder.
- print - The function prints the file specified by lpFile. The file has to be a document file. If the file is an executable file, the function opens the file, as if "open" had been specified.
- explore - The function explores the folder specified by lpFile. This parameter can be "". In that case, the function opens the file specified by lpFile.

#iShowCmd:

If sFile specifies an executable file, iShowCmd specifies how the application is to be shown when it is opened. This parameter can be one of the following values:

- SW_HIDE (=0) - Hides the window and activates another window.
- SW_MAXIMIZE (=3) - Maximizes the specified window.
- SW_MINIMIZE (=6) - Minimizes the specified window and activates the next top-level window in the z-order.
- SW_RESTORE (=9) - Activates and displays the window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when restoring a minimized window.
- SW_SHOW (=5) - Activates the window and displays it in its current size and position.
- SW_SHOWDEFAULT (=10) - Sets the show state based on the SW_ flag specified in the STARTUPINFO structure passed to theCreateProcess function by the program that started the application. An application should call ShowWindow with this flag to set the initial show state of its main window.
- SW_SHOWMAXIMIZED (=3) - Activates the window and displays it as a maximized window.
- SW_SHOWMINIMIZED (=2) - Activates the window and displays it as a minimized window.
- SW_SHOWMINNOACTIVE (=7) - Displays the window as a minimized window. The active window remains active.
- SW_SHOWNA (=8) - Displays the window in its current state. The active window remains active.
- SW_SHOWNOACTIVATE (=4) - Displays a window in its last size and position. The active window remains active.
- SW_SHOWNORMAL (=1) - Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when displaying the window for the first time. If sFile specifies a document file, nShowCmd should be zero.

Return Value

Returns a value greater than 32 if successful, or an error value that is less than or equal to 32 otherwise. The following table lists the error values.

- ERROR_FILE_NOT_FOUND (=2) - The specified file was not found.
- ERROR_PATH_NOT_FOUND (=3) - The specified path was not found.
- ERROR_BAD_FORMAT (=17) - The .exe file is invalid (non-Win32® .exe or error in .exe image).
- SE_ERR_ACCESSDENIED (=5) - The operating system denied access to the specified file.
- SE_ERR_ASSOCINCOMPLETE (=27) - The file name association is incomplete or invalid.
- SE_ERR_DDEBUSY (=30) - The DDE transaction could not be completed because other DDE transactions were being processed.
- SE_ERR_DDEFAIL (=29) - The DDE transaction did not succeed.
- SE_ERR_DDETIMEOUT (=28) - The DDE transaction could not be completed because the request timed out.
- SE_ERR_DLLNOTFOUND (=32) - The specified dynamic-link library was not found.
- SE_ERR_FNF (=2) - The specified file was not found.
- SE_ERR_NOASSOC (=31) - There is no application associated with the given file name extension.
- SE_ERR_OOM (=8) - There was not enough memory to complete the operation.
- SE_ERR_PNF (=3) - The specified path was not found.
- SE_ERR_SHARE (=26) - A sharing violation occurred.

CSV_WinUtl_UpdateTotalCpuUsage()

Updates the total percent CPU usage at minimum of 0.5 second intervals. Called from the Admin Tools page.

CSV_WinUtl_WaitCursor

Loads the wait/busy cursor for a specified window.

Syntax

CSV_WinUtl_WaitCursor(*hWnd*)

#hWnd:

Handle of window to change cursor for.

Return Value

Handle to wait cursor.

Chapter: 6 Graphics Builder Automation Interface

The CitectSCADA Graphics Builder now offers support for "automation," an OLE service that allows applications to expose their functionality, or to control the functionality of other applications on the same computer or across a network. As a result, applications can be integrated and automated with programming code.

The two key elements of automation are:

- Applications or software components, called **automation Servers**, that can be controlled because their functionality has been exposed and made accessible to other applications. Examples of Microsoft Automation servers are Microsoft Office applications and Microsoft Project. These Automation servers expose their functionality through object models.
- Other applications or development tools, called **automation controllers**, that can control OLE Automation servers through programming code, by accessing the functionality exposed by the Automation servers. Examples of Microsoft Automation controllers are Microsoft Visual Basic, Microsoft Visual C++, and Microsoft Visual Basic for Applications (which is built into Microsoft Access, Microsoft Excel, and Microsoft Project).

Automation is the umbrella term for the process by which an automation controller sends instructions to an automation server (using the functionality exposed by the automation server), where they are run.

The CitectSCADA Graphics Builder automation interface enables the CitectSCADA Graphics Builder to act as an automation server, as it exposes many Graphics Builder functions as well as some Project Editor and Citect Explorer functions.

The interface supports a simple object model: functions are on the root level. Names are structured and contain a group identifier and a function name; for example, DrawLine, DrawRectangle, PositionAt, PositionRotate, ProjectSelect, ProjectUpgrade. These functions can be called from a Visual Basic (VB) program.

Note: In the VB development environment, the reference GraphicsBuilder Type Library needs to have previously been selected. If it hasn't, choose **References** from the Project menu in the VB and check the Graphics Builder Type Library.

Example

The following sample VB code allows you to create a new CitectSCADA page, place a Genie at a specific location, set one of its parameter, draw a line, and then save the page with the name "TEST".

```
Dim GraphicsBuilder As IGraphicsBuilder2
Set GraphicsBuilder = New GraphicsBuilder.GraphicsBuilder
With GraphicsBuilder
        .Visible = True
        .PageNew "include", "standard", "normal", 0, True, True
        .LibraryObjectPlace "include", "motors", "motor_1_east", 0, True
        .PositionAt 300, 500
        .LibraryObjectPutProperty "Tag", "Test_Tag"
        .DrawLine 100, 100, 300, 300
        .AttributeLineColour = 120
        .PageSaveAs "Example", "TEST"
        .PageClose
        .Visible = False
End With
Set GraphicsBuilder = Nothing
```

See Also

Error Handling
Automation Events

Error Handling

Functions, when called from VB, throw an exception on error. The following table lists the possible HRESULT errors that may be encountered:

C++ define	Hex value	Hex codes in VB	Description
S_OK	0	No exception	Successful execution
E_INVAL-IDARG	80070057	5	One or more arguments are out of range
E_HANDLE	80070006	80070006	No active object (page or graphical object)
E_POINTER	80004003	80004003	Missing or broken link encountered

E_ABORT	80000007	11F	Enumeration terminated or function manually canceled (for example ProjectUpdate)
E_FAIL	80004005	80004005	Every other error

The following VB code can be used to process the error code:

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibObjectName Project, File, Page, Type
If Err.Number <> 0 Then
        Debug.Print "Error occurred in LibObjectName"
End If
```

Note the following points:

- VB sets the Err variable only in the erroneous case. It will not be set to 0 if the function succeeds.
- When VB handles an exception, it ignores the functions parameters. Hence when a function like ProjectNext does not succeed, the returned string is undefined and not an empty string.

The functions in the groups Page Functions, Options Functions, Object Drawing and Property Functions, Text Property Functions and the individual functions LibSelectionHooksEnabled, SelectionEventEnabled, BrokenLinkCancelEnabled and Visible are treated as variables in VB.

When calling these functions from C++, you need to use a "put_" or "get_" prefix, for example, "put_Visible(TRUE)", "get_Visible(bValue)" to set or fetch the values, except if the Attribute is read-only. In this case the function is the same in C++; for example, PageName.

To evaluate the correct function name for C++ reference the Type library CTDRAW32.TLB, which can be found in CitectSCADA's BIN directory. You can use Microsoft's Visual Studio Tool OLE / COM Object Viewer (select menu File | View Typelib...) to look at a type library.

See Also

Automation Events

Automation Events

The graphics builder also provides event based notification of actions, which an Automation client can intercept and react to accordingly. The following example creates a form, creates a graphics builder automation object with event capability and performs actions on two events that the graphics builder might generate, pasting a symbol and saving a page.

To enable this:

- The Graphics Builder object needs to be declared "WithEvents"
- The event handler subroutine needs to have the correct name and signature. Note how the event handler function names are gb, the graphics builder object, followed by _<eventName> e.g gb_PasteSymbol. This is consistent with standard Visual Basic event handling subroutine naming.

For details, see the individual event subroutine description.

```
Private WithEvents gb As GraphicsBuilder.GraphicsBuilder
Public Sub Form_Load()
       Set gb = New GraphicsBuilder.GraphicsBuilder
       gb.LibrarySelectionHooksEnabled = True
       gb.Visible = True
End Sub

Public Sub gb_PasteSymbol()
       MsgBox ("PasteSymbol")
End Sub

Private Sub gb_PageSaved(ByVal Project As String, ByVal Page As String,
        ByVal LastPage As Boolean)
       MsgBox "PageSaved: " + Project + "." + Page + "--"
End Sub
```

See Also

Error Handling

Function Categories

This table lists the CitectSCADA functions exposed through the Graphics Builder automation interface, grouped into the following categories:

Arrange and Position Functions	Allow you to modify the position of a selected object in three dimensions (X,Y and Z order).

Events Functions	Allow you to use the automation dispatch mechanism to fire events in specific situations.
Specific Functions	Currently include only the Visible function.
Dynamic Properties Functions	Allow you to modify the dynamic properties of the graphics objects in your project (movement, scaling, rotation, sliders, dynamic color fill).
Library Object Functions	Allow you to use and manipulate the objects stored in libraries in your project. This includes such objects as Genies, Super Genies, Symbols, and so on.
Miscellaneous Functions	Used for special interactions with the Graphics Builder, for example an external drag-and-drop action could be performed by requesting the active window handle.
Object Drawing and Property Functions	Allow you to draw objects and manipulate the properties of objects.
Options Functions	Relate to the options found under the Graphics Builder 's Tools menu.
Page Functions	Allow you to manipulate the pages in your project (for example open, close, save, delete), and select objects on those pages. This includes templates, symbols, Genies, Super Genies.
Page Properties Functions	Allow you to manipulate the properties of the pages in your project.
Project Functions	These functions operate on the project level. Some are actually initiated within Citect Project Editor or the Project Explorer.
Text Property Functions	Allow you to read and modify the properties of the text objects in your project.

For details and a VB example on handling return and error values, see Error Handling.

Arrange and Position Functions

The following functions modify the position of a selected object in three dimensions (X, Y and Z order).

PositionAt	Positions the active object at the specified location.
PositionBringForwards	Moves the last object addressed one step forward in the layering of objects on a page, creating the appearance of moving forward.
PositionBringToFront	Positions the last object addressed as the closest layer on a graphics page, giving it the appearance of being in front of the other objects.
PositionMirrorHorizontal	Turns the last object addressed into a mirror image of itself across a horizontal axis.
PositionMirrorVertical	Turns the last object addressed into a mirror image of itself across a vertical axis.
PositionRotate	Rotates the last object addressed by 90 degrees clockwise.
PositionSendBackwards	Moves the last object addressed one step backwards in the layering of objects on a page, creating the appearance of moving backwards.
PositionSendToBack	Positions the last object addressed as the lowest layer on a graphics page, giving it the appearance of being behind the other objects.

For details and a VB example on handling return and error values, see Error Handling.

PositionAt

Positions the active object at the specified location. The destination coordinates is adjusted if OptionSnapToGrid or OptionSnapToGuidelines are set to TRUE.

Syntax

PositionAt(*XPosition*, *YPosition*)

XPosition:

Absolute X position in pixels from the left side of the page.

YPosition:

Absolute Y position in pixels from the top of the page.

Return Value

0 (zero) if successful; otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PositionRotate, PositionMirrorVertical, PositionMirrorHorizontal, PositionSendToBack, PositionBringToFront, PositionBringForwards, PositionSendBackwards

Example

```
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt "200,200"
```

PositionBringForwards

Moves the last object addressed one step forward in the layering of objects on a page, creating the appearance of moving forward.

Syntax

PositionBringForwards

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PositionAt, PositionRotate, PositionMirrorVertical, PositionMirrorHorizontal, PositionSendToBack, PositionBringToFront, PositionSendBackwards

Example

```
' Moves an object forward in the layering of objects on a graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionBringForwards
```

PositionBringToFront

Positions the last object addressed as the closest layer on a graphics page, giving it the appearance of being in front of other objects.

Syntax

PositionBringToFront

Return Value

0 (zero) if successful; otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PositionAt, PositionRotate, PositionMirrorVertical, PositionMirrorHorizontal, PositionSendToBack, PositionBringForwards, PositionSendBackwards

Example

```
' Places an object in front of other objects on a graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionBringToFront
```

PositionMirrorHorizontal

Turns the last object addressed into a mirror image of itself across a horizontal axis.

Syntax

PositionMirrorHorizontal

Return Value

0 (zero) if successful; otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PositionAt, PositionRotate, PositionMirrorVertical, PositionSendToBack, PositionBringToFront, PositionBringForwards, PositionSendBackwards

Example

```
' Mirrors an object across a horizontal access
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionMirrorHorizontal
```

PositionMirrorVertical

Turns the last object addressed into a mirror image of itself across a vertical axis.

Syntax

PositionMirrorVertical

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PositionAt, PositionRotate, PositionMirrorHorizontal, PositionSendToBack, PositionBringToFront, PositionBringForwards, PositionSendBackwards

Example

```
' Mirrors an object across a vertical access
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionMirrorVertical
```

PositionRotate

Rotates the last object addressed by 90 degrees clockwise.

Syntax

PositionRotate

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PositionAt, PositionMirrorVertical, PositionMirrorHorizontal, PositionSendToBack, PositionBringToFront, PositionBringForwards, PositionSendBackwards

Example

```
' Rotates an object 90 degrees
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionRotate
```

PositionSendBackwards

Moves the last object addressed one step backwards in the layering of objects on a page, creating the appearance of moving backwards.

Syntax

PositionSendBackwards

Return Value

0 (zero) if successful; otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PositionAt, PositionMirrorVertical, PositionMirrorHorizontal, PositionSendToBack, PositionBringToFront, PositionBringForwards, PositionRotate

Example

```
' Moves an object backwards in the layering of objects on a graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionSendBackwards
```

PositionSendToBack

Positions the last object addressed as the lowest layer on a graphics page, giving it the appearance of being behind other objects.

Syntax

PositionSendToBack

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PositionAt, PositionMirrorVertical, PositionMirrorHorizontal, PositionSendBackwards, PositionBringToFront, PositionBringForwards, PositionRotate

Example

```
' Places an object behind other objects on a graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
GraphicsBuilder.PositionSendToBack
```

Events Functions

The following events use the automation Idispatch mechanism to fire events in specific situations.

BrokenLink	This event is fired if a missing link is encountered while executing the functions `ProjectUpdatePages()` or `PageOpen()`. Details of the missing object are provided through the parameters Project, Library, Object, GenieOrSymbol.
PasteGenie	When the `LibrarySelectionHooksEnabled()` attribute is set to TRUE, this event is fired when: the paste Genie menu item is selected; the paste genie toolbar button is pressed; or F11 is pressed.
PasteSymbol	When the `LibrarySelectionHooksEnabled()` attribute is set to TRUE, this event is fired when the paste symbol menu item is selected, the paste symbol toolbar button is pressed, or F6 is pressed.
ProjectChange	This event is fired whenever a new project is selected in Citect Explorer.
Selection	When `SelectionEventEnabled()` is set to TRUE, this event is fired every time a selection is made within a graphics page. The dimension of the selection rectangle is passed as parameters.
SwapObject	When the `LibrarySelectionHooksEnabled()` attribute is set to TRUE, this event is fired when pressing the CTRL+SHIFT keys and double-clicking on the object in the graphics page.

Note: For details on handling return and error values, see Error Handling.

BrokenLink

This event is fired if a missing link is encountered while executing the functions ProjectUpdatePages or PageOpen. Details of the missing object are provided through the parameters *Project, Library, Object, GenieOrSymbol*.

Syntax

BrokenLink(*Project, Library, Object, GenieOrSymbol*)

Project:

The name of the project.

Library:

The name of the library.

Object:

The name of the symbol or Genie.

GenieOrSymbol:

Identifies if the object is a symbol or Genie: 1 = Genie; 2 = symbol.

See Also

Automation Events

PasteGenie

When the LibrarySelectionHooksEnabled attribute is set to TRUE, this event is fired when the paste Genie menu item is selected, the paste genie toolbar button is pressed, or when F11 is pressed.

Syntax

PasteGenie

See Also

Automation Events

PasteSymbol

When the LibrarySelectionHooksEnabled attribute is set to TRUE, this event is fired when the paste symbol menu item is selected, the paste symbol toolbar button is pressed, or F6 is pressed.

Syntax

PasteSymbol

See Also

Automation Events

ProjectChange

This event is fired whenever a new project is selected in Citect Explorer.

Syntax

ProjectChange

See Also

Automation Events

Selection

When SelectionEventEnabled is set to TRUE, this event is fired every time a selection is made within a graphics page. The dimension of the selection rectangle is passed as parameters.

Syntax

Selection (*FromXPosition*, *FromYPosition*, *ToXPosition*, *ToYPosition*)

FromXPosition:

Distance from the left-hand side of the page to top-left hand corner of the selection rectangle (in pixels).

FromYPosition:

Distance from the top of the page to the top-left hand corner of the selection rectangle (in pixels).

ToXPosition:

Distance from the left-hand side of the page to the bottom-right hand corner of the selection rectangle (in pixels).

ToYPosition:

Distance from the top of the page to the bottom-right hand corner of the selection rectangle (in pixels).

See Also

Automation Events

SwapObject

When the LibSelectionHooksEnabled attribute is set to TRUE, this event is fired when pressing the CTRL+SHIFT keys and double-clicking the object in the graphics page.

Syntax

SwapObject

See Also

Automation Events

Specific Functions

The specific functions category currently includes only the Visible function.

Visible	Controls visibility of the CitectSCADA Graphics Builder, or retrieves its current visible state.

> **Note:** For details on handling return and error values, see Error Handling.

Visible

Controls visibility of the CitectSCADA Graphics Builder, or retrieves its current visible state.

Syntax

Visible

Return Value

If determining the current visible state of the Graphics Builder, TRUE or FALSE is returned. If applying a setting to this function, 0 (zero) is returned if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
' Make CitectSCADA Graphics Builder appear
GraphicsBuilder.Visible = TRUE

' Retrieve the current visible state of the Graphics Builder
MyVariable = GraphicsBuilder.Visible
```

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_Visible` sets the visible state of the Graphics Builder, and `get_Visible` retrieves the current state of the Graphics Builder (TRUE = visible).

Dynamic Properties Functions

With these functions, you can modify the dynamic properties of the graphics objects in your project (movement, scaling, rotation, sliders, dynamic color fill).

The error E_HANDLE is returned if there is no selected or active object, or if an object does not support this type of property. E_INVALIDARG is returned if an argument is out of range.

PropertiesAccessDisableGet	Reads the current values set on the Access | Disable tab of the Object Properties dialog.
PropertiesAccessDisablePut	Sets the values on the Access | Disable tab of the Object Properties dialog.
PropertiesAccessGeneralGet	Reads the values on the Access | General tab of the Object Properties dialog.
PropertiesAccessGeneralPut	Sets the values on the Access | General tab of the Object Properties dialog.
PropertiesButtonGet	Reads the values for a button object from the Appearance | General tab of the Object Properties dialog.
PropertiesButtonPut	Sets the values on the Appearance | General tab of the Object Properties dialog for a button object.
PropertiesCicodeObjectGet	Reads the values set for a Cicode object on the Cicode | General tab of the Object Properties dialog.
PropertiesCicodeObjectPut	Sets the values for a Cicode object on the Cicode | General tab of the Object Properties dialog.
PropertiesDisplayValueGet	Reads the type and expressions configured on the Appearance | Display Value tab of the Object Properties dialog.
PropertiesDisplayValuePut	Sets the values and expressions on the Appearance | Display Value tab of the Object Properties dialog.
PropertiesDisplayValueTextGet	Reads the text for a specific index from the Appearance | Display Value tab of the Object Properties dialog.
PropertiesDisplayValueTextPut	Sets the text for a specific index on the Appearance | Display Value tab of the Object Properties dialog.
PropertiesFillColourColourGet	Reads the current color value set for the specified index point on the Fill | Color tab of the Object Properties dialog. This function has been superseded by the function PropertiesFillColourColourGetEx.

PropertiesFillColourColourGetEx	Reads the current color value set for the specified index point on the Fill | Color tab of the Object Properties dialog.
PropertiesFillColourColourPut	Sets the color at the specific index on the Fill | Color tab of the Object Properties dialog. This function has been superseded by the function PropertiesFillColourColourPutEx.
PropertiesFillColourColourPutEx	Sets the color at the specific index on the Fill | Color tab of the Object Properties dialog.
PropertiesFillColourGet	Reads the values set on the Fill | Color tab of the Object Properties dialog for the current object.
PropertiesFillColourPut	Sets the values on the Fill | Color tab of the Object Properties dialog.
PropertiesFillLevelGet	Reads the values set on the Fill | Level tab of the Object Properties dialog. This function has been superseded by the function PropertiesFillLevelGetEx.
PropertiesFillLevelGetEx	Reads the values set on the Fill | Level tab of the Object Properties dialog.
PropertiesFillLevelPut	Sets the values on the Fill | Level tab of the Object Properties dialog. This function has been superseded by the function PropertiesFillLevelPutEx.
PropertiesFillLevelPutEx	Sets the values on the Fill | Level tab of the Object Properties dialog.
PropertiesInputKeyboardGet	Reads the values set on the Input | Keyboard Command tab of the Object Properties dialog
PropertiesInputKeyboardPut	Sets the values on the Input | Keyboard Commands tab of the Object Properties dialog
PropertiesInputTouchGet	Reads the values set on the Input | Touch tab of the Object Properties dialog
PropertiesInputTouchPut	Sets the values on the Input | Touch tab of the Object Properties dialog.
PropertiesShowDialog	Shows the property dialog for an object or a form for Genies.

PropertiesSymbolSetGet	Reads the type and expressions configured on the Appearance | General tab of the Object Properties dialog.
PropertiesSymbolSetPut	Sets the type defined for a symbol set on the Appearance | General tab of the Object Properties dialog.
PropertiesSymbolSetSymbolGet	Retrieves the Element name and Library name of the "Index" element of the currently selected object.
PropertiesSymbolSetSymbolPut	Sets the Element name and Library name of the "Index" element of the currently selected object.
PropertiesTransCentreOffsetExpressGet	Retrieve the express properties.
PropertiesTransCentreOffsetExpressPut	Set the express properties.
PropertiesTransformationGet	Reads the property values set on the Movement, Scaling and Slider tabs of the Object Properties dialog.
PropertiesTransformationPut	Sets values for the properties on the Movement, Scaling and Slider tabs of the Object Properties dialog.
PropertiesTrendGet	Reads the values for a trend object as set on the Appearance | General tab of the Object Properties dialog. This function has been superseded by the function PropertiesTrendGetEx.
PropertiesTrendGetEx	Reads the values for a trend object as set on the Appearance | General tab of the Object Properties dialog.
PropertiesTrendPut	Sets the values for a trend object that appear on the Appearance | General tab of the Object Properties dialog. This function has been superseded by the function PropertiesTrendPutEx.
PropertiesTrendPutEx	Sets the values for a trend object that appear on the Appearance | General tab of the Object Properties dialog
PropertyVisibility	Sets the Hidden when argument on the Appearance | Visibility tab of the Object Properties dialog.

Note: For details on handling return and error values, see Error Handling.

PropertiesAccessDisableGet

Reads the current values set on the Access | Disable tab of the Object Properties dialog for the current object.

Syntax

PropertiesAccessDisableGet(*Expression*, *DisableFlag*, *DisableStyle*)

Expression:

The string for the Disable when command.

DisableFlag:

TRUE if the object is configured to disable when an insufficient area or privilege setting is encountered.

DisableStyle:

The disable style setting:

- 0 = Embossed
- 1 = Grayed
- 2 = Hidden

Return Value

The requested values, as a string.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PropertiesAccessDisablePut

PropertiesAccessDisablePut

Sets the values on the Access | Disable tab of the Object Properties dialog for the current object.

Syntax

PropertiesAccessDisablePut(*Expression*, *DisableFlag*, *DisableStyle*)

Expression:

The string for the Disable when command.

DisableFlag:

TRUE if the object is configured to disable when an insufficient area or privilege setting is encountered.

DisableStyle:

The disable style setting:

- 0 = Embossed
- 1 = Grayed
- 2 = Hidden

Return Value

0 (zero) if successful, otherwise an error is returned

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesAccessDisableGet

PropertiesAccessGeneralGet

Reads the values on the **Access | General** tab of the Object Properties dialog for the current object.

Syntax

PropertiesAccessGeneralGet(*Description*, *Tooltip*, *Area*, *Privilege*, *LogDevice*)

Description:

Description string for the object.

Tooltip:

Tooltip string for the object.

Area:

1 to 255 representing the current area setting, or 0 if the Same area as page check box is ticked.

Privilege:

1 to 255 representing the current privilege setting, or 0 if the No privilege restrictions checkbox is ticked.

LogDevice:

The name of the log device as a string.

Return Value

The requested values, as a string

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesAccessGeneralPut

PropertiesAccessGeneralPut

Sets the values on the Access | General tab of the Object Properties dialog for the current object.

Syntax

PropertiesAccessGeneralPut(*Description*, *Tooltip*, *Area*, *Privilege*, *LogDevice*)

Description:

Description string for the object.

Tooltip:

Tooltip string for the object.

Area:

1 to 255 representing the current area setting, or 0 if the Same area as page check box is ticked.

Privilege:

1 to 255 representing the current privilege setting, or 0 if the No privilege restrictions checkbox is ticked.

LogDevice:

The name of the log device as a string.

Return Value

0 (zero) if successful, otherwise an error is returned

> **Note:** For details on handling return and error values, see Automation Error Handling.

Related Functions

PropertiesAccessGeneralGet

PropertiesButtonGet

Reads the values for a button object from the **Appearance | General** tab of the Object Properties dialog.

Syntax

PropertiesButtonGet(*ButtonType*, *Text*, *TextFont*, *Library*, *SymbolName*)

ButtonType:

Defines the button type:

- 0 = Text
- 1 = Border 3D Target
- 2 = Border Target
- 3 = Target
- 4 = Symbol
- 5 = XP Style button with text
- 6 = XP Style Button with Symbol

Text:

Button text. This argument is only valid for ButtonType = 0 and 5 (text).

TextFont:

The font use for the button text. This argument is only valid for ButtonType = 0 and 5 (text).

Library:

Library where the button symbol can be found. This argument is only valid for ButtonType = 4 and 6 (symbol).

SymbolName:

Name of the symbol to be displayed for a button. This argument is only valid for ButtonType = 4 and 6 (symbol).

Return Value

The requested values, as a string.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PropertiesButtonPut

PropertiesButtonPut

Sets the values on the **Appearance | General** tab of the Object Properties dialog for a button object.

Syntax

PropertiesButtonPut(*Type*, *Text*, *TextFont*, *Library*, *SymbolName*)

ButtonType:

Defines the button type:

- 0 = Text
- 1 = Border 3D Target
- 2 = Border Target
- 3 = Target
- 4 = Symbol
- 5 = XP Style button with text
- 6 = XP Style Button with Symbol

Text:

Button text. This argument is only valid for ButtonType = 0 and 5 (text).

TextFont:

The font use for the button text. This argument is only valid for ButtonType = 0 and 5 (text).

Library:

Library where the button symbol can be found. This argument is only valid for ButtonType = 4 and 6 (symbol).

SymbolName:

Name of the symbol to be displayed for a button. This argument is only valid for ButtonType = 4 and 6 (symbol).

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PropertiesButtonGet

PropertiesCicodeObjectGet

Reads the values set for a Cicode object on the **Cicode | General** tab of the Object Properties dialog.

Syntax

PropertiesCicodeObjectGet(*Expression*, *Library*, *SymbolName*)

Expression:

The command expression.

Library:

Name of the library where the symbol used can be found.

SymbolName:

Name of the symbol used.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesCicodeObjectPut

PropertiesCicodeObjectPut

Sets the values for a Cicode object on the Cicode | General tab of the Object Properties dialog.

Syntax

PropertiesCicodeObjectPut(*Expression*, *Library*, *SymbolName*)

Expression:

The command expression.

Library:

Name of the library where the symbol used can be found.

SymbolName:

Name of the symbol used.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesCicodeObjectGet

PropertiesDisplayValueGet

Reads the type and expressions configured on the **Appearance | Display Value** tab of the Object Properties dialog for a number or text object.

Syntax

PropertiesDisplayValueGet(*SymbolSetType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*)

SymbolSetType:

Defines the symbol set type:

- 0 = On / Off
- 1 = Multi-state
- 2 = Array
- 3 = Numeric
- 4 = String

ExpressionA:

This is the main expression:

- ON text when for type On / Off.
- Conditions A for type Multi-state.
- Array expression for type Array.
- Numeric Expression for type Numeric.
- String Expression for type String.

ExpressionB:

Conditions B, only used for multistate type.

ExpressionC:

Conditions C, only used for multistate type.

ExpressionD:

Conditions D, only used for multistate type.

ExpressionE:

Conditions E, only used for multistate type.

Return Value

The requested values, as a string

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesDisplayValuePut

Example

```
` Gets the properties on the Appearance/DisplayValue sheet for a
number or text object

GraphicsBuilder.PropertiesDisplayValueGet nType, Expression1,
Expression2, Expression3, Expression4, Expression5
```

PropertiesDisplayValuePut

Sets the fields that appear on the **Appearance | Display Value** tab of the Object Properties dialog for a number or text object. This includes the type setting and related expressions.

Syntax

PropertiesDisplayValueGet(*SymbolSetType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*)

SymbolSetType:

Defines the symbol set type:

- 0 = On / Off
- 1 = Multi-state
- 2 = Array

- 3 = Numeric
- 4 = String

ExpressionA:

This is the main expression:

- ON text when for type On / Off.
- Conditions A for type Multi-state.
- Array expression for type Array.
- Numeric Expression for type Numeric.
- String Expression for type String.

ExpressionB:

Conditions B, only used for multistate type.

ExpressionC:

Conditions C, only used for multistate type.

ExpressionD:

Conditions D, only used for multistate type.

ExpressionE:

Conditions E, only used for multistate type.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesDisplayValueGet

PropertiesDisplayValueTextGet

Reads the text for a specific index from the Appearance | Display Value tab of the Object Properties dialog for a number or text object of type Multistate, Array or Numeric.

Syntax

PropertiesDisplayValueTextGet(*Index*, *Text*)

Index:

The position of the text:

- 0..31 for type Multistate.
- 0..255 for type Array.
- 0 for type Numeric.

Text:

The text written to the field:

- State text for type Multi-state.
- Array text for type Array.
- Format for type Numeric.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesDisplayValueTextPut, PropertiesDisplayValuePut, PropertiesCicodeObjectPut

PropertiesDisplayValueTextPut

Sets the text for a specific index on the **Appearance | Display Value** tab of the Object Properties dialog for a number or text object of type Multistate, Array, or Numeric.

Syntax

PropertiesDisplayValueTextGet(*Index*, *Text*)

Index:

The position of the text:

- 0..31 for type Multistate.
- 0..255 for type Array.
- 0 for type Numeric.

Text:

The text written to the field:

- State text for type Multi-state.
- Array text for type Array.
- Format for type Numeric.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesDisplayValueTextGet, PropertiesSymbolSetPut, PropertiesSymbolSetGet

PropertiesFillColourColourGet

Reads the current color value set for the specified index point on the **Fill | Color** tab of the Object Properties dialog for Array, Threshold and Gradient types.

> **Note:** As this function does not supportTrue Color functionality, it has been superseded by the function PropertiesFillColourColourGetEx.

Syntax

PropertiesFillColourColourGet(*Index*, *ColourNo*, *Limit*, *Operator)*

Index:

Specify the index you would like to read the current color for. This values depends on the type of color fill selected:

- 0 - 31 for type Multi-state
- 0 - 255 for type Array
- 0 - 255 for type Threshold
- 0- 1 for Gradient

ColourNo:

A value between 0 and 255 representing the color applied to the Index setting.

Limit:

A value between 0 and 100 representing the threshold limit. Used for type Threshold only.

Operator:

The value representing the current operator used for the threshold limit setting:

- 0 : < (less than)
- 1 : > (greater than)
- 2 : <= (less than or equal to)
- 3 : >= (greater than or equal to)

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillColourColourPut, PropertiesFillColourGet, PropertiesFillColourPut

PropertiesFillColourColourGetEx

Reads the current color value set for the specified index point on the **Fill | Color** tab of the Object Properties dialog for Array, Threshold and Gradient types.

Syntax

PropertiesFillColourColourGet(*Index*, *OnColourNo*, *OffColourNo*, *Limit*, *Operator)*

Index:

Specify the index you would like to read the current color for. This values depends on the type of color fill selected:

- 0 - 31 for type Multi-state
- 0 - 255 for type Array
- 0 - 255 for type Threshold
- 0- 1 for Gradient

OnColourNo:

An RGB value representing the "on" color applied to the Index setting.

OffColourNo:

An RGB value representing the "off" color applied to the Index setting.

Limit:

A value between 0 and 100 representing the threshold limit. Used for type Threshold only.

Operator:

The value representing the current operator used for the threshold limit setting:

- 0 : < (less than)
- 1 : > (greater than)
- 2 : <= (less than or equal to)
- 3 : >= (greater than or equal to)

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillColourColourPutEx, PropertiesFillColourGet, PropertiesFillColourPut

PropertiesFillColourColourPut

Sets the color at the specific index on the **Fill | Color** tab of the Object Properties dialog for type Array, Threshold and Gradient.

> **Note:** As this function does not support True Color functionality, it has been superseded by the function PropertiesFillColourColourPutEx.

Syntax

PropertiesFillColourColourPut(*Index*, *ColourNo*, *Limit*, *Operator*)

Index:

Specify the index you would like to read the current color for. This values depends on the type of color fill selected:

- 0 - 31 for type Multi-state
- 0 - 255 for type Array
- 0 - 255 for type Threshold
- 0- 1 for Gradient

ColourNo:

A value between 0 and 255 representing the color applied to the Index setting.

Limit:

A value between 0 and 100 representing the threshold limit. Used for type Threshold only.

Operator:

The value representing the current operator used for the threshold limit setting:

- 0 : < (less than)
- 1 : > (greater than)
- 2 : <= (less than or equal to)
- 3 : >= (greater than or equal to)

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillColourColourGet, PropertiesFillColourGet, PropertiesFillColourPut

PropertiesFillColourColourPutEx

Sets the color at the specific index on the **Fill | Color** tab of the Object Properties dialog for type Array, Threshold and Gradient.

Syntax

PropertiesFillColourColourPutEx(*Index*, *OnColourNo*, *OffColourNo*, *Limit*, *Operator*)

Index:

Specify the index you want to read the current color for. This values depends on the type of color fill selected:

- 0 - 31 for type Multi-state
- 0 - 255 for type Array
- 0 - 255 for type Threshold
- 0- 1 for Gradient

OnColourNo:

An RGB value representing the "on" color applied to the Index setting.

OffColourNo:

An RGB value representing the "off" color applied to the Index setting.

Limit:

A value between 0 and 100 representing the threshold limit. Used for type Threshold only.

Operator:

The value representing the current operator used for the threshold limit setting:

- 0 : < (less than)
- 1 : > (greater than)
- 2 : <= (less than or equal to)
- 3 : >= (greater than or equal to)

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillColourColourGetEx, PropertiesFillColourPut

PropertiesFillColourGet

Reads the values set on the **Fill** | **Color** tab of the Object Properties dialog for the current object.

Syntax

PropertiesFillColourGet(*FillColourType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*, *RangeFlag*, *RangeMin*, *RangeMax*)

FillColourType:

The fill color type:

- 0 = On / Off
- 1 = Multi-state
- 2 = Array
- 3 = Threshold
- 4 = Gradient

ExpressionA:

This is the main expression:

- ON color when for type On / Off
- Conditions A for type Multi-state
- Array expression for type Array
- Color expression for type Animated

ExpressionB:

Conditions B, only used for multistate symbol sets.

ExpressionC:

Conditions C, only used for multistate symbol sets.

ExpressionD:

Conditions D, only used for multistate symbol sets.

ExpressionE:

Conditions E, only used for multistate symbol sets.

RangeFlag:

If set to TRUE, checks the Specify range checkbox. Flag is only valid for Threshold and Gradient types.

RangeMin:

This floating point value sets the minimum range of the tag value. Only necessary if the argument RangeFlag is set to TRUE.

RangeMax:

This floating point value sets the maximum range of the tag value. Only necessary, if the argument RangeFlag is set to TRUE.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillColourPut, PropertiesFillColourColourGet, PropertiesFillColourColourPut

PropertiesFillColourPut

Sets the values on the Fill | Color tab of the Object Properties dialog for the current object.

Syntax

PropertiesFillColourPut(*FillColourType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*, *RangeFlag*, *RangeMin*, *RangeMax*)

FillColourType:

The fill color type:

- 0 = On / Off
- 1 = Multi-state
- 2 = Array
- 3 = Threshold
- 4 = Gradient

ExpressionA:

This is the main expression:

- ON color when for type On / Off
- Conditions A for type Multi-state
- Array expression for type Array
- Color expression for type Animated

ExpressionB:

Conditions B, only used for multistate symbol sets.

ExpressionC:

Conditions C, only used for multistate symbol sets.

ExpressionD:

Conditions D, only used for multistate symbol sets.

ExpressionE:

Conditions E, only used for multistate symbol sets.

RangeFlag:

If set to TRUE, checks the Specify range checkbox. Flag is only valid for Threshold and Gradient types.

RangeMin:

This floating point value sets the minimum range of the tag value. Only necessary if the argument RangeFlag is set to TRUE.

RangeMax:

This floating point value sets the maximum range of the tag value. Only necessary, if the argument RangeFlag is set to TRUE.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillColourGet, PropertiesFillColourColourGet, PropertiesFillColourColourPut

PropertiesFillLevelGet

Reads the values set on the **Fill | Level** tab of the Object Properties dialog for the current object.

> **Note:** As this function does not support True Color functionality, it has been superseded by the function PropertiesFillLevelGetEx.

Syntax

PropertiesFillLevelGet(*Expression, RangeFlag, RangeMin, RangeMax, OffsetMin, OffsetMax, FillDirection, BackgroundColour*)

Expression:

The level expression.

RangeFlag:

TRUE if the Specify range checkbox is selected.

RangeMin:

The minimum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

RangeMax:

The maximum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

OffsetMin:

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its minimum.

OffsetMax:

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its maximum.

FillDirection:

The current fill direction setting:

- 0 = up
- 1 = down
- 2 = left
- 3 = right

BackgroundColour:

A value between 0 and 255 representing the background color setting.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillColourPut

PropertiesFillLevelGetEx

Reads the values set on the **Fill | Level** tab of the Object Properties dialog for the current object.

Syntax

PropertiesFillLevelGetEx(*Expression, RangeFlag, RangeMin, RangeMax, OffsetMin, OffsetMax, FillDirection, OnColour, OffColour*)

Expression:

The level expression.

RangeFlag:

TRUE if the Specify range checkbox is selected.

RangeMin:

The minimum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

RangeMax:

The maximum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

OffsetMin:

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its minimum.

OffsetMax:

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its maximum.

FillDirection:

The current fill direction setting:

- 0 = up
- 1 = down

- 2 = left
- 3 = right

OnColour:

An RGB value representing the background "on" color setting.

OffColour:

An RGB value representing the background "off" color setting.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillColourPut

PropertiesFillLevelPut

Sets the values on the Fill | Level tab of the Object Properties dialog for the current object.

> **Note:** As this function does not support True Color functionality, it is superseded by the function PropertiesFillLevelPutEx.

Syntax

PropertiesFillLevelPut(*Expression*, *RangeFlag*, *RangeMin*, *RangeMax*, *OffsetMin*, *OffsetMax*, *FillDirection*, *BackgroundColour*)

Expression:

The level expression.

RangeFlag:

TRUE if the Specify range checkbox is selected.

RangeMin:

The minimum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

RangeMax:

The maximum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

OffsetMin:

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its minimum.

OffsetMax:

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its maximum.

FillDirection:

The current fill direction setting:

- 0 = up
- 1 = down
- 2 = left
- 3 = right

BackgroundColour:

A value between 0 and 255 representing the background color setting.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillLevelPutEx,PropertiesFillLevelGet

PropertiesFillLevelPutEx

Sets the values on the **Fill | Level** tab of the Object Properties dialog for the current object.

Syntax

PropertiesFillLevelPutEx(*Expression, RangeFlag, RangeMin, RangeMax, OffsetMin, OffsetMax, FillDirection, OnColour, OffColour*)

Expression:

The level expression.

RangeFlag:

TRUE if the Specify range checkbox is selected.

RangeMin:

The minimum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

RangeMax:

The maximum floating point value in the range of the tag. This argument is only valid if RangeFlag is set to TRUE.

OffsetMin:

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its minimum.

OffsetMax:

The value between 0 and 100 representing the percentage of the area displayed as filled when the tag value is at its maximum.

FillDirection:

The current fill direction setting:

- 0 = up
- 1 = down
- 2 = left
- 3 = right

OnColour:

An RGB value representing the background "on" color setting.

OffColour:

An RGB value representing the background "off" color setting.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesFillLevelGetEx

PropertiesInputKeyboardGet

Reads the values set on the **Input | Keyboard Command** tab of the Object Properties dialog for the current object.

Syntax

PropertiesInputKeyboardGet(*Index*, *KeySequence*, *Command*, *Area*, *Privilege*, *LogMessage*)

Index:

0 to 255 for the key sequence.

KeySequence:

String of the keys to be pressed.

Command:

Expression for the key sequence command.

Area:

0 to 255 for the area, where 0 ticks the checkbox Same area as object.

Privilege:

0 to 255 for the privilege, where 0 ticks the checkbox Same privilege as object.

LogMessage:

The message text to be logged.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesInputKeyboardPut

PropertiesInputKeyboardPut

Sets the values on the Input | Keyboard Commands tab of the Object Properties dialog for the current object.

Syntax

PropertiesInputKeyboardPut(*Index*, *KeySequence*, *Command*, *Area*, *Privilege*, *LogMessage*)

Index:

0 to 255 for the key sequence.

KeySequence:

String of the keys to be pressed.

Command:

Expression for the key sequence command.

Area:

0 to 255 for the area, where 0 ticks the checkbox Same area as object.

Privilege:

0 to 255 for the privilege, where 0 ticks the checkbox Same privilege as object.

LogMessage:

The message text to be logged.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesInputKeyboardGet

PropertiesInputTouchGet

Reads the values set on the **Input | Touch** tab of the Object Properties dialog for the current object.

Syntax

PropertiesInputTouchGet(*Action*, *Expression*, *LogMessage*, *RepeatRate*)

Action:

The type of keyboard action:

- 0 = Up
- 1 = Down
- 2 = Repeat

Expression:

The expression configured for the selected keyboard action (either up, down or repeat).

LogMessage:

The message text to be logged.

RepeatRate:

A value between 1 and 32000 representing the repeat rate in milliseconds.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesInputTouchPut

PropertiesInputTouchPut

Sets the values on the **Input | Touch** tab of the Object Properties dialog for the current object.

Syntax

PropertiesInputTouchPut(*Action*, *Expression*, *LogMessage*, *RepeatRate*)

Action:

The type of keyboard action:

- 0 = Up
- 1 = Down
- 2 = Repeat

Expression:

The expression configured for the selected keyboard action (either up, down or repeat).

LogMessage:

The message text to be logged.

RepeatRate:

A value between 1 and 32000 representing the repeat rate in milliseconds.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesInputTouchGet

PropertiesShowDialog

Shows the properties dialog for an object or a form for Genies.

Syntax

PropertiesShowDialog

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

PropertiesSymbolSetGet

Reads the type and expressions configured on the **Appearance** | **General** tab of the Object Properties dialog for a symbol set.

Syntax

PropertiesSymbolSetGet(*SymbolSetType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*)

SymbolSetType:

Defines the symbol set type:

- 0 = On / Off
- 1 = Multi-state
- 2 = Array
- 3 = Animated

ExpressionA:

This is the main expression:

- ON symbol when for type On / Off
- Conditions A for type Multi-state
- Array expression for type Array
- Animate when for type Animated

ExpressionB:

Conditions B, only used for multistate symbol sets.

ExpressionC:

Conditions C, only used for multistate symbol sets.

ExpressionD:

Conditions D, only used for multistate symbol sets.

ExpressionE:

Conditions E, only used for multistate symbol sets.

Return Value

The requested values, as a string.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PropertiesSymbolSetPut

Example

```
` Gets the properties on the Appearance/General sheet for a symbol set
GraphicsBuilder.PropertiesSymbolSetGet nType, Expression1, Expression2, Expression3,
 Expression4, Expression5
```

PropertiesSymbolSetPut

Sets the type defined for a symbol set on the **Appearance | General** tab of the Object Properties dialog, as well any expressions used

Syntax

PropertiesSymbolSetPut(*SymbolSetType*, *ExpressionA*, *ExpressionB*, *ExpressionC*, *ExpressionD*, *ExpressionE*)

SymbolSetType:

Defines the symbol set type:

- 0 = On / Off
- 1 = Multi-state
- 2 = Array
- 3 = Animated

ExpressionA:

This is the main expression:

- ON symbol when for type On / Off
- Conditions A for type Multi-state
- Array expression for type Array
- Animate when for type Animated

ExpressionB:

Conditions B, only used for multistate symbol sets.

ExpressionC:

Conditions C, only used for multistate symbol sets.

ExpressionD:

Conditions D, only used for multistate symbol sets.

ExpressionE:

Conditions E, only used for multistate symbol sets.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PropertiesSymbolSetGet

Example

```
` Sets the properties on the Appearance General sheet for a symbol set
GraphicsBuilder.PropertiesSymbolSetPut 0, "ON / OFF", "", "", "", ""
```

PropertiesSymbolSetSymbolGet

Retrieves the Element name and Library name of the "Index" element of the currently selected object.

"Index" refers to the element within the currently selected object. For example:

- If the currently selected object is an On/Off symbol set, index can be a value in the range 0..1
- If the currently selected object is a multistate symbol set, index can be a value in the range 0..31
- If the currently selected object is an array symbol set, index can be a value in the range 0..255
- If the currently selected object is an animated symbol set, index can be a value in the range 0..255
- On return, "Element" will contain the name of the symbol set element name for the "Index" element
- On return, "Library" will contain the name of the symbol set library name for the "Index" element
  e.g.
  Index=0, Element="detail_entrycoil1_grey_01", Library="steelmill"
  Index=1, Element="detail_entrycoil1_green_01", Library="steelmill"

Syntax

PropertiesSymbolSetSymbolGet(Index, Library, Element)

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Test()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PropertiesSymbolSetSymbolPut(1, "
Library", "
Element")
Dim sLibrary As String
Dim sElement As String
gb.PropertiesSymbolSetSymbolGet(1, sLibrary, sElement)
End Sub
```

Related Functions

PropertiesSymbolSetSymbolPut

PropertiesSymbolSetSymbolPut

Sets the Element name and Library name of the "Index" element of the currently selected object.

"Index" refers to the element within the currently selected object. For example:

- If the currently selected object is an On/Off symbol set, index can be a value in the range 0..1
- If the currently selected object is a multistate symbol set, index can be a value in the range 0..31
- If the currently selected object is an array symbol set, index can be a value in the range 0..255
- If the currently selected object is an animated symbol set, index can be a value in the range 0..255
- On return, "Element" will contain the name of the symbol set element name for the "Index" element
- On return, "Library" will contain the name of the symbol set library name for the "Index" element
  e.g.
  Index=0, Element="detail_entrycoil1_grey_01", Library="steelmill"
  Index=1, Element="detail_entrycoil1_green_01", Library="steelmill"

Syntax

PropertiesSymbolSetSymbolPut(Index, Library, Element)

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Test()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PropertiesSymbolSetSymbolPut(1, "
Library", "
Element")
Dim sLibrary As String
Dim sElement As String
gb.PropertiesSymbolSetSymbolGet(1, sLibrary, sElement)
End Sub
```

Related Functions

PropertiesSymbolSetSymbolGet()

PropertiesTransCentreOffsetExpressGet

Retrieve the express properties.

Syntax

Prop-
ert-
iesTransCentreOffsetExpressGet(movementRotationalExpress,scalingHorizontalExpress,scalingVertica

movementRotationalExpress - Movement Rotational Express
scalingHorizontalExpress - Scaling Horizontal Express
scalingVerticalExpress - Scaling Vertical Express
sliderRotationalExpress - Slider Rotational Express

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Test()
Dim gb As GraphicsBuilder.GraphicsBuilder
gb = New GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PropertiesTransCentreOffsetExpressPut(0, 0, 0, 0)
Dim nMovRot As Short
Dim nScaleHorz As Short
Dim nScaleVert As Short
Dim nSliderRot As Short
gb.PropertiesTransCentreOffsetExpressGet(nMovRot, nScaleHorz, nScaleVert, nS
erRot)
End Sub
```

Related Functions

PropertiesTransCentreOffsetExpressPut

PropertiesTransCentreOffsetExpressPut

Sets the express properties.

Syntax

Prop-
ert-
iesTransCentreOffsetExpressPut(movementRotationalExpress,scalingHorizontalExpress,scalingVertica

movementRotationalExpress - Movement Rotational Express
scalingHorizontalExpress - Scaling Horizontal Express
scalingVerticalExpress - Scaling Vertical Express
sliderRotationalExpress - Slider Rotational Express

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Test()
Dim gb As GraphicsBuilder.GraphicsBuilder
gb = New GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PropertiesTransCentreOffsetExpressPut(0, 0, 0, 0)
Dim nMovRot As Short
Dim nScaleHorz As Short
Dim nScaleVert As Short
Dim nSliderRot As Short
gb.PropertiesTransCentreOffsetExpressGet(nMovRot, nScaleHorz, nScaleVert, nS
erRot)
End Sub
```

Related Functions

PropertiesTransCentreOffsetExpressGet

PropertiesTransformationGet

Reads the property values set on the **Movement**, **Scaling** and **Slider** tabs of the Object Properties dialog for the current object.

Syntax

PropertiesTransformationGet(*Action, Expression, RangeFlag, RangeMin, RangeMax, OffsetMin, OffsetMax, CustomFlag, CentreOffsetRight, CentreOffsetDown*)

Action:

Selects the tab on the Object Properties dialog that data will be read from:

- 0 = MovementHorizontal
- 1 = MovementVertical
- 2 = MovementRotational
- 3 = ScalingHorizontal
- 4 = ScalingVertical

- 5 = SliderHorizontal
- 6 = SliderVertical
- 7 = SliderRotational

Expression:

The main expression in Field:

- **Movement expression** for the actions MovementHorizontal or MovementVertical
- **Angle expression** for action MovementRotational
- **Scaling expression** for actions ScalingHorizontal or ScalingVertical
- **Tag** for actions SliderHorizontal, SliderVertical or SliderRotational

RangeFlag:

TRUE if Specify range is checked

RangeMin:

The minimum floating point value. 0 (zero) if RangeFlag is not set.

RangeMax:

This maximum floating point value. 0 (zero) if RangeFlag is set to TRUE.

OffsetMin:

The value of Angle at minimum for the actions MovementRotational and SliderRotational, or Offset at minimum for other actions.

OffsetMax:

The value of Angle at maximum for the actions MovementRotational and SliderRotational, or Offset at maximum for other actions.

CustomFlag:

TRUE if custom is selected for the center axis offset setting for the actions MovementRotational, SliderRotational, Scaling Horizontal or ScalingVertical.

CentreOffsetRight:

A value between 0 and 32767 representing the customized setting for center offset right. 0 (zero) if CustomFlag is not set.

CentreOffsetDown:

A value between 0 and 32767 representing the customized setting for center offset down. 0 (zero) if CustomFlag is not set.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesTransformationPut

PropertiesTransformationPut

Sets values for the properties on the **Movement**, **Scaling** and **Slider** tabs of the Object Properties dialog.

Syntax

PropertiesTransformationGet(*Action*, *Expression*, *RangeFlag*, *RangeMin*, *RangeMax*, *OffsetMin*, *OffsetMax*, *CustomFlag*, *CentreOffsetRight*, *CentreOffsetDown*)

Action:

Selects the tab on the Object Properties dialog that data will be read from:

- 0 = MovementHorizontal
- 1 = MovementVertical
- 2 = MovementRotational
- 3 = ScalingHorizontal
- 4 = ScalingVertical
- 5 = SliderHorizontal
- 6 = SliderVertical
- 7 = SliderRotational

Expression:

The main expression in Field:

- **Movement expression** for the actions MovementHorizontal or MovementVertical
- **Angle expression** for action MovementRotational
- **Scaling expression** for actions ScalingHorizontal or ScalingVertical
- **Tag** for actions SliderHorizontal, SliderVertical or SliderRotational

RangeFlag:

TRUE if Specify range is checked

RangeMin:

The minimum floating point value. 0 (zero) if RangeFlag is not set.

RangeMax:

This maximum floating point value. 0 (zero) if RangeFlag is set to TRUE.

OffsetMin:

The value of Angle at minimum for the actions MovementRotational and SliderRotational, or Offset at minimum for other actions.

OffsetMax:

The value of Angle at maximum for the actions MovementRotational and SliderRotational, or Offset at maximum for other actions.

CustomFlag:

TRUE if custom is selected for the center axis offset setting for the actions MovementRotational, SliderRotational, Scaling Horizontal or ScalingVertical.

CentreOffsetRight:

A value between 0 and 32767 representing the customized setting for center offset right. 0 (zero) if CustomFlag is not set.

CentreOffsetDown:

A value between 0 and 32767 representing the customized setting for center offset down. 0 (zero) if CustomFlag is not set.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesTransformationGet

PropertiesTrendGet

Reads the values for a trend object as set on the **Appearance | General** tab of the Object Properties dialog.

> **Note:** As this function does not support True Color functionality, it has been superseded by the function PropertiesTrendGetEx.

Syntax

PropertiesTrendGet(*NumberOfSamples, PixelPerSample, Expression1, Colour1, Expression2, Colour2, Expression3, Colour3, Expression4, Colour4, Expression5, Colour5, Expression6, Colour6, Expression7, Colour7, Expression8, Colour8*)

NumberOfSamples:

A value between 0 and 32767 representing the number of samples in a trend display.

PixelPerSample:

A value between 1 and 32, representing the width of each sample in pixels.

Expression1:

String argument for the field Pen1.

Colour1:

A value between 0 and 255 representing the color of trend Pen1.

Expression2:

String argument for the field Pen2.

Colour2:

A value between 0 and 255 representing the color of trend Pen2.

Expression3:

String argument for the field Pen3.

Colour3:

A value between 0 and 255 representing the color of trend Pen3.

Expression4:

String argument for the field Pen4.

Colour4:

A value between 0 and 255 representing the color of trend Pen4.

Expression5:

String argument for the field Pen5.

Colour5:

A value between 0 and 255 representing the color of trend Pen5.

Expression6:

String argument for the field Pen6.

Colour6:

A value between 0 and 255 representing the color of trend Pen6.

Expression7:

String argument for the field Pen7.

Colour7:

A value between 0 and 255 representing the color of trend Pen7.

Expression8:

String argument for the field Pen8.

Colour8:

A value between 0 and 255 representing the color of trend Pen8.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesTrendPut

PropertiesTrendGetEx

Reads the values for a trend object as set on the **Appearance** | **General** tab of the Object Properties dialog.

Syntax

PropertiesTrendGetEx(*NumberOfSamples, PixelPerSample, Expression1, OnColour1, OffColour1, Expression2, OnColour2, OffColour2, Expression3, OnColour3, OffColour3, Expression4, OnColour4, OffColour4, Expression5, OnColour5, OffColour5, Expression6, OnColour6, OffColour6, Expression7, OnColour7, OffColour7, Expression8, OnColour8, OffColour8*)

NumberOfSamples:

A value between 0 and 32767 representing the number of samples in a trend display.

PixelPerSample:

A value between 1 and 32, representing the width of each sample in pixels.

Expression1:

String argument for the field Pen1.

OnColour1:

An RGB value representing the "on" color of trend Pen1.

OffColour1:

An RGB value representing the "off" color of trend Pen1.

Expression2:

String argument for the field Pen2.

OnColour2:

An RGB value representing the "on" color of trend Pen2.

OffColour2:

An RGB value representing the "off" color of trend Pen2.

Expression3:

String argument for the field Pen3.

OnColour3:

An RGB value representing the "on" color of trend Pen3.

OffColour3:

An RGB value representing the "off" color of trend Pen3.

Expression4:

String argument for the field Pen4.

OnColour4:

An RGB value representing the "on" color of trend Pen4.

OffColour4:

An RGB value representing the "off" color of trend Pen4.

Expression5:

String argument for the field Pen5.

OnColour5:

An RGB value representing the "on" color of trend Pen5.

OffColour5:

An RGB value representing the "off" color of trend Pen5.

Expression6:

String argument for the field Pen6.

OnColour6:

An RGB value representing the "on" color of trend Pen6.

OffColour6:

An RGB value representing the "off" color of trend Pen6.

Expression7:

String argument for the field Pen7.

OnColour7:

An RGB value representing the "on" color of trend Pen7.

OffColour7:

An RGB value representing the "off" color of trend Pen7.

Expression8:

String argument for the field Pen8.

OnColour8:

An RGB value representing the "on" color of trend Pen8.

OffColour8:

An RGB value representing the "off" color of trend Pen8.

Return Value

The requested values, as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesTrendPutEx

PropertiesTrendPut

Sets the values for a trend object that appear on the **Appearance** | **General** tab of the Object Properties dialog.

> **Note:** As this function does not support True Color functionality, it has been

superseded by the functions PropertiesTrendPutEx.

Syntax

PropertiesTrendGet(*NumberOfSamples, PixelPerSample, Expression1, Colour1, Expression2, Colour2, Expression3, Colour3, Expression4, Colour4, Expression5, Colour5, Expression6, Colour6, Expression7, Colour7, Expression8, Colour8*)

NumberOfSamples:

A value between 0 and 32767 representing the number of samples in a trend display.

PixelPerSample:

A value between 1 and 32, representing the width of each sample in pixels.

Expression1:

String argument for the field Pen1.

Colour1:

A value between 0 and 255 representing the color of trend Pen1

Expression2:

String argument for the field Pen2.

Colour2:

A value between 0 and 255 representing the color of trend Pen2.

Expression3:

String argument for the field Pen3.

Colour3:

A value between 0 and 255 representing the color of trend Pen3.

Expression4:

String argument for the field Pen4.

Colour4:

A value between 0 and 255 representing the color of trend Pen4.

Expression5:

String argument for the field Pen5.

Colour5:

A value between 0 and 255 representing the color of trend Pen5.

Expression6:

String argument for the field Pen6.

Colour6:

A value between 0 and 255 representing the color of trend Pen6.

Expression7:

String argument for the field Pen7.

Colour7:

A value between 0 and 255 representing the color of trend Pen7.

Expression8:

String argument for the field Pen8.

Colour8:

A value between 0 and 255 representing the color of trend Pen8.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesTrendGet

PropertiesTrendPutEx

Sets the values for a trend object that appear on the **Appearance | General** tab of the Object Properties dialog.

Syntax

PropertiesTrendPutEx(*NumberOfSamples, PixelPerSample, Expression1, OnColour1, OffColour1, Expression2, OnColour2, OffColour2 , Expression3, OnColour3, OffColour3 , Expression4, OnColour4, OffColour4 , Expression5, OnColour5, OffColour5, Expression6, OnColour6, OffColour6 , Expression7, OnColour7, OffColour7, Expression8, OnColour8, OffColour8,*)

NumberOfSamples:

A value between 0 and 32767 representing the number of samples in a trend display.

PixelPerSample:

A value between 1 and 32, representing the width of each sample in pixels.

Expression1:

String argument for the field Pen1.

OnColour1:

An RGB value representing the "on" color of trend Pen1.

OffColour1:

An RGB value representing the "off" color of trend Pen1.

Expression2:

String argument for the field Pen2.

OnColour2:

An RGB value representing the "on" color of trend Pen2.

OffColour2:

An RGB value representing the "off" color of trend Pen2.

Expression3:

String argument for the field Pen3.

OnColour3:

An RGB value representing the "on" color of trend Pen3.

OffColour3:

An RGB value representing the "off" color of trend Pen3.

Expression4:

String argument for the field Pen4.

OnColour4:

An RGB value representing the "on" color of trend Pen4.

OffColour4:

An RGB value representing the "off" color of trend Pen4.

Expression5:

String argument for the field Pen5.

OnColour5:

An RGB value representing the "on" color of trend Pen5.

OffColour5:

An RGB value representing the "off" color of trend Pen5.

Expression6:

String argument for the field Pen6.

OnColour6:

An RGB value representing the "on" color of trend Pen6.

OffColour6:

An RGB value representing the "off" color of trend Pen6.

Expression7:

String argument for the field Pen7.

OnColour7:

An RGB value representing the "on" color of trend Pen7.

OffColour7:

An RGB value representing the "off" color of trend Pen7.

Expression8:

String argument for the field Pen8.

OnColour8:

An RGB value representing the "on" color of trend Pen8.

OffColour8:

An RGB value representing the "off" color of trend Pen8.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PropertiesTrendGetEx

PropertyVisibility

Sets the Hidden when argument on the **Appearance | Visibility** tab of the Object Properties dialog.

Syntax

PropertyVisibility(*Text*)

Text:

The argument string.

Return Value

If retrieving the current setting, the argument string. If enabling or disabling the option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

Note: For details on handling return and error values, see Error Handling.

Note: This function is implemented in the C++ environment as two separate functions: `put_PropertyVisibility` enables or disables this option, and `get_PropertyVisiblity` retrieves the current option setting.

Library Object Functions

With Library Object functions you can use and manipulate the objects stored in libraries in your project. This includes such objects as Genies, Super Genies, Symbols, and so on.

LibraryObjectFirstProperty	Returns the name and value of the active Genie's first property.
LibraryObjectFirstPropertyEx	Returns the name and value of a specified Genie's first property.
LibraryObjectHotspotGet	Retrieves the hotspot marker in a Genie or symbol page.
LibraryObjectHotspotPut	Positions the hotspot marker in a Genie or symbol page.
LibraryObjectName	Returns the name of the selected object.
LibraryObjectNextProperty	Returns the name and value of the active Genie's "next" property when implemented following a call

	of the function LibraryObjectFirstProperty.
LibraryObjectNextPropertyEx	Returns the name and value of the "next" property of the Genie specified by the implementation of LibraryObjectFirstPropertyEx.
LibraryObjectPlace	Places a library object (a symbol or genie) on the active CitectSCADA graphics page at the default location (top left corner).
LibraryObjectPlaceEx	Places a library object (a symbol or genie) on the active CitectSCADA graphics page at the specified location.
LibraryObjectPutProperty	Sets the value of a specified property for the active genie.
LibSelectionHooksEnabled	Writing a TRUE value with this function enables library selection hooks. When enabled, selecting Paste Genie or Paste Symbol (or their equivalent function key of toolbar button) will not show the standard selection dialog, but will fire the automation event PasteSymbol or PasteGenie instead.
LibraryShowPasteDialog	Shows either the paste Genie or Paste Symbol dialog.

For details on handling return and error values, see Error Handling.

LibraryObjectFirstProperty

Returns the name and value of the active Genie's first property. Can be used with LibraryObjectNextProperty to step through a genie's properties.

Syntax

LibraryObjectFirstProperty(*PropertyName, PropertyValue*)

PropertyName:

Returns the name of the active genie's first property as a string.

PropertyValue:

Returns the value of the active genie's first property as a string.

Return Value

The name and value of the Genie's first property as string values

Note: For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectPlace, LibraryObjectNextProperty, LibraryObjectPutProperty, LibraryObjectName

Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryObjectFirstProperty PropName, PropValue
While Err.Number = 0
        Debug.Print PropName, PropValue
        GraphicsBuilder.LibraryObjectNextProperty PropName, PropValue
Wend
```

LibraryObjectFirstPropertyEx

Returns the name and value of a specified Genie's first property. Can be used in conjunction with LibraryObjectNextPropertyEx to step through the specified Genie's properties.

Syntax

LibraryObjectFirstPropertyEx(*Project, Library, Object, PropertyName, PropertyValue*)

Project:

The name of the project where the Genie is located.

Library:

The name of the library where the Genie is located.

Object:

The name of the genie.

PropertyName:

Returns the name of the active genie's first property as a string.

PropertyValue:

Returns the value of the active genie's first property as a string.

Return Value

The name and value of the specified Genie's first property as string values.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectPlace, LibraryObjectNextProperty, LibraryObjectPutProperty, LibraryObjectName

Example

```
' Retrieves the first property of the specified Genie
GraphicsBuilder.LibraryObjectFirstPropertyEx "include", "motors", "Motor_2_east",
 PropName, PropValue
```

LibraryObjectHotspotGet

Retrieves the hotspot marker in a Genie or symbol page. Fails if not a Genie or symbol page.

Syntax

LibraryObjectHotspotGet(*Xposition, YPosition*)

Xposition:

Absolute X position in pixels from the left side of the page.

YPosition:

Absolute Y position in pixels from the top of the page.

Return Value

X and Y values for the hotspot, where X represents the number of pixels from the left hand side of the page, and Y represents the number of pixels from the top of the page.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectHotspotPut

LibraryObjectHotspotPut

Positions the hotspot marker in a Genie or symbol page. Fails if not a Genie or symbol page.

Syntax

LibraryObjectHotspotPut(*Xposition, YPosition*)

Xposition:

Absolute X position in pixels from the left side of the page.

YPosition:

Absolute Y position in pixels from the top of the page.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectHotspotGet

LibraryObjectName

Returns the name of the selected object, if it is a library object.

Syntax

LibraryObjectName(*Project, Library, Object, GenieOrSymbol*)

Project:

The name of the project that contains the object library you would like to source.

Library:

Specifies the library that contains the symbol or genie you would like to retrieve the name of.

Object:

The name of the symbol or genie as a string.

GenieOrSymbol:

Indicates whether the object you want to retrieve the name for is a symbol or a genie.

- 1 = Genie
- 2 = Symbol

Return Value

The name of the specified object as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectFirstProperty, LibraryObjectNextProperty, LibraryObjectPutProperty

Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryObjectName Project, File, Page, LibType
If Err.Number = 0 Then
        Debug.Print Project; "."; File; "."; Page
Else
        Debug.Print "not a library object"
End If
```

LibraryObjectNextProperty

Returns the name and value of the active Genie's "next" property when implemented following a call of the function LibraryObjectFirstProperty. By using multiple calls of this function, you can iterate through an object's properties.

Syntax

LibraryObjectNextPropertyEx(*PropertyName, PropertyValue*)

PropertyName:

Returns the name of the active genie's next property as a string.

PropertyValue:

Returns the value of the active genie's next property as a string.

Return Value

The name and value of the Genie's next property as string values

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectPlace, LibraryObjectFirstProperty, LibraryObjectPutProperty, LibraryObjectName

Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryObjectFirstProperty PropName, PropValue
While Err.Number = 0
        Debug.Print PropName, PropValue
        GraphicsBuilder.LibraryObjectNextProperty PropName, PropValue
Wend
```

LibraryObjectNextPropertyEx

Returns the name and value of the "next" property of the Genie specified by the implementation of LibraryObjectFirstPropertyEx. By using multiple calls of this function, you can iterate through the specified genie's properties.

Syntax

LibraryObjectNextPropertyEx(*PropertyName, PropertyValue*)

PropertyName:

Returns the name of the active genie's next property as a string.

PropertyValue:

Returns the value of the active genie's next property as a string.

Return Value

The name and value of the Genie's next property as string values.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectFirstPropertyEx

Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.LibraryObjectFirstPropertyEx "include", "motors", "Motor_2_east",
 PropName, PropValue
While Err.Number = 0
        Debug.Print PropName, PropValue
        GraphicsBuilder.LibraryObjectNextProperty PropName, PropValue
Wend
```

LibraryObjectPlace

Places a library object (a symbol or genie) on the active CitectSCADA graphics page at the default location (top left corner). This function will not succeed if the specified object is not found.

Syntax

LibraryObjectPlace(*Project, Library, Object, GenieOrSymbol, Linked*)

Project:

The name of the project that contains the object library you would like to source.

Library:

Specifies the library that contains the symbol or genie you would like to place on the active Citect-SCADA graphics page.

Object:

The name of the symbol or genie you would like to place on the active CitectSCADA graphics page.

GenieOrSymbol:

Indicates whether the object you want to use is a symbol of a genie.

- 0 = Library type unknown (will automatically select genie or symbol)
- 1 = Genie
- 2 = Symbol

Linked:

If set to TRUE, the object will remain linked to the library it came from. (select TRUE for Genies). Can only be set to FALSE if GenieOrSymbol is set to 2 (Symbol).

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectFirstProperty, LibraryObjectNextProperty, LibraryObjectPutProperty, LibraryObjectName

Example

```
' Adds an object to the current CitectSCADA graphics page
GraphicsBuilder.LibraryObjectPlace "include", "agitator", "agit_1_Pos1_g", 2, True
GraphicsBuilder.PositionAt 200, 200
```

LibraryObjectPlaceEx

Places a library object (a symbol or genie) on the active CitectSCADA graphics page at the specified location. This function will not succeed if the specified object is not found.

Syntax

LibraryObjectPlaceEx(*Project, Library, Object, GenieOrSymbol, Linked, Xposition, YPosition*)

Project:

The name of the project that contains the object library you would like to source.

Library:

Specifies the library that contains the symbol or genie you would like to place on the active CitectSCADA graphics page.

Object:

The name of the symbol or genie you would like to place on the active CitectSCADA graphics page.

GenieOrSymbol:

Indicates whether the object you want to use is a symbol of a genie.

- 0 = Library type unknown (will automatically select genie or symbol)
- 1 = Genie
- 2 = Symbol

Linked:

If set to TRUE, the object will remain linked to the library it came from. (Select TRUE for Genies).

Xposition:

Absolute X position in pixels from the left hand side of the page.

YPosition:

Absolute Y position in pixels from the top of the page.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectFirstProperty, LibraryObjectNextProperty, LibraryObjectPutProperty, LibraryObjectName

Example

```
' Adds an object to the current graphics page at 200 pixels from the left and top
GraphicsBuilder.LibraryObjectPlaceEx "include", "agitator", "agit_1_Pos1_g",
 2, True, 200, 200
```

LibraryObjectPutProperty

Sets the value of a specified property for the active genie. The field name is case-sensitive.

Syntax

LibraryObjectPutProperty(*PropertyName, PropertyValue*)

PropertyName:

The name of the property to be modified, as returned by the function LibraryObjectFirstProperty or LibraryObjectNextProperty.

PropertyValue:

The value to be written to the property as a string.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

LibraryObjectPlace, LibraryObjectFirstProperty, LibraryObjectNextProperty, LibraryObjectName

Example

```
GraphicsBuilder.LibraryObjectPlace "include", "motors", "Motor_1_east", 1, True
GraphicsBuilder.LibraryObjectPutProperty "Tag", "My test genie"
```

LibraryShowPasteDialog

Shows either the Paste Genie or Paste Symbol dialog.

Syntax

LibraryShowPasteDialog(*GenieOrSymbol*)

GenieOrSymbol:

Indicates whether the object you want to use is a symbol of a genie.

- 1 = Genie
- 2 = Symbol

Note: For details on handling return and error values, see Error Handling.

Related Functions

PasteSymbol, PasteGenie

LibSelectionHooksEnabled

Writing a TRUE value with this function enables library selection hooks. When enabled, selecting **Paste Genie** or **Paste Symbol** (or their equivalent function key of toolbar button) will not show the standard selection dialog, but will fire the automation event PasteSymbol or PasteGenie instead.

Additionally, when hooks are enabled, pressing CTRL + SHIFT and double-clicking a CitectSCADA page will fire the event SwapObject.

Syntax

LibSelectionHooksEnabled(*HooksEnabled*)

HooksEnabled:

A setting of TRUE enables library selection hooks.

Return Value

Enables library selection hooks, or retrieves the current library selection hooks setting.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PasteSymbol, PasteGenie

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_LibSelectionHooksEnabled` enables selection hooks, and `get_ LibSelectionHooksEnabled` retrieves the current selection hooks setting.

Metadata Functions

Use these Metadata functions to configure the metadata of the current object on a page.

PropertiesAddMetadata	Adds a new Metadata entry to the properties of the current object.
ProperitiesSelectFirstMetadata	Selects the first metadata entry from the properties of the current object.
PropertiesSelectNextMetadata	Selects the next metadata entry from the properties of the current object.
PropertiesSelectMetadataByName	Selects the specified metadata entry in the current page.
PropertiesDeleteMetadata	Deletes the selected metadata from the properties of the current object.
PropertiesMetadataName	Sets or retrieves the name of the currently selected object metadata.
PropertiesMetadataValue	Sets or retrieves the value of the currently selected object metadata.

For details and a VB example on handling return and error values, see Error Handling.

PropertiesAddMetadata

Adds a new metadata entry to the current object properties. This function will return an error if metadata with the specified name already exists.

Syntax

PropertiesAddMetadata (Name, Value)

Name:

The name of the new metadata entry to be added to the properties of the current object

Value:

The value of the new metadata to be added to the current object properties.

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Adding a new element and setting its properties:

```
GraphicsBuilder.PropertiesAddMetadata("MyName","MyValue")
```

Related Functions

PropertiesMetadataValue, PropertiesMetadataName, PropertiesSelectNextMetadata, PropertiesSelectFirstMetadata, PropertiesDeleteMetadata, PropertiesSelectMetadataByName

PropertiesDeleteMetadata

Deletes the selected metadata from the properties of the current object. After an item has been deleted, a call to PropertiesSelectNextMetadata will select the item immediately following the deleted item.

Syntax

PropertiesDeleteMetadata()

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

delete any metadata starting with “a”:

```
Dim name As String
```

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.PropertiesSelectFirstMetadata()
While (Err.Number = 0)
    name = GraphicsBuilder.PropertiesMetadataName
    If (name.ToLower().StartsWith("a")) Then
        GraphicsBuilder.PropertiesDeleteMetadata()
    End If
    GraphicsBuilder.PropertiesSelectNextMetadata()
End While
```

Related Functions

PropertiesMetadataValue, PropertiesMetadataName, PropertiesSelectNextMetadata, PropertiesSelectFirstMetadata, PropertiesAddMetadata, PropertiesSelectMetadataByName

PropertiesMetadataName

Sets or retrieves the name of the currently selected object metadata.

Syntax

Name = PropertiesMetadataName
PropertiesMetadataName (Name)

Return Value

The name of the currently selected metadata item (as a string). An error is returned if unsuccessful.

Related Functions

PropertiesMetadataValue, PropertiesDeleteMetadata, PropertiesSelectNextMetadata, PropertiesSelectFirstMetadata, PropertiesAddMetadata, PropertiesSelectMetadataByName

PropertiesMetadataValue

Sets or retrieves the value of the currently selected object metadata.

Syntax

Val = PropertiesMetadataValue

PropertiesMetadataValue(*Def*)

Return Value

The value of the currently selected metadata (as a string), or 0 (zero) if successfully used to set the default. An error is returned if unsuccessful.

Related Functions

PropertiesMetadataName, PropertiesDeleteMetadata, PropertiesSelectNextMetadata, PropertiesSelectFirstMetadata, PropertiesAddMetadata, PropertiesSelectMetadataByName

PropertiesSelectFirstMetadata

Selects the first metadata entry from the properties of the current object.

Syntax

PropertiesSelectFirstMetadata()

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Determines whether the page properties of thecurrent object has defined metadata:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.PropertiesSelectFirstMetadata()
If (Err.Number <> 0)
    ' The object has no metadata
End If
```

Related Functions

PropertiesMetadataValue, PropertiesMetadataName, PropertiesSelectNextMetadata,PropertiesDeleteMetadata, PropertiesAddMetadata, PropertiesSelectMetadataByName

PropertiesSelectMetadataByName

Selects the specified metadata in the current page.

Syntax

PropertiesSelectMetadataByName(BSTR Name)

Name:

The name of the metadata to be selected.

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Determining whether an metadata with a particular name exists:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.PropertiesSelectMetadataByName("MyName")
If (Err.Number <> 0)
    ' The metadata does not exist
End If
```

Related Functions

PropertiesDeleteMetadata, PropertiesMetadataValue, PropertiesMetadataName, PropertiesSelectNextMetadata, PropertiesSelectFirstMetadata

PropertiesSelectNextMetadata

Selects the next metadata entry from the properties of the current object.

Syntax

PropertiesSelectNextMetadata()

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Print metadata entries in the current object properties:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.PropertiesSelectFirstMetadata()
While (Err.Number = 0)
    Console.Out.WriteLine(GraphicsBuilder.PropertiesMetadataName)
    GraphicsBuilder.PropertiesSelectNextMetadata()
End While
```

Related Functions

PropertiesMetadataValue, PropertiesMetadataName, PropertiesSelectNextMetadata, PropertiesSelectFirstMetadata, PropertiesAddMetadata, PropertiesSelectMetadataByName

Miscellaneous Functions

These functions are used for special interactions with the Graphics Builder; for example, an external drag-and-drop action could be performed by requesting the active window handle.

BrokenLinkCancelEnabled	Writing a TRUE value enables the functions ProjectUpdatePages or PageOpen to exit and report the error E-POINTER when encountering the first broken link (missing reference) during execution.
ClipboardCopy	Copies the selected object(s) to the Windows Clipboard.
ClipboardCut	Cuts the selected object(s) to the Windows Clipboard.
ClipboardPaste	Paste the elements of the Windows Clipboard on to the active page.
ConvertToBitmap	Converts the active object to a bitmap. Unable to convert if no active object.
Quit	Exits the CitectSCADA development environment.
SelectionEventEnabled	Writing a true value with this function enables an event to be fired for every selection performed on a graphics page. You can also use this function to retrieve the current setting for this option.
UnLockObject	Make an object selectable.

For details and a VB example on handling return and error values, see Error Handling.

BrokenLinkCancelEnabled

Writing a TRUE value enables the functions ProjectUpdatePages or PageOpen to exit and report the error E-POINTER when encountering the first broken link (missing reference) during execution. If set to FALSE, these functions will succeed, but will issue a BrokenLink event for every unresolved reference on a page.

Syntax

BrokenLinkCancelEnabled(*CancelEnabled*)

CancelEnabled:

TRUE if enabled.

Return Value

If retrieving the current setting, TRUE or FALSE. If setting this option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

For details and a VB example on handling return and error values, see Error Handling.

Related Functions

BrokenLink, ProjectUpdatePages, PageOpen

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_BrokenLinkCancelEnabled` enables or disables this option, and `get_BrokenLinkCancelEnabled` retrieves the current setting.

ClipboardCopy

Copies the selected object(s) to the Windows clipboard.

Syntax

ClipboardCopy

Related Functions

ClipboardCut, ClipboardPaste

ClipboardCut

Cuts the selected object(s) to the Windows clipboard.

Syntax

ClipboardCut

Related Functions

ClipboardCopy, ClipboardPaste

ClipboardPaste

Paste the elements of the Windows Clipboard on to the active page.

Syntax

ClipboardPaste

Related Functions

ClipboardCut, ClipboardCopy

ConvertToBitmap

Converts the active object to a bitmap. Fails if no active object.

Syntax

ConvertToBitmap

Quit

Exits the CitectSCADA development environment.

SelectionEventEnabled

Writing a true value with this function enables an event to be fired for every selection performed on a graphics page. You can also use this function to retrieve the current setting for this option.

Syntax

SelectionEventEnabled(*EventEnabled*)

EventEnabled:

Set to TRUE to enable selection events.

Return Value

If retrieving the current setting, TRUE or FALSE. If setting this option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Selection

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_SelectionEventEnabled` enables or disables this option, and `get_SelectionEventEnabled` retrieves the current option setting.

UnLockObject

Make an object selectable.

Syntax

UnLockObject

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
 .
 .
 .
 gb.PageSelectFirstObjectInGenie()
 gb.PageSelectNextObjectInGenie()
 gb.PageTemplateSelectFirstObject()
 gb.PageTemplateSelectNextObject()
 gb.UnLockObject()
 End Sub
```

Related Functions

N/A

Object Drawing and Property Functions

With these functions, you can draw objects and manipulate the properties of objects.

> **Note:** Freehand line drawing is not supported, as the same output can be achieved using the DrawPolygon function.

Only General and 3D properties are supported. Movement, Scaling, Fill, and so on are not accessible.

The settings are applied to or read from the selected object. Typically, the last placed object is the selected object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Attribute3dEffects	Applies a 3D effect to an object, or retrieves the current 3D effect setting.
Attribute3dEffectDepth	Applies a level of depth to a 3D effect, or retrieves the current depth setting.
AttributeAN	Retrieves the animation number (AN) of the active object.
AttributeBaseCoordinates	Returns the base coordinates of an object.
AttributeClass	Retrieves the class of the active object as a string.
AttributeCornerRadius	Sets or retrieves the corner radius value for the current object.
AttributeEllipseStyle	Applies a style to an ellipse, or retrieves the current ellipse style setting.
AttributeEndAngle	Sets the end angle of an arc or pie-slice, or retrieves the end angle.
AttributeExtentX	Retrieves the X coordinate that represents the extent of the active object.
AttributeExtentY	Retrieves the Y coordinate that represents the extent of the active object.
AttributeFillColour	Sets the fill color for an object, or retrieves a value representing the current fill color.
AttributeFillOffColourEx	Sets the fill color for an object, or retrieves a value representing the current fill color.
AttributeFillOnColourEx	Sets the fill color for an object, or retrieves a value representing the current fill color.
AttributeGradientMode	Sets or retrieves the direction of the gradient for the current object.

AttributeGradientOffColour	Sets or retrieves the "Off" portion of the gradient colour for the current object.
AttributeGradientOnColour	Sets or retrieves the "On" portion of the gradient colour for the current object.
AttributeHiLightColour	Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.
AttributeLineColour	Applies a color to a line, or retrieves the current color setting. This function has been replaced by the functions AttributeLineOnColourEx and AttributeLineOffColourEx.
AttributeLineOnColourEx	This function supports True Color functionality and replaces AttributeLineColour.
AttributeLineOffColourEx	This function supports True Color functionality and replaces AttributeLineColour.
AttributeLineStyle	Applies a style to a line, or retrieves the current style setting.
AttributeLineWidth	Sets the width of a line, or retrieves its current width.
AttributeLoLightColour	Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting. As this function does not support True Colour functionality, it has been superseded by the functions AttributeLoLightOffColourEx and AttributeLoLightOnColourEx.
AttributeLoLightOffColourEx	Sets the lowlight "off" color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.
AttributeLoLightOnColourEx	Sets the lowlight "on" color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.
AttributeNodeCoordinatesFirst	Returns the coordinates of the first node of a free hand line, polygon or pipe.
AttributeNodeCoordinatesNext	Returns the coordinates of any following nodes of a free hand line, polygon or pipe when implemented after AttributeNodeCoordinatesFirst.
AttributePolygonOpen	Defines whether a polygon (polyline) is set to open

	mode (i.e. its two end points are not joined) or closed (its two ends are joined).
AttributeRectangleStyle	Sets the rectangle style, or retrieves the rectangle style setting.
AttributeSetFill	Displays the object as filled, or retrieves the current fill value.
AttributeShadowColour	Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting. As this function does not support True Color functionality, it has been superseded by the functions AttributeShadowOffColourEx and AttributeShadownOnColourEx.
AttributeShadowOffColourEx	Sets the "off" shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.
AttributeShadowOnColourEx	Sets the "on" shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.
AttributeStartAngle	Sets the start angle of an arc or pie-slice, or retrieves the start angle.
AttributeTransformationMatrixGet	Reads the elements of the transformation matrix.
AttributeTransformationMatrixPut	Sets the elements of the transformation matrix.
AttributeX	Retrieves the X coordinate of the active object.
AttributeY	Retrieves the Y coordinate of the active object.
DrawButton	Draws a button on the active page.
DrawCicodeObject	Places a Cicode object on the page at the specified location.
DrawEllipse	Draws an ellipse on the active page.
DrawLine	Draws a line on the active page.
DrawNumber	Places a number object on the page at the specified location.

DrawPipeEnd	Terminates the drawing of a pipe on the active page.
DrawPipeSection	Draws a section of pipe on the active page.
DrawPipeStart	Initiates the process of drawing a pipe on the active page by defining a starting point that DrawPipeSection() can be applied to.
DrawPolygonEnd	Terminates the drawing of a polygon on the active page.
DrawPolygonLine	Draws a line on the active page that forms part of a polygon.
DrawPolygonStart	Initiates the process of drawing a polygon on the active page by defining a starting point that DrawPolygonLine() can be applied to.
DrawRectangle	Draws a rectangle on the active page.
DrawSymbolSet	Places a Symbol Set object on the page at the specified location.
DrawText	Draws an alphanumeric string at the specified location.
DrawTrend	Draws a trend object on the active page.

For details and a VB example on handling return and error values, see Error Handling.

Attribute3dEffects

Applies a 3D effect to an object, or retrieves the current 3D effect setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

Attribute3dEffects(*Effects*)

Effects:

A value between 0 and 4 representing the 3D effect type.

- 0 = none
- 1 = raised

- 2 = lowered
- 3 = shadowed
- 4 = embossed

Return Value

If retrieving the current 3D effect setting, a value between 0 and 4 representing the effect type. If applying a 3D effect, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report with the error E_INVALIDARG. If there is no active object, they will exit with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffectDepth, AttributeShadowColour, AttributeHiLightColour, AttributeLoLightColour

Example

```
' Applies a 3D effect (embossed) to an object
GraphicsBuilder.Attribute3dEffects = 4

' Retrieves the current 3D effect applied to an object
MyVariable = GraphicsBuilder.Attribute3dEffects
```

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_Attribute3dEffect` applies a 3D effect, and `get_Attribute3dEffect` retrieves the current 3D effect setting.

Attribute3dEffectDepth

Applies a level of depth to a 3D effect, or retrieves the current depth setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

Attribute3dEffectDepth(*EffectDepth*)

EffectDepth:

A value between 0 and 32 representing the depth of the 3D effect used.

Return Value

If retrieving the current depth setting for a 3D effect, a value between 0 and 32. If applying depth to a 3D effect, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, AttributeShadowColour, AttributeHiLightColour, AttributeLoLightColour

Example

```
' Applies depth to a 3D effect for the current object
GraphicsBuilder.Attribute3dEffectDepth = 28

' Retrieves the 3D depth for the current object
MyVariable = GraphicsBuilder.Attribute3dEffectDepth
```

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_Attribute3dEffectDepth` applies depth to 3D effect, and `get_Attribute3dEffectDepth` retrieves the current 3D depth setting.

AttributeAN

Retrieves the animation number (AN) of the active object. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeAN(*AN*)

AN:

A value between 0 and 65536.

Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeX, AttributeY

Example

```
' Retrieves the AN for the current object
MyVariable = GraphicsBuilder.AttributeAN
```

AttributeBaseCoordinates

Returns the base coordinates of an object. If you use these coordinates, also apply the transformation matrix. Refer to functions AttributeTransformationMatrixPut and AttributeTransformationMatrixGet.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeBaseCoordinates(*FromXPosition, FromYPosition, ToXPosition, ToYPosition*)

FromXPosition:

Distance from the left hand side of the page to top left hand corner of the object, measured in pixels.

FromYPosition:

Distance from the top of the page to the top left hand corner of the object, measured in pixels.

ToXPosition:

Distance from the left hand side of the page to the bottom right hand corner of the object, measured in pixels.

ToYPosition:

Distance from the top of the page to the bottom right hand corner of the object, measured in pixels.

Return Value

The base coordinates of the current object. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeTransformationMatrixPut, AttributeTransformationMatrixGet

AttributeClass

Retrieves the class of the active object as a string. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeClass(*Class*)

Class:

A string depicting the class of the object. The class options include: "Draw", "Line", "Square", "Circle", "Polyline", "Pipe", "Text", "Button", "Set", "Trend", "Advanced Animation", "Bitmap", "Group", "ActiveX", "Symbol" and "Genie".

Return Value

A string depicting the class of the object. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
' Retrieves the Class for the current object
MyVariable = GraphicsBuilder.AttributeClass
```

AttributeCornerRadius

Sets or retrieves the corner radius value on the General | Appearance tab of the Object Properties dialog for the current object. This is only supported on rectangle objects.

Syntax

AttributeCornerRadius(*nRadius*)

nRadius:

Defines the radius of the corner. Values from 0- 32 pixels are permitted.

Return Value

If retrieving the current corner radius, a value between 0 and 32. If applying a corner radius, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit with a return value of E_HANDLE.

AttributeEllipseStyle

Applies a style to an ellipse, or retrieves the current ellipse style setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeEllipseStyle(*EllipseStyle*)

EllipseStyle:

A value representing the current ellipse style.

- 0 = normal ellipse
- 1 = pie slice
- 2 = arc

Return Value

If retrieving the current ellipse style setting, a value between 0 and 2 representing one of three style options. If applying a style setting, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit with a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Example

```
' Applies a style (arc) to an ellipse
GraphicsBuilder.AttributeEllipseStyle = 2

' Retrieves a value representing the style applied to an ellipse
MyVariable = GraphicsBuilder.AttributeEllipseStyle
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeEllipseStyle` applies a style to an ellipse, and `get_AttributeEllipseStyle` retrieves the current ellipse style setting.

AttributeEndAngle

Sets the end angle of an arc or pie-slice, or retrieves the end angle.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeEndAngle(*Angle*)

Angle:

A value between 0 and 360 representing the end angle (in degrees).

Return Value

If retrieving the end angle, a value between 0 and 360. If applying an end angle, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeStartAngle

Example

```
' Sets the end angle of an arc
GraphicsBuilder.AttributeEndAngle = 45

' Retrieves the start angle for an arc
MyVariable = GraphicsBuilder.AttributeEndAngle
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeEndAngle` applies an end angle setting, and `get_AttributeEndAngle` retrieves the current end angle setting.

AttributeExtentX

Retrieves the X coordinate that represents the extent of the active object. For example, if the active object were a line, it would be the end coordinate. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeExtentX(*XPosition*)

XPosition:

A value between 0 and 65536.

Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeExtentY, AttributeAN

Example

```
' Retrieves the X coordinate for the extent of the current object
MyVariable = GraphicsBuilder.AttributeExtentX
```

AttributeExtentY

Retrieves the Y coordinate that represents the extent of the active object. For example, if the active object were a line, it would be the end coordinate. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeExtentY(*YPosition*)

YPosition:

A value between 0 and 65536

Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeExtentX, AttributeAN

Example

```
' Retrieves the Y coordinate for the extent of the current object
MyVariable = GraphicsBuilder.AttributeExtentY
```

AttributeFillColour

Sets the fill color for an object, or retrieves a value representing the current fill color.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

> **Note:** As this function does not support True Color functionality, it has been superseded by the functions AttributeFillOffColourEx and AttributeFillOnColourEx.

Syntax

AttributeFillColour(*FillColour*)

FillColour:

A value between 0 and 255 representing the fill color.

Return Value

If retrieving the current fill color, a value between 0 and 255. If applying a fill color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeSetFill

Example

```
' Sets the fill color for an object
GraphicsBuilder.AttributeFillColour = 125

' Retrieves the value of the fill color
MyVariable = GraphicsBuilder.AttributeFillColour
```

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_AttributeFillColour` applies a fill color, and `get_AttributeFillColour` retrieves the current fill color setting.

AttributeFillOffColourEx

Sets the fill color for an object, or retrieves a value representing the current fill color.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects, and change or read their properties.

Syntax

AttributeFillOffColourEx(*FillColour*)

FillColour:

An RGB value.

Return Value

If retrieving the current fill color, an RGB value. If applying a fill color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeSetFill

Example

```
' Sets the fill color for an object
GraphicsBuilder.AttributeFillOffColourEx = &hFF0000

' Retrieves the value of the fill color
MyVariable = GraphicsBuilder.AttributeFillOffColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeFillOffColourEx` applies a fill color, and `get_AttributeFillOffColourEx` retrieves the current fill color setting.

AttributeFillOnColourEx

Sets the fill color for an object, or retrieves a value representing the current fill color.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeFillOnColourEx(*FillColour*)

FillColour:

An RGB value.

Return Value

If retrieving the current fill color, an RGB value. If applying a fill color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeSetFill

Example

```
' Sets the fill color for an object
GraphicsBuilder.AttributeFillOnColourEx = &hFF0000

' Retrieves the value of the fill color
MyVariable = GraphicsBuilder.AttributeFillOnColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeFillOnColourEx` applies a fill color, and `get_AttributeFillOnColourEx` retrieves the current fill color setting.

AttributeGradientMode

Sets or retrieves the direction of the gradient on the **General| Appearance** tab of the Object Properties dialog for the current object.

This function is only supported on rectangle objects.

Syntax

AttributeGradientMode(*Mode*)

Mode:

Direction of the gradient:

0 - Off
1 - Left To Right
2 - Right To Left
3 - Top To Bottom
4 - Bottom To Top
5 - Horizontal Edge To Middle
6 - Middle To Horizontal Edge
7 - Vertical Edge To Middle
8 - Middle To Vertical Edge

Return Value

If retrieving the gradient mode, the direction of the gradient as specified above. If applying a gradient mode, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

AttributeGradientOffColour

Sets or retrieves the "Off" portion of the gradient color on the **General| Appearance** tab of the Object Properties dialog for the current object.

This is only supported on rectangle objects.

Syntax

AttributeGradientOffColour(*Color*)

Color:

Off portion of the gradient color.

Return Value

If retrieving the gradient color, an RGB encoded color. If applying a gradient color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

AttributeGradientOnColour

Sets or retrieves the "On" portion of the gradient color on the **General | Appearance** tab of the Object Properties dialog for the current object.

This function is only supported on rectangle objects.

Syntax

AttributeGradientOnColour(*Color*)

Color:

On portion of the gradient color.

Return Value

If retrieving the gradient color, an RGB encoded color. If applying a gradient color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

AttributeHiLightColour

Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access objects and change or read their properties.

> **Note:** As this function does not support True Color functionality, it has been superseded by the functions AttributeHiLightOnColourEx and AttributeHiLightOffColourEx.

Syntax

AttributeHiLightColour(*HiLightColour*)

HiLightColour:

A value between 0 and 255 representing the highlight color.

Return Value

If retrieving the current highlight color setting, a value between 0 and 255. If applying a highlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeShadowColour, AttributeLoLightColour

Example

```
' Applies a highlight color to a 3D effect
GraphicsBuilder.AttributeHiLightColour = 125

' Retrieves a value representing a 3D effect's highlight color
MyVariable = GraphicsBuilder.AttributeHiLightColour
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeHiLightColour` applies a highlight color setting, and `get_AttributeHiLightColour` retrieves the current highlight color setting.

AttributeHiLightOffColourEx

Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects, and change or read their properties.

Syntax

AttributeHiLightOffColourEx(*HiLightColour*)

HiLightColour:

An RGB value.

Return Value

If retrieving the current highlight color setting, an RGB value. If applying a highlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeShadowOffColourEx, AttributeShadowOnColourEx, AttributeLoLightOffColourEx, AttributeLoLightOnColourEx

Example

```
' Applies a highlight color to a 3D effect
GraphicsBuilder.AttributeHiLightOffColourEx = &hFF0000

' Retrieves a value representing a 3D effect's highlight color
MyVariable = GraphicsBuilder.AttributeHiLightOffColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeHiLightOffColourEx` applies a highlight color setting, and `get_AttributeHiLightOffColourEx` retrieves the current highlight color setting.

AttributeHiLightOnColourEx

Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects, and change or read their properties.

Syntax

AttributeHiLightOnColourEx(*HiLightColour*)

HiLightColour:

An RGB value.

Return Value

If retrieving the current highlight color setting, an RGB value. If applying a highlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeShadowOffColourEx, AttributeShadowOnColourEx, AttributeLoLightOffColourEx, AttributeLoLightOnColourEx

Example

```
' Applies a highlight color to a 3D effect
GraphicsBuilder.AttributeHiLightOnColourEx = &hFF0000

' Retrieves a value representing a 3D effect's highlight color
MyVariable = GraphicsBuilder.AttributeHiLightOnColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeHiLightOnColourEx` applies a highlight color setting, and `get_AttributeHiLightOnColourEx` retrieves the current highlight color setting.

AttributeLineColour

Applies a color to a line, or retrieves the current color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

> **Note:** This function, as it does not support True Color functionality, has been superseded by the functions AttributeLineOnColourEx and AttributeLineOffColourEx.

Syntax

AttributeLineColour(*LineColour*)

LineColour:

A value between 0 and 255 representing a particular color.

Return Value

If retrieving the current line color, a value between 0 and 255. If setting the line color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeLineWidth, AttributeLineStyle

Example

```
' Applies a color to the current line
GraphicsBuilder.AttributeLineColour = 125

' Retrieves the value of the color applied to the current line
MyVariable = GraphicsBuilder.AttributeLineColour
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeLineColour` applies a particular color to a line, and `get_AttributeLineColour` retrieves the current color setting.

AttributeLineOffColourEx

Applies a color to a line, or retrieves the current "off" color setting. The function uses RGB colors for each state of a color instead of a palette index.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects, and change or read their properties.

Syntax

AttributeLineOffColourEx(*LineColour*)

LineColour:

An RGB value.

Return Value

If retrieving the current line color, an RGB value. If setting the line color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeLineWidth, AttributeLineStyle

Example

```
' Applies a color to the current line
GraphicsBuilder.AttributeLineOffColourEx = &hFF0000

' Retrieves the value of the color applied to the current line
MyVariable = GraphicsBuilder.AttributeLineOffColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeLineOffColourEx` applies a particular color to a line, and `get_AttributeLineOffColourEx` retrieves the current color setting.

AttributeLineOnColourEx

Applies a color to a line, or retrieves the current "on" color setting. The function uses RGB colors for each state of a color instead of a palette index.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects, and change or read their properties.

Syntax

AttributeLineOnColourEx(*LineColour*)

LineColour:

An RGB value.

Return Value

If retrieving the current line color, an RGB value. If setting the line color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeLineWidth, AttributeLineStyle

Example

```
' Applies a color to the current line
```

```
GraphicsBuilder.AttributeLineOnColourEx = &hFF0000

' Retrieves the value of the color applied to the current line
MyVariable = GraphicsBuilder.AttributeLineOnColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeLineOnColourEx` applies a particular color to a line, and `get_AttributeLineOnColourEx` retrieves the current color setting.

AttributeLineStyle

Applies a style to a line, or retrieves the current style setting. You can only apply a line style if the line width is set to 1.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeLineStyle(*LineStyle*)

LineStyle:

A value between 0 and 4 representing the style applied to a line. Line style only works if line width is set to 1.

- 0 = solid
- 1 = dashed
- 2 = dot
- 3 = dash dot
- 4 = dash dot dot

Return Value

If retrieving the current line style, a value between 0 and 4 that represents a particular style. If setting the line style, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeLineWidth, AttributeLineColour

Example

```
' Applies a style (dash dot) to the current line
GraphicsBuilder.AttributeLineStyle = 3

' Retrieves the style applied to the current line
MyVariable = GraphicsBuilder.AttributeLineStyle
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeLineStyle` applies a particular line style, and `get_AttributeLineStyle` retrieves the current style setting.

AttributeLineWidth

Sets the width of a line, or retrieves its current width.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeLineWidth(*LineWidth*)

LineWidth:

A value between 0 and 32 representing the line width in pixels.

Return Value

If retrieving the current width of a line, a value between 1 and 32 (representing pixels) is returned. If setting the line width, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeLineStyle, AttributeLoLightColour

Example

```
' Sets the width for the current line
GraphicsBuilder.AttributeLineWidth = 1

' Retrieves the width of the current line
MyVariable = GraphicsBuilder.AttributeLineWidth
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeLineWidth` sets the value for the width of a line, and `get_AttributeLineWidth` retrieves the current line width setting.

AttributeLoLightColour

Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeLoLightColour(*LoLightColour*)

LoLightColour:

A value between 0 and 255 representing the lowlight color.

Return Value

If retrieving the current lowlight color setting, a value between 0 and 255. If applying a lowlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeShadowColour, AttributeHiLightColour

Example

```
' Applies a lowlight color to a 3D effect
GraphicsBuilder.AttributeLoLightColour = 45

' Retrieves a value representing a 3D effect's lowlight color
MyVariable = GraphicsBuilder.AttributeLoLightColour
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeLoLightColour` applies a lowlight color setting, and `get_AttributeLoLightColour` retrieves the current lowlight color setting.

AttributeLoLightOffColourEx

Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeLoLightOffColourEx(*LoLightColour*)

LoLightColour:

An RGB value.

Return Value

If retrieving the current lowlight color setting, an RGB value. If applying a lowlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeShadowOffColourEx, AttributeShadowOnColourEx, AttributeHiLightOffColourEx, AttributeHiLightOnColourEx

Example

```
' Applies a lowlight color to a 3D effect
GraphicsBuilder.AttributeLoLightOffColourEx = &hFF0000

' Retrieves a value representing a 3D effect's lowlight color
MyVariable = GraphicsBuilder.AttributeLoLightOffColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeLoLightOffColourEx` applies a lowlight color setting, and `get_AttributeLoLightOffColourEx` retrieves the current lowlight color setting.

AttributeLoLightOnColourEx

Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeLoLightOnColourEx(*LoLightColour*)

LoLightColour:

An RGB value.

Return Value

If retrieving the current lowlight color setting, an RGB value. If applying a lowlight color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeShadowOffColourEx,AttributeShadowOnColourEx, AttributeHiLightOffColourEx, AttributeHiLightOnColourEx

Example

```
' Applies a lowlight color to a 3D effect
GraphicsBuilder.AttributeLoLightOnColourEx = &hFF0000

' Retrieves a value representing a 3D effect's lowlight color
MyVariable = GraphicsBuilder.AttributeLoLightOnColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeLoLightOnColourEx` applies a lowlight color setting, and `get_AttributeLoLightOnColourEx` retrieves the current lowlight color setting.

AttributeNodeCoordinatesFirst

Returns the coordinates of the first node of a free hand line, polygon or pipe.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeNodeCoordinatesFirst(*XPosition, YPosition*)

XPosition:

Distance from the left-hand side of the page to the first node of an object, measured in pixels.

YPosition:

Distance from the top of the page to the first node of an object, measured in pixels.

Return Value

The coordinates of the current object's first node. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeNodeCoordinatesNext

AttributeNodeCoordinatesNext

Returns the coordinates of any following nodes of a free hand line, polygon or pipe when implemented after AttributeNodeCoordinatesFirst.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeNodeCoordinatesNext(*XPosition, YPosition*)

XPosition:

Distance from the left-hand side of the page to the first node of an object, measured in pixels.

YPosition:

Distance from the top of the page to the first node of an object, measured in pixels.

The coordinates of the object's following nodes, or E_ABORT if no more nodes are left. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeNodeCoordinatesFirst

AttributePolygonOpen

Defines whether a polygon (polyline) is set to open mode (that is, its two end points are not joined) or closed (its two ends are joined). It can also be used to retrieve the current open mode setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributePolygonOpen(*OpenClose*)

OpenClose:

TRUE = Polygon is drawn in open mode; FALSE = Polygon is drawn in closed mode.

Return Value

If retrieving the current open mode setting for a polygon, TRUE or FALSE is returned. If setting the open mode, 0 (zero) is returned if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Example

```
' Sets a polygon to Open mode
GraphicsBuilder.AttributePolygonOpen = TRUE

' Determines if the current polygon is defined as Open
MyVariable = GraphicsBuilder.AttributePolygonOpen
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributePolygonOpen` sets the open mode for a polygon, and `get_AttributePolygonOpen` retrieves the current open mode setting.

AttributeRectangleStyle

Sets the rectangle style, or retrieves the rectangle style setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeRectangleStyle(*Style*)

Style:

0 = none

1 = border

2 = extra line

3 = border and an extra line

Return Value

If retrieving the current rectangle style setting, a value between 0 and 3. If applying a rectangle style, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeShadowColour, AttributeHiLightColour

Example

```
' Applies a style to a rectangle
GraphicsBuilder.AttributeRectangleStyle = 1

' Retrieves a value representing a rectangle style
MyVariable = GraphicsBuilder.AttributeRectangleStyle
```

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_AttributeRectangleStyle` applies a rectangle style, and `get_AttributeRectangleStyle` retrieves the current rectangle style setting.

AttributeSetFill

Displays the object as filled, or retrieves the current fill value.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeSetFill(*SetFill*)

SetFill:

TRUE if the object drawn filled.

Return Value

If retrieving the current fill setting, TRUE if the object is displayed as filled. If applying a fill setting, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeFillColour

Example

```
' Displays an object as filled
GraphicsBuilder.AttributeSetFill = TRUE

' Retrieves the current fill setting
MyVariable = GraphicsBuilder.AttributeSetFill
```

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_AttributeSetFill` applies a fill setting, and `get_AttributeSetFill` retrieves the current fill setting.

AttributeShadowColour

Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

> **Note:** As this function does not support True Color functionality, it has been superseded by the functions AttributeShadowOffColourEx and AttributeShadowOnColourEx.

Syntax

AttributeShadowColour(ShadowColour)

ShadowColour:

A value between 0 and 255 representing the shadow color.

Return Value

If retrieving the current shadow color setting, a value between 0 and 255. If applying a shadow color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeHiLightColour, AttributeLoLightColour

Example

```
' Applies a shadow color to a shadowed 3D effect
GraphicsBuilder.AttributeShadowColour = 125

' Retrieves the current shadow color
MyVariable = GraphicsBuilder.AttributeShadowColour
```

> **Note:** This function is implemented in the C++ environment as two separate functions: put_AttributeShadowColour applies a shadow color setting, and `get_AttributeShadowColour` retrieves the current shadow color setting.

AttributeShadowOffColourEx

Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeShadowOffColourEx(ShadowColour)

ShadowColour:

An RGB value.

Return Value

If retrieving the current shadow color setting, an RGB value. If applying a shadow color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active object, they exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeHiLightOffColourEx, AttributeHiLightOnColourEx, AttributeLoLightOffColourEx, AttributeLoLightOnColourEx

Example

```
' Applies a shadow color to a shadowed 3D effect
GraphicsBuilder.AttributeShadowOffColourEx = &hFF0000

' Retrieves the current shadow color
MyVariable = GraphicsBuilder.AttributeShadowOffColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeShadowOffColourEx` applies a shadow color setting, and `get_AttributeShadowOffColourEx` retrieves the current shadow color setting.

AttributeShadowOnColourEx

Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects, and change or read their properties.

Syntax

AttributeShadowOnColourEx(ShadowColour)

ShadowColour:

An RGB value.

Return Value

If retrieving the current shadow color setting, an RGB value. If applying a shadow color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

Attribute3dEffects, Attribute3dEffectDepth, AttributeHiLightOffColourEx, AttributeHiLightOnColourEx, AttributeLoLightOffColourEx, AttributeLoLightOnColourEx

Example

```
' Applies a shadow color to a shadowed 3D effect
GraphicsBuilder.AttributeShadowOnColourEx = &hFF0000

' Retrieves the current shadow color
MyVariable = GraphicsBuilder.AttributeShadowOnColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeShadowOnColourEx` applies a shadow color setting, and `get_AttributeShadowOnColourEx` retrieves the current shadow color setting.

AttributeStartAngle

Sets the start angle of an arc or pie-slice, or retrieves the start angle.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeStartAngle(*Angle*)

Angle:

A value between 0 and 360 representing the start angle (in degrees).

If retrieving the start angle, a value between 0 and 360. If applying a start angle, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeEndAngle

Example

```
' Sets the start angle of an arc
GraphicsBuilder.AttributeStartAngle = 45

' Retrieves the start angle for an arc
MyVariable = GraphicsBuilder.AttributeStartAngle
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeStartAngle` applies a start angle setting, and `get_AttributeStartAngle` retrieves the current start angle setting.

AttributeTransformationMatrixGet

Reads the elements of the transformation matrix. If A and D are both 1, and others are 0, the object is not transformed (identity matrix).

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeTransformationMatrixGet(*A, B, C, D, H, K*)

A:

Element A of the transformation matrix

B:

Element A of the transformation matrix

C:

Element A of the transformation matrix

D:

Element A of the transformation matrix

H:

Element A of the transformation matrix

K:

Element A of the transformation matrix

Return Value

The elements of the transformation matrix. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeTransformationMatrixPut

AttributeTransformationMatrixPut

Sets the elements of the transformation matrix.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeTransformationMatrixPut(*A, B, C, D, H, K*)

A:

Element A of the transformation matrix

B:

Element A of the transformation matrix

C:

Element A of the transformation matrix

D:

Element A of the transformation matrix

H:

Element A of the transformation matrix

K:

Element A of the transformation matrix

Return Value

0 (zero) if successful, otherwise an error is returned. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeTransformationMatrixGet

AttributeX

Retrieves the X coordinate of the active object. This is a read only function.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeX(*XPosition*)

XPosition:

A value between 0 and 65536.

Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeY, AttributeAN

Example

```
' Retrieves the X coordinate for the current object
MyVariable = GraphicsBuilder.AttributeX
```

AttributeY

Retrieves the Y coordinate of the active object. This is a read-only function.

This function applies to the selected object, which is typically the last placed object. By using the `PageSelectFirstObject()` and `PageSelectNextObject()` functions, you can access your objects and change or read their properties.

Syntax

AttributeY(*YPosition*)

YPosition:

A value between 0 and 65536.

Return Value

A value between 0 and 65536. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active object, they will exit and report a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeX, AttributeAN

Example

```
' Retrieves the Y coordinate for the current object
MyVariable = GraphicsBuilder.AttributeY
```

DrawButton

Draws a button on the active page.

Syntax

DrawButton(*FromXPosition, FromYPosition, ToXPosition, ToYPosition*)

FromXPosition:

Distance from the left hand side of the page to top left hand corner of the button to be drawn, measured in pixels.

FromYPosition:

Distance from the top of the page to the top left hand corner of the button to be drawn, measured in pixels.

ToXPosition:

Distance from the left hand side of the page to the bottom right hand corner of the button to be drawn, measured in pixels.

ToYPosition:

Distance from the top of the page to the bottom right hand corner of the button to be drawn, measured in pixels.

Return Value

0 (zero) if successful; otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

DrawLine, DrawEllipse, DrawRectangle, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawText, DrawNumber, DrawSymbolSet, DrawTrend, DrawCicodeObject

Example

```
GraphicsBuilder.DrawButton 50, 70, 400, 200
```

DrawCicodeObject

Places a Cicode object on the page at the specified location.

Syntax

DrawCicodeObject(*XPosition, YPosition*)

XPosition:

Distance in pixels from the left of the page to the point where you would like the Cicode object to be placed.

YPosition:

Distance in pixels from the top of the page to the point where you would like the Cicode object to be placed.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawButton, DrawLine, DrawEllipse, DrawRectangle, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawText, DrawNumber, DrawSymbolSet, DrawTrend

Example

```
GraphicsBuilder.DrawCicodeObject 500, 100
```

DrawEllipse

Draws an ellipse on the active page.

Syntax

DrawEllipse(*FromXPosition, FromYPosition, ToXPosition, ToYPosition*)

FromXPosition:

Distance in pixels from the left hand side of the page to top left hand corner of the rectangle that will enclose the ellipse.

FromYPosition:

Distance in pixels from the top of the page to the top left hand corner of the rectangle that will enclose the ellipse.

ToXPosition:

Distance in pixels from the left hand side of the page to the bottom right hand corner of the rectangle that will enclose the ellipse.

ToYPosition:

Distance in pixels from the top of the page to the bottom-right hand corner of the rectangle that will enclose the ellipse.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawButton, DrawLine, DrawRectangle, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawText, DrawNumber, DrawSymbolSet, DrawTrend, DrawCicodeObject

Example

```
GraphicsBuilder.DrawEllipse 50, 70, 400, 200
```

DrawLine

Draws a line on the active page.

Syntax

DrawLine(*FromXPosition, FromYPosition, ToXPosition, ToYPosition*)

FromXPosition:

Distance in pixels from the left-hand side of the page to top left-hand corner of the rectangle that will enclose the ellipse.

FromYPosition:

Distance in pixels from the top of the page to the top-left hand corner of the rectangle that will enclose the ellipse.

ToXPosition:

Distance in pixels from the left-hand side of the page to the bottom right-hand corner of the rectangle that will enclose the ellipse.

ToYPosition:

Distance in pixels from the top of the page to the bottom-right hand corner of the rectangle that will enclose the ellipse.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawButton, DrawEllipse, DrawRectangle, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawText, DrawNumber, DrawSymbolSet, DrawTrend, DrawCicodeObject

Example

```
GraphicsBuilder.DrawLine 50, 70, 400, 70
```

DrawNumber

Places a number object on the page at the specified location.

Syntax

DrawNumber(*XPosition, YPosition*)

XPosition:

Distance in pixels from the left of the page to the point where you would like the number object to be placed.

YPosition:

Distance in pixels from the top of the page to the point where you would like the number object to be placed.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawLine, DrawEllipse, DrawRectangle, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawText, DrawButton, DrawSymbolSet, DrawTrend, DrawCicodeObject

Example

```
GraphicsBuilder.DrawNumber 500, 100
```

DrawPipeEnd

Terminates the drawing of a pipe on the active page. To work successfully, this function needs to follow an instance of DrawPipeSection.

Syntax

DrawPipeEnd

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawPipeSection, DrawPipeStart

Example

```
GraphicsBuilder.DrawPipeStart 50, 290
GraphicsBuilder.DrawPipeSection 200, 350
GraphicsBuilder.DrawPipeSection 350, 250
GraphicsBuilder.DrawPipeSection 400, 350
GraphicsBuilder.DrawPipeEnd
```

DrawPipeSection

Draws a section of pipe on the active page. To work successfully, this function needs to have a starting point defined by the function DrawPipeStart, or it needs to follow a previous incidence of itself.

Syntax

DrawPipeSection(*XPosition, YPosition*)

XPosition:

Distance in pixels from the left of the page to the point where you would like the section of pipe to end.

YPosition:

Distance in pixels from the top of the page to the point where you would like the section of pipe to end.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawPipeStart, DrawPipeEnd

Example

```
GraphicsBuilder.DrawPipeStart 50, 290
GraphicsBuilder.DrawPipeSection 200, 350
GraphicsBuilder.DrawPipeSection 350, 250
GraphicsBuilder.DrawPipeSection 400, 350
GraphicsBuilder.DrawPipeEnd
```

DrawPipeStart

Initiates the process of drawing a pipe on the active page by defining a starting point that DrawPipeSection can be applied to.

Syntax

DrawPipeStart(*XPosition, YPosition*)

XPosition:

Distance in pixels from the left of the page to the point where you would like the section of pipe to start.

YPosition:

Distance in pixels from the top of the page to the point where you would like the section of pipe to start.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawPipeSection, DrawPipeEnd

Example

```
GraphicsBuilder.DrawPipeStart 50, 290
GraphicsBuilder.DrawPipeSection 200, 350
GraphicsBuilder.DrawPipeSection 350, 250
GraphicsBuilder.DrawPipeSection 400, 350
GraphicsBuilder.DrawPipeEnd
```

DrawPolygonEnd

Terminates the drawing of a polygon on the active page. To work successfully, this function needs to follow an instance of DrawPolygonLine.

Syntax

DrawPolygonEnd

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawPolygonLine, DrawPolygonStart

Example

```
GraphicsBuilder.DrawPolygonStart 50, 290
```

```
GraphicsBuilder.DrawPolygonLine 200, 350
GraphicsBuilder.DrawPolygonLine 350, 250
GraphicsBuilder.DrawPolygonLine 400, 350
GraphicsBuilder.DrawPolygonEnd
```

DrawPolygonLine

Draws a line on the active page that forms part of a polygon. To work successfully, this function needs to have a starting point defined by the function DrawPolygonStart or a previous incidence of itself.

Syntax

DrawPolygonLine(*XPosition, YPosition*)

XPosition:

Distance in pixels from the left of the page to the point where you would like the line to end.

YPosition:

Distance in pixels from the top of the page to the point where you would like the line to end.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

DrawPolygonStart, DrawPolygonEnd

Example

```
GraphicsBuilder.DrawPolygonStart 50, 290
GraphicsBuilder.DrawPolygonLine 200, 350
GraphicsBuilder.DrawPolygonLine 350, 250
GraphicsBuilder.DrawPolygonLine 400, 350
GraphicsBuilder.DrawPolygonEnd
```

DrawPolygonStart

Initiates the process of drawing a polygon on the active page by defining a starting point that DrawPolygonLine can be applied to.

Syntax

DrawPolygonStart(*XPosition, YPosition*)

XPosition:

Distance in pixels from the left of the page to the point where you would like the start the polygon.

YPosition:

Distance in pixels from the top of the page to the point where you would like the start the polygon.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawPolygonLine, DrawPolygonEnd

Example

```
GraphicsBuilder.DrawPolygonStart 50, 290
GraphicsBuilder.DrawPolygonLine 200, 350
GraphicsBuilder.DrawPolygonLine 350, 250
GraphicsBuilder.DrawPolygonLine 400, 350
GraphicsBuilder.DrawPolygonEnd
```

DrawRectangle

Draws a rectangle on the active page.

Syntax

DrawRectangle(*FromXPosition, FromYPosition, ToXPosition, ToYPosition*)

FromXPosition:

Distance from the left hand side of the page to top left hand corner of the rectangle to be drawn, measured in pixels.

FromYPosition:

Distance from the top of the page to the top left hand corner of the rectangle to be drawn, measured in pixels.

ToXPosition:

Distance from the left hand side of the page to the bottom right hand corner of the rectangle to be drawn, measured in pixels.

ToYPosition:

Distance from the top of the page to the bottom right hand corner of the rectangle to be drawn, measured in pixels.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawLine, DrawEllipse, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawText, DrawNumber, DrawButton, DrawSymbolSet, DrawTrend, DrawCicodeObject

Example

```
GraphicsBuilder.DrawRectangle 50, 70, 400, 200
```

DrawSymbolSet

Places a symbol set object on the page at the specified location.

Syntax

DrawSymbolSet(*XPosition, YPosition*)

XPosition:

Distance in pixels from the left of the page to the point where you would like the object to be placed.

YPosition:

Distance in pixels from the top of the page to the point where you would like the object to be placed.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawLine, DrawEllipse, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawText, DrawNumber, DrawButton, DrawTrend, DrawCicodeObject, DrawRectangle

Example

```
GraphicsBuilder.DrawSymbolSet 500, 100
```

DrawText

Draws an alphanumeric string at the specified location.

Syntax

DrawText(*Text, XPosition, YPosition*)

Text:

The text to be pasted on to the active graphics page.

XPosition:

Distance in pixels from the left of the page to the point where you would like the text to be placed.

YPosition:

Distance in pixels from the top of the page to the point where you would like the text to be placed.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

DrawLine, DrawEllipse, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawNumber, DrawButton, DrawTrend, DrawCicodeObject, DrawRectangle, DrawSymbolSet

Example

```
GraphicsBuilder.DrawText "My Text", 500, 100
```

DrawTrend

Draws a trend object on the active page.

Syntax

DrawTrend(*FromXPosition, FromYPosition, ToXPosition, ToYPosition*)

FromXPosition:

Distance from the left hand side of the page to top left hand corner of the trend object, measured in pixels.

FromYPosition:

Distance from the top of the page to the top left hand corner of the trend object, measured in pixels.

ToXPosition:

Distance from the left hand side of the page to the bottom right hand corner of the trend object, measured in pixels.

ToYPosition:

Distance from the top of the page to the bottom right hand corner of the trend object, measured in pixels.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

DrawLine, DrawEllipse, DrawPolygonStart, DrawPolygonLine, DrawPolygonEnd, DrawPipeStart, DrawPipeSection, DrawPipeEnd, DrawNumber, DrawButton, DrawCicodeObject, DrawRectangle, DrawSymbolSet, DrawText

Example

```
GraphicsBuilder.DrawTrend 50, 70, 400, 200
```

Options Functions

These relate to the options found under the Graphics Builder's Tools menu. They do not throw an exception in Visual Basic.

Option-DisplayPropertiesOnNew	Simulates the option available on the Graphics Builder Tools menu that automatically shows the properties for a new object when inserted. You can use this function to set this option, or you can retrieve its current setting.
OptionSnapToGrid	Simulates the option available on the Graphics Builder Tools menu that automatically snaps objects to a grid. You can use this function to set this option, or you can retrieve its current setting.
OptionSnapToGuidelines	Simulates the option available on the Graphics Builder Tools menu that automatically snaps objects to guide-lines. You can use this function to set this option, or you can retrieve its current setting.

For details and a VB example on handling return and error values, see Automation Error Handling.

OptionDisplayPropertiesOnNew

Simulates the option available on the Graphics Builder Tools menu that automatically shows the properties for a new object when inserted. You can use this function to set this option, or you can retrieve its current setting.

Syntax

OptionDisplayPropertiesOnNew(*OptionValue*)

OptionValue:

TRUE activates the option, FALSE deactivates the option.

Return Value

If retrieving the current setting, TRUE or FALSE. If enabling or disabling the option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

Note: For details on handling return and error values, see Error Handling.

Related Functions

OptionSnapToGrid, OptionSnapToGuidelines

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_OptionDisplayPropertiesOnNew` enables or disables this option, and `get_OptionDisplayPropertiesOnNew` retrieves the current option setting.

OptionSnapToGrid

Simulates the option available on the Graphics Builder Tools menu that automatically snaps objects to a grid. You can use this function to set this option, or you can retrieve its current setting.

Syntax

OptionSnapToGrid(*OptionValue*)

OptionValue:

TRUE activates the option, FALSE deactivates the option.

Return Value

If retrieving the current setting, TRUE or FALSE. If enabling or disabling the option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

OptionDisplayPropertiesOnNew, OptionSnapToGuidelines

> **Note:** This function is implemented in the C++ environment as two separate functions: `put_OptionSnapToGrid` enables or disables this option, and `get_OptionSnapToGrid` retrieves the current option setting.

OptionSnapToGuidelines

Simulates the option available on the Graphics Builder Tools menu that automatically snaps objects to guidelines. You can use this function to set this option, or you can retrieve its current setting.

Syntax

OptionSnapToGuidelines(*OptionValue*)

OptionValue:

TRUE activates the option, FALSE deactivates the option.

Return Value

If retrieving the current setting, TRUE or FALSE. If enabling or disabling the option, 0 (zero) if successful. In both cases, an error is returned if unsuccessful.

Note: For details on handling return and error values, see Error Handling.

Related Functions

OptionDisplayPropertiesOnNew, OptionSnapToGrid

Note: This function is implemented in the C++ environment as two separate functions: `put_OptionSnapToGuidelines` enables or disables this option, and `get_OptionSnapToGuidelines` retrieves the current option setting.

Page Functions

Using the page functions, you can manipulate the pages in your project (for example, open, close, save, delete), and select objects on those pages. This includes templates, symbols, Genies, Super Genies.

PageActiveWindowHandle	Retrieves the window handle of the active page.
PageClose	Closes the current page. This function will also close an unsaved page.
PageConvertWindowCoordinates	Converts the raw window coordinates to page coordinates which account for the scroll bar position.
PageDelete	Deletes the specified page.
PageDeleteEx	Deletes a specified symbol, Genie or Supergenie from a library.

PageDeleteObject	Deletes the currently selected object.
PageDeleteTemplate	Deletes a specified graphics page template.
PageGroupSelectedObjects	Groups the currently selected objects.
PageImport	Imports a graphics file on to the page as a bitmap.
PageNew	Creates a new CitectSCADA graphics page.
PageNewEx	Creates a new symbol, Genie or Supergenie page.
PageNewLibrary	Creates a new library. Does not succeed, if project is read-only or not valid.
PageNewTemplate	Creates a new CitectSCADA graphics page template.
PageOpen	Opens an existing CitectSCADA graphics page.
PageOpenEx	Opens the specified symbol, Genie or Supergenie page, if it is found.
PageOpenTemplate	Opens a specified graphics page template.
PagePrint	Prints the current page.
PageUpdated	The event fired when a Graphics Builder page is updated.
PageSave	Saves the page using its current name.
PageSaveAs	Saves the page with a new name within a specified project.
PageSaveAsEx	Saves a symbol, Genie, Supergenie or template page to the specified location.
PageSelect	Select an opened page.
PageSelectFirst	Selects the first page currently open in the Graphics Builder. Does not succeed if there are no pages open.
PageSelectFirstObject	Selects the first object on the active page, based on its z-order number. This will not succeed if no object exists. Note that an object can also be a group object, in which case this function will iterate through the items of a group.

PageSelectFirstObjectEx	Selects the first object on the active page, based on its z-order number. This will not succeed if no object exists. This function will not iterate through the items of a group.
PageSelectFirstObjectInGenie	Select the first sub-object in the currently selected genie
PageSe-lectFirstObjectInGroup	Selects the first object in a group.
PageSelectNext	Selects the next page currently open in the Graphics Builder. Does not succeed if there are no pages open.
PageSelectNextObject	Selects the next object on the active page, based on its z-order number. This function will not succeed if no object exists. Note that an object can also be a group object, in which case this function will iterate through the items of a group.
PageSelectNextObjectEx	Selects the next object on the active page, based on its z-order number. This function will not succeed if no object exists. This function will not iterate through the items of a group.
PageSelectNextObjectInGenie	Select the next sub-object in the currently selected genie.
PageSe-lectNextObjectInGroup	Selects the next object in a group.
PageSelectObject	Selects an object using a specified AN number.
PageSelectObjectAdd	Selects an additional object using a specified AN number. This can be used to select multiple objects for a succeeding PageGroupSelectedObjects operation.
PageTemplateSelectFirstObject	Restart the template enumeration sequence.
PageTemplateSelectNextObject	Select the next object in the template.
PageThumbnailToClipboard	Creates a thumbnail image of the current page and copies it to the clipboard.
PageUngroupSelectedObject	Ungroups the currently selected grouped object.

For details and a VB example on handling return and error values, see Automation Error Handling.

PageActiveWindowHandle

Retrieves the window handle of the active page.

Syntax

PageActiveWindowHandle(*WindowHandle*)

WindowHandle:

The active window handle. The handle is NULL if there is no active window. In VB, this is the return value.

Return Value

The active window handle. The handle is NULL if there is no active window.

Note: For details on handling return and error values, see Error Handling.

Example

```
' Retrieves the window handle of the active page
MyVariable = GraphicsBuilder.PageActiveWindowHandle
```

PageClose

Closes the current page. This function will also close an unsaved page.

Syntax

PageClose

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageNew, PageOpen, PageSaveAs, PageSave

Example

```
' Closes the current CitectSCADA graphics page
GraphicsBuilder.PageClose
```

PageConvertWindowCoordinates

Converts the raw window coordinates to page coordinates which account for the scroll bar position.

Syntax

PageConvertWindowCoordinates(*XPosition, YPosition*)

XPosition:

The raw window X coordinate as input. The page X coordinates as output.

YPosition:

The raw window Y coordinate as input. The page Y coordinate as output.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

PageDelete

Deletes the specified page.

Syntax

PageDelete(*Project, Page, Flag*)

Project:

The name of the project where the page can be found.

Page:

The name of the page to be deleted.

Flag:

If the flag is set, associated records are deleted.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageNew, PageOpen, PageSaveAs, PageSave

Example

```
' Deletes the current CitectSCADA graphics page
GraphicsBuilder.PageDelete "Example", "TestPage", True
```

PageDeleteEx

Deletes a specified symbol, Genie or Supergenie from a library.

Syntax

PageDeleteEx(*Project*, *Library*, *Element*, *PageType*)

Project:

The name of the project where the element can be found.

Library:

The name of the library where the element can be found.

Element:

Name of the symbol, Genie or Supergenie.

PageType:

- 0 = Symbol
- 1 = Genie
- 2 = Supergenie

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageNew, PageOpen, PageSaveAs, PageSave

Example

```
' Deletes the specified symbol
GraphicsBuilder.PageDeleteEx "Example", "TestLibrary", "TestObject", 0
' Deletes the specified Genie
GraphicsBuilder.PageDeleteEx "Example", "TestLibrary", "TestObject", 1
' Deletes the specified Supergenie
GraphicsBuilder.PageDeleteEx "Example", "TestLibrary", "TestObject", 2
```

PageDeleteObject

Deletes the currently selected object.

Syntax

PageDeleteObject

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObject, PageSelectFirstObject, PageSelectNextObject

PageDeleteTemplate

Deletes a specified graphics page template.

Syntax

PageDeleteTemplate(*Project, Style, Template, Resolution, Titlebar*)

Project:

The name of the project that contains the template.

Style:

The style of the template you would like to delete.

Template:

The name of the template you would like to delete.

Resolution:

The resolution of the template.

- 0 = Default
- 1 = VGA
- 2 = SVGA
- 3 = XGA
- 4 = SXGA
- 5 = User

Titlebar:

Set to TRUE to select a titlebar.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageSave, PageSaveAs, PageClose

Example

```
' Deletes a graphics page template
GraphicsBuilder.PageDeleteTemplate "include", "standard", "blank", 2, True
```

PageGroupSelectedObjects

Groups the currently selected objects.

Syntax

PageGroupSelectedObjects

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObject, PageSelectObjectAdd, PageSelectFirstObject, PageSelectNextObject, PageUngroupSelectedObject

PageImport

Imports a graphics file on to the page as a bitmap.

Syntax

PageImport(*FileName*)

FileName:

The name of the graphic file, including the complete path.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Example

```
' Imports the graphic file splash.bmp as a bitmap
GraphicsBuilder.PageImport "C:Program Files\Citect\CitectSCADA 7.10\Bin\splash.bmp"
```

PageNew

Creates a new CitectSCADA graphics page.

Syntax

PageNew(*Project, Style, Template, Resolution, Titlebar, Linked*)

Project:

The name of the project that contains the template you would like to apply to the page.

Style:

The style you would like to apply to your new CitectSCADA graphics page. CitectSCADA templates are grouped into styles.

Template:

Specifies the template you would like to apply to your new CitectSCADA graphics page.

Resolution:

Sets the appropriate resolution for the page being created.

- 0 = Default
- 1 = VGA
- 2 = SVGA
- 3 = XGA
- 4 = SXGA
- 5 = User

Titlebar:

Set to TRUE to include a titlebar on your new CitectSCADA graphics page.

Linked:

Set to TRUE to link the page to the library.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageSave, PageSaveAs, PageClose

Example

```
' Creates a new CitectSCADA graphics page
GraphicsBuilder.PageNew "include", "standard", "blank", 2, True, True
```

PageNewEx

Creates a new symbol, Genie or Supergenie page.

Syntax

PageNewEx(*PageType*)

PageType:

Specifies the type of page you would like to create:

- 0 = Symbol
- 1 = Genie
- 2 = Supergenie

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageSave, PageSaveAs, PageClose

Example

```
' Creates a symbol as a new graphics page
GraphicsBuilder.PageOpenEx "Example", "boiler", "tubes1", 0

' Creates a Genie as a new graphics page
GraphicsBuilder.PageOpenEx "Example", "example", "dial", 1

' Creates a Supergenie as a new graphics page
GraphicsBuilder.PageOpenEx "Example", "utility", "!sysinfo", 2
```

PageNewLibrary

Creates a new library. Fails, if project is read-only or not valid.

Syntax

PageNewLibrary(*Project, Library, LibraryType*)

Project:

The name of the project where the library is created.

Library:

The new library name (or style for templates).

LibraryType:

Type:

- 0 = Symbol
- 1 = Genie

- 2 = Supergenie
- 3 = Template

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageSave, PageSaveAs, PageClose

Example

```
' Creates a new symbol library
GraphicsBuilder.PageNewLibrary "Example", "newlibrary", 0

' Creates a new Genie library
GraphicsBuilder.PageNewLibrary "demo", "newlibrary", 1

' Creates a new Supergenie library
GraphicsBuilder.PageNewLibrary "Example", "newlibrary", 2

' Creates a new template style
GraphicsBuilder.PageNewLibrary "Example", "newstyle", 3
```

PageNewTemplate

Creates a new CitectSCADA graphics page template.

PageNewTemplate(*Project, Style, Template, Resolution, Titlebar, Linked*)

Project:

The name of the project that will contain the template.

Style:

The style you would like to apply to your new template.

Template:

The name you would like to give to your new template.

Resolution:

Sets the appropriate resolution for the template being created.

- 0 = Default
- 1 = VGA
- 2 = SVGA
- 3 = XGA
- 4 = SXGA
- 5 = User

Titlebar:

Set to TRUE to include a titlebar on the template.

Linked:

Set to TRUE to link the page to the library.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageSave, PageSaveAs, PageClose

Example

```
' Creates a new CitectSCADAgraphics page template
GraphicsBuilder.PageNewTemplate "include", "standard", "blank", 2, True, True
```

PageOpen

Opens an existing CitectSCADA graphics page.

Syntax

PageOpen(*Project, Page*)

Project:

The name of the project that contains the page you would like to open.

Page:

The name of the page you would like to open.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageNew, PageSave, PageSaveAs, PageClose

Example

```
' Opens an existing CitectSCADA graphics page
GraphicsBuilder.PageOpen "Example", "Genies"
```

PageOpenEx

Opens the specified symbol, Genie or Supergenie page, if it is found. See BrokenLinkCancelEnabled for more information if a missing reference is encountered.

Syntax

PageOpenEx(*Project, Library, Element, PageType*)

Project:

The name of the project where the element can be found.

Library:

The name of the library where the element can be found.

Element:

The name of the symbol, Genie or Supergenie.

PageType:

Type:

- 0 = Symbol
- 1 = Genie
- 2 = Supergenie

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageNew, PageSave, PageSaveAs, PageClose

Example

```
' Opens a symbol saved as a graphics page
GraphicsBuilder.PageOpenEx "Example", "boiler", "tubes1", 0

' Opens a Genie saved as a new graphics page
GraphicsBuilder.PageOpenEx "Example", "example", "dial", 1

' Opens a Supergenie saved as a graphics page
GraphicsBuilder.PageOpenEx "Example", "utility", "!sysinfo", 2
```

PageOpenTemplate

Opens a specified graphics page template.

Syntax

PageOpenTemplate(*Project, Style, Template, Resolution, Titlebar*)

Project:

The name of the project that contains the template.

Style:

The style of the template you would like to open.

Template:

The name of the template you would like to open.

Resolution:

The resolution of the template.

- 0 = Default
- 1 = VGA
- 2 = SVGA
- 3 = XGA
- 4 = SXGA
- 5 = User

Titlebar:

Set to TRUE to select a titlebar.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageSave, PageSaveAs, PageClose

Example

```
' Opens a graphics page template
GraphicsBuilder.PageOpenTemplate "include", "standard", "blank", 2, True
```

PagePrint

Prints the current page.

Syntax

PagePrint

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectFirstObject, PageSelectNextObject

PageSave

Saves the page using its current name.

Syntax

PageSave

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageNew, PageSaveAs, PageClose

Example

```
' Saves an existing CitectSCADA graphics page
GraphicsBuilder.PageSave
```

PageSaveAs

Saves the page with a new name within a specified project.

Syntax

PageSaveAs(*Project, Page*)

Project:

The name of the project you would like to save the page to.

Page:

The name you would like to apply to the page.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageNew, PageOpen, PageSave, PageClose

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Saves a CitectSCADA graphics page
GraphicsBuilder.PageSaveAs "Example", "MyPage"
```

PageSaveAsEx

Saves a symbol, Genie, Supergenie or template page to the specified location.

Syntax

PageSaveAsEx(*Project, Library, Element*)

Project:

The name of the project where the element is to be saved.

Library:

The name of the library (or style for templates) where the element is to be saved.

Element:

The new name for the symbol, Genie, Supergenie, or template to be saved.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageNew, PageSave, PageSaveAs, PageClose

Example

```
' Renames and saves the currently selected element to the specified location
GraphicsBuilder.PageSaveAsEx "Example", "TestLibrary", "TestObject"
```

PageSelect

Selects an opened page.

Syntax

PageSelectFirst

sBase - Base name
sFile - File name

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Example()
 Dim gb As GraphicsBuilder.GraphicsBuilder
 .
 .
 .
 gb.PageSelect("BaseName", "Filename")
 End Sub
```

Related Functions

PageSelectNext

PageSelectFirst

Selects the first page currently open in the Graphics Builder. Fails if there are no pages open.

Syntax

PageSelectFirst

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectNext

Example

```
' Selects the first page in Graphics Builder
GraphicsBuilder.PageSelectFirst
If Err.Number <> 0 Then
        Output.Print "PageSelectFirst Error" + "; Err.Number = " + Hex(Err.Number)
Else
        Output.Print "PageSelectFirst OK"

End If
```

PageSelectFirstObject

Selects the first object on the active page, based on its z-order number. This will exit and return an error if no object exists.

Syntax

PageSelectFirstObject

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

```
PageSelectObject, PageSelectNextObject, PageDeleteObject, PageSelectFirstObjectEx
```

PageSelectFirstObjectEx

Selects the first object on the active page, based on its z-order number. This will exit and return an error if no object exists. This function will not iterate through the items of a group.

Syntax

PageSelectFirstObjectEx

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObject, PageSelectNextObject, PageDeleteObject, PageSelectFirstObject

PageSelectFirstObjectInGenie

Select the first sub-object in the currently selected genie

Syntax

PageSelectFirstObjectInGenie

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PageSelectFirstObjectInGenie()
gb.PageSelectNextObjectInGenie()
gb.PageTemplateSelectFirstObject()
gb.PageTemplateSelectNextObject()
gb.UnLockObject()
End Sub
```

Related Functions

PageSelectNextObjectInGenie

PageSelectFirstObjectInGroup

Selects the first object in a group.

Syntax

PageSelectFirstObjectInGroup

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObject, PageSelectNextObject, PageDeleteObject, PageSelectFirstObject

PageSelectNext

Selects the next page currently open in the Graphics Builder. Fails if there are no pages open.

Syntax

PageSelectNext

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectFirst

Example

```
' Selects the next page in Graphics Builder
GraphicsBuilder.PageSelectNext
If Err.Number <> 0 Then
        Output.Print "PageSelectNext Error" + "; Err.Number = " +
       Hex(Err.Number)
Else
        Output.Print "PageSelectNext OK"
End If
```

PageSelectNextObject

Selects the next object on the active page, based on its z-order number. This function will exit and return an error if no object exists. An object can also be a group object, in which case this function will iterate through the items of a group.

Syntax

PageSelectNextObject

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObject, PageSelectFirstObject, PageDeleteObject, PageSelectNextObjectEx

PageSelectNextObjectEx

Selects the next object on the active page, based on its z-order number. This function will exit and return an error if no object exists. This function will not iterate through the items of a group.

Syntax

PageSelectNextObjectEx

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObject, PageSelectFirstObject, PageDeleteObject, PageSelectNextObjectEx

PageSelectNextObjectInGenie

Select the next sub-object in the currently selected genie.

Syntax

PageSelectNextObjectInGenie

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PageSelectFirstObjectInGenie()
gb.PageSelectNextObjectInGenie()
gb.PageTemplateSelectFirstObject()
gb.PageTemplateSelectNextObject()
gb.UnLockObject()
End Sub
```

Related Functions

PageSelectFirstObjectInGenie

PageSelectNextObjectInGroup

Selects the next object in a group.

Syntax

PageSelectNextObjectInGroup

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectFirstObjectInGroup

PageSelectObject

Selects an object using a specified AN number.

Syntax

PageSelectObject(*AN*)

AN:

The AN number of the object you would like to select.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObjectAdd, PageSelectFirstObject, PageSelectNextObject, PageDeleteObject

PageSelectObjectAdd

Selects an additional object using a specified AN number. This can be used to select multiple objects for a succeeding PageGroupSelectedObjects operation.

Syntax

PageSelectObjectAdd(*AN*)

AN:

The AN number of the object you would like to select.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObject, PageSelectFirstObject, PageSelectNextObject, PageGroupSelectedObjects

PageTemplateSelectFirstObject

Restart the template enumeration sequence.

Syntax

PageTemplateSelectFirstObject

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PageSelectFirstObjectInGenie()
gb.PageSelectNextObjectInGenie()
gb.PageTemplateSelectFirstObject()
gb.PageTemplateSelectNextObject()
gb.UnLockObject()
End Sub
```

Related Functions

PageTemplateSelectNextObject

PageTemplateSelectNextObject

Select the next object in the template.

Syntax

PageTemplateSelectNextObject

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Example

```
Public Sub Example()
Dim gb As GraphicsBuilder.GraphicsBuilder
.
.
.
gb.PageSelectFirstObjectInGenie()
gb.PageSelectNextObjectInGenie()
gb.PageTemplateSelectFirstObject()
gb.PageTemplateSelectNextObject()
gb.UnLockObject()
End Sub
```

Related Functions

PageTemplateSelectFirstObject

PageThumbnailToClipboard

Creates a thumbnail image of the current page and copies it to the clipboard.

Syntax

PageThumbnailToClipboard(*Size*)

Size:

The size of the thumbnail image in pixels.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

PageUngroupSelectedObject

Ungroups the currently selected grouped object.

Syntax

PageUngroupSelectedObject

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectObject, PageSelectObjectAdd, PageSelectFirstObject, PageSelectNextObject, PageGroupSelectedObjects

PageUpdated

The event fired when a Graphics Builder page is updated.

Syntax

PageUpdated(sProject, sPage)

sProject - the name of the project
sPage - the name of the page

Return Value

N/A

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageSelectNext

Example

```
Sub PageUpdated(ByVal bstrProject As String, ByVal bstrPage As String, ByVal bLastPage As Boolean)
' Add your code here
End sub
```

Page Properties Functions

Using the page properties functions, you can manipulate the properties of the pages in your project.

PageAddAssociation	Adds a new association to the current page. This function will return an error if an association with the specified name already exists.
PageAssociationDefault	Sets or retrieves the default for the currently selected page association.

PageAssociationDescription	Sets or retrieves the description for the currently selected page association.
PageAssociationName	Retrieves the name of the currently selected page association.
PageAssociationValueOnError	Sets or retrieves the value-on-error for the currently selected page association.
PageAppearanceGet	Retrieves the appearance properties of a page. This function, since it does not support True Color functionality, has been superseded by PageAppearanceGetEx.
PageAppearanceGetEx	Retrieves the appearance properties of a page. This function supports True Color functionality and replaces PageAppearanceGet.
PageArea	Retrieves or sets the PageArea property for the current graphics page.
PageClusterInherit	Retrieves or sets the cluster context inherit flag setting for current graphics page.
PageClusterName	Retrieves or sets the cluster context name property for the current graphics page.
PageDeleteAssociation	Deletes the selected association from the current page. After an item has been deleted, a call to PageSelectNextAssociation will select the item immediately following the deleted item.
PageDescription	Sets or retrieves the description attached to the active CitectSCADA graphics page.
PageEnvironmentAdd	Adds a new environment variable to the current page. This function will return an error if an environment variable with the specified name already exists.
PageEnvironmentFirst	Retrieves the first environment variable in the current page. This function will return an error if there are no environment variables in the page.
PageEnvironmentNext	Retrieves the next environment variable in the current page. This function will return an error if there are no more environment variables in the page.
PageEnvironmentRemove	Removes an environment variable from the current page. This function will return an error if an environment variable with the specified name does not exist.
PageLogDevice	Retrieves or sets the LogDevice property setting for the current graphics page.
PageName	Returns the name of the active page. This is a read only attribute.

PageNext	Retrieves the name of the page currently defined as "next" for the active graphics page, or sets the page you would like defined as next.
PagePrevious	Retrieves the name of the page currently defined as "previous" to the active graphics page, or sets the page you would like defined as previous to the current page.
PageScanTime	Retrieves or sets the PageScanTime property for the current graphics page.
PageSelectAssociationByName	Selects the specified association in the current page.
PageSelectFirstAssociation	Selects the first association in the current page.
PageSelectNextAssociation	Selects the next association in the current page.
PageTitle	Sets or retrieves the title of the active CitectSCADA graphics page.

For details and a VB example on handling return and error values, see Error Handling.

PageAddAssociation

Adds a new association to the current page. This function will return an error if an association with the specified name already exists.

Syntax

PageAddAssociation(*Name*)

Name:

The name of the new association to be added to the current page.

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Example

Adding a new element and setting its properties:

```
GraphicsBuilder.PageAddAssociation("MyAssociation")
GraphicsBuilder.SelectAssociationByName("MyAssociation")
```

```
GraphicsBuilder.PageAssociationsDefault = "TAG0"
GraphicsBuilder.PageAssociationValueOnError = "Oops"
GraphicsBuilder.PageAssociationDescription = "My Association"
```

Related Functions

PageAssociationDefault,PageAssociationDescription

PageAppearanceGet

Retrieves the appearance properties of a page.

> **Note:** As this function does not support True Color functionality, this function has been superseded by the function PageAppearanceGetEx.

Syntax

PageAppearanceGet(*Project, Style, Template, Resolution, Titlebar, Width, Height, Colour*)

Project:

The name of the project that contains the template.

Style:

The style of the template.

Template:

The name of the template.

Resolution:

The resolution of the template.

- 0 = Default
- 1 = VGA
- 2 = SVGA
- 3 = XGA
- 4 = SXGA
- 5 = User

Titlebar:

TRUE if titlebar is selected.

Width:

The width of the page in pixels.

Height:

The height of the page in pixels.

Colour:

The color of the page background.

Return Value

The requested values, as a string

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageSave, PageSaveAs, PageClose

PageAppearanceGetEx

Retrieves the appearance properties of a page.

Syntax

PageAppearanceGetEx(*Project, Style, Template, Resolution, Titlebar, Width, Height, OnColour, OffColour*)

Project:

Name of project that contains the template.

Style:

Style of template.

Template:

Name of template.

Resolution:

Resolution of template.

- 0 = Default
- 1 = VGA
- 2 = SVGA
- 3 = XGA
- 4 = SXGA
- 5 = User

Titlebar:

TRUE if titlebar is selected.

Width:

Width of the page in pixels.

Height:

Height of the page in pixels.

OnColour:

"On" color of the page background as an RGB value.

OffColour:

"Off" color of the page background as an RGB value.

Return Value

The requested values, as a string

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageOpen, PageSave, PageSaveAs, PageClose

PageArea

Retrieves or sets the PageArea property for the current graphics page.

Syntax

PageArea(*Area*)

Area:

1... 255 as a string, or blank to assign the page to every area.

Return Value

If retrieving the current PageArea setting, 1... 255 is returned as a string if a numeric value is used, or a group name is returned as a string if security has been set up using a preconfigured group. A blank string is returned if the active page is assigned to every area.

If you are using the function to apply an area setting, 0 (zero) is returned if successful, or an E_INVALIDARG error if the value you want to apply is out of range.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageName, PageTitle, PageDescription, PagePrevious, ProjectNext, PageScanTime, PageLogDevice

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Assigns a page to one of the areas defined in a CitectSCADA project
GraphicsBuilder.PageArea = "1"

' Retrieves the name of the area the current page is assigned to
MyVariable = GraphicsBuilder.PageArea
```

Note: This function is implemented in the C++ environment as two separate functions: `put_PageArea` applies an Area setting for active graphics page, and `get_PageArea` retrieves the current Area setting.

PageAssociationDefault

Sets or retrieves the default for the currently selected page association.

Syntax

Def=**PageAssociationDefault**

PageAssociationDefault(*Def*)

Def:

Default substitution string to be used if the page association has not been performed using the Ass(..) Cicode function at runtime. The default needs to be either a literal string enclosed in single quotes (e.g. 'a literal value') or a valid tag name.

Return Value

The default for the currently selected association (as a string), or 0 (zero) if successfully used to set the default. An error is returned if unsuccessful.

Related Functions

PageAddAssociation,PageAssociationDescription

PageAssociationDescription

Sets or retrieves the description for the currently selected page association.

Syntax

Description = PageAssociationDescription

PageAssociationDescription(Description)

Description:

Free text description of the association.

Return Value

The description of the currently selected association (as a string), or 0 (zero) if successfully used to set the description. An error is returned if unsuccessful.

Related Functions

PageAddAssociation

PageAssociationName

Retrieves the name of the currently selected page association.

Syntax

Name = PageAssociationName

Return Value

The name of the currently selected association (as a string). An error is returned if unsuccessful.

Related Functions

PageAddAssociation,PageAssociationDescription

PageAssociationValueOnError

Sets or retrieves the value-on-error for the currently selected page association.

Syntax

ValOnErr = PageAssociationValueOnError

PageAssociationValueOnError(ValOnErr)

ValOnErr:

Value to be used if the substitution was not performed and a default value was not defined, or a tag name was specified that did not resolve.

Return Value

The value-on-error for the currently selected association (as a string), or 0 (zero) if successfully used to set the value-on-error. An error is returned if unsuccessful.

Related Functions

PageAddAssociation,PageAssociationDescription

PageClusterInherit

Retrieves or sets the Cluster context inherit flag property setting for the current graphics page.

Syntax

PageClusterInherit(*bInherit*)

bInherit:

The setting of the cluster context inherit flag as a boolean value.

Return Value

The cluster context inherit flag for the active graphics page (as a boolean value), or 0 (zero) if successfully used to set the inherit flag. In both cases, an error is returned if unsuccessful.

Related Functions

PageClusterName

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Sets the cluster context inherit flag for current page
GraphicsBuilder.PageClusterInherit = "1"

' Retrieves the cluster context inherit flag for current page
MyVariable = GraphicsBuilder.PageClusterInherit
```

PageClusterName

Retrieves or sets the Cluster context name property setting for the current graphics page.

Syntax

PageClusterName(*ClusterName*)

ClusterName:

The name of the cluster as a string.

Return Value

The ClusterName setting for the active graphics page (as a string), or 0 (zero) if successfully used to set the ClusterName. In both cases, an error is returned if unsuccessful.

Related Functions

PageClusterInherit

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Sets the cluster context name property setting for current page
GraphicsBuilder.PageClusterName = "Cluster1"
' Retrieves the cluster context name property setting for current page
MyVariable = GraphicsBuilder.PageClusterName
```

PageDeleteAssociation

Deletes the selected association from the current page.

After an item has been deleted, a call to PageSelectNextAssociation will select the item immediately following the deleted item.

Syntax

PageDeleteAssociation()

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Example

Deleting any associations starting with “a”:

```
Dim name As String
On Error Resume Next
Err.Clear()
GraphicsBuilder.SelectFirstAssociation()
While (Err.Number = 0)
name = GraphicsBuilder.PageAssociationName
If (name.ToLower().StartsWith("a")) Then
GraphicsBuilder.PageDeleteAssociation()
End If
GraphicsBuilder.PageSelectNextAssociation()
End While
```

Related Functions

PageSelectNextAssociation

PageDescription

Sets or retrieves the description attached to the active CitectSCADA graphics page.

Syntax

PageDescription(*Description*)

Description:

The description applied to the active graphics page, as a string.

Return Value

The description of the active graphics page as a string, or 0 (zero) if successfully used to apply a description to the active graphics page. In both cases, an error is returned if unsuccessful.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageName, PageTitle, PagePrevious, ProjectNext, PageScanTime, PageLogDevice, PageNext, PageArea

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Attaches a description to the active graphics page
GraphicsBuilder.PageDescription = "MyDescription"

' Retrieves the description for the active graphics page
MyVariable = GraphicsBuilder.PageDescription
```

Note: This function is implemented in the C++ environment as two separate functions: `put_PageDescription` sets the title of the active graphics page, and `get_PageDescription` retrieves the title of the active graphics page.

PageEnvironmentAdd

Adds a new environment variable to the current page.This function will return an error if an environment variable with the specified name already exists

Syntax

PageEnvironmentAdd(name, value)

Name:

Specifies the name of the new environment variable.

Value:

Specifies the value to associate with the new environment variable

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Adding a new environment variable:

```
GraphicsBuilder.PageEnvironmentAdd("Foo", "Bar")
```

Related Functions

PageEnvironmentFirst

PageEnvironmentFirst

Retrieves the first environment variable in the current page. This function will return an error if there are no environment variables in the page.

Syntax

PageEnvironmentFirst(name, value)

Name:

Receives the name of the first environment variable in the current page.

Value:

Receives the value associated with the first environment variable.

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Printing out environment variables

```
Dim name As String
Dim value As String
Dim prevName As String
On Error Resume Next
Err.Clear()
GraphicsBuilder.PageEnvironmentFirst(name, value)
While (Err.Number = 0)
Console.Out.WriteLine(name + "=" + value)
prevName = name
GraphicsBuilder.PageEnvironmentNext(prevName, name, value)
End While
```

Related Functions

PageEnvironmentAdd

PageEnvironmentNext

Retrieves the next environment variable in the current page. This function will return an error if there are no more environment variables in the page.

Syntax

PageEnvironmentNext(*currentName, nextName, nextValue*)

currentName:

Specifies the name of the current environment variable.

nextName:

Receives the name of the next environment variable.

nextValue:

Receives the value associated with the next environment variable.

Return Value

0 (zero) if successful, otherwise an error is returned.

Related Functions

PageEnvironmentFirst

PageEnvironmentRemove

Removes an environment variable from the current page. This function will return an error if an environment variable with the specified name does not exist.

Syntax

PageEnvironmentRemove(*name*)

name:

Specifies the name of the environment variable to be removed.

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Deleting an existing environment variable

```
GraphicsBuilder.PageEnvironmentRemove("Foo")
```

Updating an existing environment variable

```
GraphicsBuilder.PageEnvironmentRemove("Foo")
GraphicsBuilder.PageEnvironmentAdd("Foo", "Bar2")
```

Related Functions

PageEnvironmentFirst

PageLogDevice

Retrieves or sets the LogDevice property setting for the current graphics page.

Syntax

PageLogDevice(*LogDevice*)

LogDevice:

The name of the log device as a string.

Return Value

The LogDevice setting for the active graphics page (as a string), or 0 (zero) if successfully used to set the LogDevice. In both cases, an error is returned if unsuccessful.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageName, PageTitle, PageDescription, PagePrevious, PageNext, PageScanTime

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Sets the LogDevice for the current graphics page
GraphicsBuilder.PageLogDevice = "MyDevice"

' Retrieves the name of the LogDevice for the current page
MyVariable = GraphicsBuilder.PageLogDevice
```

Note: This function is implemented in the C++ environment as two separate functions: `put_PageLogDevice` applies an LogDevice setting for active graphics page, and `get_PageLogDevice` retrieves the current LogDevice setting.

PageName

Returns the name of the active page. This is a read only attribute.

Syntax

PageName(*PageName*)

PageName:

Returns the name of the active page as a string.

Return Value

The name of the active CitectSCADA graphics page as a string.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageTitle, PageDescription, PagePrevious, PageNext, PageArea, PageScanTime, PageLogDevice

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
Debug.Print "PageName"; GraphicsBuilder.PageName
```

PageNext

Retrieves the name of the page currently defined as "next" for the active graphics page, or sets the page you would like defined as next.

Syntax

PageNext(*PageName*)

PageName:

The name of the page defined as next for the active graphics page, as a string.

Return Value

The name of the page defined as next for the active graphics page (as a string), or 0 (zero) if successfully used to set the page that is defined as next. In both cases, an error is returned if unsuccessful.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

PageName, PageTitle, PageDescription, PagePrevious, PageArea, PageScanTime, PageLogDevice

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Defines a page as the one that follows the current page in a browse sequence
GraphicsBuilder.PageNext = "MyPage3"
' Retrieves the name of the page that follows the current page in a browse sequence
MyVariable = GraphicsBuilder.PageNext
```

Note: This function is implemented in the C++ environment as two separate functions: `put_PageNext` sets the page defined as next for the active graphics page, and `get_PageNext` retrieves the name of the next graphics page.

PagePrevious

Retrieves the name of the page currently defined as "previous" to the active graphics page, or sets the page you would like defined as previous to the current page.

Syntax

PagePrevious(*PageName*)

PageName:

The name of the page defined as previous for the active graphics page, as a string.

Return Value

The name of the page defined as previous to the active graphics page (as a string), or 0 (zero) if successfully used to set the page that is previous to the active graphics page. In both cases, an error is returned if unsuccessful.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageName, PageTitle, PageDescription, PageNext, PageArea, PageScanTime, PageLogDevice

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Defines a page as previous to the current page in a browse sequence
GraphicsBuilder.PagePrevious = "MyPage1"
' Retrieves the name for the page defined as previous to the current page
MyVariable = GraphicsBuilder.PagePrevious
```

Note: This function is implemented in the C++ environment as two separate functions: `put_PagePrevious` sets the page defined as previous to the active graphics page, and `get_PagePrevious` retrieves the name of the previous graphics page.

PageScanTime

Retrieves or sets the PageScanTime property for the current graphics page.

Syntax

PageScanTime(*ScanTime*)

ScanTime:

A value between 1 and 60000 as a string, or blank to set to default.

Return Value

If retrieving the current PageScanTime setting, the value returned is between 1 and 60000 as a string, or a blank string if set to default.

If you are using the function to apply a ScanTime setting, 0 (zero) is returned if successful, or an E_INVALIDARG error if the value you want to apply is out of range.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageName, PageTitle, PageDescription, PagePrevious, PageNext, PageLogDevice

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Assigns a ScanTime value to the current graphics page
GraphicsBuilder.PageScanTime = "2000"
```

```
' Retrieves the ScanTime setting for the current page
MyVariable = GraphicsBuilder.PageScanTime
```

Note: This function is implemented in the C++ environment as two separate functions: `put_PageScanTime` applies an ScanTime setting to active graphics page, and `get_PageScanTime` retrieves the current ScanTime setting.

PageSelectAssociationByName

Selects the specified association in the current page.

Syntax

PageSelectAssociationByName(Name)

Name

The name of the association to be selected.

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Determining whether an association with a particular name exists:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.SelectAssociationByName("MyAssociation")
If (Err.Number <> 0)
    ' The association does not exist
End If
```

Related Functions

PageAddAssociation

PageSelectFirstAssociation

Selects the first association in the current page.

Syntax

PageSelectFirstAssociation()

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Determine whether the current page has associations in the page properties:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.SelectFirstAssociation()
If (Err.Number <> 0)
    ' The page has no associations
End If
```

Related Functions

PageAddAssociation,PageAssociationDescription

PageSelectNextAssociation

Selects the next association in the current page.

Syntax

PageSelectNextAssociation()

Return Value

0 (zero) if successful, otherwise an error is returned.

Example

Print associations in the current page's properties:

```
On Error Resume Next
Err.Clear()
GraphicsBuilder.SelectFirstAssociation()
While (Err.Number = 0)
    Console.Out.WriteLine(GraphicsBuilder.PageAssociationName)
    GraphicsBuilder.SelectNextAssociation()
End While
```

Related Functions

PageAddAssociation,PageAssociationDescription

PageTitle

Sets or retrieves the title of the active CitectSCADA graphics page.

Syntax

PageTitle(*Title*)

Title:

The title of the active page as a string.

Return Value

The title of the active graphics page as a string, or 0 (zero) if successfully used to set the title of the active graphics page. In both cases, an error is returned if unsuccessful.

Note: For details on handling return and error values, see Error Handling.

Related Functions

PageName, PageDescription, PagePrevious, PageNext, PageArea, PageScanTime, PageLogDevice

See Also

"Page Properties - General" in the CitectSCADA User Guide

Example

```
' Sets the title of the active graphics page
GraphicsBuilder.PageTitle = "MyTitle"

' Retrieves the title of the active graphics page
MyVariable = GraphicsBuilder.PageTitle
```

Note: This function is implemented in the C++ environment as two separate functions: `put_PageTitle` sets the title of the active graphics page, and `get_PageTitle` retrieves the title of the active graphics page.

Project Functions

These functions operate on the project level. Some are actually initiated within Citect Project Editor or the Project Explorer. If they experience an error in Visual Basic, they throw an exception with a return value E_FAIL.

ProjectCompile	This function starts the CitectSCADA compiler with the current project.
ProjectFirst	Retrieves the name of the first project defined in CitectSCADA.
ProjectFirstInclude	Retrieves the name of the first included project defined for the current CitectSCADA project.
ProjectNext	Retrieves the name of the next project defined in CitectSCADA.
ProjectNextInclude	Retrieves the name of the next included project defined for the current CitectSCADA project.
ProjectPackDatabase	Packs the current project's database files.
ProjectPackLibraries	Packs the library files for the current CitectSCADA project.
ProjectSelect	Selects the passed project as the current project within Citect Explorer.
ProjectSelected	Retrieves the name of the project that is currently selected in CitectSCADA.
ProjectUpdatePages	Updates the pages for the current CitectSCADA project.
ProjectUpgrade	Performs a project upgrade on the current CitectSCADA project.
ProjectUpgradeAll	Performs a project upgrade on the CitectSCADA projects.

For details and a VB example on handling return and error values, see Error Handling.

ProjectCompile

This function starts the CitectSCADA compiler with the current project. There are currently no functions to check errors or trigger the compiler's cancel function.

Syntax

ProjectCompile

Return Value

0 (zero) if successful, otherwise an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase

Example

```
GraphicsBuilder.ProjectCompile
If Err.Number <> 0 Then
        Debug.Print "Error in ProjectCompile"
        Err.Clear
Else
        Debug.Print "ProjectCompile OK"
End If
```

ProjectFirst

Retrieves the name of the first project defined in CitectSCADA. Can be used in conjunction with ProjectNext to call the projects currently defined in CitectSCADA.

Syntax

ProjectFirst(*Project*)

Project:

The name of the project.

Return Value

The name of the first CitectSCADA project. If no project exists, an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

Example

```
On Error Resume Next
sProject = GraphicsBuilder.ProjectSelected
If Err.Number <> 0 Then
        Debug.Print "Error in ProjectSelected"
        Err.Clear
Else
        Debug.Print "Selected project:", sProject
End If
Debug.Print "list of projects:"
sProject = GraphicsBuilder.ProjectFirst
While Err.Number = 0
        Debug.Print sProject
        sProject = GraphicsBuilder.ProjectNext
Wend
```

ProjectFirstInclude

Retrieves the name of the first included project defined for the current CitectSCADA project. Can be used in conjunction with ProjectNextInclude to call the projects defined as included for current CitectSCADA project.

Syntax

ProjectFirstInclude(*Project*)

Project:

The name of the project.

Return Value

The name of the first include project for the current CitectSCADA project. If no project exists, an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

ProjectNext

Retrieves the name of the next project defined in CitectSCADA. Can be used with ProjectFirst to call projects currently defined in CitectSCADA.

Syntax

ProjectNext(*Project*)

Project:

The name of the project.

Return Value

The name of the next CitectSCADA project. If no project exists, an error is returned.

Note: For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

Example

```
On Error Resume Next
sProject = GraphicsBuilder.ProjectSelected
If Err.Number <> 0 Then
        Debug.Print "Error in ProjectSelected"
        Err.Clear
Else
        Debug.Print "Selected project:", sProject
End If
Debug.Print "list of projects:"
sProject = GraphicsBuilder.ProjectFirst
While Err.Number = 0
        Debug.Print sProject
        sProject = GraphicsBuilder.ProjectNext
Wend
```

ProjectNextInclude

Retrieves the name of the next included project defined for the current CitectSCADA project. Can be used with ProjectFirstInclude to call the projects defined as included for current CitectSCADA project.

Syntax

ProjectNextInclude(*Project*)

Project:

The name of the project.

Return Value

The name of the next include project for the current CitectSCADA project. If no project exists, an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

ProjectPackDatabase

Packs the current project's database files.

Syntax

ProjectPackDatabase

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** This function displays a cancel dialog. It will exit and report error code E_ABORT, if the cancel button is pressed. For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectCompile

Example

```
GraphicsBuilder.ProjectPackDatabase
If Err.Number <> 0 Then
        Debug.Print "Error in ProjectPackDatabase"
        Err.Clear
Else
        Debug.Print "ProjectPackDatabase OK"
End If
```

ProjectPackLibraries

Packs the library files for the current CitectSCADA project.

Syntax

ProjectPackLibraries

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** This function displays a cancel dialog. It will exit and report error code E_ABORT, if the cancel button is pressed. For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

Example

```
GraphicsBuilder.ProjectPackLibraries
If Err.Number <> 0 Then
        Debug.Print "Error in ProjectPackLibraries"
        Err.Clear
Else
        Debug.Print "ProjectPackLibraries OK"
End If
```

ProjectSelect

Selects the passed project as the current project within Citect Explorer.

Syntax

ProjectSelect(*Project*)

Project:

The name of the project.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

Example

```
GraphicsBuilder.ProjectSelect "Example"
```

ProjectSelected

Retrieves the name of the project that is currently selected in CitectSCADA.

Syntax

ProjectSelected(*Project*)

Project:

The name of the project.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

Example

```
On Error Resume Next
sProject = GraphicsBuilder.ProjectSelected
If Err.Number <> 0 Then
        Debug.Print "Error in ProjectSelected"
        Err.Clear
Else
        Debug.Print "Selected project:", sProject
End If
Debug.Print "list of projects:"
sProject = GraphicsBuilder.ProjectFirst
While Err.Number = 0
        Debug.Print sProject
        sProject = GraphicsBuilder.ProjectNext
Wend
```

ProjectUpdatePages

Updates the pages for the current CitectSCADA project. If you encounter missing references during the update, see BrokenLinkCancelEnabled.

Syntax

ProjectUpdatePages(*FastUpdate*)

FastUpdate:

Set to TRUE to enable a fast update.

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** This function displays a cancel dialog. It will exit and report error code E_ABORT, if the cancel button is pressed. For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectUpgradeAll, ProjectPackLibraries, ProjectPackDatabase, ProjectCompile

Example

```
GraphicsBuilder.ProjectUpdatePages True
If Err.Number <> 0 Then
        Debug.Print "Error in ProjectUpdatePages"
        Err.Clear
Else
        Debug.Print "ProjectUpdatePages OK"
End If
```

ProjectUpgrade

Performs a project upgrade on the current CitectSCADA project.

Syntax

ProjectUpgradeAll

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** This function displays a cancel dialog. It will exit and report error code E_ABORT, if the cancel button is pressed. For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgradeAll, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.ProjectUpgrade
If Err.Number <> 0 Then
```

```
        Debug.Print "Error in ProjectUpgrade"
        Err.Clear
Else
        Debug.Print "ProjectUpgrade OK"
End If
```

ProjectUpgradeAll

Performs a project upgrade on CitectSCADA projects. This function produces the same result as setting Upgrade=1 in the Citect.ini file.

Syntax

ProjectUpgradeAll

Return Value

0 (zero) if successful, otherwise an error is returned.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

ProjectSelect, ProjectSelected, ProjectFirst, ProjectNext, ProjectFirstInclude, ProjectNextInclude, ProjectUpgrade, ProjectPackLibraries, ProjectUpdatePages, ProjectPackDatabase, ProjectCompile

Example

```
On Error Resume Next
Err.Clear
GraphicsBuilder.ProjectUpgrade
If Err.Number <> 0 Then
        Debug.Print "Error in ProjectUpgrade"
        Err.Clear
Else
        Debug.Print "ProjectUpgrade OK"
End If
```

Text Property Functions

These functions allow you to read and modify the properties of the text objects in your project.

AttributeText	Sets the text for a text object, or retrieves the current text.
AttributeTextColour	Applies a color to the selected text, or retrieves the current font color setting.
AttributeTextOffColourEx	Applies the "off" color to the selected text, or retrieves the current font color setting.
AttributeTextOnColourEx	Applies the "on" color to the selected text, or retrieves the current font color setting.
AttributeTextFont	Applies a specific font to the selected text, or retrieves the font setting.
AttributeTextFontSize	Applies a font size to the selected text, or retrieves the current font size.
AttributeTextJustification	Applies a specific justification setting to selected text, or retrieves the current text justification value.
AttributeTextStyle	Sets a specific text style, or retrieves the current text style setting.

The following object functions are also valid for text objects:

Attribute3dEffects	Applies a 3D effect to an object, or retrieves the current 3D effect setting.
Attribute3dEffectDepth	Applies a level of depth to a 3D effect, or retrieves the current depth setting.
AttributeHiLightColour	Sets the highlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current highlight color setting.
AttributeLoLightColour	Sets the lowlight color applied to the 3D effects raised, lowered or embossed, or retrieves the current lowlight color setting.
AttributeShadowColour	Sets the shadow color when a shadowed 3D effect is used, or retrieves the current shadow color setting.

For details and a VB example on handling return and error values, see Error Handling.

AttributeText

Sets the text for a text object, or retrieves the current text.

Syntax

AttributeText(*Text*)

Text:

The text object's text as a string.

Return Value

If retrieving the current text for the object, the text is returned as a string. If setting the text, a 0 (zero) is returned if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeTextStyle, AttributeTextJustification, AttributeTextFont, AttributeTextFontSize, AttributeTextColour

Example

```
' Sets the text for the currently text object
GraphicsBuilder.AttributeText = "TestText"

' Retrieves text for the current text object
MyVariable = GraphicsBuilder.AttributeText
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeText` sets the text for the currently selected text object, and `get_AttributeText` retrieves the text for the current text object.

AttributeTextColour

Applies a color to the selected text, or retrieves the current font color setting.

> **Note:** As this function does not support True Color functionality, it has been superseded by the functions AttributeTextOnColourEx and AttributeTextOffColourEx.

Syntax

AttributeTextColour(*TextColour*)

TextColour:

A value between 0 and 255 representing the font color.

Return Value

If retrieving the current font color, a value between 0 and 255. If applying a particular font color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeText, AttributeTextStyle, AttributeTextJustification, AttributeTextFont, AttributeTextFontSize

Example

```
' Applies a color to the selected text
GraphicsBuilder.AttributeTextColour = 255

`Retrieves the current font color setting
MyVariable = GraphicsBuilder.AttributeTextColour
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeTextColour` applies a color to the currently selected text, and `get_AttributeTextColour` retrieves the current text color.

AttributeTextOffColourEx

Applies the "off" color to the selected text, or retrieves the current font color setting.

Syntax

AttributeTextOffColourEx(*TextColour*)

TextColour:

An RGB value.

Return Value

If retrieving the current font color, an RGB value. If applying a particular font color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeText, AttributeTextStyle, AttributeTextJustification, AttributeTextFont, AttributeTextFontSize

Example

```
' Applies a color to the selected text
GraphicsBuilder.AttributeTextOffColourEx = &hFF0000

`Retrieves the current font color setting
MyVariable = GraphicsBuilder.AttributeTextOffColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeTextOffColourEx` applies a color to the currently selected text, and `get_AttributeTextOffColourEx` retrieves the current text color.

AttributeTextOnColourEx

Applies the "on" color to the selected text, or retrieves the current font color setting.

Syntax

AttributeTextOnColourEx(*TextColour*)

TextColour:

An RGB value.

Return Value

If retrieving the current font color, an RGB value. If applying a particular font color, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function exits and reports the error E_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E_HANDLE.

> **Note:** For details on handling return and error values, see Error Handling.

Related Functions

AttributeText, AttributeTextStyle, AttributeTextJustification, AttributeTextFont, AttributeTextFontSize

Example

```
' Applies a color to the selected text
GraphicsBuilder.AttributeTextOnColourEx = &hFF0000

`Retrieves the current font color setting
MyVariable = GraphicsBuilder.AttributeTextOnColourEx
```

This function is implemented in the C++ environment as two separate functions: `put_AttributeTextOnColourEx` applies a color to the currently selected text, and `get_AttributeTextOnColourEx` retrieves the current text color.

AttributeTextFont

Applies a specific font to the selected text, or retrieves the font setting.

Syntax

AttributeTextFont(*TextFont*)

TextFont:

The font name as a string.

Return Value

If retrieving the current font, the name of the font as a string, for example "courier". If applying a particular font, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeText, AttributeTextStyle, AttributeTextJustification, AttributeTextFontSize, AttributeTextColour

Example

```
' Applies the font Courier to the selected text
GraphicsBuilder.AttributeTextFont = "Courier"

' Retrieves the font setting
MyVariable = GraphicsBuilder.AttributeTextFont
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeTextFont` applies a font to the currently selected text, and `get_AttributeTextFont` retrieves the current font setting.

AttributeTextFontSize

Applies a font size to the selected text, or retrieves the current font size.

Syntax

AttributeTextFontSize(*TextFontSize*)

TextFontSize:

A value between 0 and 65535 representing the font size.

Return Value

If retrieving the current font size, a value between 0 and 65535. If applying a particular font size, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeText, AttributeTextStyle, AttributeTextJustification, AttributeTextFont, AttributeTextColour

Example

```
' Applies the font size to the selected text
GraphicsBuilder.AttributeTextFontSize = 12

' Retrieves the font size
MyVariable = GraphicsBuilder.AttributeTextFontSize
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeTextFontSize` sets the font size, and `get_AttributeTextFontSize` retrieves the current font size.

AttributeTextJustification

Applies a specific justification setting to selected text, or retrieves the current text justification value.

Syntax

AttributeTextJustification(*TextJustification*)

TextJustification:

A value depicting the type of justification used:

- 0 = left justified
- 1 = right justified
- 2 = centered

Return Value

If retrieving the current text justification, a value between 0 and 2 depicting the type of justification used. If applying justification, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeText, AttributeTextStyle, AttributeTextFont, AttributeTextFontSize, AttributeTextColour

Example

```
' Applies right justification to the selected text
GraphicsBuilder.AttributeTextJustification = 1

' Retrieves the current text justification value
MyVariable = GraphicsBuilder.AttributeTextJustification
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeTextJustification` applies justification to the currently selected text, and `get_AttributeTextJustification` retrieves the current justification setting.

AttributeTextStyle

Sets a specific text style, or retrieves the current text style setting.

Syntax

AttributeTextStyle(*TextStyle*)

TextStyle:

A value depicting text style:

- 0 = normal
- 1 = bold
- 2 = italic
- 4 = underline
- 8 = strikeout

You can superimpose styles by adding the above values.

Return Value

If retrieving the current text style, a value between 0 and 8 depicting the applied style. If applying a text style, 0 (zero) if successful. In both cases, an error is returned if unsuccessful. If values are out of range on writing to the attribute, the function will exit and report the error E_INVALIDARG. If there is no active text object, these functions throw an exception with a return value of E_HANDLE.

Note: For details on handling return and error values, see Error Handling.

Related Functions

AttributeText, AttributeTextJustification, AttributeTextFont, AttributeTextFontSize, AttributeTextColour

Example

```
' Sets the normal text style
GraphicsBuilder.AttributeTextStyle = 0

' Retrieves the current text style setting
MyVariable = GraphicsBuilder.AttributeTextStyle
```

Note: This function is implemented in the C++ environment as two separate functions: `put_AttributeTextStyle` applies a style to the currently selected text, and `get_AttributeTextStyle` retrieves the current text style setting.

Chapter: 7 Frequently Asked Questions

This section contains answers to some commonly asked questions about CitectSCADA functionality. The FAQs have been divided into several categories:

- Pages
- Graphics
- Runtime
- Trends
- Controls
- Alarms
- Miscellaneous

Pages

Q: How do I open a page on startup?

A:CitectSCADA searches for a page called "Startup" when it starts up. If CitectSCADA locates this page, it is opened automatically. You can change the name of the default startup page with the Computer Setup Wizard, run in Custom mode. See "General Options Setup" in the CitectSCADA User Guide for details.

Q: How do I get the Page Next and Page Previous buttons to work on new graphics pages?

A: Use the page Properties (File menu in the Graphics Builder) to define the next page and previous page.

Q: How do I display pages starting with '!'?

A: Turn on the List system pages option, under **Options** in the **Tools** menu of the Graphics Builder.

Q: What does the '!' mean when it is the first character in a page name?

A: The '!' means the page is a System page. Pages that begin with '!' will not appear on the default menu page or in the Page Select combo box.

Q: Is it possible to associate more than 8 variable tags to a Super Genie. I am using the`AssPopup()`function but it only allows me to use 8?

A: You can use the AssVarTags function to associate up to 256 variable tags, or use arrays.

Q: I have just created a Genie and want to add a privilege to it. In versions 3 and 4, I was able to hold down CTRL and double-click, but it doesn't seem to work in this version?

A: This is an effect of the addition of version 5 property-based objects. To add a privilege field, you need to add another field to the Genie form by adding `%privilege%` in the **Privilege** field of the Genie.

Graphics

Q: With the version 4 graphic objects, you could display the on/off state of strings in different colors. How can I do this with the new objects?

A: You can create the same effect by using the **On/Off** type in the Fill tab of a text object. Just add the digital tag again and specify the colors you want.

Q: Why are there dots displayed everywhere on my bar graphs and symbols?

A: The dots indicate a communication error for this object display. If the I/O Device associated with the data is offline, then it is displayed with dots over it.

Runtime

Q: How do I run a Cicode function on startup?

A: Specify the Cicode function with the Computer Setup Wizard - run in Custom mode. Use the **Startup FunctionsSetup** page. See "Startup Functions Configuration" in the CitectSCADA User Guide for details.

Q: How do I run a report on startup?

A:CitectSCADA searches for a report called "Startup" when it starts up. If CitectSCADA locates this report, it is run automatically. You can change the name of the default startup report with the Computer Setup Wizard - run in Custom mode. Use the Startup report field on the Reports Setup page. See "Reports Configuration" in the CitectSCADA User Guide for details.

Q: How do I disable operator reboot?

A: You can disable the Ctrl+Alt+Del command by using a third-party utility. See the Citect knowledge base or contact Technical Support for this product. to obtain the latest recommended software.

Q: How do I remove the Cancel button from the Startup Message Box?

A: Use the Computer Setup Wizard - run in Custom mode. De-select the **DisplayCancel** button on startup option on the Security Setup - Miscellaneous page. See "Miscellaneous Security Configuration" in the CitectSCADA User Guide for details.

Q: How do I remove the Shutdown command from the Control menu?

A: Use the Computer Setup Wizard - run in Custom mode. De-select the Shutdown on menu option on the Security Setup - Control Menu page. See "Control Menu Security Configuration" in the CitectSCADA User Guide for details.

Q: How do I remove the Project Editor and Graphics Builder commands from the Control Menu?

A: Use the Computer Setup Wizard - run in Custom mode. De-select the Project Editor/Graphics Builder on menu option on the Security Setup - Control Menu page. See "Control Menu Security Configuration" in the CitectSCADA User Guide for details.

Trends

Q: How do I get trend data into a dBASE database?

A: Display the trend on screen and select the File Save/As tool. This tool displays a Save/As dialog box for you to enter the name of the file, and calls the TrnExportDBF function to save the data. (For more complex procedures, call this function directly.)

Q: How do I get trend data into Excel?

A: You can get data into Excel in three ways:

- Display the trend on the screen, select the Clipboard tool to copy the data, and use the Excel paste command to paste the data into Excel.
- Display the trend on the screen and select the File Save/As tool. Save the data in CSV format, and open the CSV file in Excel.
- Display the trend on the screen and select the File Save/As tool. Save the data in DBF format and open the DBF file in Excel.

(For other procedures, call the TrnExportCSV or TrnExportDBF function.)

Q: How do I get trend data into MS Access?

A: You can get data into Access in three ways:

- Display the trend on the screen, select the Clipboard tool to copy the data, and use the Access Paste command to paste the data into Access.
- Display the trend on the screen and select the File Save/As tool. Save the data in CSV format, and open the CSV file in Access.
- Display the trend on the screen and select the File Save/As tool. Save the data in DBF format and open the DBF file in Access.

(For other procedures, call the TrnExportCSV or TrnExportDBF function.)

Q: How do I update trend data?

A: Use the TrnSetTable function to write data back to the trend system.

Q: How do I archive trend data?

A: Use the trend archive Cicode functions from the Examples database. Use the Find Function command to search for the `TrendArchive()` function.

Q: How do I restore archived trend data to the system?

A: Use the trend archive Cicode functions from the "Examples" database. Use the Find Function command to search for the `TrendArchive()` function.

Q: How do I get trend data into a report?

A: Use the TrnGetTable function.

Controls

Q: How do I start a motor with the keyboard?

A: Define a Page Keyboard command that sets the value of the digital variable to 1, for example:

- Key Sequence: **ENTER** or **F5**
- Command: `Conv_Motor = 1`

Q: How do I start a motor with a button?

A: Define a Button command that sets the value of the digital variable to 1 (for example Conv_Motor = 1).

Q: How do I adjust a setpoint from a keyboard entry?

A: Define a Page Keyboard command to set the setpoint to a new value, for example:

- Key Sequence: #### **ENTER**
- Command: **SP1 = Arg1**

Q: How do I enter a command that is bigger than the width of the field?

A: Use an Include file or write a Cicode function and call that function. For details, see Using Include (Text) Files and Writing Functions respectively.

Alarms

Q: How do I allow the operator to go to the alarm page from any page in the system using the keyboard?

A: Define a Global keyboard command (for example, the F3 key) to display the page with the PageAlarm function, or use a page template that has an Alarm Page button. Global keyboard commands are defined in System Keyboard Commands.

Q: How do I call a function when an alarm trips?

A: Define alarms as an alarm category, and call the function in the Alarm Action field.

Q: How do I send alarms to a dBASE file?

A: Specify the dBASE file in the Log Device field on the Alarm Category form.

Q: How do I display alarms?

A: Use the PageAlarm function to display the standard alarm page. Alternatively, for more control, draw your own alarm page and use the AlarmDsp function.

Q: How do I create a standard alarm page?

A: When you configure the project, create a default alarm page (with the Graphic Builder) based on the alarm template. Save the page with the name "Alarm". Alarms will automatically display on this page.

Q: How do I display the alarm summary?

A: Use the PageSummary function to display the standard alarm summary page. Alternatively, for more control, draw your own alarm summary page and use the AlarmDsp function.

Q: How do I get alarms into a report?

A: You can do either of the following:

- Read the alarm log (.DBF) files (logged for the alarm category).
- Use alarm functions such as AlarmFirstCatRec. (You can only use these functions if the alarms server and Reports server are on the same computer.)

Q: How do I disable groups of alarms from the I/O Device?

A: Program the I/O Device to set a bit when it wants to disable alarms. Use an event to monitor this bit (Trigger is Bit = 1), and call the AlarmDisable function as the Action (when the bit is set).

Q: How do I display alarm summaries?

A: When you configure the project, create a default alarm summary page (with the Graphic Builder) based on the alarm template. Save the page with the name "Summary". Alarms will display on this page.

Q: How do I acknowledge alarms?

A: Display the standard alarm page and click the left mouse button over the alarm, or use the AlarmAckRec function.

Q: How do I set up alarm redundancy?

A: Use the Computer Setup Wizard (run in Custom mode) to set up a second Alarm Server. No project reconfiguration is necessary.

Q: How do I attach comments to alarms at runtime?

A: Define a Page Keyboard command that calls the AlarmComment function.

Q: How do I sound a bell when alarm occurs?

A: Define the alarm in an Alarm Category and call the Beep function in the Alarm Action field.

Q: How do I display the last alarm on every page?

A: Define a continuous animation on each page (or template) - a standard feature of alarm templates.

Q: How do I advise operators that alarms are active?

A: If you are using standard templates, the clock animates. Alternatively, for more control, call the AlarmActive function.

Q: How do I change the analogue alarms limits at runtime?

A: Define a Page Keyboard command that calls the AlarmSetThreshold function.

Miscellaneous

Q: Why is #COM displayed on my pages?

A: The #COM indicates a communication error for this animation. If the I/O Device associated with the data is offline, #COM is displayed.

Q: I'm getting communication errors with my PLC (hardware alarms, #COMS, missing symbols or missing trend data on my pages). What do I do now?

A: Use the CitectSCADA Kernel window. (See "Using the CitectSCADA Kernel" in the CitectSCADA User Guide for details.)

1. Once the project is started, invoke the Kernel window on the client process.
2. In the Kernel Main window, type **page table CSAtoPSI.Subs** to bring up the list of client tag subscriptions.
3. Find the tag(s) that are causing #COM on the screen and look at the quality column to determine the error(s) involved.

Q: I get the error "Citect low on Physical memory" when I startup CitectSCADA**. What can I do about this error?**

A: On startup, CitectSCADA checks that you have enough available physical memory (real physical memory not virtual memory) to run your system. If CitectSCADA starts to use lots of virtual memory your system performance will be seriously affected. Under some conditions CitectSCADA cannot correctly detect the amount of physical memory and this alert message displays when in fact you do have enough memory. See the alert message Low physical memory for details.

Q: Why is my menu in VGA on an XGA resolution?

A: The default menu is a simple menu that puts buttons to every one of your pages on a VGA size page. If you want a better menu, configure your own page using the menu templates.

Q: I have a spare dongle but I do not know what type, point count or how many users it supports.

A: When running CitectSCADA, start the kernel and type **PAGE GENERAL** and view the bottom of the screen. There is information on the above questions.

Q: I have configured a few Events that are not running, but run on another machine with exactly the same project. What am I doing wrong?

A:CitectSCADA computers will only run Events if they are set up to do so. Use the Computer Setup Wizard to enable or disable events on each computer.

Q: How do I reset accumulators. I tried writing to the tag directly but it just keeps counting up?

A: Use the AccControl function.

Q: I have deleted a lot variable tags from my project but the number of records remain the same. It seems that even though I have deleted them they still exist?

A: This is true. You need to pack your database by selecting **Pack** from the File menu in the Project Editor. This deletes records marked for deletion and reindexes those that remain. pack regularly if you have been deleting or editing the Variables database file using third-party database editors (like MS-Excel).

Q: I want to set up a file server, but I would like the client connections to be as robust as possible if the server experiences an outage. What steps can I take?

A: Use the Computer Setup Wizard - run in Custom mode. Enter a standby location in the Backup project path field of the General Options Setup page. See "General Options Setup" in the CitectSCADA User Guide for details.

Q: How do I quickly set up communications to an I/O Device?

A: Use the Express Communications Wizard to select the I/O Server, manufacturer, then the I/O Device, then the communication method. This quickly sets up the basic options necessary, but does not set up advanced features. See "Using the Communications Express Wizard" in the CitectSCADA User Guide for details.

Glossary

1

10base2
Ethernet implementation on thin coaxial cable. Typically uses a BNC connection.

10base5
Ethernet implementation on thick coaxial cable.

10baseT
Ethernet implementation on unshielded twisted pair. Typically uses as RJ45 connection.

A

Accredited - Level 1
Drivers developed under the CiTDriversQA96 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 Driver Quality and Accreditation System.

accumulator
A 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 alarm
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.

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.

Alarms Server

Monitors all alarms and displays an alarm on the appropriate control client(s) when an alarm condition becomes active.

analog alarms

Triggered when an analog variable reaches a specified value. 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, 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.

Association

An association is the name or number you use when defining a Super Genie substitution, the value or values of which are dynamically generated at runtime.

attachment unit interface (AUI)

Typically used to interface to a transceiver through what is often known as a drop cable.

automation component (ActiveX object)

ActiveX objects typically consist of a visual component (which you see on your screen) and an automation component. The automation component allows the interaction between the container object and the ActiveX object.

B

baud rate

The number of times per second a signal changes in a communication channel. While the baud rate directly affects the speed of data transmission, the term is often erroneously used to describe the data transfer rate. The correct measure for the data rate is bits per second (bps).

BCD variable (I/O device)

BCD (Binary Coded Decimal) is a two-byte (16-bit) data type, allowing values from 0 to 9,999. The two bytes are divided into four lots of four bits, with each lot of four bits representing a decimal number. For example the binary number 0010 represents decimal 2. Thus the BCD 0010 0010 0010 0010 represents 2,222.

bottleneck

A bottleneck occurs when too many requests are being sent to a PLC communication link/data highway. It can occur with all types of protocols, and is dependent on several factors, including the frequency of requests, the number of duplicated (and hence wasteful) requests, whether the protocol supports multiple outstanding requests, as well as other network traffic.

browse sequence

A series of graphics pages linked by a browse sequence, which is a linear navigation sequence within your runtime system that uses Page Previous and Page Next commands.

byte variable (I/O device)

Byte is a one-byte data type, allowing values from 0 to 255. One byte consists of 8 bits. Each ASCII character is usually represented by one byte.

C

cache (I/O device data cache)

When caching is enabled, all data read from a I/O device is stored temporarily in the memory of the I/O server. If another request is made (from the same or another control client) for the same data within the cache time, the I/O server returns the value in its memory, rather than read the I/O device a second time.

callback function

A function that is passed as an argument in another function. Callback functions must be user-written functions.

Cicode

Programming language designed for plant monitoring and control applications. Similar to languages such as Pascal.

Cicode blocking function

A Cicode function that blocks, or waits, for an asynchronous event to complete before returning.

CiNet

CiNet is no longer supported. CiNet was designed as a low speed wide area network (for remote monitoring applications). If you have a widely-distributed application where computers are separated by vast distances, using a LAN to connect your control clients can be expensive. To connect control clients in this instance, use Microsoft's remote access server (RAS) or a Microsoft-approved solution, such as Shiva LanRover.

citect.ini file

A text file that stores information about how each computer (servers and control clients) operates in the configuration and runtime environments. The Citect.INI file stores parameters specific to each computer and therefore cannot be configured as part of the project.

CiUSAFE

CiUSAFE is the application used to manage the hardware key that authorizes use of your software within the agreed limitations.

client

A computer that accesses shared network resources provided by another computer called a server. '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 control 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).

computer

A computer running . Other common industry terms for this computer could be node, machine or workstation.

Control Client

The interface between the runtime system and an operator. If you are using on a network, all computers (on the network) are control clients.

control inhibit mode

Prohibits writing to the Field VQT tag element of a tag extension.

custom alarm filter

Custom alarm filters provide a way to filter and display active alarms. Up to eight custom filter strings can be assigned to a configured alarm. In conjunction with a user-defined query function, the custom filters enable operators to identify and display active alarms of interest.

D

data acquisition board

Data acquisition boards communicate directly with field equipment (sensors, controllers, and so on). You can install a data acquisition board in your 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 communications equipment (DCE)

Devices that establish, maintain, and terminate a data transmission connection. Normally referred to as a modem.

data terminal equipment (DTE)

Devices acting as data source, data sink, or both.

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 . You must specify the correct data type whenever I/O device variables are defined or referenced in your 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.

debug.log

The debug.log file stores information about an unexpected system shut down or other internal issues. If an unexpected shutdown occurs, it will identify the version and path of each DLL being used at the time.

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 modem

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. then returns the call using a dial-back modem.

dial-out modem

Makes calls to remote I/O devices in response to a 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 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 control client.Note: Don't use distributed servers to split up a single section or process into discrete areas. A single cluster system with distributed processing would be better used here since it would not be hampered by the maintenance overhead of a distributed server system (such as extra project compilations, and so on).

dither (imported bitmaps)

A method of approximating colors in imported or pasted bitmaps that involves combining pixels of different or colors from a color palette.

domain name server (DNS)

Database server that translates URL names into IP addresses.

dot notation

Used for Internet addresses. Dot notation consists of four fields (called octets), each containing a decimal number between 0 and 255 and separated by a full stop (.).

driver

A driver is used to communicate with control and monitoring devices, allowing the run-time system to interact directly with different types of equipment. Communication with an I/O device requires a device driver which implements the communication protocol(s).

driver logs

Driver logs relate to the operation of a particular driver and are named accordingly. For example, the OPC driver is logged in 'OPC.dat'.

duplex

The ability to send and receive data over the same communication line.

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 . The 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.

E

empty value

Indicates that the variant has not yet been initialized (assigned a value). Variants that are empty return a VarType of 0. Variables containing zero-length strings (" ") aren't empty, nor are numeric variables having a value of 0.

Ethernet

Widely used type of local area network based on the CSMA/CD bus access method (IEEE 802.3).

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, 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. .

expression

A statement (or group of statements) that returns a value. An expression can be a single variable, a mathematical formula, or a function.

F

Field element

The latest tag field data received from a device.

file server

A computer with a large data storage capacity dedicated to file storage and accessed by other client computers via a network. On larger networks, the file server runs a special network operating system. On smaller installations, the file server may run a PC operating system supplemented by peer-to-peer networking software.

full duplex

Simultaneous two-way (in both directions) independent transmission (4 Wires).

G

generic protocol

A pseudo-protocol supported by disk I/O devices that provides a convenient way to represent disk data. The generic protocol is not a real protocol (communicates with no physical I/O device).

Genie

If you have numerous devices of the same type (e.g., 100 centrifugal pumps), the display graphics for each will behave in much the same way. Using Genies, you only have to configure common behavior once. The graphics can then be saved as a Genie and pasted once for each device.

global Cicode variable

Can be shared across all Cicode files in the system (as well as across include projects).

global client

A control 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)

allows you to group multiple objects together. Each group has a unique set of properties, which determine the runtime behavior of the group as a whole.

H

half duplex

Transmission in either direction, but not simultaneously.

hardware alarm

A hardware alarm indicates that an error has been detected in your system. Typically displayed on a dedicated hardware alarms page, this type of alarm may indicate that a loss of communication has occurred, that Cicode can not execute, that a graphics page is not updating correctly, or that a server has become inoperative. A description and error code are provided to help decipher the cause of the problem.

histogram

A bar graph that shows frequency of occurrence versus value. Quite often the data is fitted to a distribution such as a normal distribution. .

I

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, 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). 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 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 . Variables are stored in memory registers in an I/O device. exchanges information with an I/O device by reading and writing variables. 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 control clients. No data processing is performed by the I/O server (except for its local display). Data is collected and passed to the control clients for display, or to another server for further processing. All data sent to an I/O device from any 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.

imestamp (T)

The timestamp of when the element was last updated on a tag extension.

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 commands or expressions that would otherwise be too long or complicated to type into the command or expression field within . The include file name is entered instead, and the whole file is activated when called.

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 projects over the Internet from a remote location. It is basically a "runtime-only" version of : you can run your project from that computer, just as you would from any normal client.

interrupt

An external event indicating that the CPU should suspend its current task to service a designated activity.

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.

K

Kernel

The Kernel allows you to perform low-level diagnostic and debugging operations for runtime analysis of your system. A set of diagnostic windows display low-level data structures, runtime databases, statistics, debug traces, network traffic, I/O device traffic and so on.

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 technical documentation that is updated regularly and available at http://www.citect.com.

kurtosis

An index indicating the degree of peakedness of a frequency distribution (usually in relation to a normal distribution). Kurtosis < 3 indicates a thin distribution with a relatively high peak. Kurtosis > 3 indicates a distribution that is wide and flat topped.

L

language database

When a project is compiled, 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 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.

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).

local variable

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

log files

Log files are a record of time-stamped system data that can be analyzed to determine the cause of a problem. The available log files include syslog.dat, tracelog.dat, debug.log, kernel.dat, and dedicated driver logs.

long BCD variable (I/O device)

A 4-byte (32-bit) data type, allowing values from 0 to 99,999,999. The four bytes are divided into eight lots of four bits, with each lot of four bits representing a decimal number. For example the binary number 0011 represents decimal 3. Thus the BCD 0011 0011 0011 0011 0011 0011 0011 0011 represents 33,333,333.

long variable (I/O device)

A 4-byte (32-bit) data type allowing values from 2,147,483,648 to 2,147,483,647.

low and low low alarms

Defined by specifying the values of the variable that trigger each of these alarms. As a low alarm must precede a low low alarm, the low alarm no longer exists when the low low alarm is triggered. Note that the variable must rise above the deadband before the alarm becomes inactive. .

M

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.

Metadata

Metadata is a list of names with corresponding values that is attached to an objects animation point.

millisecond trending

Allows you to use a trends sample period of less than one second.

mimic

A visual representation of a production system using an organised set of graphical pages. .

minimum update rate

A pre-defined period of time after which tag update value notifications are sent to subscription clients

module Cicode variable

Specific to the file in which the variable is declared. This means that it can be used by any function in that file, but not by functions in other files. By default, Cicode variables are defined as module, therefore prefixing is not required (though a prefix of MODULE could be added if desired). Module variables should be declared at the start of the file.

multi-digital alarms

Use combinations of values from three digital variables to define eight states. For each state, you specify a description (e.g., healthy or stopped), and whether or not the state triggers an alarm.

N

native language

Generally the language of the project developer. Display items such as alarm descriptions, button text, keyboard/alarm logs, graphic text, Cicode strings and so on can be configured in the native language, and displayed, at runtime, in the language of the end-user (local language).

network

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 that is connected to a LAN through a network adaptor card and network software. .

Network Dynamic Data Exchange (NetDDE)

Enables communication between Windows applications on separate computers connected across a common network.

nodes

A structural anchor point for a graphic object, usually visible as a small square box superimposed over a graphic. Nodes will be 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 = skewness = 1.0 2 = 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 return false.

O

object

Basic building blocks of a graphics page. Most objects possess properties that allow them to change dynamically under user-definable runtime conditions allowing them to provide animated display of conditions within the plant.

object ID (OID)

An object ID associated with every tag in a 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).

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.

override mode

A state where an invalid tag quality value is overridden by a manually added value.

P

pack

Packing a database re-indexes database records and deletes records marked for deletion. If you edit your databases externally to , 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, 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 be the same (even or odd) for each group of bits transmitted without error.

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).

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 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.

point limit
An individual digital (or analog) variable read from an I/O device. 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 license. When you run the point count of your project is checked against the point limit specified by your Hardware Key.

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.

Privileges
Level of access applied to system elements within your project. A user assigned a role that possesses the matching privilege can control it.

project
The elements of a 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 compiler to compile the project into a runtime system.

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).

protocol
Messaging format consisting of a set of messages and guidelines used for communication between the server and an I/O device. The communication protocol determines how and the I/O device communicate; the type of data to exchange; rules governing communication initiation and termination; and error detection.

proxi/proxy server
Caches internet transactions to improve performance by reducing the average transaction times by storing query and retrieved information for re-use when the same request is made again. When an Internet display client (IDC) connects to a proxy server, that server provides the TCP/IP addresses necessary to access report server session information.

PSTN

A public switched telephone network is the network of all the world's public switched telephone networks. It is now primarily digital and includes mobile as well as fixed telephones.

Q

qualified tag reference

Referencing tag data by using the tag name, element name and the item name.

Quality (Q)

The quality of the value of a tag extension.

QualityTimestamp (QT)

The timestamp of when the quality last changed on a tag extension

R

rate of change alarms

Triggered when the value of the variable changes faster than a specified rate. The alarm remains active until the rate of change falls below the specified rate. Deadband does not apply to a rate of change alarm.

real variable (Cicode)

Real (floating point) is a 4-byte (32-bit) data type allowing values from 3.4E38 to 3.4E38. Use a real variable to store numbers that contain a decimal place.

real variable (I/O device)

Real (floating point) is a 4-byte (32-bit) data type, allowing values from 3.4E38 to 3.4E38. Use a real variable to store numbers that contain a decimal place.

record name

Usually the primary property of a database record, referenced in system through its name. Database record names must be unique for each type of database record. Sometimes you can use identical names for different record types. However, to avoid confusion, you should use a unique name for each database record in your application.When you specify a name for a database record, the name must begin with an alphabetic character (A-Z, a-z) and cn only include alphanumeric characters (A-Z, a-z, 0-9) and the underscore character (_). For example, "Pressure," "Motor_10," and "SV122_Open" are all valid database record names. Each database record name can contain up to 16 characters.Database record names are not case-sensitive, so "MOTOR_1," "Motor_1" and "motor_1" are all identical database record names. For this reason use a meaningful name for any database record as well as the necessary naming conventions.

redundancy
A method of using the hardware in a system such that if one component in the system becomes inoperative, 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. 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.

Roles
A defined set of permissions (privileges and areas) that are assigned to users.

RS-232
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.

runtime system

The 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 project.

S

scalable architecture

A system architecture that can be resized without having to modify existing system hardware or software. lets you re-allocate tasks as more computers are added, as well as distribute the processing load.

schedule period

Determines how often the I/O server contacts a scheduled I/O device to read data from it. .

serial communication

Uses the communication port on your computer or a high speed serial board (or boards) installed inside your computer.

server

A computer connected to an I/O device (or number of I/O devices). When is running, the server exchanges data with the I/O device(s) and distributes information to the other control clients as required. A local area network (LAN) computer that perform processing tasks or makes resources available to other client computers. In , 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), then the tail is extended on that side, and there is more data on the left side of the graph than would be expected from a normal

distribution. Positive skew indicates the distribution's mean (and tail) is skewed to the right. Negative skew indicates the distribution's mean (and tail) is skewed to the left.

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.

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

uses a hardware key that plugs into the printer port of your computer to protect against license infringement. The hardware key contains the details of your user license. When you run , the point count in your project is checked against the point limit specified in the hardware key.

staleness period

Represents the total number of seconds that will elapse after the last update before extended quality of the tag element is set to “Stale”.

standby Alarms Server

The Server that processes alarms if the primary alarms server is unavailable.

standby Reports Server

The server that processes reports if the primary reports server is unavailable.

standby Trends Server

The 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.

substatus value

The underlying details of a QUALITY tag.

Substitution

A Super Genie substitution is comprised of the data type (optional) and association that you use to define an object or group of object’s properties when creating a Super Genie.

Super Genies

Dynamic pages (usually pop-ups), to which you pass information when the page displays at run-time. You can use Super Genies for pop-up type controllers (to control a process, or a single piece of plant floor equipment).

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.

syslog.dat

Syslog.dat is the primary log file. It contains useful system information, from low-level driver traffic and Kernel messages, to user defined messages. Trace options (except some CTAPI traces) are sent to this file.

T

tag extension

Additional information for a tag that represents data as a collection of elements, and a collection of items in a tag.

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. provides templates for all common page types.

text box

When text is added to a graphics page, it is placed in a text box. A text box has a number of handles, which can be used to manipulate the text object.

thread

Used to manage simultaneous execution of tasks in multitasking operating systems, enabling the operating system to determine priorities and schedule CPU access.

timeout

The period of time during which a task must be completed. If the timeout period is reached before a task completes, the task is terminated.

time-stamped alarms

An alarm triggered by a state change in a digital variable. Time-stamped alarms have an associated register in the I/O device to record the exact time when the alarm changes to active. Use time-stamped alarms when you need to know the exact order in which alarms occur.

time-stamped analog alarms

Time stamped analog alarms work in the same way as analog alarms except that they are time stamped (with the Alarm On and Alarm Off times) using millisecond precision from the time kept by the field device (i.e. the RTU or PLC). The configuration details for time stamped analog alarms are exactly the same as for analog alarms.

time-stamped digital alarms

Time stamped digital alarms work in the same way as digital alarms except that they are time stamped (with the Alarm On and Alarm Off times) using millisecond precision from the time kept by the field device (i.e. the RTU or PLC). The configuration details for time stamped digital alarms are exactly the same as for digital alarms.

tool tip

A help message that displays in a pop-up window when an operator holds the mouse stationary over an object.

touch (object at runtime)

An object is considered touched if an operator clicks it.

Touch command

Can be assigned to objects on graphics pages. Touch commands allow you to send commands to the runtime system by clicking an object.

tracelog.dat

The tracelog.dat file contains managed code logging, mainly in relation to data subscriptions and updates. Note that field traces and requests to native drivers go to the syslog.dat or a specific driver log file.

trend

A graphical representation of the changing values of a plant-floor variable (or expression), or a number of variables. .

trend line

The actual line on a trend that represents the changing values of a plant-floor variable (or expression). .

trend plot

Consists of a trend (or a number of trends), a title, a comment, scales, times and so on.

Trends Server

Controls the accumulation and logging of trend information. This information provides a current and historical view of the plant, and can be processed for display on a graphics page or printed in a report.

U

UAC

User Account Control. Security technology introduced in Windows Vista to enable users to run with standard user rights more easily. .

unqualified tag reference

Reference to tag data by using only the tag name.

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.

Users

A person or group of persons that require access to the runtime system

V

Valid element

The last field data which had “Good” quality in a tag extension.

Value (V)

The value of the extension of a tag.

ValueTimestamp (VT)

The timestamp of when the value last changed on a tag extension

variable type (Cicode)

The type of the variable (INT (32 bits), REAL (32 bits), STRING (256 bytes), OBJECT (32 bits)).

view-only 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.

virtual

Behavioral identification rather than a physical one. For example, Windows 95 is a virtual desktop.

W

wizard

A facility that simplifies an otherwise complex procedure by presenting the procedure as a series of simple steps.

Index

T